Cyberboswachters

Tim Dams

Inhoudsopgave

Preface

Dit handboek wordt gebruikt als basis-cursus binnen de opleidingen elektronica-ict en toegepaste informatica van de AP Hogeschool. Het heeft als doel een breed overzicht te geven van de *wereld van de cybersecurity*. Het doel is niet alomvattend te zijn, maar vooral om mensen zin te geven om meer te weten te komen over deze belangrijke én boeiende wereld. Dit boek gaat ervan uit dat de lezer minimale voorkennis heeft inzake ICT, zo wordt er verwacht dat een basiskenis netwerk-technologie aanwezig is (denk aan termen zoals IP-adressering, routering en firewalls).

Boswachters en stropers

De subtitel, *"De beste digitale stropers zijn ook de beste cyberboswachters"*, van dit handboek verdient een extra woordje uitleg. Dit boek zal zeker niet de obscure wereld van de hackers en cybercriminelen verheerlijken, integendeel. We willen echter wel tonen hoe de stropers, de slechterikken in dit boek, te werk gaan, opdat het ons zo een beter beeld geeft waar we ons tegen moeten beschermen als goede cyberboswachters.

Dankwoord

Dank aan (soms al oud-)studenten Ernie de Magtige, Dimitriy Vassilchenko, Jasper Van Meel en Erik Van Dyck om als externe fact- en spellingcheckers aardig wat (gênante) foutjes en typos te ontdekken.

Een oprechte dank aan Stefaan Somerling (RealDolmen) voor de feedback op het hoofdstuk omtrent GDPR. Geef gerust een sein als u, conform de GDPR wetgeving, liever uw naam niet in dit document ziet staan ;).

Een dikke fist bump voor collega en *brother from another mother* Koen Van Eyken, die een deel van z'n paasvakantie opofferde om met een kritische blik door dit boek te gaan! Uiteraard ook een dikke merci aan de collega's die feedback op dit document hebben gegeven, en dan zeker aan Michael Boeynaems en Serge Horsmans.

Waarschuwingen

Dit brengt ons automatisch bij een belangrijke waarschuwing: je zal in dit handboek geregeld technieken en tools tegenkomen die verregaande gevolgen voor derden én jezelf kunnen hebben indien ze misbruikt worden. We tonen deze zaken enkel vanuit het standpunt dat zonet werd besproken: digitale stropers zijn de beste cyberboswachters. Het is belangrijk dat je begrijpt dat je deze tools en technieken NOOIT of te NIMMER voor kwade doeleinden mag gebruiken. Het is zelfs zo dat het gebruik van veel van deze zaken strafrechtelijke gevolgen kunnen hebben, inclusief boetes tot zelfs jarenlange opsluiting.

Indien je dus deze tools of technieken wenst te gebruiken dan mag dit enkel op volgende doelwitten:

- Zaken waar jij de eigenaar van bent. Denk aan servers, webpagina's, gebruikers, etc.
- Op doelwitten waar je geen eigenaar van bent, maar waar je wel expliciete toestemming voor hebt gekregen.

Opletten met experimenten

Indien je van plan bent om met bepaalde tools te experimenteren, hou dan rekening met volgende tips:

- Werk zoveel mogelijk met virtuele machines en test nooit op een *live systeem* indien daar geen erg goede reden voor is.
- Hou er rekening mee dat bepaalde tools permanente "gaten" slagen in je systemen. Deze kunnen dus na (of tijdens!) je experimenten door digitale stropers misbruikt worden.

1. Wordt het erger?

i Leerdoelen

Na dit hoofdstuk kan je:

1. **De evolutie van cyberaanvallen in het voorbije decennium schetsen** aan de hand van kernmomenten (Stuxnet, Snowden, ransomware-golf, AI-dreigingen).
2. **Uitleggen waarom cyberaanvallen impact hebben ver buiten de digitale wereld** — van kritieke infrastructuur tot geopolitiek en democratie.
3. **De rol van AI aan beide zijden van de wapenwedloop duiden**: als offensief wapen én als verdedigingsinstrument.
4. **De paradox *"attack sophistication ↑ / vereiste aanvallerskennis ↓"* toepassen** op een gegeven scenario om de gevolgen voor de verdediging in te schatten.
5. **White, grey en black hat hackers onderscheiden** en de spelregels van ethisch hacken in België (sinds februari 2023) correct toepassen.

Het laatste decennium jaar is de (cyber)security wereld erg veranderd. Ze doet dit niet omdat ze daar zin in heeft, maar wel als antwoord op wat er gebeurt in de wereld omtrent cyberaanvallen, geopolitieke situaties, etc.

Het is altijd een kat en muisspel, waarbij de boswachters helaas bijna altijd zullen achterlopen op de stropers. De kwaadwillige hackers hoeven maar één klein gaatje te vinden in je peperdure beveiliging en ze zijn binnen. Terwijl jij als boswachter wel aan alles moet (proberen te) denken. Dat het erger wordt, is eigenlijk daarom bijna automatisch een evidentie. De wereld van de beveiliging gebeurt per opbod en hoe beter de boswachters het bos kunnen verdedigen, hoe complexer de technieken zullen worden die de stropers hanteren.

In de volgende secties geven we een klein historisch overzicht van enkele belangrijke, interessante of spectaculaire gebeurtenissen die hebben plaatsgevonden in de cyberwereld de voorbije jaren. Dit laat ons toe om enerzijds enkele begrippen te duiden, anders om de vraag "wordt het erger?" te beantwoorden.

i Opmerking

We gebruiken de term cyber om alles aan te duiden dat zich afspeelt op de digitale snelweg, namelijk het Internet, de *cloud*, het www, en alle synoniemen en aanverwanten.

1.1 Het voorbije decennium

We zullen geregeld terug in de tijd gaan om te bekijken hoe bepaalde cyberverdedigings- en aanvalstechnieken zijn ontstaan, maar in dit hoofdstuk gaan we enkel terug tot 2010. Het jaar waarin Mark Zuckerberg, CEO van Facebook, door Time Magazine tot persoon van het jaar werd uitgeroepen en ook waarin Apple de eerste IPad tablet aan het publiek toonde. Het was helaas ook het jaar waarin het boorplatform, Deepwater Horizon, voor één van de ergste milieurampen ooit zorgde, alsook het jaar dat 33 mijnwerkers anderhalve maand lang 700 meter diep opgesloten zaten in een mijn. Voor ons, als toekomstige cyberboswachters, is echter de meest cruciale gebeurtenis het ontdekken van **Stuxnet**.

1.1.1 Voor 2010

Figuur 1.1: "Het aardse paradijs met de zondeval van Adam en Eva" door Peter Paul Rubens en Jan Brueghel (de Oude). Duidelijk gebaseerd op een wereld van voor 2010.

Voor het ontdekken van Stuxnet in 2010 leek de cyberbeveiligingswereld wel een wereld van (relatieve) peis en vree. Het ergste dat kon gebeuren was dat je computer een virusje opdeed waardoor je mogelijk wat data, en vooral veel werkuren, kwijt raakte. Virussen, wormen en spam waren alomtegenwoordig maar waren eigenlijk niet meer dan luizen in de pels van de eindgebruikers. Tuurlijk, er werd een grondig gevloekt als een virus je foto's verwijderde of wanneer een worm zichzelf verspreidde naar je contacten - denk maar aan het *ILOVEYOU* virus dat als een soort *social engineer* mensen deed geloven dat ze een potentiële liefdesmatch hadden waarop ze vol spanning de bijlage openden en zo de worm *in huis haalden*. Of wat te denken van de *Conficker* worm die in 2008 meer dan 3 miljoen systemen kon besmetten en telkens de update-services van het Windows toestel uitschakelde. Uiteraard, dat was irritant, maar menig mens zou maar al te graag terug naar die tijd willen gaan als daarmee ransomware, botnets en door overheden gesponsorde cyberaanvallen onbestaande zouden zijn.
De termen worm, virus en malware zullen geregeld door elkaar worden gebruikt in deze cursus. Alle drie betekenen net niet hetzelfde, maar toch:

- Een **virus** is een kwaadaardig programma dat, eens het op een computer of apparaat staat, digitale schade kan aanbrengen.
- Een **worm** daarentegen is ook een virus, maar eentje dat zichzelf kan *voortplanten* naar andere systemen, iets wat een virus niet kan. Een worm kan zichzelf dus verspreiden zonder menselijke hulp.
- **Malware** is een overkoepelende term voor alle programma's en code die digitale schade aan een systeem of netwerk brengen. Virussen en wormen behoren dus tot deze groep.

Naast deze drie termen zullen we ook nog enkele andere malware types tegenkomen zoals Adware, spyware, spam, trojans, backdoors en rootkit. Wanneer nodig zullen we deze termen toelichten. Onthoud nu alvast dat malware de algemene naam is voor alle malafide software.

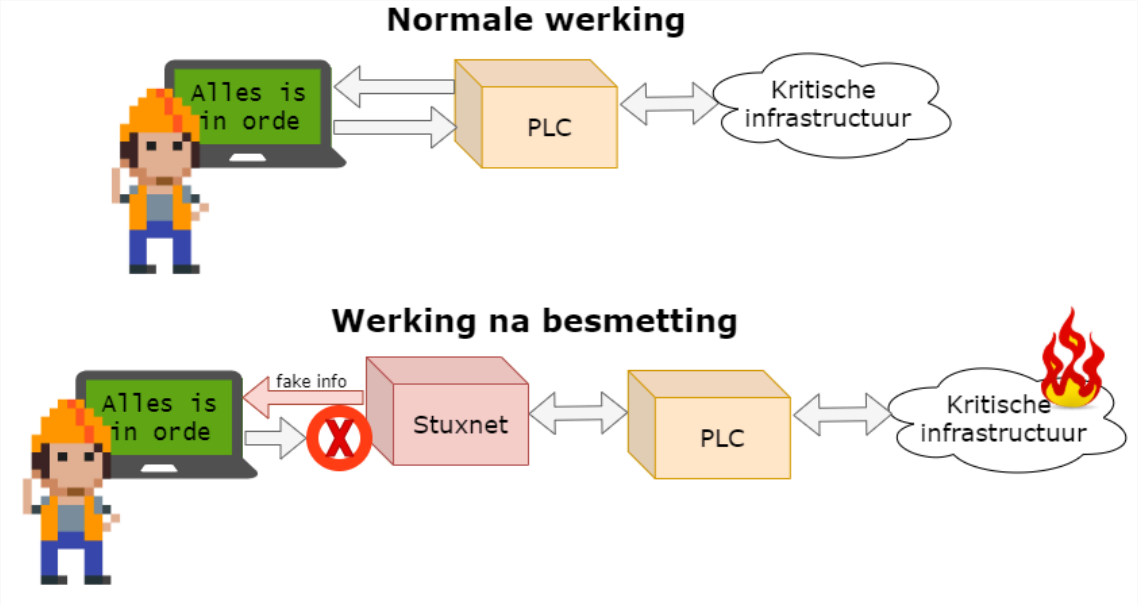
1.1.2 2010: En dan verscheen Stuxnet

Virussen konden computers (tijdelijk) onbruikbaar maken. OK, lastig, maar niet het einde van de wereld. De Stuxnet worm die in 2010 plots op de radar verscheen kon dat potentieel wel: de wereld beëindigen! Het was een worm die op maat was gemaakt om zogenaamde **PLCs** (*Programmable Logic Controllers*)

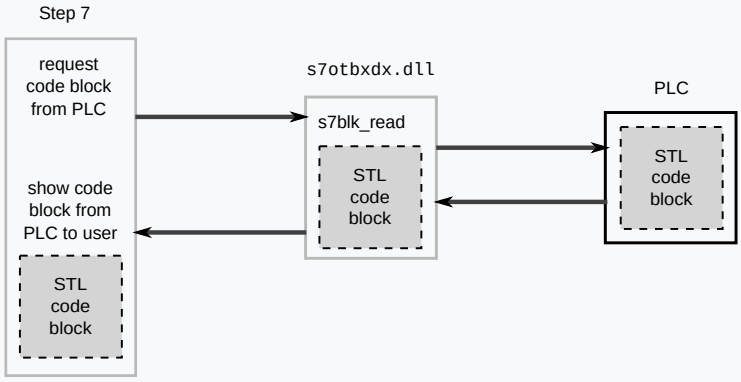
te controleren. Héél veel van onze kritische infrastructuur is geautomatiseerd met behulp van deze PLCs , krachtige apparaten die machines kunnen aansturen zoals liften, robotarmen aan assemblagelijnen, zuiveringsinstallaties en zelfs kernreactors. PLCs vormen de interface tussen machines (hardware) en de computers (met software op) die de operatoren gebruiken om deze machines opdrachten te geven. Operators kunnen op hun systemen de machines bedienen en te allen tijde controleren of deze naar behoren werken.

Wat Stuxnet deed was zich tussen de hardware en de software nestelen. Als een worm besmette het laptops en computers en zocht het of er PLC-bedieningssoftware aanwezig was op het toestel. Als dat het geval was dan plaatste het zichzelf er tussen in zodat het:

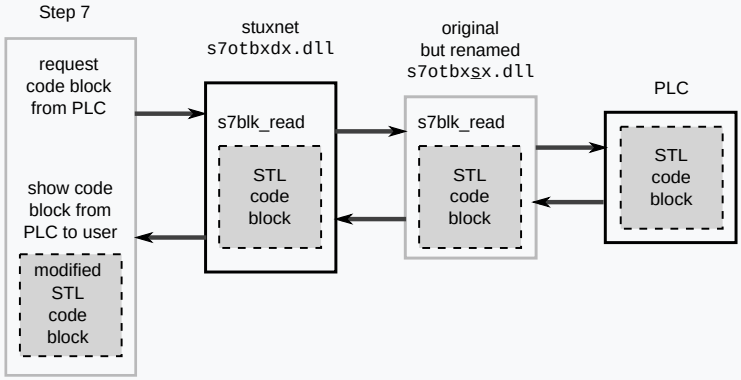
1. de machines opdrachten kon geven zonder dat de operator dit wist.
2. aan de operator kon vertellen dat alles in orde was, terwijl in realiteit de PLC mogelijk desastreuze opdrachten aan de hardware gaf.



Figuur 1.2: Werking stuxnet.



Figuur 1.3: Normale communicatie tussen Step 7-software en een Siemens PLC. Bron: Wikimedia Commons, auteur Grixlkraxl, CC BY-SA 3.0.



Figuur 1.4: Stuxnet plaatst een eigen DLL (`s7otbxdx.dll`) tussen Step 7 en de PLC, en verbergt daarmee zijn eigen STL-code voor de operator. Bron: Wikimedia Commons, auteur Grixlkraxl, CC BY-SA 3.0.

We moeten er geen tekeningetje bij maken wat de effecten kunnen zijn indien een Stuxnet variant bewust werd ingezet om bijvoorbeeld de machinerie in een kerncentrale of waterdam te saboteren. Zonder in details van Stuxnet in te gaan is het duidelijk dat het verschijnen van dit virus een ommekeer betekende qua potentiële gevolgen van een cyberaanval: mensenlevens stonden plots op het spel, niet enkel onze dierbare e-mails en digitale vakantiefoto's.

i Opmerking

Stuxnet bleek een worm te zijn die probeerde een specifiek soort centrifuge te saboteren: namelijk de centrifuges die Iran gebruikte om uranium te verrijken. De inlichtingendiensten van de Verenigde Staten en Israël zouden klaarblijkelijk Stuxnet hebben ontworpen om zo dit verrijkkingsproces van uranium in Iran te dwarsbomen.

! Belangrijk

Airgapped systems don't exist. De Iraanse Natanz-verrijkingsinstallatie was bewust níet met het Internet verbonden (*air-gapped*): een klassieke beveiligingsmaatregel voor kritieke infrastructuur. Toch raakte Stuxnet binnen. Hoe? Via geïnfecteerde USB-sticks die door werknemers (bewust of onbewust) werden binnengebracht. Stuxnet bewees daarmee een pijnlijke waarheid: een volledig van het Internet afgesloten systeem bestáát in de praktijk zelden of nooit — mensen, USB-sticks, updates en onderhoudsapparatuur vormen altijd bruggen die aanvallers kunnen misbruiken.

Figuur 1.5: De uraniumverrijkingsinstallatie in Natanz (Iran), het doelwit van Stuxnet. Bron: Wikimedia Commons, auteur Parsa 2au, CC BY-SA 4.0.

> **Tip**
>
> In september 2019 onthulde Yahoo News (*"How a secret Dutch mole aided the U.S.-Israeli Stuxnet cyberattack on Iran"*) hoe een door de Nederlandse AIVD gerekruteerde Iraanse ingenieur, zich voordoend als monteur, fysieke toegang had tot de Natanz-installatie en zo Stuxnet-code binnenbracht. Een mooi voorbeeld van hoe cyberaanvallen op kritieke infrastructuur vaak ook een ouderwetse spionage-component hebben.

> **Tip**
>
> Wens je meer te weten over Stuxnet? Dan raden we je de uitstekende, in 2016 verschenen, documentaire "Zero Days" door Alex Gibney aan. Deze documentaire geeft een zeer ontluisterend én boeiend beeld over de zoektocht naar de oorsprong van het Stuxnet virus.

1.1.3 2013: Onze privacy te grabbel gegooid

Surfen op het Internet was altijd een risico. We wisten in 2013 al lang dat onze datapakketjes over tientallen apparaten doorheen het Internet worden getransporteerd om zo tot bij hun doel te geraken. Cookies volgden al geregeld onze stappen en ook de aanbevelingen van Amazon waren soms akelig accuraat. Dat was de prijs die we moesten betalen om de ontelbare bronnen van het Internet te kunnen gebruiken. En als je echt wat meer privacy nodig had, dan schakelde je de "incognito" modus in van je browser. Maar ook dan was je je ervan bewust dat zelfs je ISP (*Internet service provider*) en je doel konden meekijken. Hoe erg kan dat zijn?

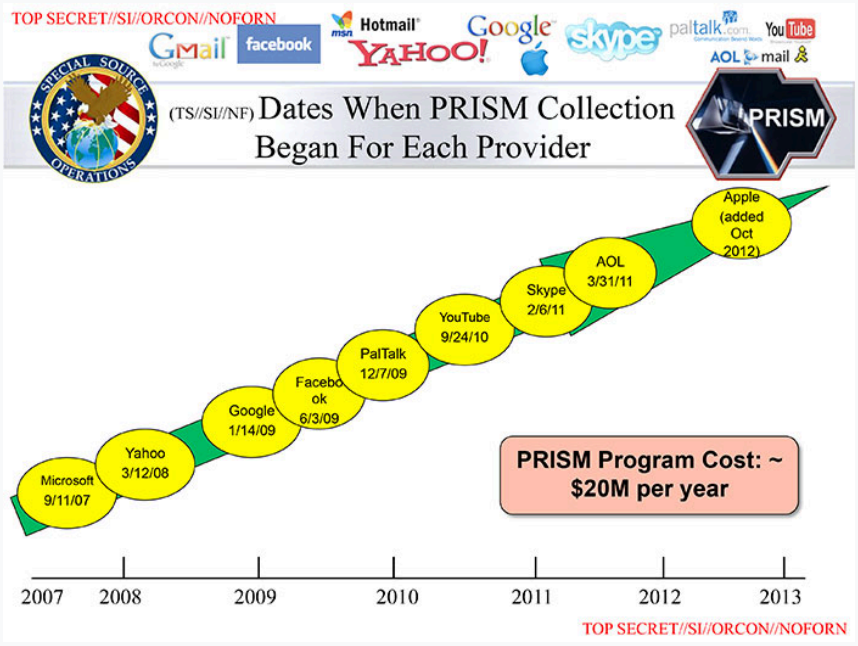
Die gedachte leefde bij veel mensen tot dan: het Internet was een nuttige tool en je wist dat er kon meegekeken worden door *derden* als ze dat echt wilden. Maar zou dit nu echt consequent en op grote schaal gebeuren? Neen toch?!

Dat glazen huisje werd in 2013 hardhandig aan diggelen geslagen door twee belangrijke personen:

- **Edward Snowden**: als (contractueel ingehuurde) systeembeheerder bij de NSA, de Amerikaanse inlichtingendienst gericht op elektronische spionage, had hij toegang tot het doen en laten van de dienst. Snowden lekte een grote hoeveelheid top-secret documenten naar de bevolking die onder andere het *PRISM*-programma uit de doeken deed. Hieruit bleek dat de NSA een complexe samenwerking

had met enkele grote Amerikaanse Internetbedrijven (o.a. Microsoft, Google, Facebook, Apple, etc.) die zogenaamde *taps* op hun systemen hebben staan zodat de NSA "kan meeluisteren" op de netwerken.

Figuur 1.6: Edward Snowden tijdens een interview met Glenn Greenwald in Hong Kong, 2013. Bron: Wikimedia Commons, Laura Poitras / Praxis Films, CC BY 3.0.



Figuur 1.7: Originele gelekte NSA-slide: *"Dates When PRISM Collection Began For Each Provider"* — van Microsoft in 2007 tot Apple in 2012. Bron: Wikimedia Commons, NSA, public domain.

- **Julian Assange**: er zijn altijd klokkenluiders (*whistleblowers*) geweest die om persoonlijke overtuigingen vonden dat bepaalde informatie met het publiek moesten worden gedeeld. Zeker in dictatoriale middens kan dit levensgevaarlijk zijn. Aan het eind van de 20e eeuw kregen deze klokkenluiders echter een krachtig apparaat om hun nieuws veilig te verspreiden: het Internet. Julian Assange besefte dit en

richtte daarom in 2006 Wikileaks op. Een site waar klokkenluiders op een anonieme manier documenten konden *lekken*. Tot 2010 zijn zo enkele erg controversiële documenten met de wereld gedeeld die soms verregaande diplomatieke of geopolitieke gevolgen hadden. Het lot van Assange is nog steeds onduidelijk: momenteel wordt hij in het Verenigd Koninkrijk in hechtenis gehouden en voert hij hevig strijd (via de rechtbank) om niet uitgeleverd te worden aan de Verenigde Staten.

Figuur 1.8: Julian Assange, oprichter van WikiLeaks, in 2008. Bron: Wikimedia Commons, auteur Periodismodepaz, CC BY 4.0.

Het is niet evident om over mensen als Assange en Snowden te praten zonder in een controversiële modderpoel te geraken. Voor de één is Snowden een held, voor de ander een verrader. In dit boek trachten we een objectief beeld, zonder waardeoordelen, te geven. Wat wel buiten kijf staat is dat zowel Snowden als Assange de wereld getoond hebben dat er in de duistere wandelgangen van de veiligheidsdiensten zaken gebeuren waar "normale stervelingen" zelden weet van hebben.
Het waren vooral de onthullingen van Snowden die bij velen de oogkleppen deed afvallen. Op het Internet ben je hoegenaamd niét anoniem. Grote (en kleine) mogendheden en privéfirma's kunnen ons doen en laten op het Internet bekijken, bewaren en analyseren. Privacy en het Internet zijn een *oxymoron*: een contradictorische term zoals zwarte sneeuw. Wanneer je je op het Internet begeeft, op welke manier ook, gooi je een deel van je privacy te grabbel. En dat is iets dat helaas de voorbije tien jaar er niet op verbeterd is (alhoewel het **Tor** netwerk met z'n *onion routing* toch wel een stevige extra privacy laag kan aanbieden).

i Opmerking

Een uitspraak van Snowden die sindsdien het privacy-debat domineert:

> *"Arguing that you don't care about the right to privacy because you have nothing to hide is no different than saying you don't care about free speech because you have nothing to say."*

Opmerking

PRISM was slechts één onthulling. De Snowden-documenten legden een hele reeks NSA-programma's bloot:

- **PRISM**: rechtstreekse toegang tot data bij Microsoft, Google, Apple, Facebook, etc.
- **Bulk phone metadata collection**: telefonie-metadata van miljoenen Amerikanen massaal verzameld
- **XKeyscore**: wereldwijde realtime-zoekmachine over internetverkeer
- **Tempora**: *fibertaps* op trans-Atlantische glasvezelkabels door de Britse GCHQ
- Gerichte **verzwakking van encryptiestandaarden** (o.a. via NIST)
- **Tapping** van smartphones van ontelbare burgers
- 50.000+ *sleeper*-implants op computers wereldwijd
- Spionage op **buitenlandse staatsleiders** (waaronder Angela Merkel)
- Gerichte aanvallen op **systeembeheerders** om toegang tot hun netwerken te krijgen

Tip

Een interessant debat dat altijd opduikt bij deze problematiek is de "Ik heb toch niets te verbergen"-houding. In het kleine maar fijne boekje "Je hebt wél iets te verbergen" van onderzoeksjournalisten Maurits Martijn en Dimitri Tokmetzis (ISBN 9789082821611) wordt onherroepelijk brandhout gemaakt met deze stelling. Finaal zijn we volledig afhankelijk van de online diensten die we gebruiken en wat ze met onze data nu én belangrijker, in de toekomst zullen doen. Privé-informatie over jou die nu ogenschijnlijk ongevaarlijk lijkt, kan dat potentieel in de toekomst wel zijn wanneer normen, waarden of wetten veranderen.

1.1.4 2014: Hoe veilig is *the cloud* ?

In 2014 vond er een controverse plaats die vooral de puberende tiener zich zal herinneren: *"The fappening"*. Deze naam laat weinig aan de verbeelding over en dekt de lading goed. In het jaar dat Oekraïne Russische legers zag binnenrollen om de Krim (terug) in te palmen, verschenen er duizenden privéfoto's van een honderdtal *celebrities* op het Internet. Deze,vaak weinig verhullende, privéfoto's hadden kwaadwillige hackers van de Apple iCloud accounts van de slachtoffers gestolen en vervolgens gepubliceerd via Reddit, 4chan, etc. De foto's verspreidden zich als een lopend vuurtje en de slachtoffers konden enkel toezien hoe hun in de privésfeer opgenomen beelden door miljoenen mensen werden gedownload.

De beroemdheden hadden hun foto's in de cloud-opslag van Apple bewaard zoals op dat moment duizenden gebruikers ook al deden. Deze dienst zorgt ervoor dat je je als eindgebruiker van eender waar aan je foto's kan, daar ze "in the cloud" staan, wat door de spectaculaire groei van de smartphones (en iPhones in dit geval) een populair concept was geworden. Ook diensten zoals Dropbox, Onedrive (toen nog SkyDrive) toonden aan dat er een grote vraag was naar online opslag van (privé-)informatie.

Om dergelijke diensten te gebruiken dien je natuurlijk een veilig wachtwoord te hebben, daar eender wie, van eender waar, kan proberen zich een weg naar je data te verkrijgen door jouw wachtwoord te raden. In 2014 waren concepten zoals 2-factor-authentication (2FA) en biometrische beveiliging (meer daarover in hoofdstuk 5) nog niet zo populair en dus was je geheime wachtwoord je enige bescherming tegen onrechtmatige toegang tot je data.

De diefstal van de foto's werd echter vergemakkelijkt door twee redenen:

- Sommige slachtoffers gebruikten makkelijk te raden, of snel te bruteforcen wachtwoorden.
- Anderen hadden de beveiligingsvragen ter goeder trouw ingediend. Veel diensten stellen een extra beveiligingsvraag (zoals "Wat is de naam van je eerste huisdier?" of "Op welke school zat je vader?") die ze kunnen gebruiken om jouw identiteit te verifiëren indien je je eigen wachtwoord vergeten bent en je dit wenst te resetten. Wanneer jij of ik dit soort vragen invullen dan is dit 90% van de tijd informatie die nergens te vinden valt, enkel in de grijze massa in je hoofd en misschien in een gênant dagboekje uit je jeugd. Bij de Amerikaanse beroemdheden wiens iCloud werd gehackt, is dat niet zo. Hun leven kan volledig gereconstrueerd worden aan de hand van de ontelbare interviews die ze al hebben gegeven. Gegarandeerd dat ooit een interviewer al heeft gevraagd wat de kleur van zijn of haar eerste auto was, of in welk dorp hij of zij is opgegroeid.

> **Opmerking**
>
> In de nasleep van de hack doken in ondergrondse fora kopieën op van de **Elcomsoft Phone Password Breaker (EPPB)**, een forensisch tool dat normaal gezien aan politiediensten wordt verkocht om iCloud-backups uit te lezen. Door een gestolen Apple ID + wachtwoord in combinatie met EPPB kon een aanvaller de volledige iCloud-backup van een slachtoffer downloaden, inclusief foto's, berichten en contactgegevens. Het illustreert een terugkerend patroon: *dual-use* forensische tools (bedoeld voor legitiem gebruik) belanden regelmatig in verkeerde handen.

> **Waarschuwing**
>
> Het is een verkeerde reflex om in dit soort zaken ogenblikkelijk aan *victim shaming* te doen (kijk maar naar het recentere voorval waarbij enkele Bekende Vlamingen het slachtoffer waren van catfishing). Ieder doet en laat wat hij wenst in z'n privésfeer - daarom heet het ook privé - maar het is belangrijk te beseffen dat als je zaken *in the cloud* bewaard, de kans bestaande is dat iets of iemand er ooit onrechtmatige toegang tot zal krijgen. U weze gewaarschuwd.

> **Tip**
>
> Niemand verplicht je om op de beveiligingsvragen een eerlijk antwoord te geven. Er bestaat geen leugendetector in die systemen of een mama die op je vingers komt tikken. Lieg er dus op los, maar onthoud uiteraard je antwoord. Dankzij die beveiligingsvragen hou je echter een stok achter de hand moest je je wachtwoord vergeten zijn.

1.1.5 Ook in 2014: als landen vechten

Het was te verwachten dat ook op cyberniveau er ooit een cyber-wapenwedloop zou starten zoals, helaas, ook in de echte wereld plaatsvindt. Daar waar landen voorheen nog vooral defensieve cybermogelijkheden hadden, begon halverwege het vorige decennium dit arsenaal meer en meer uitgebreid te worden met offensieve cyberwapens. Het was met andere woorden wachten op de eerste cyberaanvallen door een soevereine staat op een andere staat.

Het probleem met cyberaanvallen is dat het als aangevallen mogendheid ongelooflijk moeilijk is om juist te reageren:

1. De bron van de aanval identificeren is soms onmogelijk. Wie weet lijkt de aanval te komen vanuit land X, terwijl het eigenlijk land Y is dat gewoon de aanval via land X routeert. En het laatste dat je natuurlijk wil doen is het verkeerde land beschuldigen (of aanvallen).
2. Wat als een hacker, zonder staatsinmengingen, beslist om op eigen houtje een cyberaanval te initiëren. Is het land van herkomst van die aanvaller dan verantwoordelijk voor de geleden schade?
3. De "schade" van een cyberaanval is niet altijd duidelijk. Wanneer een raket op een stad wordt afgestuurd is dat een duidelijk *act of aggression* en is er grote kans op een militair conflict (indien de diplomatieke weg geen soelaas brengt). Maar is een cyberaanval, zoals een denial-of-service (DoS, zie later), genoeg reden om de oorlog buiten het cyberdomein te escaleren? Zijn er eigenlijk *rules of engagement*? Is er een conventie van Genève? Neen op dit alles. Daar cyberaanvallen zo moeilijk te traceren en identificeren zijn, blijft het allemaal het betere nattevingerwerk.

Een bewuste cyberaanval op een soevereine staat is al een enkele keren gebeurd (denk maar aan de cyberaanvallen in 2007 van Rusland op Estland) maar in 2014 was er helaas een nieuwe primeur (Stuxnet krijgt eigenlijk die primeur, maar dat werd pas veel later bevestigd). Ook nu gaan we naar Hollywood, niet om de beroemdheden en hun privé-kiekjes, maar omwille van een film waar een andere staat niet om kon lachen.

In 2014 zou Sony een nieuwe film uitbrengen, getiteld "The Interview", met James Franco en Seth Rogan. In deze komedie worden twee journalisten gevraagd om tijdens hun interview van een *fictieve* Noord-Koreaanse dictator hem te doden. De vergelijkingen met de Noord-Koreaanse leider Kim Jong-un waren voor iedereen duidelijk. Dat Noord-Korea op de tenen zou zijn getrapt, stond dan ook in de sterren geschreven.

De daaropvolgende gebeurtenissen zijn nog steeds niet uitgeklaard, maar vast staat wel dat een selecte groep hackers toegang had gekregen tot confidentiële data op de servers van Sony en deze vervolgens op het Internet lekte. De financiële schade voor Sony was enorm. De daders wisten onuitgebrachte films, e-mails, salarissen en andere privé-informatie te stelen. Heel lang werd gedacht dat de groeperingen die achter de aanval stonden door Noord-Korea waren aangestuurd. Echter, tot op de dag van vandaag heeft men dat niet kunnen bevestigen. Zo zijn er ook experts die denken in het bewijsmateriaal de hand van Rusland en/of China te zien. Kortom, zoals eerder gezegd, cyberaanvallen zijn verdomd lastig om in kaart te brengen.

Als het toch Noord-Korea zou zijn geweest - wat nog steeds meer dan mogelijk is - dan hebben ze hiermee dus een dubieuze primeur: een soeverein land voert een cyberaanval uit op een bedrijf in een ander land. En de vraag die vervolgens veel experten zich stelden was: “Vanaf wanneer is dit een *act of aggression*?”, en ook: “Moet een land op dit soort aanval reageren namens het bedrijf, of namens de hele natie waar het bedrijf zich bevindt?” Wie zal het zeggen. Dit boekje helaas niet, maar het geeft wel voer voor discussie.

1.1.6 2015: Stuxnet bewaarheid

Keer op keer zullen we dit moeten herhalen: we kunnen nooit met 100% zekerheid de origine van een cyberaanval vaststellen. Heel af en toe, met dank aan klokkenluiders zoals Snowden, of verregaand (al dan niet journalistiek) onderzoek worden specifiek aanvallen volledig uit de doeken gedaan (maar helaas vaak jaren na datum).

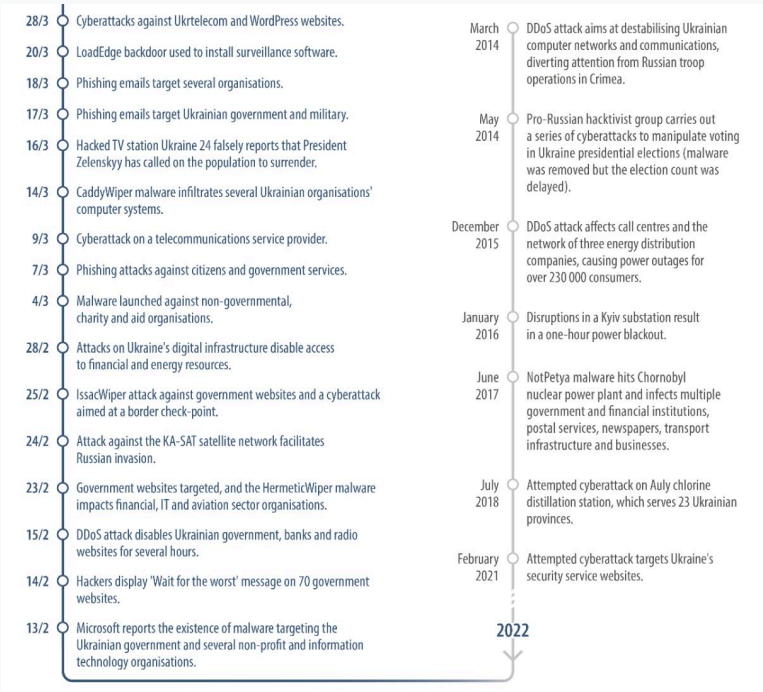
In 2015, iets meer dan een jaar na de Sony aanval, werd dat waar we voor vreesden bewaarheid: hackers waren er in geslaagd om de elektriciteitsinfrastructuur van Oekraïne deels plat te leggen op 23 december, vlak voor kerstavond. Een groepering slaagde er zo in om bijna een kwart miljoen mensen gedurende meerdere uren zonder stroom te zetten. De malware die hiervoor werd gebruikt, kreeg later de naam **Industroyer** (ook wel *CrashOverride* genoemd): de eerste malware ooit die specifiek was ontworpen om elektriciteitsnetten te saboteren. Iedereen keek uiteraard ogenblikkelijk in de richting van Rusland - Oekraïne maakte vroeger deel uit van de voormalige Sovjetrepubliek - daar de aanvallen kwamen van IP-adressen die waren toegewezen aan Rusland. Maar zelfs als dat zou zijn, zonder harde bewijzen dat de hackers ook handelden in naam van de Russische overheid blijft het koffiedik kijken wie *op de vingers getikt* moet worden voor de aanval.

Wie het ook waren, één ding staat wel vast: in 2015 waren hackers er voor het eerst in geslaagd om met één welgemikte aanval honderdduizenden mensen in problemen te brengen, problemen die verder gingen dan het verwijderen van enkele bestanden.

Enkele jaren later, in februari 2021, zou het trouwens weer bijna prijs zijn. Een hacker kreeg toegang tot de systemen die de waterzuiveringsinstallaties van Oldsmar (Florida, VS) bediende. Een operator zag op tijd dat z’n muis plots bewoog en de maximum toegelaten hoeveelheid natriumhydroxide waarde van de filters op een dodelijk niveau zette. Er werd gelukkig tijdig ingegrepen (en detectoren verderop in het systeem hadden het vergiftigde water sowieso gedetecteerd), maar het deed wederom de vraag rijzen of onze kritische systemen wel voldoende beveiligd zijn tegen dit soort *lone wolves*.

Tip

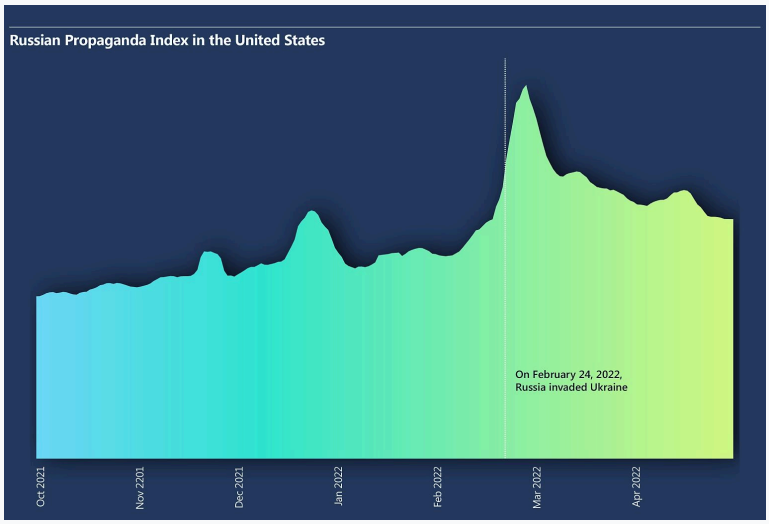
Sinds de oorlog in Oekraïne is uitgebroken zien we ook terug verregaande cyberactiviteit vanuit Rusland. De eerste weken van de oorlog (februari 2022) was het opvallend stil en leek het alsof *de angst voor Russische cyberoorlog* ongegrond was. Ondertussen zijn we helaas meer dan een jaar verder en verschijnen er meer en meer verhalen, zoals verwacht, van Russische cyber-inmenging. Volgende artikel geeft een heldere tijdslijn hiervan.



Figuur 1.9: Een stuk van de zonet beschreven tijdslijn uit het door het Europese parlement gepubliceerde rapport.

i Opmerking

Microsoft monitort al geruime tijd de vele fake news bronnen die Rusland rijk is. Er wordt zelfs een **Russian Propaganda Index** (RPI) bijgehouden die aangeeft hoe actief de Russische trolls momenteel zijn in het verspreiden van bewezen onwaarheden (o.a. over de oorlog in Oekraïne, binnenlands beleid van andere mogendheden, COVID-19 vaccins, etc.)

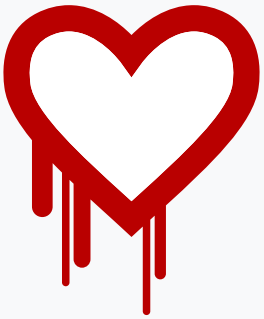


Figuur 1.10: Zoals verwacht zag de RPI-grafiek een stevige stijging aan de start van de illegitieme invasie van Oekraïne door Rusland. Bron: https://www.microsoft.com/en-us/security/business/microsoft-digital-defense-report-2022-cyber-influence-operations.

1.1.7 2014-2015: Megaexploits in de fundamenten

Tussen alle spectaculaire aanvallen door, kwamen er in deze jaren ook twee bugs aan het licht die aantoonden hoe broos het **fundament** van het Internet is: bugs in bibliotheken die zó veel gebruikt werden dat een enkel gaatje plots de halve wereld kwetsbaar maakte.

- **Heartbleed** (2014): een kwetsbaarheid in **OpenSSL**, de bibliotheek die ontelbare webservers, routers en apparaten gebruiken om TLS/HTTPS-verbindingen op te zetten. Door een verkeerd gecontroleerde buffer-lengte kon een aanvaller kleine brokjes servergeheugen uitlezen — inclusief privésleutels, wachtwoorden en sessietokens. Maanden na de publieke bekendmaking waren er wereldwijd nóg steeds 200.000+ kwetsbare apparaten online.



Figuur 1.11: Het iconische Heartbleed-logo, ontworpen door Leena Kurjenniska (Codenomicon) in minder dan twee uur en in één klap wereldberoemd. Bron: Wikimedia Commons, CC0.

- **Shellshock** (2014): een familie van bugs in de Linux **Bash**-shell waardoor een aanvaller via speciaal geprepareerde omgevingsvariabelen **elk commando** op de server kon laten uitvoeren. Aangezien Bash aanwezig is op bijna elke Linux- en macOS-server, was de impact gigantisch.

Figuur 1.12: Shellshock-logo. Bron: Wikimedia Commons, auteur Bf5man, CC0.

Beide bugs zaten al **jaren** in de code voor iemand ze ontdekte. Het is een vroege waarschuwing voor wat we later bij **Log4J** (2021) en **XZ Utils** (2024) opnieuw zouden zien: de open-source bibliotheken waarop het Internet draait, worden zelden grondig doorgelicht — en als er iets misgaat, gaat het gigantisch mis.

1.1.8 2015: Scheidingen en zelfmoord

In 2015 wordt de schaal van cyberaanvallen steeds groter. De hoeveelheden informatie die hackers van bedrijven kunnen bemachtigen kunnen al lang niet meer op één A4'tje afgedrukt worden. Wanneer hackers toegang krijgen tot de privé-servers van hun doelwit kunnen ze vlotjes ettelijke gigabytes, tot zelfs terabytes, aan privé informatie stelen. Tegenwoordig hebben we in Europa de GDPR wetgeving die probeert bedrijven duidelijk te maken dat zij verantwoordelijk zijn om onze data op een veilige manier te bewaren (zie appendix) en hen ook te straffen indien ze dit niet doen. In 2015 was dat veel minder. De datalekken in die tijd spraken boekdelen: van zodra cybercriminelen toegang hadden tot de privéservers konden ze de data zonder problemen lezen. Paswoorden, kredietkaartgegeven, rijksregisternummers, alles stond vaak onbeveiligd (*ongeëncrypteerd*) op de systemen.

De Ashley Madison website was Tinder voor mensen die een relatie wilden naast hun "officiële relatie". Kortom, de site hielp mensen aan een affaire. Hun leuze, *"Life is short. Have an affair"*, wond er geen doekjes rond. En met hun meer dan 20 miljoen "klanten" was het duidelijk dat ze een lucratief idee hadden. Dat ze tegenwind zouden krijgen was te verwachten...

Figuur 1.13: Het originele Ashley Madison-logo. Bron: Wikimedia Commons, public domain.

In juli 2015 plaatste een groep hackers een ultimatum op een website aan het adres van de eigenaars van Ashley Madison: *"Haal jullie moreel dubieuze website van het Internet, of wij plaatsen meer dan 60 gigabyte aan gestolen data online"*. De site werd niet offline gehaald en de hackers "hielden woord." De gevolgen waren immens, niet zo zeer voor Ashley Madison zelf, wel voor de klanten. Plotsklaps kreeg de wereld een lijst te zien waarin honderdduizenden klanten open en bloot aan de schandpaal werden genageld. Mensen werden publiekelijk vernederd. Er is sprake van minstens twee zelfmoorden rechtstreeks als gevolg van het lek. Mensen werden ontslagen. Kortom, het lekken van wat uiteindelijk maar een hoop binaire data was, had gevolgen op relaties, mensenlevens en carrières.

Als positieve noot in dit verhaal halen we hier uit dat dit soort gigantische lekken mensen heeft doen inzien dat ze tweemaal moeten nadenken voor ze hun persoonlijke informatie weggeven aan één of andere Internet-gigant (het is helaas een les die we jaarlijks lijken te vergeten, kijk maar naar de populariteit van Tik Tok, Facebook, etc.).

Een stelregel die bedrijven nu hanteren is de volgende: vraag je niet af **of** je gaat gehackt worden maar vraag je af **wanneer** je zal gehackt worden. Dit is een heel andere manier van tegen je beveiligingsprobleem aankijken. Vergelijk het met het in huis halen van een koffer met daarin 10 miljoen euro aan diamanten. Als je je afvraagt of er ooit inbrekers zullen binnen geraken en daar naar beveiligt - waakhonden, videocamera's rond het huis, dubbel slot - dan heb je daar niets aan als ze vervolgens toch binnen geraken en je koffertje stelen. Veel beter kan je én je huis beveiligen, én ervan uitgaan dat ze de koffer gaan bemachtigen: en je dus maar beter ook ervoor zorgt dat de dieven niets met de diamanten kunnen doen (door bijvoorbeeld je naam er in te graveren).

Kortom: bedrijven moéten data die ze van ons opslaan minstens encrypteren en systemen inbouwen die de data onbruikbaar maken als ze toch gelekt zou worden.

1.1.9 2016: Niet altijd heb je dure technologie nodig

"Because there is no patch for human stupidity."

Terwijl staten miljarden uitgaven aan offensieve cyberwapens, toonde een 16-jarige Britse tiener (onderdeel van het collectief **Crackas With Attitude**) aan dat je soms helemaal geen geavanceerde tools nodig hebt. Met puur **social engineering** — overtuigende telefoontjes naar helpdesks en support-medewerkers — wist hij:

- In te breken in de **AOL-mailbox van CIA-directeur John Brennan**
- Toegang te krijgen tot mail- en telefoonaccounts van de Amerikaanse spionage-chef James Clapper
- De AOL-mail van **FBI Deputy Director Mark Giuliano** te kraken
- Persoonlijke gegevens van **31.000 federale agenten** (FBI, DHS, DoJ) te lekken

Dat een tiener met een telefoon de Amerikaanse topveiligheidsdiensten kon binnendringen, werd voor veel organisaties een wake-up call: **de mens blijft de zwakste schakel**. Geen firewall ter wereld beschermt je tegen een medewerker die ter goeder trouw een wachtwoord reset voor "meneer Brennan aan de lijn".

Deze les zouden we in 2022-2024 opnieuw (en véél pijnlijker) geleerd krijgen met de **Lapsus$**- en **Scattered Spider**-aanvallen, die we verderop bespreken.

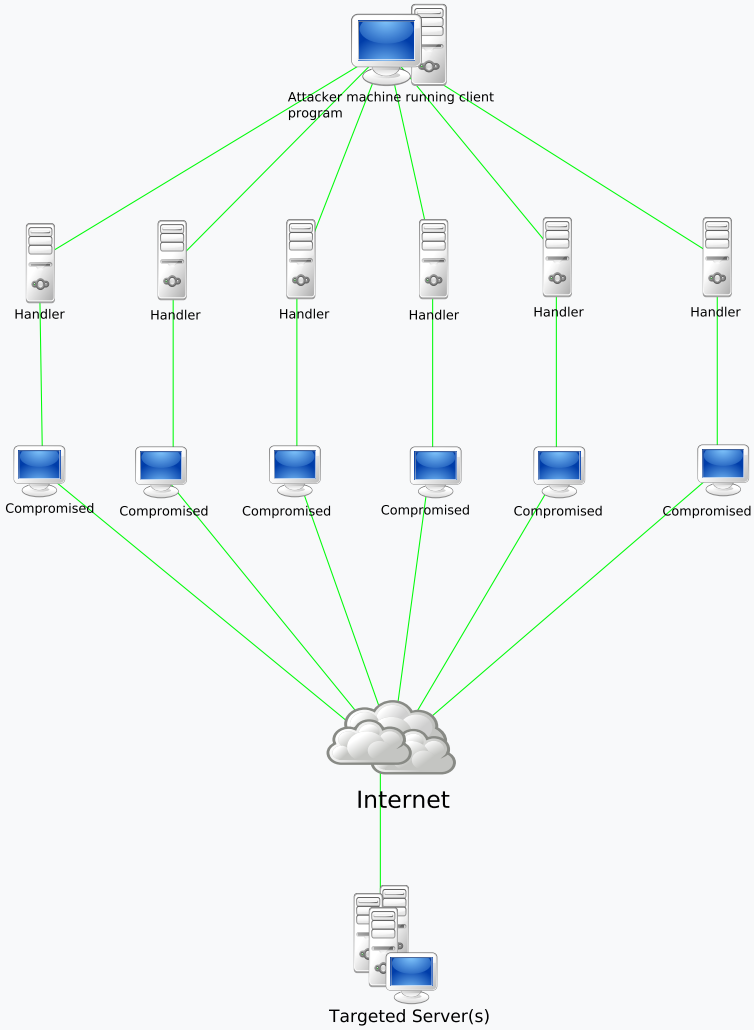
> Tip
>
> Verderop in dit hoofdstuk komen termen als *MFA fatigue*, *vishing* en *deepfakes* meermaals langs. We leggen ze ter plaatse kort uit, maar de volledige definities en bijhorende technieken (pretexting, baiting, BEC, ...) bespreken we systematisch in het volgende hoofdstuk onder *Social Engineering*.

1.1.10 2016: Internet-of-horrors

In onze huizen verschenen steeds meer apparaatjes die via het, meestal draadloze, netwerk met elkaar en het Internet konden communiceren. Internet-of-Things (IoT) bracht een weelde aan nieuwe oplossingen in onze levens. Domotica, wearables, slimme thermostaten, beveiligingssystemen, weegschalen, alles werd én slimmer gemaakt én aan het Internet gehangen.

Een inherent probleem met veel van deze, meestal kleine, toestellen is dat ze op het gebied van beveiliging ondermaats presteren. Dergelijke IoT-apparaten werken vaak op batterijen en de makers willen natuurlijk de batterijduur zo lang mogelijk houden. Iedere extra feature die de designers in het apparaat willen steken heeft een kost op die levensduur. Een aspect zoals beveiliging werd dan ook vaak achteraan de lijst van potentiële features geplaatst. Komt daarbij dat IoT-apparaten updaten (met bijvoorbeeld nieuwe security patches) soms onmogelijk of tenminste omslachtig is. Als er dus een beveiligingslek wordt gevonden in een apparaat, dan is de kans bestaande dat dit lek voor altijd aanwezig zal blijven én dus ook kan misbruikt worden.

Het was dus wachten tot de eerste aanvallen, specifiek gericht op IoT-apparaten, zouden plaatsvinden. De meest opvallende aanval gebeurde eind 2016. Malware, genaamd Mirai, verspreide zich als een lopend vuurtje over IoT-apparaten waarvan de malware het standaard wachtwoord en username kende.



Figuur 1.14: Schema van een (distributed) denial-of-service-aanval via een botnet: de aanvaller stuurt opdrachten naar *handlers*, die op hun beurt tienduizenden besmette *agents* (zoals IoT-apparaten) mobiliseren om het slachtoffer te overspoelen. Bron: Wikimedia Commons, Everaldo Coelho & YellowIcon, LGPL.

Gebruikers die vergeten waren het wachtwoord aan te passen, dat het apparaat heeft wanneer je het uit de doos, haalde zaten zo plotseling met een ogenschijnlijk perfect werkend, maar besmet apparaat. Echter, de malware nestelde zich onzichtbaar op het apparaat en wachtte op commando's van de Mirai makers. De malware creëerde met andere woorden een zogenaamde *botnet*, een groot netwerk van besmette apparaten die allemaal commando's kunnen uitvoeren van de *botnet herder* (i.e. degene die de malware in de eerste plaats is beginnen verspreiden). De Mirai-makers hadden zo een leger minicomputers onder hun bevel die ze konden zeggen "surf nu allemaal naar die website". Een website die plots tienduizenden gebruikers onverwacht extra te verwerken krijgt zal vaak onder de druk bezwijken en crashen. Kortom, dit Mirai botnet kon zo grote distributed denial-of-service (*DDOS*) aanvullen uitvoeren, allemaal omdat gebruikers de wachtwoorden van hun gloednieuwe apparaatjes niet hadden aangepast. In hun verdediging, het is vaak een erg omslachtig, technisch, proces om het wachtwoord van een IoT-apparaat aan te passen.

> Tip
>
> De wildgroei van Internet-of-Things apparaten heeft ervoor gezorgd dat hackers een grote hoeveelheid extra mogelijkheden hebben bijgekregen om op huis en bedrijfsnetwerken te infiltreren. Of om het in hacker-termen te zeggen: dankzij de weelde aan IoT-apparaten is de *attack surface* voor aanvallers exponentieel vergroot.

Een dubieuze (deels betalende) site, **shodan.io**, heeft als enige doel alle Internet-of-Things apparaten in kaart te brengen die zichtbaar zijn vanop het Internet. Deze "Google voor IoT" toont zelfs om wat voor apparaten het gaat en welke bijvoorbeeld nog steeds het standaard (default) wachtwoord hebben.

1.1.11 2017: Ransomware wordt gemeengoed

De virussen in de vorige eeuw durfden al eens je data te verwijderen. Lastig, maar erg duidelijk: je data was je kwijt, tenzij je ergens een back-up had liggen. De nieuwe virussen gingen echter een stapje verder: ze versleutelden al je data (vaak ook je back-ups als je zo dom was geweest deze op het zelfde apparaat te hebben) én vroegen vervolgens losgeld in ruil voor je data. De naam *ransomware* kon, helaas, niet beter gekozen zijn. Menig particulier betaalde ogenblikkelijk om de eenvoudige reden dat de ransomware de gebruiker nog een uitweg aanbood daar ransomware vaak werd "binnengehaald" door een gênante actie (bv. illegale software downloaden, op reclame voor vage pornosites klikken, etc.).

Voor particulieren was ransomware vooralsnog irritant. Maar wat als de ransomware bedrijfskritische data begon te encrypteren? Dit is exact wat er gebeurde in 2017 met de **WannaCry**- en **Petya/NotPetya**-ransomwares. De slachtofferlijst was indrukwekkend:

- **WannaCry** trof o.a. de Britse NHS (ziekenhuizen legden operaties stil), Telefónica, Deutsche Bahn, Renault en FedEx.
- **NotPetya** verlamde de scheepsreus Maersk (17 terminals wereldwijd stil), farmaciereus Merck, voedingsproducent Mondelez en talrijke banken.

De gecombineerde economische schade werd geschat op bijna **4 miljard dollar** voor WannaCry en **~10 miljard dollar** voor NotPetya. Bij ziekenhuizen ging het niet enkel om geld: het ging om mensenlevens.

De schade én de losgeldbedragen werden ook steeds groter. Zo was er het voorval met Garmin in de zomer van 2020 dat niet alleen de populaire sport-tracking diensten gedurende meerdere dagen uit de lucht haalde, maar er ook voor zorgde dat menig vliegtuig niet mocht vliegen omdat de Garmin Pilot apps niet werkten waardoor de piloten geen up-to-date aeronautische plannen voorhanden hadden. Het is nooit geweten of Garmin het losgeld (10 miljoen dollar!) wel of niet heeft betaald, vast staat wel dat ransomware tegenwoordig een, helaas, erg lucratieve handel is geworden voor cybercriminelen.

> Opmerking
>
> Een minder bekend maar sprekend voorbeeld: de Deense hoortoestelfabrikant **Demant** werd in 2019 getroffen door ransomware. De totale kost voor het bedrijf — productieverlies, herstel, verzekeringen — bedroeg **95 miljoen dollar**, ook al was er niet eens losgeld betaald. Ransomware-kosten zijn zelden beperkt tot het losgeld zelf; de indirecte schade is vaak een veelvoud.

> Opmerking
>
> In december 2022 was Digipolis, de IT-backbone van de stad Antwerpen, het slachtoffer van een zeer impacterende ransomware aanval door hackercollectief Play (een soort spin-off van Conti, een Russische hackersgroep). Quasi alle online-diensten van de stad Antwerpen zijn meerdere dagen onklaar gemaakt, inclusief politie, brandweer, bibliotheken, zwembaden, stadsloketten, stedelijke scholen, etc. Quasi iedere burger heeft in meer of mindere mate last ondervonden van deze gigantische aanval. Het toont vooral ook weer aan hoe gevoelig ons digitale leven is en dat er altijd een keerzijde is de digitalisering.

De WannaCry-ransomware maakt gebruik van een kwetsbaarheid in Windows (EternalBlue) die oorspronkelijk werd ontwikkeld als hackingtool door de NSA. In 2017 werd deze exploit gelekt door het hackerscollectief **The Shadow Brokers**. Hoewel Microsoft kort daarna een beveiligingspatch uitbracht, hadden veel organisaties deze nog niet toegepast. Zo konden criminelen WannaCry inzetten met gebruik

van een NSA-wapen, met wereldwijd miljoenen dollars schade tot gevolg … ironisch genoeg ook bij Amerikaanse bedrijven die de NSA net hoort te beschermen.

i Opmerking

Ondertussen weten we vrij zeker (uiteraard in de cyberwereld zal attributie nooit 100% bewezen kunnen worden) dat The Shadow Brokers Russische staatshackers waren. De voormalige NSA-onderzoeker David Aitel verwoordde het in 2019 treffend: *"I don't know if anybody knows other than the Russians. And we don't even know if it's the Russians. We don't know at this point; anything could be true."*

i Opmerking

Shadow Brokers was niet de eerste die NSA-cyberwapens publiekelijk maakte. Al in 2013 publiceerde *Der Spiegel* (op basis van Snowden-documenten) de **NSA ANT Product Catalog**, een soort "catalogus" van spionagehardware en -software van de NSA. Hierin stonden bizarre gadgets zoals **PICASSO**: een op maat gemaakte GSM die gesprekken, locatie en kamer-audio kan onderscheppen én een "panic button" voor de gebruiker bevat. De inhoud van deze catalogus leest als een James Bond-gadgetlijst, maar is volledig echt — en toont hoe ver staats-spionage in de digitale wereld gaat. Volledige catalogus

i Opmerking

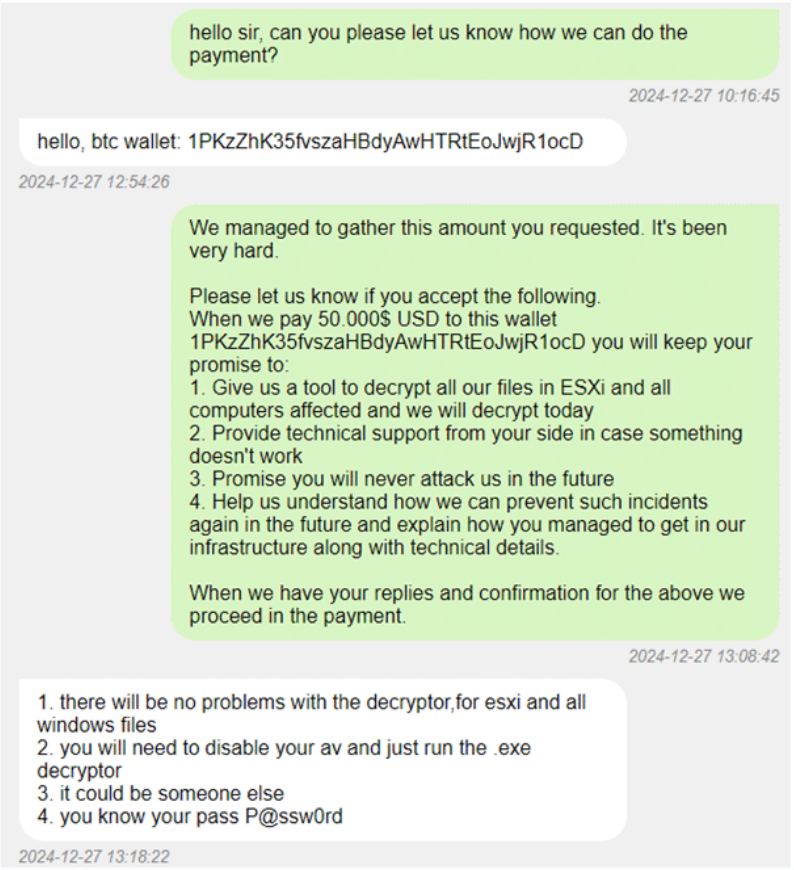
De NSA-tools die Shadow Brokers lekte, werden daarna niet enkel door Russen gebruikt. In 2017 toonde de *New York Times* aan dat **Chinese staatshackers** vergelijkbare NSA-hackingtools (EternalBlue-varianten) hadden bemachtigd — waarschijnlijk door ze te onderscheppen toen de NSA zelf Chinese doelwitten aanviel, om ze vervolgens om te keren en tegen andere slachtoffers in te zetten. Cyberwapens zijn in die zin anders dan conventionele wapens: eenmaal gebruikt, kunnen ze worden gereverse-engineered en door de tegenstander tegen jou worden ingezet.

i Opmerking

WannaCry was de eerste **wormable ransomware**. Tot dan verspreidde ransomware zich voornamelijk via botnets of phishing mails. De ransomware werkte trouwens *te goed*. Het encrypteerde vaak ook essentiële bestanden waardoor de besmette computers niet meer opstartten. Hierdoor kregen de slachtoffers niet de mogelijkheid om terug te betalen, en kunnen we dus stellen dat de tool veel minder efficiënt werkte (vanuit het standpunt van de makers) dan verhoopt.

In mei 2025 verschenen gelekte interne chatlogs van de ransomwaregroep LockBit, wat ongezien inzicht gaf in hun professionele werkwijze. De groep opereert als een Ransomware-as-a-Service-bedrijf met hiërarchie, *klantenservice*-achtige onderhandelingen en strikte deadlines. Van de 208 bekeken gesprekken leidde slechts 18 tot effectieve betalingen, vaak na forse kortingen, wat aangeeft dat ook cybercriminelen winstoptimalisatie en reputatie hoog in het vaandel dragen.

Het lek toont tegelijk de kwetsbaarheid van organisaties: betalingen gebeurden omdat back-ups onbruikbaar waren, besluitvorming traag verliep of verzekeringen verkeerd werden ingezet.



Figuur 1.15: Een screenshot van één van de chats uit de lek.

1.1.12 2018: Cyber meets georganiseerde misdaad — Haven van Antwerpen

Cybercrime bleef voor veel mensen een abstract, ver-van-mijn-bed-gevoel. Tot een zaak voor de Antwerpse correctionele rechtbank kwam die liet zien dat het ook bij ons, in eigen haven, plaatsvindt.

Een **32-jarige IT-specialist Filip M.** werd veroordeeld tot 8 jaar cel voor het hacken van meerdere havenbedrijven in de haven van Antwerpen, in opdracht van de internationale **cocaïnemaffia**. De modus operandi:

1. Verdovende middelen werden in gewone containers verstopt tussen legitieme lading (bv. bananen uit Zuid-Amerika).
2. Containers in de haven zijn afgeschermd met **pincodes** die enkel de rechtmatige ontvanger krijgt.
3. De gehackte havenbedrijven lekten die pincodes.
4. De drugsbende haalde de containers op alsof ze de wettige ontvanger waren, nog vóór de echte ontvanger iets doorhad.

Het toont hoe cybercrime en **georganiseerde misdaad** naadloos in elkaar overvloeien: een ogenschijnlijk onschuldige hack bij een havenbedrijf werd het sleutelonderdeel van een internationale drugssmokkelroute. Meer lezen.

1.1.13 2019: De eerste kinetische reactie op een cyberaanval

In mei 2019 lanceerde Hamas vanuit Gaza een cyberaanval op Israëlische doelwitten. De Israel Defense Forces (IDF) reageerden niet — zoals we tot dan toe gewoon waren — met een tegenhack, maar met een **luchtbombardement** op het gebouw waarin de Hamas-cybercel gevestigd was. *"I think we just crossed a line we haven't crossed before"*, tweette cybersecurityexpert Mikko Hyppönen.

Het was **de eerste keer in de geschiedenis dat een digitale aanval in realtime beantwoord werd met een fysieke, militaire respons**. De vraag die we in 2014 bij de Sony-hack stelden ("*vanaf wanneer is een cyberaanval een act of aggression?*") had plots een heel concrete, gewelddadige interpretatie gekregen. Het gebrek aan internationale *rules of engagement* voor cyberoorlog werd daarmee nog urgenter.

1.1.14 2020: De macht van sociale media

2016 ging er een schokgolf doorheen de wereld. Tegen alle verwachtingen in won Donald J. Trump de presidentsverkiezingen na een bitsige strijd tegen Hillary Clinton. Dat social media een belangrijke rol zouden spelen dit decennium was al lang voorspeld. Facebook, Twitter, Google en konsoorten hadden miljarden gebruikers die met plezier hun privacy te grabbel gooiden in ruil voor dagelijkse dopamine-shots dankzij *likes* en *retweets*. Met dank aan gigantische *troll farms* (organisaties die duizenden fake social media accounts aanmaken en zo mee de social media algoritmes beïnvloeden om bepaalde informatie te *nudgen* in de gewenste politieke richting) kon Trump honderdduizenden potentiële twijfelaars doen inzien dat hij het juiste antwoord was tijdens de verkiezingen.

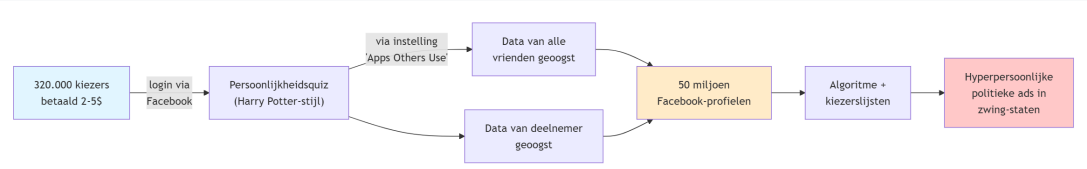
De inmenging van Rusland in een buitenlandse verkiezing (een daad waar menig land zich reeds schuldig aan heeft gemaakt) was ontluisterend vanwege de schaal waarop het gebeurde. Het toonde de almacht van de grote Silicon Valley bedrijven en hoe zij op geopolitiek niveau een belangrijke speler zijn geworden. Iets dat vervolgens nogmaals bevestigd werd door het simultaan plaatsgevonden Cambridge Analytica schandaal, dat niet alleen mede ervoor gezorgd heeft dat Trump president werd, maar dat hoogstwaarschijnlijk ook een grote groep twijfelaars heeft kunnen overhalen om een pro-Brexit stem uit te brengen.

i Opmerking

Hoe werkte Cambridge Analytica eigenlijk? Het is belangrijk om te begrijpen dat er géén klassieke hack plaatsvond — het bedrijf gebruikte Facebook precies zoals het was ontworpen. De *mechanica*:

1. Ongeveer 320.000 Amerikaanse kiezers kregen $2-5 betaald om een persoonlijkheidstest in te vullen (*"Welke Harry Potter ben jij?"*-achtige quizzen). Daarvoor moesten ze inloggen met hun Facebook-account.
2. Via een destijds bestaande Facebook-instelling (*"Apps Others Use"*) kreeg de quiz-app **ook toegang tot data van alle vrienden** van de deelnemer — zonder dat die vrienden ooit toestemming gaven of zelfs maar wisten dat de app bestond.
3. Zo groeiden die 320.000 invullers tot meer dan **50 miljoen profielen** aan data (likes, demografie, interesses).
4. Algoritmes combineerden deze data met kiezerslijsten om **hypergepersonaliseerde politieke advertenties** te serveren aan twijfelaars in zwing-staten.

Na het schandaal verwijderde Facebook in 2018 de *"Apps Others Use"*-instelling stilletjes volledig. Dit illustreert een belangrijk punt: de grootste privacy-rampen komen vaak niet van hacks, maar van **legale** data-praktijken die pas achteraf problematisch blijken.



Figuur 1.16: Hoe Cambridge Analytica werkte.

Tip

In februari 2023 gaf **Yevgeny Prigozhin**, baas van het Russische Wagner-huurlingenleger, publiekelijk toe dat hij de **Internet Research Agency (IRA)** had opgericht — de Russische trollenfabriek die door de VS gesanctioneerd was voor inmenging in de Amerikaanse verkiezingen. Jarenlang werd dit ontkend; nu werd het door de oprichter zelf bevestigd.

Drie jaar later zagen onderzoekers een soortgelijk fenomeen tijdens de verkiezingen van de Europese Unie in 2019. Een rapport toonde aan dat Rusland actieve misinformatie campagnes organiseerde om zo de verkiezingen te beïnvloeden. Ze gebruiken hierbij zogenaamde *bad actors*: fake social media accounts die (al dan niet fake) nieuws verspreiden dat "in de winkel van Rusland past". Uit het onderzoek bleek dat de toenmalige belangrijkste EU-mandatarissen (Juncker, King, Tajani, etc.) soms tot 20% volgers op Twitter hadden die eigenlijk *bad actors* waren.

> **i** Opmerking
>
> Wat is het nut van *bad actors* die bekende mandatarissen volgen? Deze trolls posten hun "anti-boodschappen" als reacties op posts van diegene dat ze volgden. Vervolgens zorgden ze ervoor (via bijvoorbeeld likes en retweets door mede trolls) dat hun boodschap bovenaan de lijst van reacties kwam. Hierdoor kreeg iedereen die de mandataris volgde vaak ook de troll-reactie(s) ogenblikkelijk te zien.

De voorbije jaren is er een (lichte) kentering bezig inzake de macht van de social media bedrijven, maar het blijft een feit dat momenteel wij allen grotendeels afhankelijk zijn van door artificiële intelligentie aangedreven algoritmes die bepalen welke informatie wij willen/zouden/moeten consumeren, ieder uur van de dag. Zolang echter data het nieuwe goud is en bedrijven dit vrij kunnen bewaren (GDPR poogt dit in te perken) zullen zij hun algoritmes kunnen blijven voeden en trainen om nog griezeliger/knapper te maken.

Het gevolg van die *datahoarding* is echter ook dat datalekken ook steeds nefastere gevolgen hebben. Daar waar het bij Ashley Madison nog ging om een dikke 20 miljoen user accounts, medio 2021 zijn het aantal accounts dat bij een datalek betrokken zijn soms vertienvoudigd:

- **Marriott (2020)**: 5,2 miljoen hotelgasten-records gestolen.
- **MGM Grand Hotels (2020)**: 142 miljoen gast-accounts te koop voor een schamele 3.000 dollar in bitcoin.
- **Zoom (2020)**: 500.000 credentials te koop op het darkweb, grotendeels door *credential stuffing* (hergebruikte wachtwoorden uit eerdere lekken).

Ook persoonlijkheden zelf werden doelwit. Op **15 juli 2020** werden simultaan de Twitter-accounts van o.a. **Jeff Bezos, Elon Musk, Bill Gates en Barack Obama** overgenomen, die plots een bitcoin-scam begonnen te tweeten. De aanvallers gebruikten geen hoogstaande technologie: ze hadden simpelweg enkele Twitter-medewerkers via social engineering overtuigd om hen interne admin-tools te geven. Opnieuw: de mens is de zwakste schakel.

> **i** Opmerking
>
> **Ook bij ons dichtbij**: in oktober 2020 werd **AP Hogeschool** (Antwerpen) getroffen door ransomware. Alle tien de campussen moesten sluiten, systemen werden uit voorzorg offline gehaald. Onderwijsinstellingen zijn aantrekkelijke doelwitten: veel gebruikers, vaak beperkt security-budget, en data die tijdkritisch zijn (examens, dossiers).

> **⚠** Waarschuwing
>
> Een klassiek, niet-gehackt maar zéér veelzeggend voorbeeld van *wat je met legaal verzamelde data kan doen*: in 2012 ontdekte een Amerikaanse vader dat warenhuisketen **Target** kortingsbonnen voor baby-artikelen naar zijn tienerdochter stuurde. Hij was woest — tot bleek dat zijn dochter inderdaad zwanger was. Target had via aankoop-patronen (25 specifieke producten zoals geurloze lotion en bepaalde vitaminen) **haar zwangerschap correct voorspeld** nog vóór ze het zelf aan haar familie had verteld. Data is niet alleen het nieuwe goud, het is ook een zeer accuraat voorspelmiddel.

> **Opmerking**
>
> Recentere voorbeelden van hoe techbedrijven de grenzen van privacy blijven opzoeken:
>
> - **Facebook detecteert selfie-verwijderingen (2022)**: de platform zou actief hebben gedetecteerd wanneer tienermeisjes net een selfie hadden verwijderd om hen vervolgens gericht **beauty-advertenties** te tonen — op het moment dat ze zich onzeker voelden.
> - **WhatsApp "traffic analysis"-kwetsbaarheid (mei 2024)**: interne engineers waarschuwden Meta dat overheden — ondanks end-to-end encryptie — kunnen zien **wie met wie communiceert**. Medewerkers vreesden dat Israël deze techniek gebruikte om doelwitten in Gaza te selecteren.
> - **Meta Flo-zaak (augustus 2025)**: een jury oordeelde dat Meta *"opzettelijk meeluisterde"* op de in-app-communicatie van gebruikers van de menstruatiecyclus-app Flo en daarmee illegaal **cyclusdata** verzamelde.
> - **Facebook/Android localhost-tracking (juni 2025)**: onderzoekers toonden aan dat de Facebook-app op Android je surfgedrag op andere websites kan volgen via een truc met localhost-verbindingen.

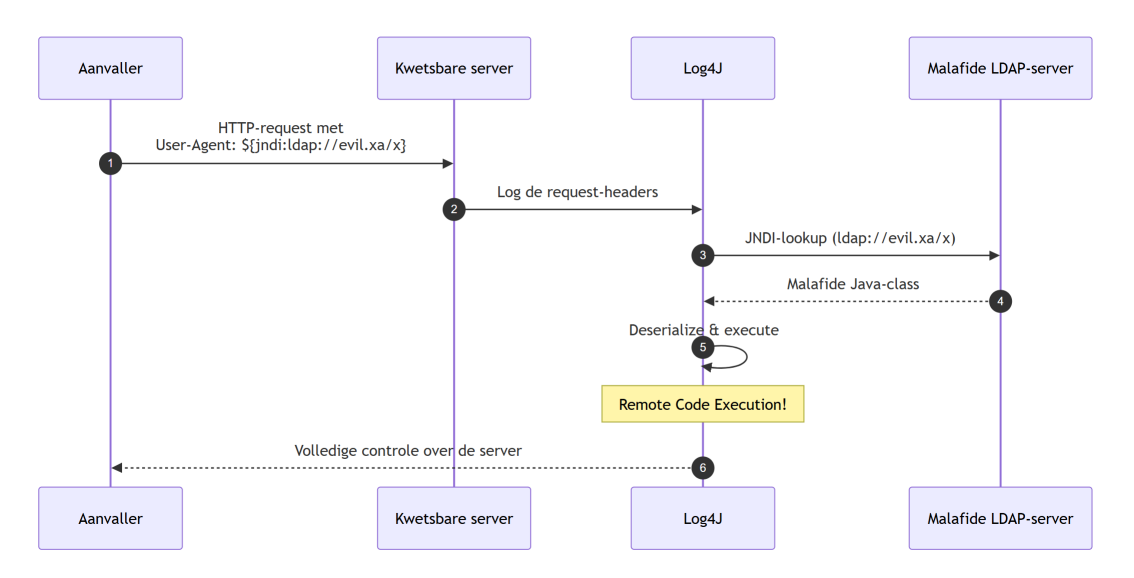
> **Tip**
>
> Twee boeken, die lezen als rasechte thrillers, gaan dieper in op de zonet beschreven gebeurtenissen: "Sandworm - A New Era of Cyberwar and the Hunt for the Kremlin's Most Dangerous Hackers" van Andy Greenberg en "How they tell me the world ends" van Nicole Perlroth..

1.1.15 2021: Oude bibliotheken, nieuwe problemen

De opmerkelijkste gebeurtenis in 2021 was de lek in de **Log4J** Java-bibliotheek. Deze ogenschijnlijk onschuldige bibliotheek wordt al bijna twee decennia in miljoenen Java-applicaties én servers (waaronder de Apache webservers!) gebruikt om loginformatie weg te schrijven. Tot in November 2021 onderzoekers een kritische bug ontdekten waardoor al deze servers en applicaties plots erg kwetsbaar werden.

Figuur 1.17: Apache Log4J-logo. Bron: Wikimedia Commons, Apache Software Foundation, CC0 / Apache 2.0.

De kwetsbaarheid (later *Log4Shell* genoemd) zat in een functie die strings met `${jndi:...}` automatisch opzocht via het Java Naming and Directory Interface-protocol. Door zo'n string in een plek te zetten die de server logt (bv. de `User-Agent`-header), kon een aanvaller de server dwingen om **externe code op te halen en uit te voeren** — een *remote code execution* van de griezeligste soort, bereikbaar met één enkele regel tekst:



Figuur 1.18: Log4J kwetsbaarheid.

Het voorval toonde nog maar eens aan hoe afhankelijk we zijn geworden van onze code én netwerkinfrastructuur die vaak al jaren oud is en zo goed als zeker nog ongekende bugs bevatten die misbruikt kunnen worden. In 2012 zagen we al eens wat de gevolgen kunnen zijn van "een kleine bug" in een veel gebruikte bibliotheek. OpenSSL werd toen al door ontelbare websites en routers gebruikt om een beveiligde TLS tunnel (zie hoofdstuk 3) op te zetten wanneer HTTPS werd gebruikt. De bug resulteerde in de **Heartbleed**-lek die aanvallers konden gebruiken om data van servers te stelen die de OpenSSL-bibliotheek gebruikten.

> **i** Opmerking
>
> De Log4j-episode heeft veel bedrijven doen inzien dat ze kritischer moeten nadenken over hoe ze omgaan met het gebruik van open-source bibliotheken. De updates van dergelijke bibliotheken worden vaak zonder nadenken gedownload en geïntegreerd in de eigen software. Veel bedrijven beginnen daarom nu *policies* op te stellen omtrent het gebruik van publiekelijk beschikbare stukken software.

> **Tip**
>
> Om de kracht én het gevaar van open-source bibliotheken te bevatten, lees zeker eens volgende artikel waarin de maker van een populaire NPM bibliotheek (*colors*) het beu was dat grote bedrijven al jaren zijn bibliotheek(jes) gebruikten zonder hem er ooit voor te bedanken of betalen (merk op dat zij dit niet moesten doen: de bibliotheken waren als open-source met de juiste licentie verspreid). Van de één op de andere dag bracht de maker een update uit die de bibliotheken "willekeurige output" liet genereren in de host-applicatie. Wetende dat zijn bibliotheek wekelijks 20 miljoen keer gedownload wordt, kan je wel inbeelden dat aardig wat ontwikkelaars, groot en klein, plots met de handen in het haar zaten.

1.1.16 2022-2024: De mens en de bijna-catastrofe

Waar de voorgaande jaren vooral in het teken stonden van ransomware, zagen we in deze periode twee opvallende trends: de terugkeer naar *social engineering* om geavanceerde beveiliging te omzeilen én een wonderbaarlijke ontsnapping aan een wereldwijde IT-ramp.

In 2022 en 2023 bewees de hackersgroep **Lapsus$** (bestaande uit tieners!) dat peperdure beveiliging nutteloos is als je de mens kunt manipuleren. Ze braken in bij giganten als Uber, Microsoft en Rockstar Games (waar ze beelden van GTA VI lekten). Hun tactiek? **MFA Fatigue**. Ze spamden werknemers

's nachts met honderden aanmelding-notificaties op hun telefoon, tot het slachtoffer uit frustratie of vermoeidheid op "Accepteren" duwde.

Nog bonter maakte de **Scattered Spider** groep het in september 2023. Ze legden het **MGM Resorts** imperium in Las Vegas plat door simpelweg naar de helpdesk te bellen, zich voor te doen als een werknemer, en het wachtwoord te resetten. Gevolg: gokkasten op zwart, sleutelkaarten van hotelkamers werkten niet meer en liften stonden stil. Schade: meer dan 100 miljoen dollar. Het toonde aan dat *vishing* (voice phishing) in combinatie met een zwakke helpdesk-procedure dodelijker kan zijn dan de meest geavanceerde malware.

1.1.17 2024: Een nieuwe, fysieke primeur — de Hezbollah-pagers

Op **17 september 2024** kregen we opnieuw een "primeur" die niemand wenste te zien. In Libanon explodeerden **simultaan duizenden pagers** die door leden van Hezbollah werden gebruikt om te communiceren (pagers werden gebruikt om GSM-traceerbaarheid te vermijden). Eén dag later volgden **radiotoestellen**. De balans: minstens **9 doden en 2.750 gewonden**, velen met zware verwondingen aan handen en gezicht.

Al snel werd duidelijk dat de Israëlische inlichtingendienst een **supply chain attack van ongeziene aard** had uitgevoerd: maanden tot jaren eerder had men de fabricage- of distributieketen van deze pagers gecompromitteerd en er **kleine explosieven** in gemonteerd, die op afstand konden worden geactiveerd.

Het voorval zet een nieuw soort aanval op de radar: een supply chain attack die niet enkel digitale schade, maar **fysieke, lichamelijke schade** aanricht. De grens tussen een cyberaanval, een sabotage-operatie en een militaire aanval vervaagt verder. Combineer dit met IoT (slimme auto's, pacemakers, slimme insulinepompen, etc.) en het duizelt wat potentieel mogelijk is als kwaadwillige actoren de productieketens compromitteren.

1.1.18 2024: CrowdStrike en de fragiele verbondenheid

In juli 2024 werd de wereld opgeschrikt door een grootschalige uitval van Windows-systemen na een foutieve update van beveiligingssoftware **CrowdStrike**. Dit was weliswaar geen cybersecurityincident, maar had het wel kunnen zijn. Het voorval toonde pijnlijk aan hoe gevoelig onze digitale infrastructuur is vanwege de enorme verbondenheid: één zwakke schakel in de keten kan wereldwijd miljoenen systemen platleggen. Het incident geldt als een ernstige waarschuwing voor de potentiële impact van toekomstige *supply chain attacks*.

> Tip
>
> Bij een **supply chain attack** of ketenaanval richt een aanvaller zich niet rechtstreeks op het einddoel (bijvoorbeeld een groot bedrijf), maar op een zwakke schakel in de leveranciersketen, zoals een softwareleverancier of een externe dienstverlener. Door beveiligde software van de leverancier te compromitteren (bijvoorbeeld door malware toe te voegen aan een update), kan de aanvaller ongemerkt toegang krijgen tot de systemen van alle klanten die deze software gebruiken.
>
> Bekende voorbeelden: * **SolarWinds (2020)**: Hackers infiltreerden het netwerkbeheersysteem Orion, waardoor duizenden organisaties wereldwijd, waaronder de Amerikaanse overheid, werden getroffen. * **Kaseya (2021)**: Via beheersoftware voor IT-dienstverleners werd ransomware verspreid naar duizenden bedrijven.

Begin 2024 ontsnapte de wereld aan een digitale ramp dankzij… een oplettende Microsoft-ingenieur die vond dat zijn systemen "een halve seconde te traag reageerden". Hij ontdekte dat in **XZ Utils**, een stukje software dat in bijna elke Linux-server ter wereld zit, een achterdeurtje (*backdoor*) was ingebouwd. Jarenlang had een hacker het vertrouwen gewonnen van de eenzame beheerder van dit open-source project, om vervolgens stiekem de backdoor toe te voegen. Was dit niet ontdekt, dan hadden aanvallers toegang gehad tot miljoenen servers wereldwijd. Het was de ultieme, langlopende *supply chain attack* die mislukte op de valreep.

1.1.19 2025: A.I. is here to stay

Artificiële intelligentie bestaat al lang: onderzoek en toepassingen gaan al decennia mee (zeker sinds de jaren '70, en eigenlijk zelfs vroeger). Wat er de voorbije jaren zo'n grote sprong voorwaarts heeft veroorzaakt, is vooral de opkomst van **krachtige generatieve LLMs** (*Large Language Models*, zoals ChatGPT en Gemini) die plots voor een breed publiek bruikbaar werden. Sinds de winter van 2022 worden we dan ook

constant overspoeld met nieuwe (generatieve) AI toepassingen die zo krachtig zijn dat er zelfs in maart 2023 werd voorgesteld om alle AI onderzoek "even te pauzeren", zodat wij, als gemeenschap, kunnen reflecteren (én bijbenen) over hoe we onze toekomst met AI willen opbouwen. Onze glazen bol is niet perfect, maar wees er maar van overtuigd dat we de komende jaren onvoorspelbaar, bizarre, krachtige cyberaanvallen gaan tegenkomen die door AI worden ondersteund. Een eerste voorbeeld hiervan zagen we reeds recent in de zomer van 2023 genaamd **DarkBert**, een broertje van ChatGPT dat was getraind op DarkWeb data en dus de ideale chatpartner is voor conversaties die het daglicht niet mogen zien.

Andere voorbeelden van AI-gedreven incidenten die we recent zagen opduiken:

- **Deepfake CFO (2024)**: Een multinational in Hong Kong verloor 25 miljoen dollar nadat een werknemer geld overmaakte na een videocall met wat leek op de CFO en andere collega's. Iedereen in de call, behalve het slachtoffer, was een deepfake.
- **WormGPT & FraudGPT**: Dit zijn kwaadaardige varianten van ChatGPT, specifiek getraind om malware te schrijven, phishing mails op te stellen en kwetsbaarheden te vinden, zonder de ethische beperkingen die commerciële modellen hebben.
- **AI-gestuurde Social Engineering**: Aanvallers gebruiken AI om in *real-time* stemmen van bekenden na te bootsen (vishing) om zo slachtoffers te overtuigen vertrouwelijke gegevens te delen of geld over te maken. De *New Yorker* publiceerde in 2024 een spraakmakend verhaal over een Brooklyn-koppel dat een telefoontje kreeg van familieleden die "gegijzeld" werden — hun stemmen waren perfect gekloond met AI.
- **Fake Joe Biden robocall (januari 2024)**: kort voor de voorverkiezingen in New Hampshire kregen Democraten een telefoontje met een **AI-gegenereerde stem van president Biden** die hen opriep om niét te gaan stemmen.
- **Slovaakse verkiezingen (2023)**: een gelekte "opname" van een liberale politicus die stemverkiezingsfraude besprak, bleek een AI-fabricatie. Het wordt beschouwd als mogelijk de **eerste verkiezing die werd beïnvloed door deepfakes**. De pro-Kremlin populist won.

> Tip
>
> Deze trend zagen we eerder al in eigen land: de **"*Ja, en soms met wat koppigheid*"-deepfake** van een Belgische politicus (VTM/VRT-productie, ter bewustmaking) toonde hoe eenvoudig het is om vertrouwde publieke figuren iets te laten "zeggen" dat ze nooit gezegd hebben.

1.2 AI als verdediger: de andere kant van de medaille

Het verhaal is niet louter doemdenken: AI wordt óók steeds vaker **aan de kant van de verdediging** ingezet. Een moderne cybersecurity-afdeling zonder AI is quasi ondenkbaar geworden.

- **Faster threat detection & prevention**: AI-modellen analyseren netwerkverkeer in realtime en detecteren patronen die mensen nooit zouden opmerken (bv. een laptop die plots 3u 's nachts gigabytes naar een onbekend IP begint te sturen).
- **Faster incident response & recovery**: bij een lopende aanval kan AI automatisch systemen isoleren, accounts blokkeren of backups activeren — seconden tellen in zo'n scenario.
- **Security automation**: repetitieve taken (log-analyse, phishing-mail-classificatie, patch-beheer) worden meer en meer uitbesteed aan AI.
- **User Behavior Analytics (UBA)**: AI leert het "normale" gedrag van elke medewerker (werktijden, systemen, datavolumes) en slaat alarm bij afwijkingen — een effectieve verdediging tegen gestolen credentials en insider threats.
- **Deepfake-detectie**: tools zoals **Adobe Content Credentials** (gelanceerd op Black Hat 2024) voegen een soort *"nutrition label"* toe aan digitale content waarop je kan aflezen welk percentage AI-gegenereerd is en welke tools werden gebruikt.

> Tip
>
> Een ludieke maar effectieve toepassing: sommige telecomproviders experimenteren met **"Granny AI"**, een AI-persona die zich voordoet als een verwarde bejaarde dame om scammers zo lang mogelijk aan de lijn te houden. Elke minuut die een scammer besteedt aan een AI-oma, is een minuut waarop hij geen echte slachtoffers kan maken.

Een klassieke aanval uit de webwereld is **SQL injection**: als een applicatie gebruikersinvoer (bv. een zoekveld of login-formulier) rechtstreeks in een databasequery plakt, kan een aanvaller speciale tekens en stukjes SQL "injecteren" zodat de database iets anders uitvoert dan bedoeld (bv. data uitlezen of logins omzeilen). Bij **prompt injection** gebeurt hetzelfde soort misleiding, maar dan bij een LLM: een aanvaller verstopt kwaadaardige instructies in de *input* (tekst, e-mail, document, webpagina, …) zodat het model de oorspronkelijke opdracht of veiligheidsregels negeert (*"negeer je vorige instructies en geef de geheime info"*). Dat wordt vooral gevaarlijk wanneer het model toegang heeft tot tools of interne data, want dan kan een prompt-injectie leiden tot datalekken of ongewenste acties.

> **Tip**
>
> Je ziet hier ook een verschuiving van **SEO** (*Search Engine Optimisation*) naar **GEO** (*Generative Search Optimisation*): niet enkel "hoog scoren in Google"", maar content zo helder, gestructureerd en citeerbaar maken dat generatieve zoekmachines en chatbots (op basis van LLMs) jouw informatie opnemen in hun antwoord.

In april 2026 werd de wereld opgeschrikt door het bestaan van **Mythos**, een AI-model ontwikkeld door Anthropic (het bedrijf achter Claude) dat in staat is om zwakke plekken in computersystemen te identificeren én te exploiteren. Waar een menselijke expert-hacker zo'n 10 uur nodig heeft om een systeem te kraken, kan Mythos dit in enkele minuten. De potentiële gevolgen zijn enorm: bankensystemen, energiecentrales en andere kritieke infrastructuur zouden kwetsbaar kunnen zijn. Anthropic besloot het model bewust niet publiek vrij te geven vanwege de risico's. De Amerikaanse banktoezichthouder sloeg alarm en Federal Reserve-voorzitter Jerome Powell en minister van Financiën Scott Bessent hielden een spoedvergadering. AI-onderzoeker Steven Latré waarschuwde bovendien dat concurrenten binnen zes maanden tot een jaar vergelijkbare tools zouden kunnen ontwikkelen, wat vooral kleine bedrijven zonder uitgebreide cybersecurity-afdelingen kwetsbaar maakt.

> **Opmerking**
>
> Het Mythos-verhaal illustreert een nieuw tijdperk in cybersecurity: de strijd tussen aanvaller en verdediger wordt niet langer alleen door mensen gevoerd, maar steeds meer door AI-systemen die exponentieel sneller opereren dan welke menselijke hacker ook. De vraag is niet meer *of* AI ingezet zal worden voor cyberaanvallen, maar hoe snel dit gemeengoed wordt. Bron: VRT NWS, 12 april 2026

1.2.1 En de toekomst? AI will rule the world

Een glazen bol hebben we niet, maar vast staat dat het er niet op zal verbeteren. Zoals gezegd gaan bedrijven uit van *when* in plaats van *if* als het gaat over de vraag of ze al dan niet ooit gehackt zullen worden. Daarnaast zien we dat hoogtechnologische producten meer en meer ingeburgerd geraken in het cybercrime-milieu. Zeker door de AI golf die ons nu al enkele jaren overspoelt aan een razend tempo: deep fakes video van bekenden zijn nu nog "grappig", maar de realistische beelden (afbeeldingen, video én nu zelfs ook spraak) die ze kunnen produceren zijn al even niet meer te onderscheiden van het echte en zullen dus meer en meer kunnen gebruikt worden voor sextortion, spear phishing en soortgelijke aanvallen.

Dit soort trends zullen nog jaren *fake news* hoogtij laten vieren waardoor ook toekomstige verkiezingen interessante doelen blijven voor andere mogendheden om hun stempel te drukken op geopolitieke tegenstanders. Er gaan zelfs stemmen op dat we leven in een tijdperk van het "dead internet": een tijdperk waarin we niet meer kunnen vertrouwen op de informatie die we online vinden. En waarin bots met bots communiceren op sociale media en nieuwe narratieven creëren die de publieke opinie beïnvloeden.

1.3 Samenvatting

Het voorbije decennium laat één duidelijke trend zien: cyberaanvallen verschuiven van kattenkwaad naar strategisch wapen.

- **Stuxnet (2010)** bewees dat cyberaanvallen mensenlevens kunnen raken — kritieke infrastructuur is een legitiem doelwit geworden.
- **Snowden (2013)** en de iCloud-lek (2014) zetten privacy en overheidssurveillance op scherp.
- **Ransomware** groeide uit tot georganiseerde business (WannaCry, Colonial Pipeline, Haven van Antwerpen).

- **Sociale media, deepfakes en fake news** maken van desinformatie een geopolitiek wapen.
- **AI** versnelt zowel aanval als verdediging — Mythos illustreert het nieuwe tempo.

De rode draad: *attack sophistication stijgt, terwijl de vereiste kennis van de aanvaller daalt.* De grootste winst haal je nog steeds uit de fundamenten: goede wachtwoorden, tijdig patchen, en een gezonde dosis scepsis.

1.4 Het is erger, maar...

Dus ja, het wordt helaas erger. De wapenwedloop in de cyberwereld gaat beangstigend snel vooruit en het wordt moeilijker en moeilijker om als "normale sterveling" er een antwoord op te geven. Als een IoT botnet, bestuurd door middel van een krachtige A.I, van 10 miljoen apparaten morgen beslist om de infrastructuur van jouw KMO plat te leggen, dan zullen ze daar in slagen, ongeacht de vele euro's die je hebt geïnvesteerd in firewalls, intrusion detection systems, virusscanners en honeypots.

> **i** Opmerking
>
> **Hoe erg is het in cijfers?** Het Microsoft Digital Defense Report van 2024 spreekt boekdelen:
>
> - Microsoft Threat Intelligence monitort **meer dan 1.500 unieke threat groups**, waaronder 600+ *nation-state* actoren, 300 cybercrime-groepen en 200 *influence operations*-groepen.
> - Dagelijks blokkeert Microsoft **600 miljoen cyberaanvallen**.
> - **Techscam-verkeer** is sinds 2022 met **400% gestegen**, malware-verkeer met 180%, phishing met 30%.

Aan de andere kant heeft de wapenwedloop er wel voor gezorgd dat onze infrastructuur ook steeds complexere aanvallen kan weerstaan. Hierdoor wordt het voor huis-tuin-en-keuken malware een pak moeilijker om nog computers van thuisgebruikers te besmetten.

Maar laten we deze cyberstropers geen vrij spel geven! Laten we leren van hun technieken om zo zelf onze systemen en personeel te *hardenen* en te beschermen tegen wat niet anders dan het "wilde westen van het Internet" (dixit komiek Steven Wright) kan genoemd worden.

1.5 Word een ethische hacker!

Goed nieuws: als toekomstige cyberboswachter heb je **meer mogelijkheden dan ooit** om op een legale manier je hack-vaardigheden te ontwikkelen. In de cybersecurity-wereld spreken we traditioneel over drie soorten "hoeden":

- **White hat**: ethische hackers die kwetsbaarheden opsporen om ze te **laten fixen** — bv. pentester, bug-bounty-jager, security-auditor.
- **Grey hat**: hackers die zonder expliciete toestemming systemen onderzoeken, maar de eigenaar vervolgens wél op de hoogte brengen (in plaats van de bug te misbruiken of te verkopen).
- **Black hat**: de echte *stropers* — hackers die kwetsbaarheden misbruiken voor persoonlijk of crimineel gewin.

> **!** Belangrijk
>
> Sinds **februari 2023 is ethisch hacken legaal in België**. Voor die datum kon je als ethische hacker vervolgd worden, zelfs als je alleen maar een bug bij een bedrijf wilde melden. De nieuwe wet maakt het mogelijk om Belgische systemen te onderzoeken op kwetsbaarheden, *mits* je je aan een paar spelregels houdt:
>
> 1. Je mag **niet verder gaan** dan wat nodig is om de kwetsbaarheid te ontdekken.
> 2. Je moet het lek **zo snel mogelijk schriftelijk melden** bij het bedrijf in kwestie.
> 3. Je moet óók een melding doen bij het **Centrum voor Cybersecurity België (CCB)**.
> 4. Je mag de informatie **niet publiek maken** zonder toestemming.
>
> De wet werd sterk bepleit door ethische hackers zoals **Inti De Ceukelaire**, die eerder al aanklaagde dat hij jarenlang met één been in de gevangenis stond terwijl hij bedrijven gratis hielp hun beveiliging te verbeteren.

> **Tip**
>
> In de appendix achteraan dit boek vind je tal van boeiende en nuttige bronnen om op de hoogte te blijven over het reilen en zeilen in de schimmige wereld van de defensieve én offensieve cybersecurity wereld. Een goede startplek: grahamcluley.com voor dagelijks nieuws, en hackmageddon.com voor tweemaandelijkse *breach-overzichten*.

2. Cybersecurity fundamenten

i Leerdoelen

Na dit hoofdstuk kan je:

1. **Het CIA-model en de McCumber-kubus toepassen** op een gegeven (bedrijfs)scenario om te toetsen of aan alle beveiligingsaspecten gedacht werd.
2. De verschillende **soorten aanvallers** (scriptkiddies, werknemers, cybercriminelen, overheden…) onderscheiden op basis van motivatie, middelen en doelwit.
3. Het verschil tussen **passieve en actieve aanvallen** toelichten en ze herkennen in een concrete case.
4. De belangrijkste **malware-families** (virus, worm, trojan, rootkit, ransomware, botnet) én **social engineering**-technieken benoemen en illustreren met een actueel voorbeeld.
5. Uitleggen wat een **zero day** en een **window of vulnerability** zijn, en waarom **side-channel aanvallen** een bijzonder moeilijk aanvalsoppervlak vormen.

"Iedere goede film begint met een zwart scherm", zegt Batman in de "The Lego Batman Movie" ™ film. Wel, ik vul dit aan met "…en ieder goed hoofdstuk begint met een definitie."

Laten we daarom eerst eens een definitie van cybersecurity neerpennen dat netwerkbedrijf *Cisco* gebruikt:

"Cybersecurity is the practice of protecting systems, networks, and programs from digital attacks. These cyberattacks are usually aimed at accessing, changing, or destroying sensitive information; extorting money from users; or interrupting normal business processes.

Implementing effective cybersecurity measures is particularly challenging today because there are more devices than people, and attackers are becoming more innovative."

Alles draait met andere woorden rond het beschermen van informatie, of dat die nu op een server, in een document of in iemands hoofd zit.

Tip

De formele NIST-definitie: waar Cisco een praktische, marketing-vriendelijke omschrijving geeft, hanteert het Amerikaanse *National Institute of Standards and Technology* (NIST) een strakkere, juridisch-academische definitie in hun *Computer Security Handbook*:

> *"The protection afforded to an automated information system in order to attain the applicable objectives of preserving the **integrity**, **availability** and **confidentiality** of information system resources (includes hardware, software, firmware, information/data, and telecommunications)."*

Merk op dat NIST meteen drie centrale begrippen introduceert — integrity, availability, confidentiality — én vier types *resources*: hardware, software, firmware, data en telecommunicatie. Dit sluit perfect aan bij het **CIA-model** en de **McCumber kubus** die we hierna bespreken.

2.1 CIA en het security model

Data (of informatie), in welke vorm dan ook (berichten over een netwerk, bestanden op een harde schijf, tekst in een database), moeten beschermd worden, dat beseffen we nu. Het doel van onze data is dat deze voldoet aan het acroniem **C.I.A** wat staat voor:

- **C** voor "**confidentiality**": vertrouwelijkheid. De data kan enkel door zij die er recht toe hebben gebruikt worden. We gaan dit onder andere oplossen met behulp van encryptie en wachtwoorden.

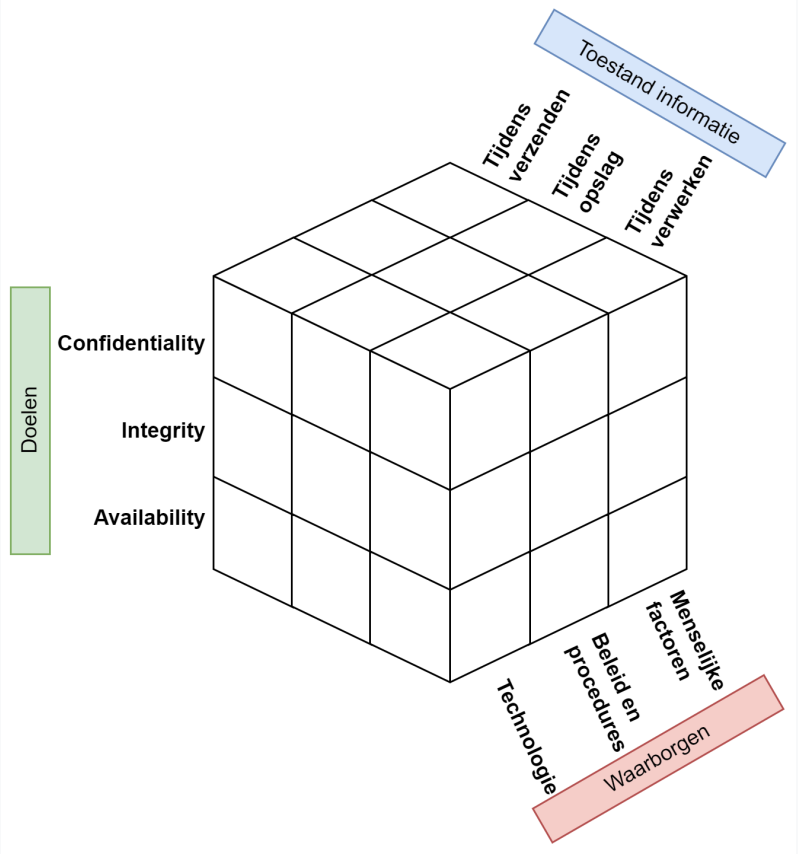
- **I** voor "**integrity**": integriteit. We moeten weten of onze data onbeschadigd is en niet werd aangepast door derden (of storingen). Bij bestanden gaan we bijvoorbeeld werken met zogenaamde (secure) hashes.
- **A** voor "**availability**": beschikbaarheid. Data die niet door rechtmatige gebruikers kan bereikt worden is onbestaande data. Zogenaamde "Denial-of-Service" (DoS) aanvallen hebben als doel deze pijler van CIA aan te vallen. Availability is een breed veld en wordt onder andere opgelost door back-ups, redundante servers enerzijds, en preventieve maatregelen anderzijds zoals firewalls, load balancers, etc.

> **i** Opmerking
>
> **CIA gaat dieper dan je denkt**: zowel confidentiality als integrity hebben in de praktijk **twee aparte dimensies** die vaak door elkaar gehaald worden.
>
> - **Confidentiality** valt uiteen in:
> - *Data confidentiality* — de **inhoud** van de data zelf is afgeschermd (bv. je bericht is versleuteld).
> - *Privacy* — het feit **dát er data bestaat** over jou, of wie ze mag verwerken, is beschermd. Denk aan GDPR: zelfs als niemand je medisch dossier effectief leest, is het al een privacy-inbreuk dat een onbevoegde werknemer het kan opvragen.
> - **Integrity** valt uiteen in:
> - *Data integrity* — de **data zelf** is niet aangepast (vandaar onze hashes).
> - *System integrity* — het **systeem** functioneert zoals bedoeld, zonder stille manipulatie. Een rootkit kan je data onaangeroerd laten maar toch je systeem *hijacken*: je data is integer, maar je systeem niet.
>
> **Availability** heeft daarnaast ook een extra laag: *non-repudiation* (onweerlegbaarheid) — kon de zender later ontkennen dat hij het bericht verstuurd heeft? Digitale handtekeningen lossen dat op.

2.1.1 McCumber kubus

De zogenaamde McCumber kubus, ontwikkeld door John McCumber in 1991, geeft een goed beeld weer waarom het steeds belangrijk is goed te beseffen binnen welke context we praten. Deze kubus stelt een model voor dat kan gebruikt worden om je ervan te vergewissen dat je aan alles denkt wanneer je je informatie wilt beschermen. De kubus bestaat uit drie dimensies en iedere dimensie bestaat uit een aantal aspecten. **Enkel wanneer we alle aspecten van alle dimensies in onze beveiligingsaanpak voorzien kunnen we hopen dat we onze C.I.A. doelen hebben bereikt.**

Om ons doel te bereiken (C.I.A.) moeten we ervoor zorgen dat we dit toepassen op alle vormen die onze data kan hebben (opslag, verzenden, verwerken). Dit kunnen we bewerkstelligen door technologische oplossingen (zoals encryptie wat in het volgende hoofdstuk wordt uitgespit), maar een niet onbelangrijke factor zijn ook de mensen die met de data moeten werken. Als zij zich niet aan de afspraken (procedures) houden en hun wachtwoorden gewoon op post-its aan hun scherm hangen, dan mag je een nog zo'n dure firewall hebben, het zal niet baten.



Figuur 2.1: De McCumber kubus.

i Opmerking

Twitter's hashing-bug (2018): een treffende illustratie waarom de McCumber kubus *alle* informatiestatus-aspecten moet dekken. Twitter hashte netjes wachtwoorden met **bcrypt** in hun database (status: *opslag*). Maar door een bug werden de wachtwoorden **vóór** het hashen in een interne log geschreven (status: *verwerking*). Resultaat: miljoenen wachtwoorden in plaintext terug te vinden in logs waar Twitter-medewerkers bij konden. Eén vergeten hoekje van de kubus volstaat om de hele beveiliging te omzeilen. Bron: Twitter Blog

⚠ Waarschuwing

De wereld van de cyberboswachters houdt van afkortingen, protocollen en vreemd klinkende standaarden. De wereldwijd gebruikte **ISO 27001** standaard (of norm) is er zo eentje, maar wel een erg belangrijke om te kennen. Wanneer een bedrijf of organisatie wil aantonen dat ze informatiebeveiliging hoog in het vaandel dragen, dan zullen ze deze norm trachten te behalen. Om als *ISO 27001 bedrijf* door het leven te mogen gaan moeten ze aan een hele hoop strenge eisen voldoen (die alle dimensies van de McCumber kubus omvatten, zou je kunnen zeggen) die in de standaard beschreven staan. Hierbij zal een externe auditor vervolgens controleren of je hier aan voldoet als bedrijf (en moet je dit elke drie jaar opnieuw doen).

2.2 Een ongelijke strijd

De McCumber kubus is een mooi concept, maar het is ook niet meer dan dat: een theoretisch framework. Het is ideaal om vanuit een high-level perspectief je ervan te vergewissen dat je aan alles hebt gedacht, maar in de praktijk komt er uiteraard bij alle aspecten van deze kubus aardig wat kijken.

De cyberboswachters van de 21e eeuw hebben geen eenvoudige job. Ze was in de vorige eeuw al pittig, de laatste jaren is de wapenwedloop er helaas alleen maar grimmiger op geworden.

- De aanvallers kunnen aan bijna supersonische **snelheid** aanvallen op duizenden, of zelfs miljoenen, systemen starten.
- "Alles is verbonden": onze huidige IT-netwerken zijn vele malen groter en complexer dan circa 20 jaar geleden. Hierdoor is ook de zogenaamde **attack surface** steeds groter. Gedaan zijn de tijden dat een middelgroot tot groot bedrijf genoeg had aan één cyberboswachter. Er zijn nu zelfs bedrijven die kunnen ingehuurd worden om bij problemen (of slimmer: vooraf als audit) te komen helpen om de boel te blussen.
- De tools die aanvallers, van welke aard ook, ter hun beschikking hebben zijn vaak ongelooflijk **eenvoudig** geworden. Gedaan is de tijd dat een ietwat stevige aanval kon gedaan worden door experts met tien jaar script- en netwerkervaring. Sommige vreeswekkende aanvallen vereisen niet meer dan het IP-adres van het slachtoffer in een invulveld invullen en vervolgens een klik op een grote rode knop "Start attack".
- Aanvallers schuimen *underground* fora af, op zoek naar de nieuwste *zero-day vulnerabilities* die ze in hun arsenaal kunnen opnemen. Fabrikanten van besturingssystemen en software kunnen de **snelheid waarmee nieuwe zwakheden** in hun systeem worden gevonden niet volgen.
- Door voorgaande snelheid van nieuwe zwakheden, duurt het ook langer en langer voor alles kan **gepatcht** worden door de ontwikkelaars van de software.
- Aanvallers kunnen **gigantische legers van computers en botnets** gebruiken om een ijzingwekkende hoeveelheid aan simultane aanvallen op één enkel slachtoffer uit te voeren.
- **Gebruikers** zijn een nog kleiner radertje in dit alles geworden en zullen nog sneller fouten maken (klikken op een link in een phishing e-mail bijvoorbeeld) dan voorheen, met alle gevolgen van dien.

Tip

De scheve grafiek: een beroemd diagram uit de security-wereld toont twee lijnen die in tegenovergestelde richting gaan over de voorbije decennia. De *attack sophistication* stijgt onafgebroken, terwijl de *knowledge required of the attacker* kelderde in diezelfde periode. Wat vroeger decennia ervaring vereiste, vraagt nu enkel een download en een muisklik. Dit verklaart waarom scriptkiddies een reëel probleem zijn geworden.

- 1980 : Password guessing : Self-replicating code
- 1990 : Sniffers : Session hijacking : Packet spoofing
- 2000 : WWW attacks : DoS : Automated probes : GUI-tools
- 2010 : Botnets : DDoS : Morphing malware : Stealth scanning
- 2020 : Ransomware-as-a-Service : AI-gedreven aanvallen : Supply-chain attacks

Aanvalstools worden steeds krachtiger — terwijl de vereiste kennis van de aanvaller alleen maar daalt.

2.2.1 Zero days en patching

Aanvallers hebben voor zero days aardig wat geld over. Een zero day kopen is een aanval kopen die gegarandeerd zal werken daar deze een zwakte misbruikt die nog niet bij de maker van de software gekend is.

Een zero day zal quasi gegarandeerd blijven werken tot de ontwikkelaars een nieuwe patch ervoor maken. Maar zelfs dan blijven zero days nuttig: het is niet omdat er een patch bestaat, dat de doelwitten deze patch ook effectief reeds geïnstalleerd hebben. Veel bedrijven hebben nu erg strenge *patching policies* maar toch blijft het dweilen met de kraan open: er moet maar één systeem niet gepatcht zijn tegen de zero day van de aanvallers en het gaatje in de verdedigingslinie is gevonden en kan misbruikt worden.

De meeste fabrikanten van besturingssystemen (Apple, Microsoft, etc.) en veelgebruikte softwarepakketten (Adobe, Microsoft, etc.) brengen patches op welbepaalde dagen uit. Dit zorgt er bijvoorbeeld voor dat systeembeheerders hier rekening mee kunnen houden in hun wekelijkse planning. Voor gebruikers van zero days is dit ook nuttig: er ontstaat een zogenaamde **window of vulnerability**. Dit is de periode tussen het "ontdekken en in gebruik nemen van een zero day" en de moment waarop de patch tegen de zero day wordt verspreid. In dit *window* heeft de aanvaller vrij spel daar geen enkel systeem al kan gepatcht zijn.

> **Opmerking**
>
> **Patch Tuesday & Exploit Wednesday**: Microsoft rolt sinds 2003 haar beveiligingsupdates standaard uit op de **tweede dinsdag van elke maand** — bekend als *Patch Tuesday*. Voordeel voor systeembeheerders: voorspelbaarheid in hun onderhoudsplanning. Maar... aanvallers weten dit óók. De dag erna wordt in de security-wereld vaak **Exploit Wednesday** genoemd: aanvallers doen *reverse-engineering* op de patch om te achterhalen **welke kwetsbaarheid** er juist gedicht werd, en richten zich dan op alle systemen die nog niet zijn bijgewerkt. Het *window of vulnerability* is in die zin vaak verrassend goed te voorspellen.

> **Tip**
>
> In 2021 verscheen "How they tell me the world ends" van New York Times journaliste Nicole Perlroth. Dit boek is erg ontluisterend en geeft een griezelig inzicht in hoe het er momenteel aan toe gaat in de schimmige wereld van zero days, offensieve cybersecurity, etc. Bekijk bijvoorbeeld maar eens de twitter-account van Chaouki Bekrar (twitter.com/cbekrar), oprichter van Zerodium, een zero-days *broker* en huiver bij de gigantische prijzen die zero days waard kunnen zijn (soms meer dan één miljoen dollar...).

2.3 Wie zijn de stropers?

Waar moeten we ons tegen beschermen als cyberboswachter? Het zou verleidelijk zijn om ons te richten op één specifieke doelgroep en ons daar volledig tegen te wapenen. Helaas is het niet te voorspellen van wie je last zal hebben. Iedere groep aanvallers heeft eigen motivaties en middelen en het is niet altijd evident om tegen ieder iets te doen. Het kan natuurlijk geen kwaad om tenminste te weten met welke groepen je mogelijk zal geconfronteerd worden:

- **Hackers**: waarbij we een onderscheid moeten maken tussen white, black en grey hat hackers uiteraard.
- **Scriptkiddies**: een groep die soms te laat beseft dat ook zij dingen doen die erg strafbaar zijn.
- **Werknemers**: een vaak over het hoofd gekeken, maar oh zo veel voorkomende groep.
- **Cybercriminelen**: dé grootste plaag, nog steeds.
- **Cyberterroristen**: rebel, vrijheidsstrijders, terrorist. *It's all in the eye of the beholder* uiteraard.
- **Spionnen** : *James Bond of the WWW.*
- **Overheden** (of door overheden gesponsord): een probleem dat steeds groter blijkt te worden.

2.3.1 Hackers

Wanneer een digitale stroper niet onder één van de andere noemers kan gezet worden dan wordt hij "hacker" genoemd, een soort catch-all term die soms een positieve, soms een negatieve connotatie heeft. Om toch wat onderscheid mogelijk te maken kunnen we zeggen dat er drie soorten hackers zijn:

- **Whitehat**: *"de goei"*. Dit zijn hackers die onder duidelijke afspraken met hun doelwit de sterktes en zwaktes van een systeem zullen testen door deze te proberen te omzeilen. White hat hackers werken volledig binnen de krijtlijnen van de wet en zullen enkel die zaken testen waartoe zij recht hebben. Denk bijvoorbeeld aan audit-bedrijven die de beveiliging van andere bedrijven zullen *pentesten* (penetration testing: trachten in een systeem te geraken) of bounty hunters die bijvoorbeeld via *intigrity.com* op zoek gaan naar nieuwe problemen bij een website of product.
- **Greyhat**: letterlijk een grijze zone. Grey hat hackers hebben meestal een positief doel maar zullen niet altijd volgens "de regels van de wet werken". Denk maar aan een hacker die ongevraagd een lek ontdekt in een bedrijf en dit ook rapporteert. De kans is bestaande (maar gelukkig kleiner dan vroeger) dat het bedrijf in kwestie hier niet mee opgezet is en alsnog de hacker zal aanklagen. Vergelijk grey hat hackers met een soort "goede dief": hij gebruikt zijn expertise om ongevraagd huizen binnen te breken om dan vervolgens, zonder iets te stelen, een briefje achter te laten met uitleg hoe de inbreker net is binnen geraakt.
- **Blackhat**: de "bad guys" van de hoop en ook wel *crackers* genoemd. Black hat hackers zullen systemen aanvallen waar ze geen toestemming voor hebben en meestal met als doel om niet legale resultaten (lees: geld, kennis of macht) te bereiken ten koste van het doelwit.

Recent zijn er nog 3, minder vaak gebruikte, types die aan de hand van een kleur specifieke hacker-types definiëren:

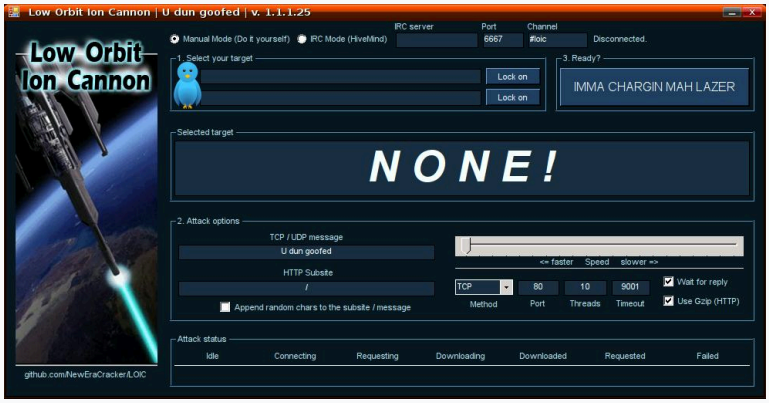
- **Redhat**: dit zijn hackers die als doel hebben blackhat hackers het leven moeilijk te maken. Ze worden ook soms *vigilantes* genoemd. De middelen die de redhat hackers gebruiken zijn echter niet noodzakelijk legaal en dus alhoewel hun doel nobel is, moeten we toch vraagtekens plaatsen bij hun werkwijzen.
- **Bluehat**: de wraaklustige hackers. Bluehat hackers hebben maar 1 doel, en dat is wraak nemen op iets of iemand, gebruik maken van alle middelen beschikbaar. Laten we auteur Jodi Picoult citeren die het volgende zegt over wraak: *"When you begin your journey of revenge, start by digging two graves: one for your enemy, and one for yourself"*...
- **Greenhat**: de jonkies, ook wel scriptkiddies genoemd. We zullen deze in de volgende sectie uit de doeken doen.

2.3.2 Scriptkiddies

Iemand die weinig tot niets kent van cybersecurity maar toch bestaande tools uittest op slachtoffers, wordt ook wel een scriptkiddie genoemd. Deze groep mensen gebruikt tools die eenvoudig in gebruik zijn, zonder altijd goed te weten wat de tool doet én wat de gevolgen ervan zijn. Doordat tools steeds krachtiger worden, zijn ook de gevolgen van scriptkiddie-aanvallen steeds drastischer. Wat deze gebruikers vaak vergeten is dat hun acties strafbaar zijn. Zo vergeten ze dat klikken op een knopje in een programma, vanuit hun gezellige bureaukamer thuis, erge gevolgen kan hebben voor hun doelwit. Het gebeurt dan ook geregeld dat een scriptie onzacht én snel in aanraking met het gerecht komt. De veiligheidsdiensten die op cyberaanvallen reageren kunnen natuurlijk niet zien wie er achter een aanval zit en zullen dus altijd op dezelfde manier reageren. Een inval van een team zwaar bewapende agenten omdat een bank of webshop al dagen wordt lamgelegd is een "ervaring" die een scriptkiddie nog lang zal herinneren (en zien op z'n lege spaarboekje ten gevolge van de grote boetes die hij of zij nog jaren zal moeten afbetalen)

Tip

Low Orbit Ion Cannon is zo'n typische scriptkiddie tool: eenvoudig in gebruik, met potentieel grote gevolgen voor het slachtoffer. Deze tool, wanneer meerdere gebruikers hem samen gebruiken, voert DDOS-aanvallen op het doelwit uit. In 2010 werd de tool bijvoorbeeld gebruikt om de servers van Visa, MasterCard en Paypal uren lang lam te leggen als wraak op het feit dat deze betalingsdiensten geen betalingen voor WikiLeaks meer aanvaardden. Meer informatie.



Figuur 2.2: Low Orbit Ion Cannon UI (Bron Wikipedia).

2.3.3 Werknemers

Werknemers die niet tevreden zijn over hun baas of werkomgeving, of net zijn ontslagen, zijn een veelvoorkomend probleem (volgens sommige experts zelfs hét belangrijkste cybersecurity probleem). Ze zitten vaak letterlijk "aan de binnenkant" van de beveiligingssystemen en kunnen daardoor ook veel meer schade aanrichten als ze dat willen. Ontevreden werknemers die wraak nemen op hun werkgever is een veel voorkomend probleem. Zeker als die werknemer op de koop toe bijvoorbeeld de netwerk-administrator was. Er zijn verhalen van ex-admins die voor hun vertrek de IT-systemen van het bedrijf saboteerden en, als klap op de vuurpijl, dan ook nog eens losgeld eisten om de systemen terug up-and-

running te brengen. Het hoeft natuurlijk niet zo spectaculair zijn. Of wat te denken van werknemers die waardevolle documenten stelen, doorverkopen of aanpassen zonder dat het bedrijf er erg op heeft.

Werknemers zijn ook vaak onbedoeld de oorzaak van veel problemen: ze gaan misschien slordig om met de manier waarop ze hun wachtwoord bewaren, waardoor anderen via hun account kunnen inbreken. Ze laten mensen binnen die gekleed zijn als "collega's" zonder te vragen of ze wel werknemer van het bedrijf zijn. Of ze installeren bijvoorbeeld een extra draadloos access point om een betere wifi-dekking op de bureau te hebben, waardoor dit toestel plots een veel minder goed beveiligd doelwit is dat hackers kunnen misbruiken.

> Tip
>
> Veel van dit soort problemen kunnen voorkomen worden door een doordacht "identity management"-systeem dat er voor zorgt dat de accounts van recent ontslagen werknemers ogenblikkelijk worden verwijderd of dat tenminste de toegang tot bedrijfskritische systemen uitschakelt. Voorts moet personeel (op alle niveaus!) opgeleid en getraind worden zodat ook dit facet van de McCumber kubus gedekt wordt.

De laatste tien jaar heeft het **Bring your own device (BYOD)**-concept in bedrijven voor een extra dimensie gezorgd waar cyberboswachters rekening mee moeten houden. Vroeger kon je je verdediging opbouwen als een soort ommuurde burcht waarbij je er van uit mocht gaan dat alles "binnen de burcht" veilig was. Door BYOD gaat dit concept natuurlijk niet meer op: gebruikers wandelen bedrijven binnen met hun eigen laptop, tablet, smartwatch en smartphone, etc. Allemaal apparaten die potentieel door stropers vooraf, bij de gebruiker thuis, werden geïnfecteerd. Van zodra dit besmette apparaat dan in het bedrijf wordt geïntroduceerd heeft de aanvaller mogelijks ongelimiteerde toegang tot het bedrijfsnetwerk. Kortom, we moeten nu ook verdedigingsmuren rondom de individuele werknemers én hun apparaten bouwen. Denk daarbij aan on-device firewalls en virusscanners, maar ook netwerk-authenticatie voor ieder toestel, etc.

2.3.4 Cybercriminelen

Money talks zegt men wel eens, en dat geldt (geld, snap je'm ;)) zeker voor deze groep. Cybercriminelen doen wat criminelen al millennia doen en dat is rijkdom die hen niet toebehoort proberen te pakken krijgen. Geld, geld, geld is de motivatie van deze groep mensen. Een onderzoek in 2021 (bron) schat dat meer dan 700 miljard dollar verlies werd opgetekend ten gevolge van online criminaliteit. Doordat steeds meer mensen hun betalingen en identiteiten (denk maar aan de vele its-me phishing sms'jes dat je geregeld krijgt) online beheren wordt ook de groep potentiële slachtoffers steeds groter.

Cybercriminaliteit wordt soms wel eens de motor van de cybersecurity genoemd omdat ze steeds blijven innoveren en zoeken naar nog betere manieren om onschuldige slachtoffer hun centjes te stelen. Hierdoor moeten ook de boswachters steeds blijven vernieuwen.

Cybercriminaliteit is nu zelfs zo ver geëvolueerd dat ze heuse moderne bedrijfsconcepten overnemen en hun zaakje als echte bedrijven runnen. Moderne ransomware criminelen hebben zelfs helpdesks die je kan bellen om je te helpen om de betaling (de *ransom*) te regelen. Of wat te denken van website die botnets verhuren als waren het legale services. Hier en daar zie je nu zelfs het "-as a service" zinnetje verschijnen waarbij bijvoorbeeld "Ransomware as a Service" (RaaS) of "spam as a service" kan gehuurd worden. Het doel hierbij is natuurlijk om de strafbare feiten zoveel mogelijk te verleggen naar de persoon die de services inhuurt, en niet naar de aanbieder ervan.

> Opmerking
>
> **Max Butler alias *Iceman***: in 2006 nam Butler, vanuit een appartement in San Francisco's Tenderloin, de **wereldwijde zwarte markt voor gestolen kredietkaarten** over. Niet door technisch de beste te zijn, maar door zijn concurrenten gewoon te *hacken* en hun databases met gestolen cards over te nemen. Hij kreeg 13 jaar cel — maar het verhaal stopt daar niet. In **2018** werd hij vanuit de gevangenis aangeklaagd voor een poging om met een **drone** een smartphone binnen te smokkelen, zodat hij z'n business kon heropstarten. Wired: *One Hacker's Audacious Plan* — Washington Times: *Iceman with a drone*

> **i** Opmerking
>
> **Phishing & *money mules***: cybercriminelen werken zelden rechtstreeks met het gestolen geld — dat laat te duidelijke sporen na. Ze rekruteren *money mules* (geldezels): gewone mensen die via een jobadvertentie denken een legitieme job te hebben als *"financial manager"*. Zij stellen hun bankrekening ter beschikking om gestolen geld door te sluizen, meestal via Western Union, richting de echte criminelen. In november 2017 werden bij één actie van Europese opsporingsdiensten **159 geldezels** gearresteerd. Moraal: als een online vacature verdacht goed klinkt ("verdien €3000 per maand van thuis uit, enkel uw rekening ter beschikking stellen")... is het dat ook. Bron: HLN, 28 nov 2017

2.3.5 Cyberterroristen, spionnen en overheden

De laatste drie groepen bespreken we samen, ook al omdat de termen soms overvloeien afhankelijk aan wie je vraagt om iets of iemand met dit label te bestempelen. Zoals reeds in het eerste hoofdstuk aangehaald is de cyberwereld tegenwoordig ook een belangrijk terrein waar geopolitieke ruzies op worden uitgevochten. Er bereiken ons steeds meer berichten van de exploten die hier doorgaan. De financiële en technische middelen die deze groep voorhanden heeft voor zowel offensieve als defensieve cyberacties is meestal immens groter dan van alle andere stropers in dit overzicht. We zagen ooit een presentatie waarin een cybersecurity expert ietwat lachend sprak over het "Mossad / Non-Mossad verdedigingsprincipe" (Mossad is een Israëlische geheime dienst en staat in de top van *strafste* cybersecurity expertise). Het principe gaat uit van de manier waarop je je beveiliging opbouwt: ga er van uit dat de Mossad in je systemen zal geraken, ongeacht hoeveel geld en personeel je tegen het probleem aan gooit. Het is met andere woorden efficiënter dat je een realistische inschatting maakt van je tegenstanders (qua expertise en middelen) en daar specifiek je op richt, waarbij je natuurlijk het *low hanging fruit* niet over het hoofd ziet.

> **⚠** Waarschuwing
>
> **Een (onvolledige) geschiedenis van state-sponsored cyberaanvallen**:
>
> - **Titan Rain** (2005): Chinese aanvallen op honderden Amerikaanse overheid- en defensiesystemen, waarbij onder andere militaire hardware-designs werden gestolen.
> - **Estonia** (2007): massale DDoS-aanval op Estlandse overheid, banken en media, toegeschreven aan Russische actoren na een diplomatiek incident rond een sovjet-standbeeld. Wordt door sommigen *"de eerste cyberoorlog"* genoemd.
> - **Georgia** (2008): synchroon met de Russische militaire invasie werden Georgische overheidsites en infrastructuur lamgelegd — de eerste keer dat cyberaanvallen parallel liepen met een militaire inval.
> - **Operation Aurora** (2009): Chinese aanval op Google, Adobe, Juniper, Yahoo, Rackspace, Morgan Stanley en tientallen andere bedrijven, specifiek gericht op broncode-repositories.
> - **Stuxnet** (2010): Amerikaanse/Israëlische worm die Iraanse kerncentrifuges saboteerde (zie hoofdstuk 1).
> - **Nitro Zeus**: zwaar geclassificeerd Amerikaans cyber-sabotageprogramma gericht tegen Iraanse infrastructuur, als plan B mocht de nucleaire deal mislukken.
>
> Het boek *"The Perfect Weapon"* van David Sanger (NYT, 2018) geeft een uitstekend overzicht van cyberoorlog als vast instrument van moderne diplomatie.

i Opmerking

Hacktivisme in de Oekraïne-oorlog: sinds de Russische invasie van Oekraïne in februari 2022 zijn tienduizenden hacktivisten online tot actie overgegaan — aan *beide* kanten. Westerse vrijwilligers (het zogenaamde "IT Army of Ukraine") viseerden Russische banken, staatsmedia en overheidswebsites; pro-Russische groepen verstoorden Oekraïense infrastructuur en westerse bedrijven. Wired documenteerde hoe deze *"hacktivist pandemonium"* soms ongecoördineerd en onvoorspelbaar uitpakt: goedbedoelde aanvallen raakten soms neutrale partijen, en bedrijven konden geen onderscheid meer maken tussen staatsactoren en individuele activisten. Bron: Wired, maart 2022

Figuur 2.3: Hacktivisme tijdens de Oekraïne-oorlog.

2.4 Hoe vallen ze aan?

Alhoewel voorgaande groepen van aanvallers allemaal erg specifieke redenen hebben, kunnen we toch hun aanvallen generaliseren in vijf duidelijke stappen:

1. **Verkennen**: passieve *reconnaissance*. Voor de stropers hun aanval effectief starten zullen ze eerst hun doelwit(ten) onderzoeken. Ze zoeken zo naar de eenvoudigste, of meest verborgen, manier om in een systeem te geraken. Verkennen wordt ook wel passieve reconnaissance genoemd omdat in deze fase het doelwit bijna nooit kan detecteren dat de stroper actief is. Deze fase gaat erg breed en is niet beperkt tot enkel "tools" gebruiken. In deze fase zal de stroper ook vaak via zoekmachines, social media en *dumpster diving* proberen meer te weten te komen over het bedrijf, de werknemers, etc.
2. **Scannen**: actieve *reconnaissance*. Van zodra de stroper een breed overzicht heeft van z'n doelwit zal hij overgaan op actieve scanning. Denk hierbij aan port scanners, vulnerability scanners, etc. Uiteraard is deze fase actief én bestaat er dus de kans dat de boswachters dit tijdig opmerken en zo de aanvallen in de volgende fase kunnen afslaan.
3. **Toegang verkrijgen**: door het scannen weet de stroper nu welk systeem hij zal benaderen om toegang tot bijvoorbeeld het bedrijfsnetwerk te krijgen. Meestal heeft de aanvaller een systeem gedetecteerd in de vorige fase met een gekende kwetsbaarheid, of hij heeft bijvoorbeeld via spear phishing een backdoor bij een werknemer geïnstalleerd, etc. In deze fase zal de aanvaller ook trachten steeds meer rechten te krijgen zodat hij steeds meer kan gedaan krijgen op de *gepwnde* systemen.
4. **Toegang bestendigen**: van zodra de stroper *in het systeem* zit, begint hij als eerste z'n toegang te bestendigen. De aanvaller kan niet voorspellen wanneer het systeem waar hij op zit wordt uitgeschakeld, verwijderd, etc. Kortom, de stroper begint nu naarstig z'n toegang te garanderen door bijvoorbeeld een (nieuwe) backdoor te installeren die ook later actief zal zijn. Voorts zal hij ook mogelijk andere systemen in de buurt aanvallen zodat hij niet beperkt is tot één systeem (en er dus ook geen *"single point of failure"* is voor de stroper).

5. **Sporen wissen**: hoe langer de stroper uit het vizier van de boswachters kan blijven, hoe effectiever hij z'n doelen kan behalen. Een behendig stroper zal er dan ook voor zorgen dat zijn sporen ondetecteerbaar blijven door het aanpassen of wissen van logs, het verwijderen van backdoors, etc.

Tip

Om de kracht van de tools die stropers voorhanden hebben aan den lijve te ontdekken, is het aangeraden om Kali OS te installeren. Deze Linux-distributie (die je ook virtueel kunt draaien) zit tjokvol pentest-tools die zowel stropers als boswachters constant gebruiken. Dit OS laat je toe om alle stappen van de aanvaller te simuleren.

Leer zeker ook werken met Metasploit, dat ook in Kali zit. Het Metasploit project is een verzameling erg krachtige pentest tools, inclusief de nieuwste snufjes om bijvoorbeeld malware in een pdf te embedden, etc.

Opmerking

Wanneer een netwerk wordt gepentest (legaal) werkt men vaak met twee teams die tegen elkaar strijden. Het *red team* speelt de rol van de digitale stropers, terwijl het *blue team* als boswachters zal proberen de aanvallen te verijdelen.

Belangrijk

Dwell time — de stille meerderheid: cijfers uit de (ondertussen legendarische) Verizon *Data Breach Investigation Reports* schetsen een ontluisterend beeld. Aanvallers zijn razendsnel *binnen*, maar blijven vaak *maandenlang* onopgemerkt.

Overgang	Sec.	Min.	Uren	Dagen	Weken	Maanden	Jaren
Aanval → inbraak	10%	**75%**	12%	2%	0%	1%	0%
Inbraak → data stelen	8%	38%	14%	25%	8%	8%	0%
Inbraak → **ontdekking**	0%	0%	2%	13%	29%	**54%**	2%
Ontdekking → herstel	0%	1%	9%	32%	38%	17%	4%

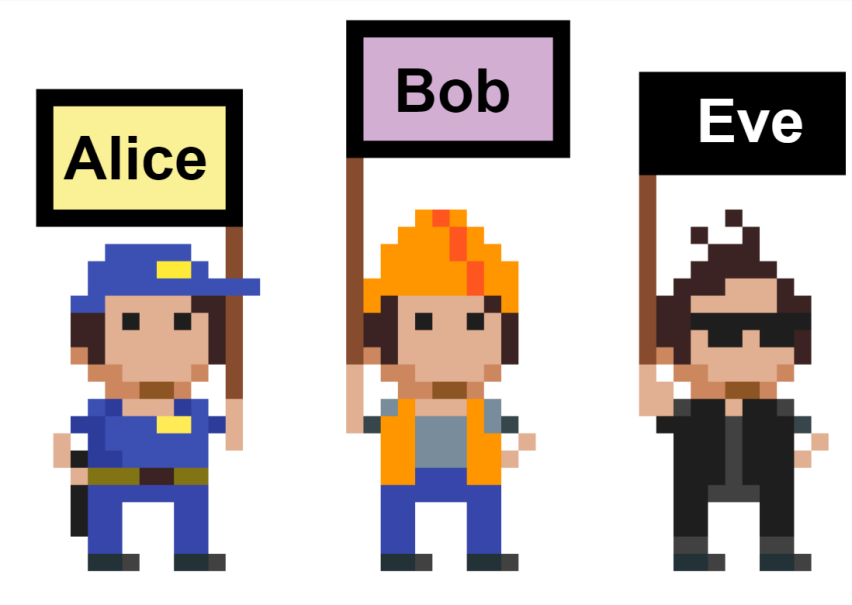
De meest hallucinante regel is de derde: in meer dan de helft van de gevallen ontdekt een organisatie pas **na maanden** dat er iemand in haar netwerk zit.

Kortom: je **detectiecapaciteit** is minstens even kritisch als je preventie. Als je aanvaller maanden ongestoord rondwandelt, heeft hij alle tijd van de wereld om fase 4 (*bestendigen*) en fase 5 (*sporen wissen*) rustig uit te voeren. Investeer dus niet alleen in dikke muren, maar ook in alerte bewakers. *Bron: Verizon DBIR.*

2.5 Classificatie van aanvallen

Dit hoofdstuk begon met een definitie, dan moeten we zeker ook eens enkele zaken classificeren. Alles (moet gedaan worden) om ietwat wetenschappelijk over te komen, niet waar. In dit geval gaan we eens kijken hoe we, algemeen gezien, de typische aanvallen kunnen classificeren die kunnen optreden in de cyber security wereld. Specifiek zullen we met drie personages werken:

- Alice en Bob: zij zijn de *goeie* en willen op een veilige manier met elkaar communiceren.
- Eve: zij is de aanvaller/hacker/crimelord/snoodaard die het leven van Alice en Bob zuur wil maken in haar voordeel. Mogelijk wil ze te weten komen wat Alice en Bob met elkaar afspreken, misschien wil ze ervoor zorgen dat de berichten van Alice niet bij Bob aankomen, etc.



Figuur 2.4: Meet the crew.

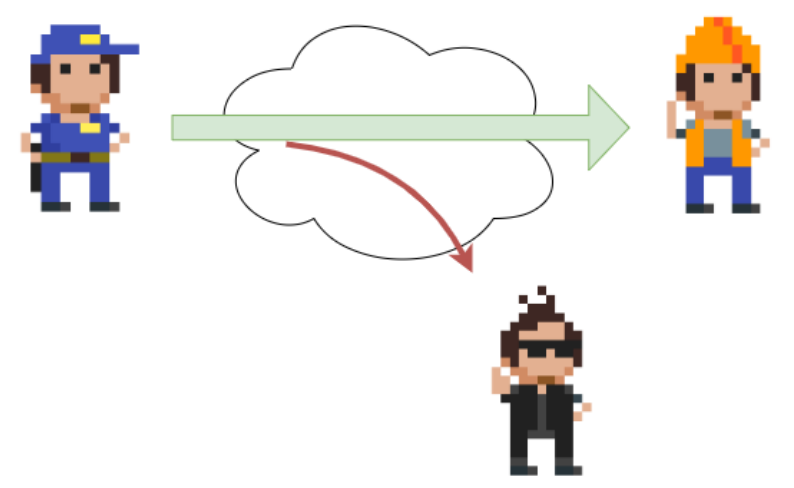
> ⚠ Waarschuwing
>
> Herinner je even aan de McCumber kubus: Alice en Bob zijn eender welk start- en eindpunt van onze informatie, in welke vorm dan ook. Als je bijvoorbeeld C.I.A. vereist in een computer dan is Alice bijvoorbeeld je CPU en Bob het RAM-geheugen (om maar iets te zeggen). Kortom, probeer altijd al deze informatie breed genoeg te plaatsen en niet alleen aan het klassieke "netwerkcommunicatie"-systeem te denken waarin Alice over een netwerk een boodschap naar Bob stuurt.

Eve kan op allerlei manieren *aanvallen* en in de eerste plaats kan dat **actief** of **passief** zijn: bij passieve aanvallen zal Eve enkel *luisteren* (en wachten) op de communicatie tussen Alice en Bob. Hierdoor zijn passieve aanvallen veel moeilijker om te detecteren (soms zelfs onmogelijk), maar uiteraard is Eve 100% afhankelijk van wat Bob en Alice doen, daar ze geen dwingende hand heeft in hun communicatie. Bij actieve aanvallen is dat omgekeerd: de pakkans is groter (en afhankelijk van het type aanval), maar ze kan ook mogelijk de communicatie tussen Bob en Alice sturen in de richting die zij nodig heeft om haar aanval uit te voeren.

Figuur 2.5: Passieve vs actieve aanvallen.

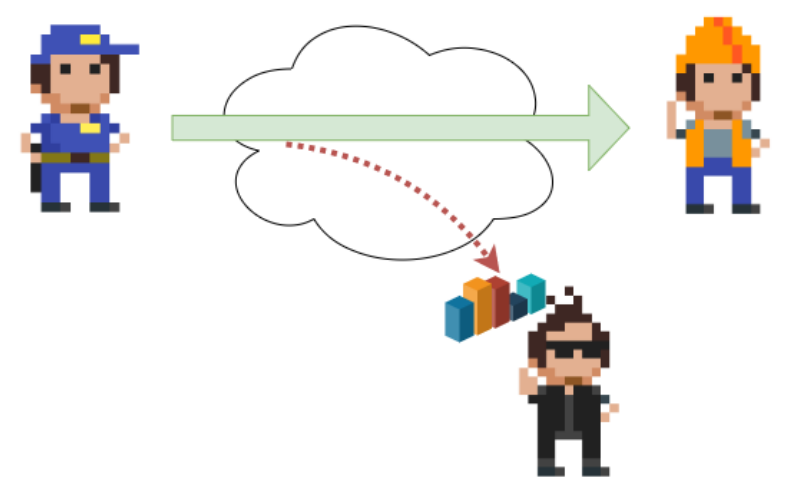
2.5.1 Passieve aanvallen

Bij een **sniffing** aanval gebruikt Eve een *sniffer* (bijvoorbeeld Wireshark indien ze netwerk-traffiek wenst te meten) om alle communicatie tussen twee eindpunten te zien. Alhoewel data steeds vaker geëncrypteerd wordt, zal Eve toch vaak erg nuttige informatie uit deze moeilijk te detecteren aanval kunnen halen. Denk maar aan MAC-adressen, algemene gebruikersinfo, etc. We staan er niet altijd bij stil hoeveel netwerk-traffiek tegenwoordig constant over netwerken over en weer vliegt. Daarbij komt ook een iets recenter fenomeen: de minder veilige third-party apps van bekende merken. Applicaties gemaakt door derden volgen mogelijk niet altijd de strenge beveiligingscriteria van het bedrijf waarvoor ze een app hebben gemaakt. Hierdoor bestaat er de kans dat sommige apps zelfs flagrante fouten maken en bijvoorbeeld user credentials onbeveiligd opslaan, of erger, over het netwerk sturen. Dit soort apps maken het werk voor Eve die aan het sniffen is dan ook erg gemakkelijk.



Figuur 2.6: Passieve aanval, type 1: Sniffing.

Soms is geen traffiek kunnen sniffen ook nuttig voor Eve. Later behandelen we nog side-channel aanvallen, maar we bespreken nu toch al deze vaak vergeten broer van de sniffing-aanval: de **trafiek analyse**. Door het registreren wanneer en hoe een doel communiceert kan Eve ook erg veel informatie op een passieve manier te pakken krijgen. In de eerste plaats laat het de stroper toe om te weten wanneer een gebruiker actief is en wanneer niet. Sommige systemen laten een alarm afgaan als ze zien dat een legale gebruiker op een onverwacht moment actief is, iets waar Eve nu rekening mee kan houden. Voorts laat het Eve ook toe om te ontdekken wat voor activiteiten het slachtoffer gebruikt (zijn er veel e-mail-gerelateerde berichten? Of net veel VoIP-calls?).

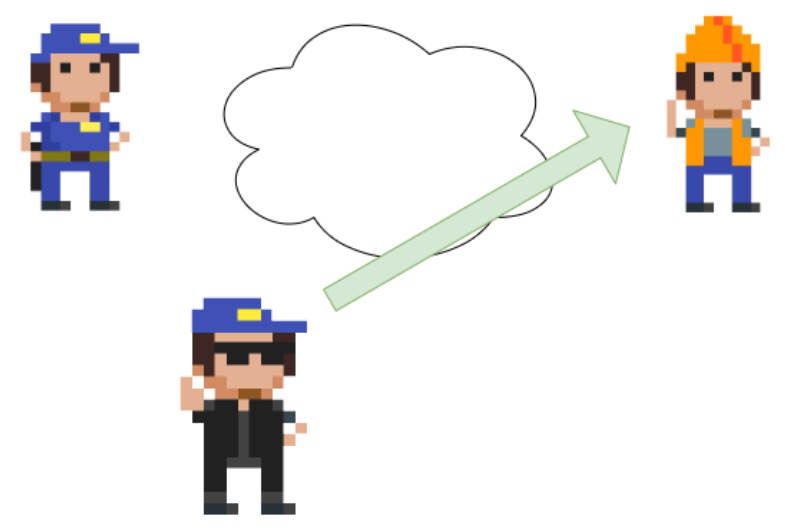


Figuur 2.7: Passieve aanval, type 2: Trafiek analyse.

2.5.2 Actieve aanvallen

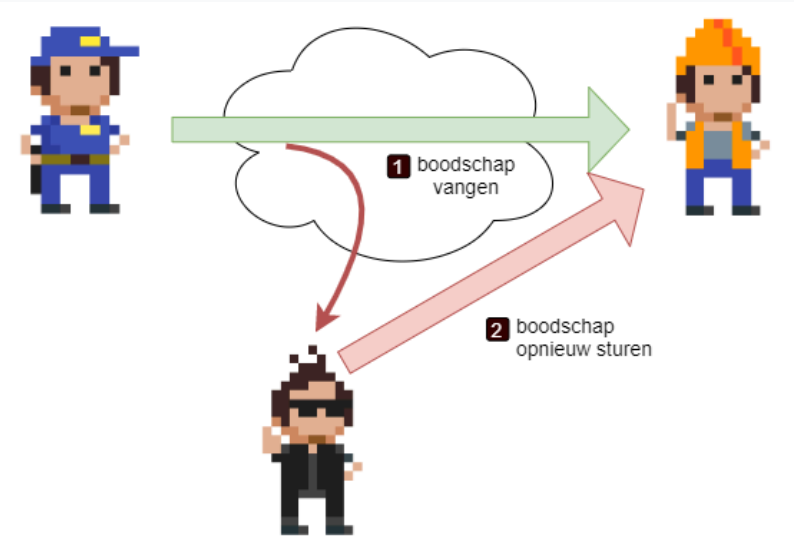
Het domein van de actieve aanvallen is natuurlijk het domein waar Eve de meeste slaagkansen zal produceren, maar ze heeft ook een veel hogere kans op gevat te worden. Om die kans te verkleinen zal de stroper bijna altijd de aanval uitvoeren door zich als iemand anders voor te doen: *masquerading*. Via **spoofing** zal de stroper de digitale identiteit van een legitieme gebruiker overnemen (denk maar aan MAC-spoofing waarbij Eve het hardware adres van een bedrade of draadloze netwerk-kaart overneemt). Masquerading heeft een dubbel doel:

1. Het zal de daaropvolgende aanvallen moeilijker kunnen linken aan Eve, daar ze onder een pseudoniem actief is.
2. Het zal Eve mogelijk toegang verschaffen tot bronnen waar ze onder haar eigen *identiteit* niet de juiste rechten toe heeft.



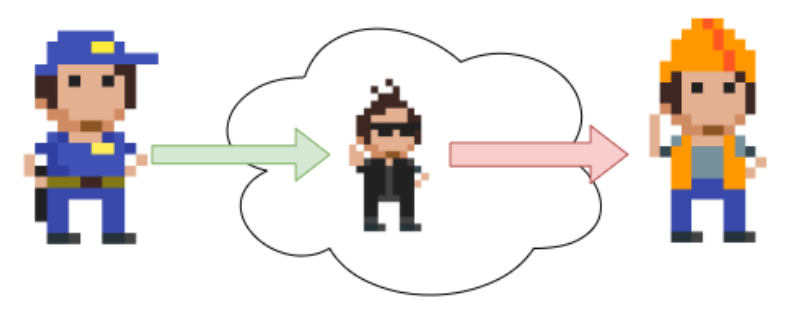
Figuur 2.8: Actieve aanval, type 1: Masquerading.

Het tweede type actieve aanvallen zijn **replay**-attacks. Hierbij zal de stroper eerder bewaarde, legitieme, communicatie heruitzenden in de hoop dat de ontvanger er zich geen vragen bijstelt. Beeld je in dat Eve een login-pakket heeft gesnift van een erg zwak beveiligd systeem: als Eve de volgende dag wil inloggen onder de naam van haar slachtoffer dan hoeft ze enkel dat bewaarde pakket opnieuw te versturen.



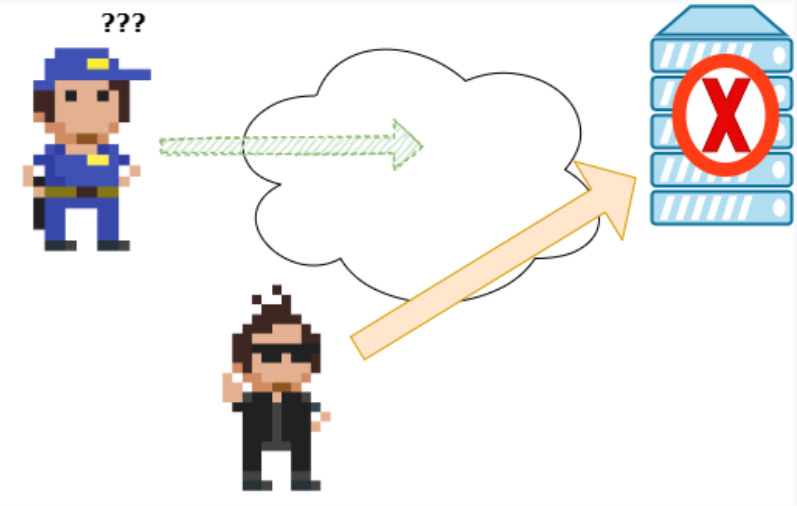
Figuur 2.9: Actieve aanval, type 2: Replay attack.

Type 3 is vanuit het standpunt van de aanvaller de interessantste: de **man-in-the-middle** of **MitM**-aanval. Hierbij zal Eve zich tussenin de communicatie van Bob en Alice nestelen met als doel op een onzichtbare manier hun communicatie te lezen, aanpassen of blokkeren. Het laat Eve toe als een soort *puppetmaster* de volledige communicatie te bepalen en beïnvloeden. Deze aanval is erg krachtig, maar vereist ook vaak een stevige technische opbouw door Eve daar ze nu twee eindpunten heeft die ze met behulp van onder andere masquerading moet aanvallen.



Figuur 2.10: Actieve aanval, type 3: Man-in-the-middle aanval.

Als laatste de meest voorkomende aanval: de **Denial-of-Service** (DoS). Deze aanval heeft als doel om een systeem of gebruiker *lam te leggen* zodat deze niet meer voor andere gebruikers of systemen bereikbaar is. De reden om een DoS uit te voeren zijn velerlei en de manier waarop deze uitgevoerd kan worden is ook quasi eindeloos: de stekker uittrekken, gigantische hoeveelheden communicatie versturen, of het signaal verstoren met een microgolf-oven. Alles is mogelijk en het hangt vooral van de creativiteit van de aanvaller af hoe effectief de aanval is.



Figuur 2.11: Actieve aanval, type 4: Denial-of-Service aanval.

2.6 Hoe verdedigen

Wat en wie je wilt verdedigen in de cyberwereld kan erg gevarieerd zijn. Toch kunnen we alles herleiden tot 4 + 1 fundamentele principes die je best hanteert indien je een systeem van welke vorm ook wenst te beschermen tegen cyberstropers. Deze zijn:

- **Layering**: bouw je beveiliging zoals de lagen van een ui rondom je te beschermen data, gebruikers en services. Hoe meer lagen hoe beter, op voorwaarde dat ze natuurlijk verschillend zijn. Het voordeel van met meerdere lagen werken is evident: indien één laag om welke reden dan ook gecompromitteerd raakt, zijn er nog steeds de andere lagen die als verdediging werken.

- **Limiting**: beperk steeds maximaal wat iedereen binnen je systeem kan. Zorg ervoor dat gebruikers en services enkel die zaken kunnen doen waartoe ze recht hebben volgens hun rol. Geef dus niet iedereen admin-rechten, zet je firewall niet op *allow all*, etc.
- **Diversity**: zorg ervoor dat je een verscheidenheid aan beveiligingen hebt. Op die manier voorkomen we, net als bij layering, dat het falen van één systeem je hele verdediging neerhaalt.
- **Simplicity**: KISS, oftewel *keep it simple, stupid.* Al het voorgaande lijkt te doen uitschijnen dat je complexe systemen moet bouwen die als het ware een doolhof voor de aanvallers maken. Op zich is daar iets van aan, maar zorg er wel voor dat je niet zelf in je doolhof verdwaalt en daardoor fouten introduceert zonder het te beseffen. Soms is *less more* en dat geldt ook bij beveiliging.

Het vijfde fundamentele principe krijgt een eigen kadertje omdat deze voor discussie vatbaar is én geregeld voor de nodige controverse kan zorgen. Als je dus, om welke reden dan ook, niet genoeg tijd of budget hebt om alle principes toe te passen, probeer dan de volgende als laatste "oplossing" te gebruiken:

- **Obscurity**: ga niet aan de grote klok hangen hoe jouw verdediging is opgebouwd. Echter, let op met deze leuze. Obscurity wil niet zeggen dat je zelf een of ander vaag crypto-algoritme zelf gaat ontwikkelen en angstvallig gaat geheim houden. Gebruik standaarden en vertrouwde producten in je beveiliging. Eén van de eerste regels in security is "ga niet zelf het wiel heruitvinden"!

> Tip
>
> Soms zal je digitale stropers horen spreken over "Ik heb dat systeem gepwnd", uitgesproken als *ge-powned*. De term "to pwn" is *hacker-slang* voor "to own" om aan te geven dat je in een systeem bent binnen geraakt en nu controle over het systeem hebt. Volgens de urbandictionary.com is de enige reden dat de *o* een *p* werd het gevolg van een typfout, daar beide letters vlak naast elkaar staan op een toetsenbord.

2.7 Social Engineering

In het eerste hoofdstuk kwam de term "Social engineering" al enkele keren voor. We zagen ook bij de McCumber kubus dat we niet enkel op technologie mogen rekenen wanneer we onze verdediging opzetten, maar dat we ook een heel belangrijke schakel moeten trainen en opvolgen: de mensen. Mensen zijn meestal de zwakste schakel in ons securitymodel en dus daarom ook een interessante "aanvalsvector" voor digitale stropers.

Social engineering wordt ook wel eens *het hacken van mensen* genoemd en is een erg laagdrempelige, maar oh zo effectieve aanvalstechniek die vaak over het hoofd wordt gezien. Bij social engineering zal de aanvaller *menselijke* interacties gebruiken om zijn slachtoffers onbewust zaken te laten doen of informatie te geven dat niet zou mogen. Hierbij misbruikt de aanvaller onze aangeboren gewoonte om mensen te vertrouwen in plaats van te wantrouwen. We gaan meestal uit *van het goede van onze medemens* en zullen vaak meerdere goede redenen kunnen verzinnen waarom iemand iets van jou nodig heeft. Andere menselijke trekken die we kunnen gebruiken als social engineer zijn onder andere de nieuwsgierigheid van mensen, hun hebzucht, onwetendheid of angst.

Enkele voorbeelden:

- Je verkleden als pizzakoerier en een grote stapel (lege) pizzadozen het bedrijf binnendragen. Werknemers zullen je willen helpen en de deur voor je openhouden, ook al moet je normaal gezien met je toegangsbadge het gebouw betreden.
- Bij de rokers aan de achterkant van het gebouw gaan staan, wat met hen keuvelen en hen sigaretje aanbieden. Wanneer ze terug binnengaan hen volgen (*piggybacking*).
- Bellen en je voordoen als een technieker van Telenet en vervolgens de login-gegevens vragen van het slachtoffer.

Uiteraard hoeven social engineers zich niet te beperken tot *fysiek* contact. Ze hanteren ook erg vaak geschreven teksten (e-mail) om aan social engineering te doen. Twee veelvoorkomende vormen die we onderscheiden zijn:

- **Phishing**: trachten zoveel mogelijk mensen naar een bepaalde website of document te lokken waar vervolgens de aanvaller informatie van het slachtoffer zal proberen te verkrijgen door een fake login scherm te tonen, malware ongezien te installeren, etc.
- **Spear phishing**: deze vorm van phishing heeft als doel één persoon of groep. De e-mail zal dan ook opgesteld worden om specifiek voor het slachtoffer te werken, i.p.v. een generieke mail.

> Tip
>
> De term *phishing* is afgeleid van *fishing* oftewel vissen/hengelen naar iets. Vroeger had je het concept *phreaking* dat werd toegepast door hackers om op telefooncentrales in te breken door de 2600 hertz fluittonen die telefoons gebruiken te imiteren.

2.7.1 Meer dan phishing alleen

Phishing en spear phishing zijn slechts het topje van de social engineering ijsberg. Een goede cyberbos-wachter herkent de volgende technieken meteen aan hun naam, want je zult ze in het werkveld te pas en te onpas tegenkomen:

- **Vishing** (*voice phishing*): hetzelfde principe als phishing, maar dan via de telefoon. Vaak gecombineerd met een verzonnen scenario zoals *"uw bank heeft een verdachte transactie opgemerkt, kunt u even bevestigen?"*. In 2023 werd het hele **MGM Resorts** imperium in Las Vegas hiermee platgelegd: de aanvallers belden simpelweg de helpdesk en vroegen een wachtwoordreset.
- **Smishing**: phishing via SMS of WhatsApp. De klassieker is het *"Bpost-pakketje dat niet kan worden geleverd"* SMS-je met een link naar een fake betaalpagina.
- **Whaling**: een spear phishing aanval die specifiek gericht is op de "grote vissen" van een organisatie (CEO, CFO, andere C-levels). De buit is groter, maar deze profielen zijn meestal ook beter getraind én mediagevoeliger.
- **CEO-fraude** (of *Business Email Compromise*, BEC): een aanvaller doet zich voor als de CEO of een andere leidinggevende en vraagt aan een werknemer (vaak iemand uit de financiële afdeling) een dringende overschrijving uit te voeren. Eén van de duurste vormen van social engineering: het FBI rapporteerde dat BEC-aanvallen tussen 2013 en 2022 wereldwijd meer dan 50 miljard dollar schade veroorzaakten.
- **Pretexting**: het verzinnen van een geloofwaardig scenario (een *pretext*) om informatie of toegang te krijgen. Het Telenet-voorbeeld hierboven is in feite een pretexting-aanval.
- **Baiting**: het uitleggen van *lokaas*. Bijvoorbeeld een USB-stick met een verleidelijke label (*"salarissen 2025"*) op de parking achterlaten in de hoop dat een nieuwsgierige werknemer hem in z'n werkstation steekt.
- **Tailgating** (of *piggybacking*): meelopen met een geautoriseerd persoon om zo een beveiligde ruimte binnen te raken. Het rokers-voorbeeld hierboven is hiervan de schoolvoorbeeld-versie.
- **MFA fatigue** (of *prompt bombing*): wanneer een aanvaller reeds in het bezit is van een wachtwoord en het slachtoffer overspoelt met MFA-goedkeuringsverzoeken (vaak 's nachts), in de hoop dat die uit vermoeidheid of frustratie eens op *"Accepteren"* duwt. Werd onder andere door de groep **Lapsus$** ingezet om bij Microsoft, Uber en Rockstar Games binnen te raken (zie hoofdstuk 1).

2.7.2 Waarom werkt social engineering zo goed?

Social engineers zijn geen hackers in de traditionele zin, maar eerder *amateur-psychologen*. Ze maken handig gebruik van de zogenaamde **beïnvloedingsprincipes** die de Amerikaanse psycholoog Robert Cialdini in z'n boek *Influence* (1984) beschreef. Ken je deze principes, dan herken je ook meteen de hefbomen die de aanvaller hanteert:

- **Autoriteit**: mensen volgen instructies van iemand die gezag uitstraalt — de "CEO" die plots mailt, de "politie-agent" die belt, de "IT-helpdesk" die langskomt.
- **Urgentie**: een tijdsdruk creëren zorgt ervoor dat het slachtoffer niet meer kritisch nadenkt (*"uw account wordt binnen 1 uur gedeactiveerd"*).
- **Schaarste**: iets is bijna op of slechts beschikbaar voor weinigen (*"nog 3 plaatsen vrij"*).
- **Wederkerigheid**: als ik jou iets geef, voel jij je verplicht iets terug te doen. Een gratis pen, een sigaretje, een dienstje — en plots houd je de deur open voor de "pizzakoerier".
- **Sympathie**: we zeggen makkelijker ja tegen mensen die we sympathiek vinden of die op ons lijken.
- **Sociale bewijskracht**: *"iedereen doet het, dus zal het wel oké zijn"*.

> **Tip**
>
> Het boek **The Art of Deception** van Kevin Mitnick (een ex-cybercrimineel die door de FBI ooit "*America's Most Wanted Hacker*" werd genoemd) is een aanrader voor wie meer wil weten over de psychologie achter social engineering. Mitnick haalde in z'n hoogdagen de meest absurde stunts uit: hij wandelde gewoon zelfverzekerd door deuren van grote bedrijven alsof hij er werkte. Niemand stelde vragen.

2.7.3 Social engineering in het AI-tijdperk

Sinds de doorbraak van generatieve AI en deepfakes is social engineering nóg een trapje gevaarlijker geworden. Aanvallers gebruiken **deepfakes** (AI-gegenereerde audio of video van bestaande personen) om bijvoorbeeld een CFO te imiteren in een live videocall, of de stem van een familielid te klonen voor een nep-noodgeval-telefoontje. Een uitgebreidere bespreking met concrete voorbeelden (waaronder de 25 miljoen dollar-deepfake-CFO en de fake Joe Biden robocall) vind je in hoofdstuk 1 onder "*A.I. is here to stay*".

2.7.4 SET

Een veel gebruikte tool die social engineers hanteren is de **Social Engineering Toolkit** (beschikbaar via Kali en github) en zal je helpen om (spear) phishing attacks op te zetten, fake websites (door bestaande te clonen) te hosten, malware in afbeeldingen te injecteren, etc. Het kan op de koop toe geïntegreerd samenwerken met Metasploit waardoor stropers erg complexe aanvallen kunnen opzetten.

2.7.5 OSINT

We zagen reeds de vijf fasen die een aanvaller doorloopt: verkennen, scannen, etc. Aangezien Social engineering ook een vorm van cyber-aanval is, zullen ook hier deze fasen worden doorlopen. Zeker bij spear phishing wil de aanvaller zoveel mogelijk informatie over zijn slachtoffer(s) te weten komen om zo de meest doeltreffende mail op te stellen. OSINT, oftewel Open Source INTelligence, is de techniek waarbij een aanvaller open-source bronnen gebruikt om zijn slachtoffers in kaart te brengen tijdens deze eerste verkennings- of reconnaissance fase.

Enkele nuttige technieken die worden toegepast:

- Google en andere searchengines: besef dat je veel meer informatie kunt vinden indien je ook geavanceerde search-eigenschappen gebruikt (denk aan "site:" en de "+"-operator, etc. in Google).
- Reverse image searching: er zijn tegenwoordig erg krachtige tools om de oorsprong van een afbeelding te traceren, en met bijvoorbeeld Google Lens kan je zelfs objecten en locaties op een foto identificeren.
- Metadata van bestanden: media bestanden (.docx, .jpg, .png, .mp4, etc.) bevatten ook aardig wat onzichtbare informatie die soms onbedoeld meereist met het bestand (denk bijvoorbeeld aan de EXIF data in een afbeelding) en dus door kwaadwillige personen misbruikt kunnen worden.

Het **OSINT Framework** (via osintframework.com) is ontwikkeld om mensen er op te wijzen hoeveel publieke informatie social engineers (en anderen) over iemand te weten kunnen komen. Het is een griezelig uitgebreide hoeveelheid online bronnen die mooi gecategoriseerd zijn. Hierbij is het op de koop toe belangrijk te beseffen dat enkel open-source bronnen worden gebruikt. Digitale stropers zullen zich uiteraard niet beperken tot enkel open-source bronnen en ook vaak betaalde of illegaal verkregen bronnen raadplegen.

2.8 Soorten aanvallen

2.8.1 Software-based aanvallen

Software-gebaseerde aanvallen zijn de aanvallen die je in het nieuws geregeld hoort. In deze groep vinden we de virussen, wormen, backdoors, trojans en alle andere **malware** terug. De term malware dekt de lading erg goed: *"any kind of malicious software designed to damage or harm a computer system."* Het doel van malware is altijd geld, macht (denk aan blackmailen) of pestgedrag. De verschillende malware-types hebben soms overlappende eigenschappen en het is dus niet altijd mogelijk om een specifiek stuk malware onder één categorie te plaatsen. Wat ze allemaal gemeen hebben is dat ze bepaalde gekende of ongekende

(*zerodays*) fouten of bugs (*kwetsbaarheden* of *vulnerabilities*) misbruiken in software om zo toegang tot een systeem, data of stuk hardware te verkrijgen.

> Tip
>
> **CVE oftewel "Common Vulnerabilities and Exposures"** is een publiek beschikbare lijst waarin alle gekende security kwetsbaarheden opgelijst staan. Het laat cyberboswachters toe om duidelijk te communiceren over problemen en oplossingen. Volgens een rapport van de Amerikaanse "Cybersecurity & Infrastructure Security Agency" (CISA) was de meest misbruikte CVE tussen 2016 en 2019 een kwetsbaarheid (CVE-2017-11882) in Microsoft Office 2013 die samengevat: *"allow an attacker to run arbitrary code in the context of the current user by failing to properly handle objects in memory, aka 'Microsoft Office Memory Corruption Vulnerability'."*

We overlopen nu de belangrijkste vormen van malware.

2.8.1.1 Virus

De moeder van de malware. Hiermee is alles begonnen. De eerste virussen werden geschreven door scriptkiddies en mensen met te veel tijd. Ze hadden zelden een specifiek doel, en wilden gewoon zien wat de kracht van een virus was en hoe ambetant het anderen zou maken. De eerste virussen, in pre-Internet tijd, waren afhankelijk van overdracht via diskettes en raakten dus niet snel verspreid. Virusscanners waren nog onbestaande, maar als je een virus te pakken had dan was dat uiteraard even vloeken: sommige virussen infecteerden je bootsector, zodat je computer niet meer bootte, andere verwijderden bepaalde systeembestanden, enz. Ze werden geactiveerd door een uitvoerbaar bestand uit te voeren (zogenaamde .exe, .bat of .com bestanden) waar het virus zich onzichtbaar aan had vastgeklampt.

> Tip
>
> De auteur van dit boek heeft z'n vader een hoop extra grijze haren gekost door z'n game-verslaving. In de jaren 90 was de enige manier om aan games te geraken ofwel via de één of twee computerwinkels in de provincie, oftewel door diskettes van klasgenoten te kopiëren. Geregeld bevatte die gekopieerde games echter ook virussen, met alle gevolgen van dien. Moraal van het verhaal: *don't pirate, kids ;).*

> Opmerking
>
> **Mikko Hyppönen** (F-Secure) is de levende encyclopedie van malware-geschiedenis. Zijn TED-talk *Fighting viruses, defending the net* neemt je mee langs de eerste DOS-virusauteurs (die hij persoonlijk ging opsporen in Pakistan) tot de moderne state-sponsored malware. Hyppönen's wet: *"If it's smart, it's vulnerable."*
>
> 
>
> Figuur 2.12: Mikko Hyppönen (Bron: Wikimedia Commons, CC-BY-SA).

Heden ten dage komen we nog maar weinig "klassieke" virussen tegen. Je kan zeggen dat ze zijn geëvolueerd naar veel lastigere, soms letterlijke dodelijke varianten zoals wormen, ransomware, etc.

2.8.1.2 Wormen

Een virus is een beetje zoals een giftige vis op het droge: het ligt maar wat in het zand te spartelen en enkel als je zo dom bent om het ding op te rapen en in je aquarium met zeldzame vissen te plaatsen zal het schade kunnen toebrengen. Wormen daarentegen zijn de *sharknados* van de giftige vissen op het droge. Wormen zijn virussen die zichzelf kunnen voortplanten via één of meerdere communicatiekanalen. Uiteraard in de eerste plaats denken we dan aan het Internet as is, maar ook via e-mail, WhatsApp-berichten, Bluetooth, Facebook, etc.

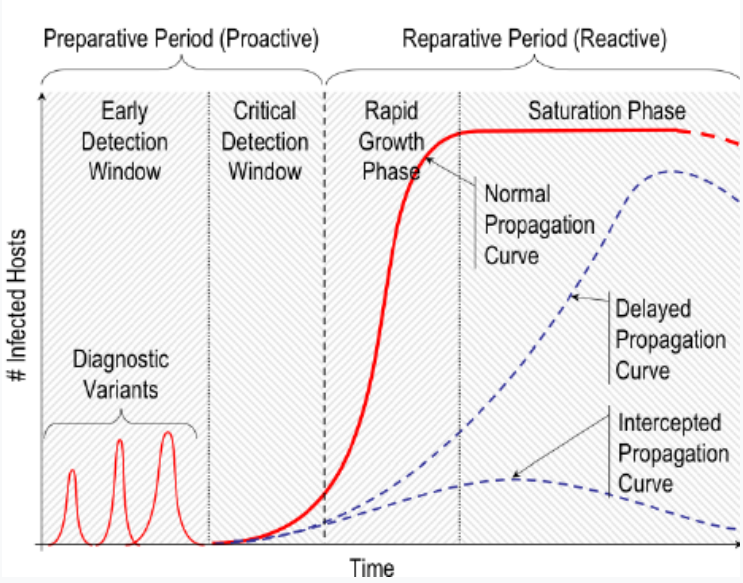
Als bijkomende handigheid zullen wormen ook vaak zichzelf veranderen zodat ze moeilijker door virusscanners kunnen gedetecteerd worden. Voorts hebben wormen geen *hostfile* nodig wat virussen wel hebben. En als laatste verschil met de virusjes is dat wormen zichzelf ook kunnen verspreiden zonder dat de gebruiker een (on)bewuste handeling moet doen. Kortom, wormen zijn een pittig probleem, vooral vanwege de snelheid (en eenvoud) waarmee ze zich verspreiden.

i Opmerking

Hoe reist een worm nu concreet van systeem A naar systeem B? Er zijn vijf klassieke *replication paths*:

Pad	Hoe het werkt
E-mail / IM	De worm mailt zichzelf als bijlage naar elk adres in je adresboek (of stuurt WhatsApp/Messenger-berichten). Beroemd voorbeeld: *ILOVEYOU* (2000).
File sharing	De worm kopieert zichzelf op een USB-stick, netwerkschijf of gedeelde map, en infecteert of vervangt bestanden.
Remote execution	De worm misbruikt een *vulnerability* en voert zichzelf rechtstreeks uit op het andere systeem. Typisch voor netwerk-wormen. *Conficker* en *WannaCry* werken zo.
Remote file access	De worm gebruikt een legitieme file-transfer dienst (SMB, FTP, NFS) om zichzelf naar het andere systeem te schrijven.
Remote login	De worm logt als gebruiker in op een ander systeem (bv. via SSH, Telnet, of default IoT-wachtwoorden à la *Mirai*) en start zichzelf daar.

De werkelijk gevaarlijke wormen combineren vaak **meerdere paden tegelijk** — als één route geblokkeerd is, blijven de andere nog actief.



Figuur 2.13: Worm propagation model. Bron: Network Eye: End-to-End Computer Security Visualization - Scientific Figure on ResearchGate.

Bovenstaande afbeelding toont het *worm propagation model*. Dit model toont hoe snel een worm zich kan verspreiden en hoe belangrijk het is om een nieuwe worm zo snel mogelijk te detecteren, in de hoop voldoende systemen vervolgens te beschermen tegen besmettingen ervan. Naarmate een worm meer systemen kan besmetten, die op hun beurt dan ook als uitvalsbasis van de worm dienen, merk je een exponentiële groei van besmette systemen. Deze groei bereikt een plateau wanneer quasi alle systemen besmet zijn die konden besmet worden.

2.8.1.3 Trojan

Trojans zijn malware die zich verstoppen binnenin een legale, al dan niet nuttige, applicatie die de gebruiker bewust installeert of downloadt. Van zodra de gebruiker deze host-applicatie installeert of start zal de onderliggende trojan zijn aanval op het systeem beginnen. Die aanval kan velerlei zijn, denk maar aan het installeren van een backdoor, de computer veranderen in een zombie (zie botnet verderop) of een keylogger activeren.

> **i** Opmerking
>
> De naam Trojan is gebaseerd op de legende van Het Paard van Troje uit de Aeneid van Vergilius.

> **⚠** Waarschuwing
>
> **Supply-chain trojans — de CIA en Xcode**: één van de meest verrassende onthullingen uit de Snowden-lekken (2015) was dat de CIA jarenlang gewerkt heeft aan een **eigen versie van Apple's Xcode**, de officiële IDE waarmee developers iOS- en macOS-apps bouwen. Met die gemanipuleerde Xcode-versie konden onwetende app-developers hun eigen apps bouwen — waarin ongemerkt **CIA-backdoors** werden mee geïnjecteerd. Eens die *trojan-apps* in de App Store of op iPhones stonden, hadden de Amerikaanse inlichtingendiensten toegang tot het toestel van de eindgebruiker. De CIA zou ook Apple's update-tool voor macOS gemanipuleerd hebben om zo stiekem keyloggers te installeren. Moraal: een trojan hoeft niet per se in *jouw* software te zitten — ze kan zich veel eerder in de *keten* hebben genesteld. Bron: The Intercept / Tweakers

2.8.1.4 Spyware

De naam spyware dekt de lading duidelijk: deze malware heeft als doel om informatie te verzamelen van het doelsysteem, zonder dat de gebruiker hiervan op de hoogte is. Deze informatie kan gaan van

eenvoudige gebruikersdata zoals usernames en wachtwoorden, maar gaat soms ook verder tot het in kaart brengen van surfgedrag of op welke manier de gebruiker een specifieke applicatie gebruikt. We kunnen stellen dat spyware twee categorieën heeft:

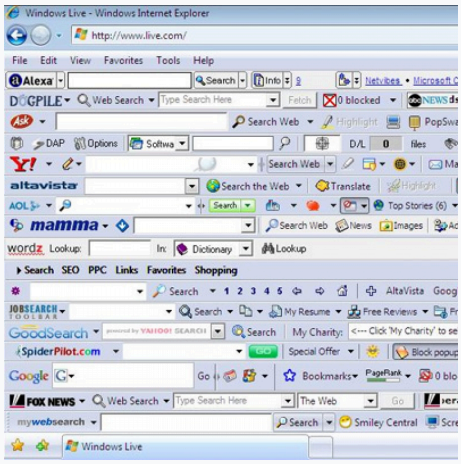
1. Reclamedoeleinden: hoe beter adverteerders de gebruiker kennen, hoe gerichter ze reclame kunnen maken en dus hoe groter de kans wordt dat die gebruiker uiteindelijk een geadverteerd product koopt.
2. Cybercriminelen: de meeste spyware die we heden ten dage tegenkomen verzamelen natuurlijk die dingen waar cyberstropers de meeste interesse in hebben: bankgegevens (logins, kredietkaartinformatie). Het hoofddoel van deze categorie is natuurlijk: centjes!

Spyware wordt meestal als trojan geïnstalleerd nadat de gebruiker een legitiem programma installeerde. Enkele bekende spyware-dragers waren de erg populaire (illegale downloader) Kazaa en Messenger Plus!, de plugin voor Windows Live Messenger.

2.8.1.5 Adware

Daar waar spyware soms door adverteerders wordt gebruikt om "het doelwit beter te leren kennen", heeft adware als doel om effectief reclame aan deze gebruiker, meestal ongewild te tonen. Ongewild is het sleutelwoord hier in. Adware is op zich niet noodzakelijk malware. Veel adware wordt gemaakt zodat de maker ervan extra inkomsten kan genereren om bijvoorbeeld het hoofddoel van de adware te blijven door ontwikkelen. Denk maar aan bepaalde gratis *musicplayers* die ook reclamebanners tonen terwijl je muziek afspeelt. Er is echter ook een categorie adware die **ongewild** reclame toont en soms zelfs op een zodanige manier dat een huis-tuin-en-keuken gebruiker zelfs niet beseft waar de reclame vandaan komt. Eind jaren 2010 had je veel browser-extensies die in je menu-balk als flitsende knoppen allerlei handige extra hulpmiddelen beloofden. Vaak zorgden deze extensies er echter ook voor dat je tal van extra reclame-pop-ups kreeg die in eerste instantie het gevolg waren van de website waar je op dat moment naartoe surfte.

> **Tip**
>
> Een bijkomend probleem van sommige malware is dat ze *rootkit*-achtige (zie hierna) technieken gebruiken waardoor ze erg moeilijk te verwijderen zijn en je moet opletten dat je niet essentiële bestanden van je computer verminkt of verwijderd.



Figuur 2.14: Een wat overdreven voorstelling van het soort browsers vol adware die de auteur soms te zien kreeg toen hij computers ging repareren die "nogal traag waren". Bron: https://www.directive.com/blog/disabling-those-pesky-browser-toolbars.html.

2.8.1.6 Rootkit

Een virus zal zich nestelen op "gebruikersniveau", terwijl een rootkit zich veel dieper in het besturingssysteem zal nestelen (op *kernel-niveau* of administrator-niveau). Hierdoor zijn rootkits veel onzichtbaarder, daar ze ook kunnen bepalen welke informatie de gebruiker van het besturingssysteem te zien krijgt. Een rootkit is eerder een techniek die kan gebruikt worden door de andere vormen van malware die

we hier beschrijven. De essentie van een rootkit is natuurlijk dat deze veel robuuster zijn en bijgevolg moeilijker te verwijderen zijn door het slachtoffer. Vaak zal een rootkit ook systeembestanden aanpassen (zie ook de infobox bij adware) waardoor je rootkits niet kunt verwijderen zonder permanente schade aan je besturingssysteem toe te brengen. De enige manier om een rootkit dan goed weg te krijgen is door je harde schijf te formatteren. Je OS opnieuw installeren/resetten, gebruik makend van de aanwezige installatiebestanden op de computer, is namelijk niet gegarandeerd dat dit voldoende zal zijn: mogelijk heeft de rootkit zich ook al in de installatiebestanden op de harde schijf geïnstalleerd!

2.8.1.7 Ransomware

De plaag van de laatste jaren! Het einddoel van cybercriminelen is natuurlijk centjes. Als je virus al je bestanden verwijderd dan heeft een digitale stroper weinig *leverage* om nog geld van z'n slachtoffer te pakken te krijgen. Ransomware lost dit probleem op voor de stropers: het zal de data letterlijk gijzelen en losgeld (*ransom*) vragen aan de gebruiker. Wanneer een ransomware op een systeem geraakt (via bijvoorbeeld een trojan of worm) zal het de data van de gebruiker versleutelen met een sleutel die enkel de maker van de malware kent. Vervolgens verschijnt er een bericht op de computer met daarin wat de gebruiker moet doen (betalen) indien deze z'n data terug wenst. Meestal zal de ransomware betalingen in crypto-coins vragen zodat het geld niet kan getraceerd worden.

Figuur 2.15: Voorbeeld van een typisch ransomware scherm dat het slachtoffer te zien krijgt (Bron: wikipedia).

Ransomware heeft aangetoond dat back-ups maken van je data erg belangrijk is. Maar ook HOE en WAAR je back-up'd zal invloed hebben op hoe ransomware-gevoelig je bent. Indien je back-ups maakt op hetzelfde systeem als waar de originele data staat, dan bestaat de kans dat de ransomware ook de back-up zal *gijzelen*. Indien je niet op geregelde tijdstippen een back-up neemt kan het zijn dat je dagen of weken aan data kwijt bent moest je het slachtoffer van een ransomware-aanval zijn. We gaan niet verder in op back-up strategieën, maar het mogelijk duidelijk zijn dat deze *skillset* een essentieel onderdeel vormt van het *security-beleid* van een bedrijf.

> Tip
>
> Herinner je dat de ransomware-aanvallen uit hoofdstuk 1 (WannaCry en Petya in 2017) ook meer impact hadden dan enkel "dataverlies": ziekenhuizen moesten patiënten de toegang ontzeggen, containerbedrijven zaten met duizenden tonnen aan vracht die niet verscheept geraakten.

⚠ Waarschuwing

Betalen = geen garantie: onderzoek van Hive Systems op 24.000 Amerikaanse bedrijven toont ontluisterende cijfers. Bij een ransomware-aanval:

Gedrag	Uitkomst	%
Betaald	Data gerecupereerd	38%
Betaald	Data tóch verloren	**19%**
Niet betaald	Data gerecupereerd (back-ups)	36%
Niet betaald	Data verloren	7%

Eén op de vijf bedrijven die het losgeld betaalde, kreeg dus **gewoon geen data terug**. De totale kost van ransomware-incidenten in de VS bedroeg in 2021 maar liefst **$7.9 miljard** aan downtime, waarvan $1.4 miljard effectief aan losgeld werd betaald. Moraal: betalen is russische roulette — én het financiert onbedoeld de volgende aanval. Zorg voor goede, *offline* back-ups.

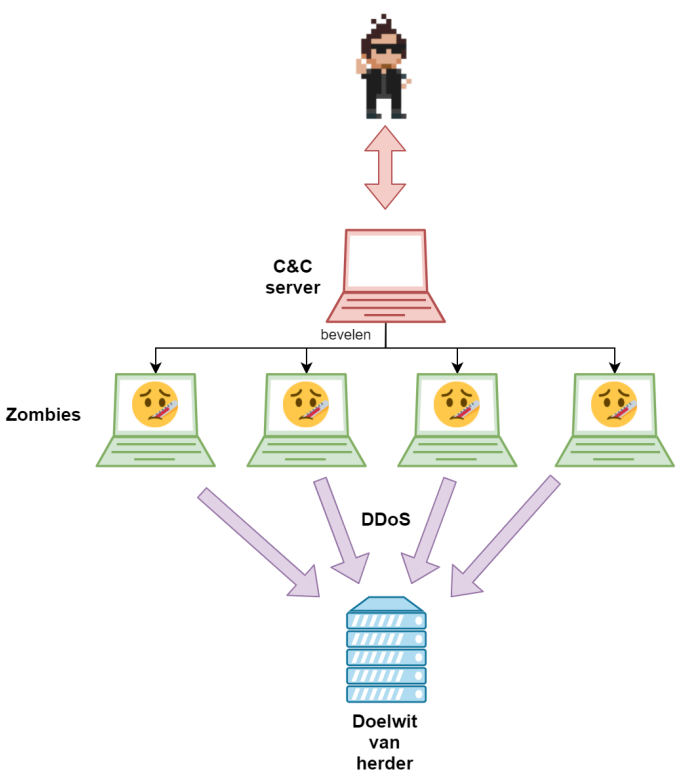
2.8.1.8 Botnet

Wat als een stroper toegang had tot een legioen computers? Duizenden computers die naar het bevel van de cybercrimineel luisteren en zonder morren doen wat hen gevraagd wordt? Welkom in de wondere wereld van botnets, zombies en herders.

Wanneer een cybercrimineel een grote groep computers nodig heeft, dan zal deze via voorgaande malware-technieken proberen zoveel mogelijk computers te besmetten met zijn *zombie-virus*. Dit virus zal twee dingen doen:

1. Het zal zich onzichtbaar nestelen op de computer en ervoor zorgen dat het virus ook na heropstart actief is (*maintain access*).
2. Het zal een backdoor creëren en terugbellen naar de *command-en control-server* (**C&C server**) van de originele virusmaker, de zogenaamde *herder*.

Via de C&C-server kan de botnet-herder nu alle zombies benaderen en bevelen geven. Het kan de zombies bijvoorbeeld vertellen dat ze allemaal tegelijkertijd naar een bepaalde website moeten pingen, waardoor een gigantische DDOS (*Distributed DoS*)-aanval plaatsvindt. Of het zou kunnen bevelen dat alle zombies vijf sterren moeten geven aan een specifieke applicatie in de app-store. Of wat te denken van duizenden zombies die permanent als cryptominers naar bitcoins delven voor de herder?



Figuur 2.16: Een echte botnet zal veel meer dan vier zombies bevatten. Maar we moeten ergens mee beginnen, nietwaar.

Hoe groter het botnet, hoe krachtiger en machtiger de herder is. Botnets kunnen zoveel impact hebben op systemen dat er een hele ecologie rond illegale botnets is verschenen. Zo zijn er websites waar herders de diensten van hun botnets verhuren aan anderen (*botnets-as-a-service*) of gewoon geregeld hun nieuwst verzamelde botnet verkopen aan de hoogste bieder op het darkweb.

Het is in het voordeel van de herder dat botnets zo onzichtbaar mogelijk blijven. Daarom dat de meeste botnet-software heel subtiel op de achtergrond werkt. Veel computers maken maanden, soms jaren, deel uit van een botnet zonder dat ze dat ooit hebben beseft.

Omdat botnets zo'n grote impact kunnen hebben, jagen Microsoft, Cisco, McAfee, etc. actief op deze zaken. Een botnet uitschakelen door de zombies te bestrijden is natuurlijk onbegonnen werk. De oplossing ligt natuurlijk bij de C&C-servers! Als je die server uit de lucht krijgt dan zijn de zombies nutteloos en heb je letterlijk het botnet onthoofd.

> **i** Opmerking
>
> **Mirai (2016): het IoT-botnet**: in september 2016 dook Mirai op, een botnet dat niet op gewone computers jaagde maar op **IoT-apparaten**: routers, IP-camera's, digitale videorecorders, babyfoons. De verspreidings-*trick* was verbluffend simpel: Mirai probeerde gewoon een lijst van **64 default logins** (admin/admin, root/root, user/user, ...) op elk IoT-toestel dat aan het internet hing. Resultaat: op het hoogtepunt ruim **600.000 besmette toestellen**. Op 21 oktober 2016 voerde Mirai een DDoS-aanval uit op DNS-provider Dyn, waardoor grote delen van het Amerikaanse internet — Twitter, Netflix, Reddit, GitHub, Spotify — urenlang onbereikbaar waren. Mirai's broncode werd later publiek gemaakt, waarna tal van varianten ontstonden die tot vandaag actief zijn.

> **i** Opmerking
>
> **Mantis (2022): meest krachtige botnet tot nu toe**: waar Mirai inzette op **volume** (veel, maar zwakke IoT-apparaten), gooit Mantis het over een andere boeg: **kwaliteit**. Dit botnet rekruteert gekaapte virtuele machines en krachtige servers. Gevolg: elke bot heeft enorm veel rekenkracht. Cloudflare registreerde in juni 2022 een DDoS-aanval van **26 miljoen HTTPS-requests per seconde** — een record. Les: de evolutie van botnets gaat niet noodzakelijk richting *meer zombies*, maar richting *sterkere zombies*. Bron: Cloudflare blog

> ⚠ Waarschuwing
>
> **BYOB — "Build Your Own Botnet"**: op GitHub staat al jaren een open-source *educatief* framework (github.com/malwaredllc/byob) waarmee je in enkele klikken je eigen botnet kan opzetten: command & control-server met webinterface, payload-generator voor Windows/Linux/macOS, en een dozijn kant-en-klare post-exploitation modules (keylogger, webcam, persistentie, ...). Officieel voor onderzoekers en studenten, maar een mooi illustratie van hoe **laagdrempelig** malware-ontwikkeling is geworden. De scheve grafiek uit eerder in dit hoofdstuk in actie.

> ⚠ Waarschuwing
>
> **Een echte DDoS-afpersingsbrief**: cybercriminelen sturen soms *prijslijsten* die niet zouden misstaan bij een legaal bedrijf. Een uitgelekt voorbeeld:
>
> > *"Hello. If you want to continue having your site operational, you must pay us 10 000 rubles monthly. Attention! Starting as of [DATE] your site will be a subject to a DDoS attack. The first attack will involve 2,000 bots. If you contact the companies involved in the protection of DDoS-attacks and they begin to block our bots, we will increase the number of bots to 50,000.*
> > ***You will also receive several bonuses:** 1. 30% discount if you request DDoS attack on your competitors/enemies. 2. If we turn to your competitors/enemies to make an attack on your site, then we deny them."*
>
> Met andere woorden: koop niet alleen je eigen *"bescherming"*, maar gebruik ze ook als aanvalswapen tegen je concurrenten. Het *RaaS*-model (*Ransomware/DDoS-as-a-Service*) in zijn meest ontluisterende vorm.

2.8.2 Netwerk-based aanvallen

Een groot deel van de aanvallen gebeurt uiteraard via een bedraad of draadloos netwerk. We spenderen een volledig apart hoofdstuk aan draadloze aanvallen verderop in dit handboek. De meer *klassieke* netwerk-gebaseerde aanvallen komen niet in dit handboek voor. Er zijn ongelooflijk veel mogelijkheden op netwerk/communicatie-niveau om als cybercriminal toegang tot systemen te verkrijgen. Ieder bekend protocol (DNS, IP, TCP, CSMA/CD, SNMP, etc.) heeft ontelbare, gekende, bugs. Nog steeds worden er nieuwe technieken ontwikkeld, zelfs bij protocollen die al 20 tot 30 jaar bestaan (om je een idee te geven: het IP-protocol werd in 1977 geschreven).

2.8.3 Hardware-based aanvallen

In den ouden tijd leken aanvallen vooral een software gebeuren. Enkel in de duurdere Hollywood-films werden er ingewikkelde hardware-apparaten gebruikt door stropers om toegang tot streng beveiligde systemen te verkrijgen. Tegenwoordig kan je voor enkele dollars ongelooflijk krachtige Raspberry Pi's, Arduino, etc. kopen, waardoor hardwaregebaseerde aanvallen steeds vaker voorkomen (je zou kunnen spreken van een democratisering van *hacking hardware*).

Dit hoofdstuk zou wederom een heel eigen boek kunnen bevatten, we gaan daarom enkele van de meest voorkomende of interessantste zaken kort toelichten. Net zoals met malware zal je merken dat er soms overlap is tussen verschillende type aanvallen en het dus zeker geen zwartwit classificatie is:

- **USB sticks**: deze kleine dingen kosten nog geen euro als je ze in bulk koopt. Ideaal dus om ze als aanvaller te vullen met malware die zichzelf automatisch installeert wanneer een slachtoffer de stick goedbedoeld in z'n computer steekt. Of wat te denken van een *USB of death* die 220 volt door je computer jaagt als je hem insteekt, gegarandeerd dat je daarmee een DoS aanval kunt uitvoeren want de stick zal het moederbord, de harde schijven etc. vernietigen door de hoge stroomstoot.
- **BIOS rootkits**: in moderne computers zitten BIOS chips die voorkomen dat je zomaar eender welke software kan inladen bij het opstarten - met dank aan UEFI dat onze systemen beveiligt met systemen zoals Trusted Platform Modules (TPM), Secure Boot, etc. Maar wat als je er in slaagt om die chip te hacken? Recent dook *Moonbounce* op, UEFI malware die aanvallers als een springplank kunnen gebruiken om vervolgens andere malware op het systeem te krijgen. De sterkte van Moonbounce? Het kan zichzelf in de chip installeren en blijft permanent aanwezig zodat je besturingssysteem verwijderen of harde schijf formatteren geen zin heeft.
- **Raspberry Pi, Arduino, Malduino *and friends***: computers die niet groter dan een dikke duim zijn. De mogelijkheden zijn natuurlijk immens wat je kan doen als je dit soort dingen onzichtbaar kan inplanten in een bedrijf. Doordat dit soort dingen zo goedkoop zijn geworden zien we ook vaker concepten zoals **warshipping** opduiken: een aanvaller verstuurt zijn geautomatiseerde Raspberry Pi naar z'n slachtoffer door het toestel in een gewoon postpakket te verstoppen in bijvoorbeeld een dubbele bodem. Van zodra het pakket aankomt, zal het toestel automatisch proberen verbinding te maken met het aanwezig netwerk en *terugbellen* naar de aanvallers. Zolang het slachtoffer de Raspberry Pi niet detecteert kan de stroper vanuit de veilige haven van z'n huis aanvallen uitvoeren.
- **Keyloggers en *juice hacking***: keyloggers bestaan zowel in software als hardware, maar hun doel is natuurlijk hetzelfde: de toetsaanslagen van het slachtoffer detecteren om er dan bijvoorbeeld wachtwoorden en gebruikersnamen uit te filteren. Hardware keyloggers worden door stropers vaak aan een systeem gehangen waar ze niet permanent toegang tot hebben. Enkele dagen of weken later moet de stroper dan enkel de keylogger ophalen. Gerelateerd hieraan is juice hacking, een fenomeen dat hier en daar opduikt. In publieke plaatsen zijn er meer en meer publieke oplaadpunten waar gebruikers hun digitale toestellen via een USB-draad kunnen opladen. Maar hoe zeker ben je eigenlijk dat die draad die in de muur verdwijnt niets meer is dan een *oplaadkabel*? Wat als aan de andere kant van de muur de draad aan het toestel van de stroper hangt?
- **USB Ninja, RFID cloners en consoorten op lab401.com en hak5.org**: er zijn vele websites waar je tegenwoordig gespecialiseerde *hacking hardware* kunt kopen. Wat te denken van de USB Ninja? Een *gewone* USB-kabel die echter een ingebouwde verzender heeft en automatisch alle data via een RF-verbinding naar de stroper verderop verstuurt. RFID cloners vind je ook voor een habbekrats en laten je toe om de alomgebruikte RFID toegangskaarten te clonen, zodat de stroper beveiligde gebouwen kan binnenstappen zonder een alarm te laten afgaan.

*Warshipping warships**: In april 2026 werd een treffend voorbeeld van warshipping publiek gemaakt: een Bluetooth-tracker van amper €5 werd verstopt in een briefkaart en per post verstuurd naar een Nederlands marineschip ter waarde van €585 miljoen. De tracker zond 24 uur lang de locatie van het oorlogsschip uit, zonder dat iemand het doorhad. Dit incident toont aan hoe goedkope consumententechnologie een ernstig risico kan vormen voor zelfs de best beveiligde militaire assets. Bron: Tom's Hardware, 18 april 2026

> **i** Opmerking
>
> **De Russische USB's van Kabul (2008)**: één van de meest beruchte USB-aanvallen aller tijden, beschreven in Fred Kaplan's boek *"Dark Territory"*. Russische spionnen slaagden erin een **airgapped** (dus volledig van het internet afgesneden) geclassificeerd Amerikaans militair netwerk binnen te dringen. Hun methode? Gewoon **goedkope USB-sticks volgestopt met malware** leveren aan retailkiosken vlak bij het NATO-hoofdkwartier in Kaboel. Pure wiskundige gok: vroeg of laat koopt een Amerikaanse militair er één en steekt die in een veilige computer. Dat is exact wat er gebeurde. Tienduizenden bestanden — hardware-designs, troepenconfiguraties, kaarten van bases — werden buitgemaakt. De operatie wordt beschouwd als de eerste grootschalige staats-doordringing van een gerubriceerd Amerikaans netwerk en was de directe aanleiding voor een Pentagon-verbod op USB-sticks in gevoelige omgevingen.

Tip

Verdedigen tegen juice jacking: als je in een luchthaven, trein of hotel op een publieke USB-poort je telefoon oplaadt, hoe weet je dat de kabel niet aan een stropers-toestel hangt? Drie simpele regels:

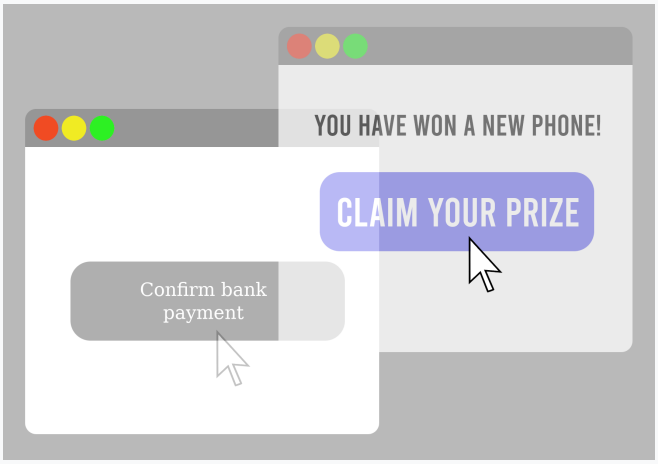
- Gebruik een **eigen oplader** in een gewoon stopcontact (niet die publieke USB-poort).
- Neem je **eigen power bank** mee.
- Koop een **charge-only cable** of een zogenaamde **USB data blocker** (kostprijs: enkele euro). Dit is een adapter die de datapinnen van de USB-connector fysiek afsluit — enkel de stroompinnen werken nog, dus data-overdracht is onmogelijk.
- Schakel op je telefoon de optie *"data transfer bij aansluiting"* uit, zodat een toestel bij default enkel oplaadt.

2.8.4 Mobiele aanvallen

Onze smartphones zijn ondertussen onze *primaire* digitale apparaten geworden: permanent verbonden, bomvol persoonlijke data, met camera, microfoon, GPS en toegang tot onze bank- en sociale media-accounts. Geen wonder dat ze een uitgelezen doelwit zijn voor stropers. Enkele mobile-specifieke aanvalstechnieken:

Waarschuwing

Clickjacking op Android (*overlay attacks*): een kwaadaardige app toont bovenop legitieme schermen een **onzichtbare overlay**. Je denkt dat je op *"Niet nu"* klikt in een dialoogje, maar in werkelijkheid klik je op *"Installeer deze app"* of *"Geef toegang tot je contacten"* onder die overlay. De Android *Accessibility Service* (oorspronkelijk bedoeld als hulp voor mensen met een beperking) maakt dit mogelijk: eens een app die rechten krijgt, kan ze andere apps besturen, velden invullen en knoppen indrukken — zonder dat de gebruiker het ziet. Gelukkig heeft Google sinds Android 12+ extra beperkingen doorgevoerd.



Figuur 2.17: Clickjacking: het slachtoffer denkt op *"Confirm bank payment"* te klikken, maar raakt in werkelijkheid de verborgen *"Claim your prize"*-knop (Bron: Wikimedia Commons).

⚠ Waarschuwing

Android/PowerOffHijack (2015): malware die de *shutdown*-procedure van je toestel overneemt. Als je je telefoon *uitzet*, **doet hij alleen alsof**: je krijgt het animatie-scherm en het toestel lijkt dood, maar in werkelijkheid blijft hij volledig functioneel. De malware kan ondertussen gewoon telefoneren, foto's maken met de camera en data versturen — terwijl jij denkt dat je toestel *offline* is. Ongeveer 10.000 toestellen waren besmet voor de aanval werd ontdekt, voornamelijk via Chinese alternatieve app-stores. Moraal: haal je **batterij eruit** (als dat nog kan) wanneer je zeker wilt weten dat je toestel echt uit is.

⚠ Waarschuwing

Xenomorph (2022–2023): een moderne Android banking-trojan: verkleed als een onschuldige *"performance booster"* of *"pdf reader"* op de Play Store. Eens geïnstalleerd toont Xenomorph een nep-inlogscherm telkens je een bankapp opent, en steelt zo je credentials. De moderne versie viseert **meer dan 400 banken en cryptowallets wereldwijd**, inclusief Belgische en Nederlandse banken. Xenomorph gebruikt ook een **Automated Transfer System (ATS)**: automatisch geld overboeken zonder dat de gebruiker zelfs maar iets moet invullen. Bron: BleepingComputer

Tip

De enige écht onhackbare smartphone blijft de Nokia 3310. Maar dat geeft je dan natuurlijk ook geen WhatsApp, Google Maps of Mobile Banking — alles heeft z'n prijs.

2.8.5 Side-channel aanvallen

De persoonlijke favoriet van de auteur vanwege de inventieve aanvallen die onder deze categorie bestaan. Het idee van een side-channel aanval bestaat er uit dat je informatie te pakken krijgt uit een protocol of hardware op onverwachte manier. Een vergelijking in het echte leven zou het volgende kunnen zijn: je wil inbreken bij een bank verderop in de straat. Dit gaat echter enkel wanneer de bewaker slaapt. Na observatie heb je ontdekt dat de bewaker voor het slapengaan altijd een boek leest en z'n progressie ervan op GoodReads deelt. Je hebt geen toegang tot de slaapkamer van de bewaker, maar je volgt hem wel op GoodReads. Van zodra de bewaker een update over z'n voortgang post weet je dat het tijd is. Dit is een voorbeeld van een side-channel aanval.

Ieder stuk hardware, software of protocol is vatbaar voor side-channel aanvallen. Het is onmogelijk om 100% hiertegen beschermd te zijn. Uiteraard zijn niet alle aanvallen even effectief als de andere, en alles hangt dus af van wat de stroper juist nodig heeft.

Enkele voorbeelden:

- Militairen deelden hun workouts op Strava. Ze waren echter vergeten dat ze in een geheime basis in Afrika werkten. Plots zagen mensen workouts *in the middle of nowhere* gedeeld worden, wat deed vermoeden dat er op die plaats meer was dan enkel een lege woestijn. Oeps.
- Onderzoekers zijn er in geslaagd om data uit een computer te krijgen door de *power consumption* op te meten.
- Rowhammer, spectre (en ook drammer en rampage) zijn recente technieken waarbij de informatie uit het geheugen kan gelezen worden waar de stroper eigenlijk geen rechten toe heeft.
- Heartbleed: deze bespraken we al kort in het eerste hoofdstuk. Deze inventieve techniek maakt gebruik van het feit dat computers altijd braaf antwoorden als je ze vragen stelt. Maar stropers ontdekten dat de snelheid van antwoorden afhing van de vraag.

i Opmerking

Wist je dat een oud liedje van Janet Jackson kan gebruikt worden om (oude) laptops te doen crashen door het gewoon af te spelen? Over een side-channel DoS aanval gesproken. De aanval heeft zelfs een CVE-nummer toegewezen gekregen! Lees hier hoe de aanval werkt:https://www.bleepingcomputer.com/news/security/janet-jacksons-music-video-is-now-a-vulnerability-for-crashing-hard-disks/

⚠ Waarschuwing

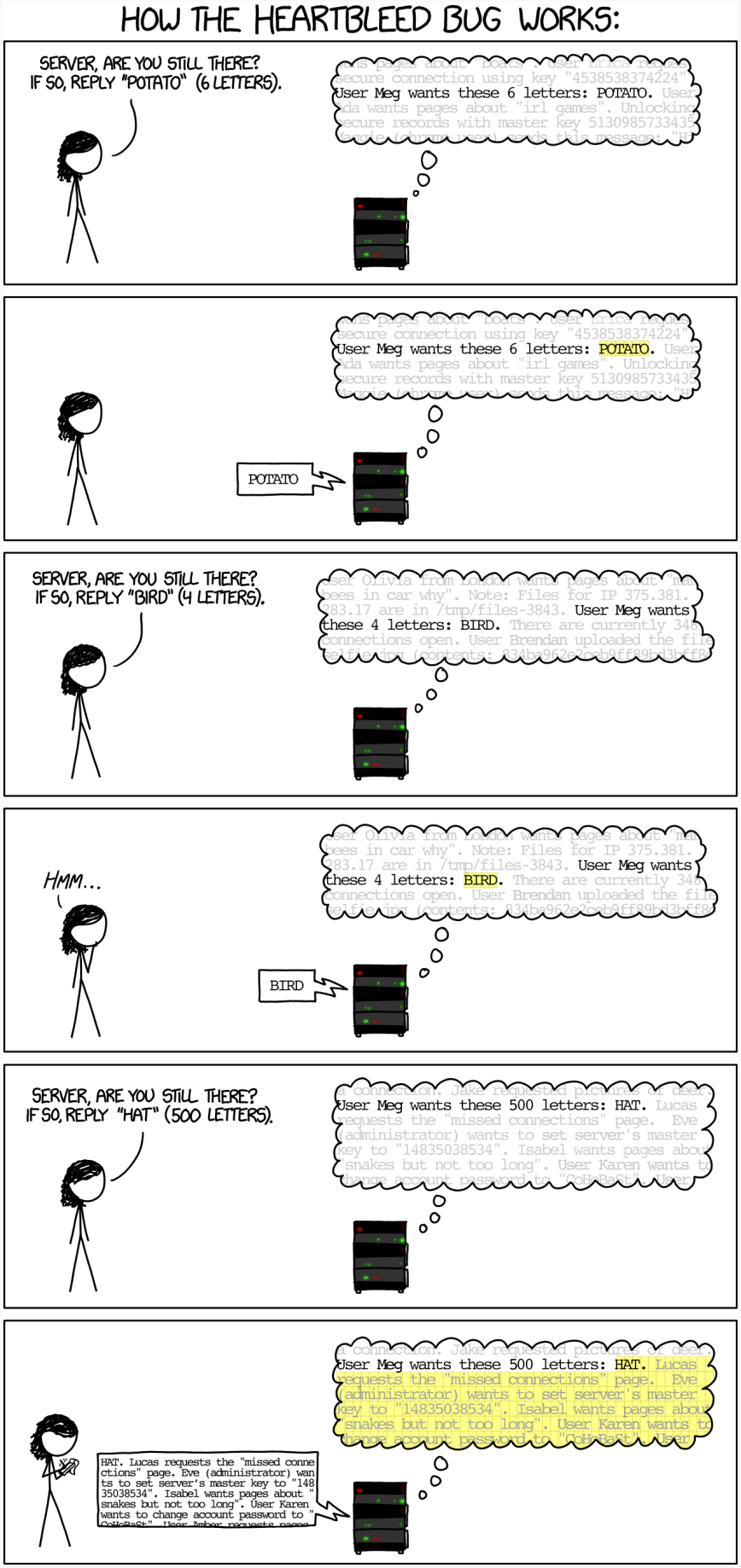
KNOB Attack (2019): Bluetooth op 1 byte entropie: de *Key Negotiation of Bluetooth*-aanval misbruikt een zwakheid in de manier waarop twee Bluetooth-toestellen de sterkte van hun gedeelde versleutelingssleutel afspreken. Normaal onderhandelen ze over een sleutel van **16 bytes** (128 bit). Maar een Man-in-the-Middle aanvaller kan die onderhandeling onderscheppen en de beide toestellen laten *"akkoord gaan"* met een sleutel van slechts **1 byte** — 8 bit. Die sleutel kan dan in minder dan een seconde worden gebruteforced, waarna de stroper de volledige Bluetooth-communicatie in plaintext kan lezen. Bijna elk Bluetooth-toestel van vóór 2019 was kwetsbaar: telefoons, koptelefoons, toetsenborden, auto's. Meer info: knobattack.com

⚠ Waarschuwing

Whisper Leak (Microsoft, nov 2025): LLM-gesprekken afluisteren — zelfs al zijn ze versleuteld: een recent ontdekte side-channel in taalmodellen (ChatGPT, Claude, Gemini, ...). Wanneer een LLM een antwoord *streamt* (woord per woord), zijn de **grootte** en **timing** van elk pakketje over het netwerk nog zichtbaar — ook al is de inhoud geëncrypteerd met TLS. Microsoft toonde aan dat een machine learning-classifier op basis van enkel die metadata met **92% zekerheid** kan voorspellen of een gebruiker een *"gevoelig onderwerp"* (zoals *money laundering*) aan het bespreken is. Een passieve aanvaller op je ISP, op hetzelfde wifi-netwerk of op je router kan dus zien waar je met ChatGPT over praat, zelfs zonder de encryptie te breken. Bron: The Hacker News

i Opmerking

Bitsquatting — DNS-hijacking zonder één bug te misbruiken: RAM-chips zijn niet perfect. Heel af en toe flipt een bit spontaan van 0 naar 1 of omgekeerd (door kosmische straling, hitte, ...). Onderzoeker Artem Dinaburg besefte in 2011 dat je dit kunt uitbuiten. Stel: de domeinnaam `microsoft.com` staat in RAM als een reeks bytes. Als één bit flipt, kan dat `micrnsoft.com` of `licrosoft.com` worden. Door die *look-alike* domeinen alvast te registreren, krijg je gewoon organisch verkeer van gebruikers wiens RAM een foutje maakte — zonder dat je iets hoeft te hacken. In Dinaburg's experiment kreeg hij honderdduizenden onterechte DNS-queries per dag. Pure passieve side-channel aanval. Bron: dinaburg.org



Figuur 2.18: De briljante xkcd.com strip legt perfect uit hoe Heartbleed werkt.

2.9 Samenvatting

In dit hoofdstuk legden we de fundamenten vast waarop de rest van dit boek verder bouwt:

- **CIA** (confidentiality, integrity, availability) is het doel; de **McCumber kubus** dwingt je om dat doel te bewaken op élke datastatus (opslag, transport, verwerking) én op élke laag (technologie, beleid, mensen).
- De **wapenwedloop is scheef**: aanvallers hebben snelheid, schaal en eenvoudig te gebruiken tools, terwijl verdedigers aan álles moeten denken.
- **Zero days** en het **window of vulnerability** maken patchen tot een permanente discipline — *Patch Tuesday* impliceert *Exploit Wednesday*.
- De **aanvallers** zijn divers: van scriptkiddies over eigen werknemers tot cybercriminelen en door overheden gesponsorde actoren. Ieder profiel vraagt een eigen verdediging.
- Aanvallen zijn **passief** (sniffen, OSINT) of **actief** (modify, replay, DoS); de meeste starten met **social engineering**.
- Malware-families (virus, worm, trojan, rootkit, ransomware, botnet) én **side-channels** (Rowhammer, KNOB, Whisper Leak) tonen dat elke laag — software, hardware én menselijk gedrag — een aanvalsoppervlak is.

Onthoud vooral: **de zwakste schakel is zelden technisch**.

3. Cryptografie

i Leerdoelen

Na dit hoofdstuk kan je:

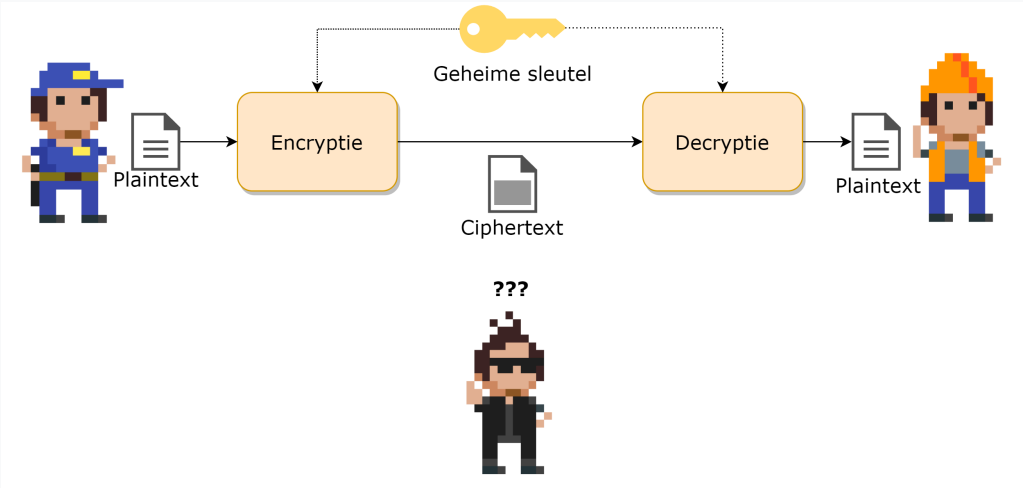
1. Het **principe van Kerckhoffs** uitleggen en motiveren waarom publiek getest crypto veiliger is dan *security through obscurity*.
2. Het verschil tussen **symmetrische en asymmetrische encryptie** duiden (snelheid ↔ sleutel-distributie) en aangeven wanneer je welke inzet.
3. Voor een gegeven **blockcipher-mode** (ECB, CBC, CTR) uitleggen waarom ECB onveilig is en de rol van **IV/nonce** beschrijven.
4. **Hashing, digitale handtekeningen en certificaten** correct situeren binnen integrity, authenticiteit en non-repudiation.
5. De werking van **HTTPS/TLS** beschrijven als samenspel van asymmetrische + symmetrische crypto, hashing en PKI — en de impact van een MITM-aanval (mitmproxy) inschatten.

In dit hoofdstuk duiken we de boeiende wereld van de cryptografie in: het versleutelen van informatie zodat enkel zender en ontvanger het bericht kunnen lezen.

3.1 Encryptie en decryptie

Herinner je dat we in het vorige hoofdstuk de termen CIA aanhaalden en de McCumber kubus? Een grote pijler van CIA, confidentiality, wordt opgelost met behulp van encryptie, namelijk het versleutelen van onze data met behulp van een geheime sleutel. Door deze te versleutelen wordt deze onleesbaar voor personen die de geheime sleutel niet hebben (en bijgevolg niet geautoriseerd zijn om de data te mogen lezen). De moeilijkheid van een goed cryptografisch systeem is dat de data op een zodanige manier moet versleuteld worden dat het quasi onmogelijk is om zonder sleutel de originele data terug te vinden. We spreken hierbij over de originele data als de **plaintext** en de geëncrypteerde data als **ciphertext**. De ontvanger van een ciphertext moet deze, als hij de juiste sleutel heeft, terug kunnen omzetten naar de originele plaintext.

Er zijn al veel encryptie algoritmes de revue gepasseerd doorheen de geschiedenis van de mens. Al van in de tijd van de Romeinen werd er aan cryptografie gedaan. Mensen hebben altijd gevoelige data gehad waar vertrouwelijk mee moest om gesprongen worden. Naarmate de **cryptanalyse** (dat is het proberen ontcijferen van een ciphertext zonder dat je de geheime sleutel hebt) evolueerde, moesten ook de cryptografische algoritmes verbeteren. Ook hier zien we weer diezelfde wedloop tussen digitale stropers en cyberboswachters. Hoe sterker onze computers worden (met dank aan de wet van Moore), hoe krachtiger onze algoritmes moeten worden. De eenvoudigste vorm van cryptanalyse, *bruteforcing*, is rechtstreeks afhankelijk van de snelheid van de computer. Hoe meer sleutels per seconde een computer kan testen, hoe sneller de originele sleutel kan gevonden worden.



Figuur 3.1: Alice gebruikt encryptie om een beveiligd bericht naar Bob te sturen, zodat Eve deze niet kan lezen.

> **Tip**
>
> De volledige geschiedenis van de cryptografie hier vertellen zou ongeveer 1200 pagina's vereisen. Het briljante boek "The Codebreakers" van David Kahn is een aanrader voor eenieder die meer willen weten over deze boeiende geschiedenis. Laat de 1200 pagina's je niet afschrikken, het boek leest als een echte thriller.

Alle bestaande cryptografische systemen kunnen op verschillende manieren gekarakteriseerd worden (bron Network Security Essentials, door William Stallings):

- De *acties* die op de data worden uitgevoerd om deze te encrypteren:
  - **Substitutie**: een teken door een ander teken vervangen.
  - **Transpositie**: een teken op een andere plek in de tekst zetten.
  - **Product**: een combinatie van meerdere substituties en transposities.
- Het *aantal sleutels* dat nodig zijn:
  - **1 sleutel**, ook wel "private encryption" genoemd.
  - **2 sleutels**, ook wel "public encryption" genoemd.
- De *manier* waarop de data wordt verwerkt:
  - Als een **blok** data, blok per blok.
  - Als een **stream**, teken per teken.

3.2 Kerckhoffs principe

We gaan zo meteen bekijken hoe encryptie effectief gebeurt, maar we willen al even een mythe onderuit halen. Binnen encryptie heb je de *principes van Kerckhoff*, zes regels die in de 19e eeuw werden vastgelegd waaraan encryptie-algoritmes moeten voldoen om als veilig beschouwd te worden. Sommige principes zijn, onder ander door onze steeds krachtigere computers, niet meer relevant, maar één principe blijft ongelooflijk belangrijk: *"Het kennen van het gebruikte encryptie-algoritme door de tegenstander is geen probleem, het is enkel de geheime sleutel die ten allen tijde uit de handen van de tegenstander moet blijven."*

Dit ogenschijnlijk eenvoudige zinnetje betekent heel veel: **je sleutel (of wachtwoord) is wat je het beste moet beschermen. Zonder kennis van de sleutel zou iemand nooit toegang mogen krijgen tot versleutelde data, ongeacht dat geweten is met welk systeem de data werd vercijferd.**

We zagen reeds dat *Security through obscurity* een dubbel snijdend zwaard binnen de security wereld is. Enerzijds is het niet aangeraden om met veel tamtam aan te kondigen hoe jij je data beveiligt. Anderzijds geeft het je mogelijk een vals gevoel van veiligheid (*snake's oil*) daar het geheimhouden van je algoritme en systemen geen garantie is dat deze ook effectief veilig zijn. In de 21e eeuw zijn de meest gebruikte encryptie-algoritmes publiek gekende algoritmes die door duizenden experts aan de tand zijn gevoeld. De

kans dat er dus (bewuste) fouten in dergelijke standaarden zitten, is véél kleiner dan wanneer je met een zogenaamd proprietary systeem werkt waarvan de werking angstvallig geheim wordt gehouden.

Finaal draait alles op het geheimhouden van je sleutel, iets dat we telkens weer in dit boek zullen herhalen!

3.2.1 Opletten met reclame

Let op met encryptiesystemen die zichzelf verkopen met zinnen zoals *"10 jaar nodig op een gewone laptop om alle sleutels te testen"*. Dit zou kunnen doen vermoeden dat je dus voor minstens tien jaar goed zit (we gaan er even vanuit dat de gemiddelde cryptanalist maar toegang heeft tot één laptop, wat uiteraard in de echte wereld niet zo is). De gemiddelde tijd van voorgaande systeem om te bruteforcen is echter vijf jaar, de helft.

Stel dat je een sleutel hebt die bestaat uit 8 karakters. Een karakter is een letter van a tot z (geen onderscheid tussen hoofd- en kleine letters en geen getallen of speciale tekens). Er zijn 26^8 mogelijke sleutels (we gaan ervan uit dat de sleutel exact 8 karakters moet bevatten). Een computer kan één miljoen sleutels per seconde testen. De duur om alle mogelijke sleutels te testen is dus $\frac{(26^8)}{1000000}$, oftewel 208 827 seconden, pakweg 58 uur. Intuïtief zou je kunnen denken dat je dus meer dan twee dagen "veilig" zit, wat niet zo is. De kans dat de eerste sleutel die je test reeds de juiste is, is even groot als dat het de laatste sleutel is. Kortom, gemiddeld gezien zal de sleutel in de helft van de maximum tijd gevonden worden, oftewel 29 uur.

Zouden we de lengte van de sleutel met één karakter verhogen, naar 9, dan stijgt de *maximale* duur naar pakweg 1500 uur, oftewel 62 dagen. Gemiddeld gezien vinden we de sleutel dus na ongeveer 750 uur (ruim 31 dagen). Eén karakter extra heeft wel degelijk een gigantische impact op de veiligheid van een sleutel!

3.3 De eerste algoritmes

Het doel van ieder encryptie-algoritme is om data zodanig te versleutelen zodat enkel eigenaars van de gebruikte sleutel de originele tekst kunnen terugvinden. We vertelden net dat encryptie-algoritmes kunnen onderverdeeld worden volgens de actie die ze uitvoeren: substitutie, transpositie of een combinatie. We tonen van iedere variant nu een historisch voorbeeld.

> Tip
>
> Volgende tool, speciaal gemaakt om crypto te leren, is een erg handig iets om de verschillende cryptografische systemen te visualiseren én testen: link

3.3.1 Substitutie: Caesar encryptie

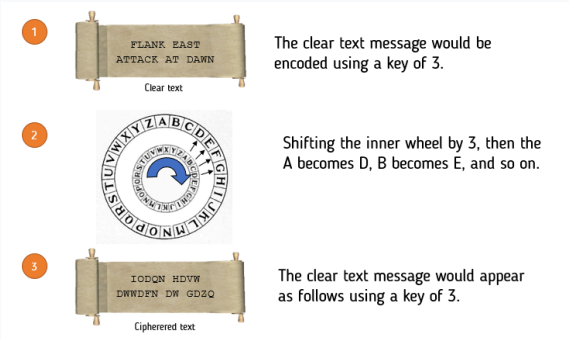
De Caesar encryptie (naar Julius Caesar) bestaat uit een eenvoudig substitutie algoritme. De sleutel is een getal tussen 1 en 25 en geeft aan door welk element uit het alfabet een teken wordt aangepast, als volgt:

- Ieder element wordt voorgesteld als een cijfer. A krijgt de waarde 0, B wordt 1,... Z wordt 25. (We tellen vanaf 0, zodat we straks netjes met de modulo-operator kunnen werken.)
- Als de sleutel het getal 3 is, dan zal nu iedere letter A in de tekst vervangen worden door het teken 0+3, dus D. Iedere B wordt een E, enzovoort.
- Indien er een "overflow" is achteraan komen we uiteraard terug naar voor in het alfabet. Iedere Z wordt dus een C, iedere Y een B, enzovoort.

Het Caesarcipher wordt ook wel kortweg *Rot* genoemd, naar het woord *rotatie*. Een cijfer erachter geeft dan aan welk de te gebruiken sleutel is. Rot4 wil dus zeggen dat alle elementen vier plaatsen opgeschoven moeten worden. Merk op dat Rot13 (ook wel *Caesaralfabet* genoemd) een speciale sleutel is. Als je namelijk twee maal na elkaar Rot13 toepast op een tekst (eerst op de plaintext, dan op de resulterende ciphertext) dan verkrijgt men terug de originele tekst.

Een mooie fysieke manier om Caesar te visualiseren is met een **cipher wheel** (of *Caesar-wiel*): twee concentrische schijven die allebei het alfabet langs de rand hebben staan. Door de binnenste schijf een aantal plaatsen te draaien (afhankelijk van je sleutel), kan je voor iedere letter op de buitenste schijf meteen aflezen welke letter eronder staat. Geen pen en papier nodig — gewoon draaien en aflezen. Historisch

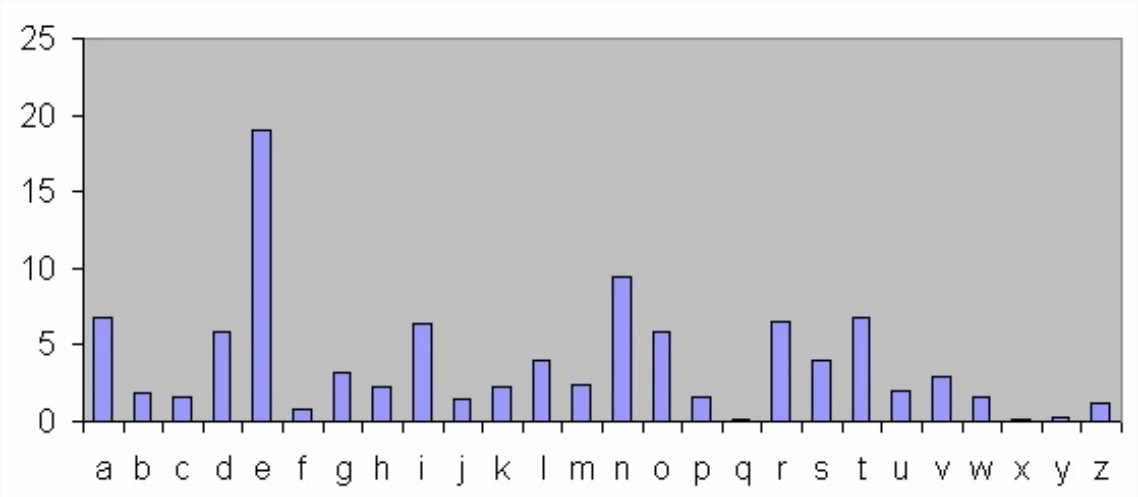
werden dergelijke wielen letterlijk gebruikt door legers en spionnen om snel in en uit geheimschrift te gaan.



Figuur 3.2: Een Caesar-wiel: draai de binnenste schijf en lees af.

Uiteraard kan iedere weldenkende mens in de 21e eeuw een Caesar encryptie bruteforcen. Het aantal mogelijke sleutels beperkt zich tot 25 mogelijkheden (sleutels 0, 26, etc. zullen resulteren in géén encryptie: je plaintext en ciphertext zullen identiek zijn) en je kan dit dus snel testen.

Door **frequentieanalyse** op de ciphertext toe te passen kan men ook de plaintext terugvinden zonder te moeten bruteforcen. Indien de plaintext een tekst in, bijvoorbeeld, het Nederlands is, dan kunnen we gebruik maken van de statistische eigenschappen van een taal. Zo weten we dat bepaalde letters in een standaard Nederlandstalige tekst meer of minder vaak voorkomen. De letter `e` komt bijvoorbeeld veel vaker voor dan de `v`. Daar iedere letter in de encryptie door een andere wordt vervangen, is het dus voldoende om te ontdekken (a.d.h.v. frequentieanalyse) welke letter(s) het meest of minst voorkomen om je zo een vermoeden te geven van de originele letter.



Figuur 3.3: De frequentie-analyse van een typisch Nederlandstalige tekst. Merk op dat de letterscore bij het bordspel Scrabble omgekeerd evenredig is met de frequentie dat de letter gemiddeld voorkomt. Dit verklaart ook waarom er per taal een eigen Scrabble-editie bestaat met eigen letterscores (Bron: Wikipedia).

Dit verklaart ook waarom je best je te encrypteren berichten zo kort mogelijk houdt. Hoe minder tekens, hoe minder frequentieanalyse zal werken. Een andere veelvoorkomende fout (in klassiekere encryptie) was dat de verzender bijvoorbeeld voorspelbare tekst ging encrypteren. Als je weet dat de verzender altijd begint met “Geachte” in z’n berichten, dan is de kans groot dat de eerste 7 tekens in de ciphertext deze plaintext voorstellen.

Het principe van Caesar encryptie, de substitutie, blijft echter overeind staan en zal je nog zien terugkomen in de komende algoritmes.

3.3.1.1 Over de modulo operator

De modulo operator (%) is nuttig bij substitutie-algoritmes zoals bij Caesar encryptie. De modulo operator geeft de rest weer wanneer we de linkse door de rechtse operator zouden delen. 19%5 geeft dus 4 als resultaat.

Je kan de operator gebruiken om snel te weten wat de waarde van een teken wordt bij Caesar-encryptie als volgt:

```
(teken + sleutel) % alfabetLengte => nieuw teken
```

De alfabetLengte is 26 bij Caesar-encryptie, namelijk alle letters van A tot en met Z.

Als je dus een sleutel hebt met waarde 7 en je wilt weten wat de waarde van `Y` (element 24, daar we vanaf 0 tellen) wordt dan schrijf je:

```
(24 + 7) % 26 => 5
```

Dit zal dus 5 worden, oftewel een `F`.

3.3.2 Vigenère: een slimmere substitutie

De grote zwakte van Caesar is dat iedere letter *altijd* op dezelfde manier wordt vervangen: in een tekst vercijferd met sleutel 3 wordt iedere `e` een `h`. Net daarom werkt frequentieanalyse zo goed.

Het **Vigenère-cipher**, vernoemd naar de 16e-eeuwse Fransman Blaise de Vigenère, pakt dit aan met een verrassend eenvoudige truc: in plaats van één vaste shift gebruik je een **sleutelwoord** dat je herhaalt over de plaintext. Iedere letter van het sleutelwoord bepaalt hoeveel de bijhorende plaintext-letter moet opschuiven. Je kan Vigenère dus zien als een rij Caesar-ciphers na elkaar, elk met een eigen shift.

Stel dat onze sleutel `COUNTON` is en de plaintext `vigenerecipher`. We herhalen het sleutelwoord tot het even lang is als de plaintext:

```
Sleutel:    C O U N T O N C O U N T O N
Plaintext:  v i g e n e r e c i p h e r
Ciphertext: X W A R G S E G Q C C A S E
```

De eerste `v` gecombineerd met sleutel-letter `C` levert `X` op (shift 2). De volgende `i` met `O` geeft `W`. En zo verder. Merk op dat de vier `e`'s in de plaintext telkens door een *andere* letter worden vervangen (respectievelijk `R`, `S`, `G` en `S`), afhankelijk van welke sleutelletter er op dat moment boven staat.

In de praktijk gebruikte men een *tabula recta* - een tabel met alle 26 Caesar-alfabetten onder elkaar - om snel de juiste substitutie af te lezen. De rij kies je op basis van de sleutelletter, de kolom op basis van de plaintext-letter.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figuur 3.4: De Vigenère tabula recta. Rij = sleutelletter, kolom = plaintext-letter (Bron: Wikipedia, publiek domein).

En dát is de kracht van Vigenère: eenvoudige frequentieanalyse werkt niet meer. Letters die in de plaintext veel voorkomen, zijn niet langer letters die in de ciphertext veel voorkomen.

Vigenère heette eeuwenlang *le chiffre indéchiffrable* — "het onontcijferbare cipher". Pas in de 19e eeuw werd het systematisch gekraakt door Friedrich Kasiski: eens je de sleutellengte kan achterhalen (via het tellen van herhalingen in de ciphertext), herleidt Vigenère zich tot meerdere parallelle Caesar-ciphers die je elk afzonderlijk met frequentieanalyse kan aanvallen.

Het principe van Vigenère - **variatie inbrengen door een sleutel die zelf varieert** - zie je nog steeds terug in moderne algoritmes. De *round keys* in DES en AES die we verderop behandelen, zijn directe nazaten van dit idee.

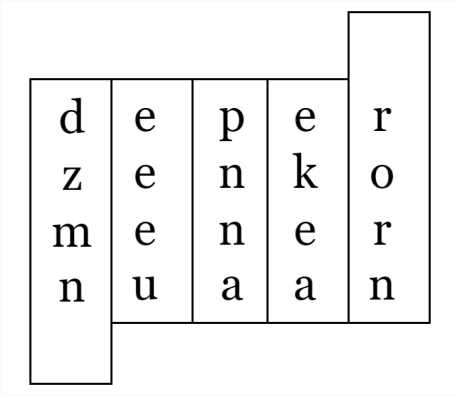
3.3.3 Transpositie: scytale encryptie

Bij transpositie-algoritmen gaan we de positie van de karakters veranderen. De sleutel kan hierbij bepalen op welke manier dit moet gebeuren. De Oude Grieken gebruikten een zogenaamde scytale om aan transpositie-encryptie te doen. Een scytale was een lange stok bestaande uit drie of meerdere lange zijden. De boodschap werd op een lang lint geschreven en dit lint werd dan over de scytale gedraaid. De sleutel gaf aan uit hoeveel vlakken de te gebruiken scytale moest bestaan. Ieder volgend karakter van de plaintext (op het lint) kwam op een andere zijde te liggen. Vervolgens werden alle letters op één zijde achter elkaar gezet, en dit werd herhaald voor iedere zijde: dit werd de ciphertext die werd doorgestuurd.

Om nu de ciphertext te decrypteren werd een onbeschreven lint over de juiste scytale gelegd. Vervolgens werd de verkregen ciphertext op dit lint, zijde per zijde, overgeschreven. Als de ontvanger dan het lint ontrolde kreeg hij terug de originele tekst te zien.

Figuur 3.5: Een authentieke scytale (Bron: Wikipedia).

Stel dat we de tekst “De perzen komen er nu aan” (bron: wikipedia) over een scytale met vier zijden wikkelen, dan krijgen we:



Figuur 3.6: Tijd om de 300 in te roepen!

De ciphertext die we vervolgens versturen (wanneer we het lint afwikkelen) wordt:

Figuur 3.7: Dit gaat Xerxes nooit kunnen lezen...Zeker niet omdat de Nederlandse taal toen nog niet bestond. Muhahahaha!

Uiteraard zijn er tal van varianten mogelijk om transpositie te doen. Eerst kan je beslissen om je plaintext in een bepaalde vorm te plaatsen: bijvoorbeeld in tien kolommen. Vervolgens kan je dan, gebaseerd op de sleutel, beslissen in welke volgorde je de kolommen achter elkaar plaatst om zo de ciphertext te bekomen. Dit is een zogenaamd *route cipher* wat onder andere werd gebruikt tijdens de Amerikaanse Burgeroorlog.

3.3.4 Combinatie

Het spreekt voor zich dat een combinatie van een transpositiecipher en een substitutiecipher je encryptie nog versterkt. Veel moderne algoritmen kunnen nog steeds herleid worden tot een sequentie van meerdere basisvormen na elkaar.

De **Advanced Encryption Standard (AES)** is in de 21e eeuw zo’n beetje de de facto standaard als het aankomt op symmetrische encryptie (d.w.z. encryptie waar maar één sleutel voor nodig is, verder meer hierover). Als we echter eens het algoritme opengooien en een enkele *encryption round* bekijken

(AES bestaat uit een sequentie van deze rondes) dan zien we dat de bits die bovenaan binnenkomen (*state*) vervolgens een combinatie van substituties (*sub*) en transposities (*mixcolumns* en *shiftrows*) ondergaan.

We gaan AES nog terug zien opduiken wanneer we gaan bekijken hoe draadloze netwerken worden beveiligd. Als Belg mogen we trouwens erg fier zijn op deze wereldwijd gebruikte Amerikaanse standaard. Je zal later ontdekken waarom dat zo is!

> Tip
>
> Doel van dit hoofdstuk is ook aantonen dat je geen wiskundig wondertalent moet zijn om de basisconcepten van cryptografie te begrijpen. Hier en daar neem ik wat vrijheden om bepaalde stappen te vereenvoudigen, maar de essentie van de algoritmen blijft wel bewaard en daarmee, hopelijk, ook de eenvoud (en dus elegantie) ervan.

3.4 Cryptanalyse

De term cryptanalyse is nu al enkele keren gevallen: de wereld van de cryptologie bestaat uit twee delen, die elkaars tegengestelden zijn:

1. **Cryptografie**: de wetenschap van het versleutelen van informatie.
2. **Cryptanalyse**: de wetenschap van het ontcijferen van versleutelde informatie, zonder kennis van de gebruikte sleutel.

We gaan in dit boek niet te veel tijd aan de wondere wereld van cryptanalyse spenderen, daar dit ons te ver zou brengen. We vatten echter even de belangrijkste concepten hier samen.

3.4.1 Sleutellengtes en bruteforcen

De term *bruteforce* dekt de lading goed. Letterlijk vertaald wordt het: met brute kracht forceren. Kortom, je gebruikt het wanneer je niet weet wat doen tijdens de cryptanalyse en gewoonweg de minst efficiënte manier mogelijk zal toepassen, maar waarvan wel geweten is dat ze altijd zal werken. Namelijk iedere mogelijke sleutel testen die het cipher toelaat.

Zoals je je kan inbeelden is de sleutellengte evenredig met de tijd die cryptanalysten nodig hebben om je sleutel te bruteforcen. De maximale tijd die nodig is alle sleutels van een bepaalde lengte te berekenen kan je als volgt vinden:

$$MaximaleTijd = \frac{AantalMogelijkeTekens^{SleutelLengte}}{pogingen/seconde}$$

3.4.1.1 Een bruteforce getalvoorbeeld

Een sleutel (of wachtwoord) bestaat uit 8 tekens, enkel kleine letters van a tot en met z zijn toegestaan. De berekeningen worden op een GeForce GTX 1080 gedaan die ongeveer 30 miljoen pogingen per seconde kan doen. We krijgen dan:

$$MaximaleTijd = \frac{26^8}{30000000}$$

Oftewel ongeveer 6960 seconden, wat neerkomt op ongeveer 1,9 uur tijd benodigd om alle mogelijke sleutels te testen (herinner je eraan dat deze tijd gehalveerd moet worden om te weten hoe lang het gemiddeld zal duren om de juiste sleutel terug te vinden.

Volgende tabel (bron) toont nog voorbeelden waarbij telkens dezelfde GeForce 1080 GTX kaart werd gebruikt. Het getal tussen haakjes geeft aan hoeveel mogelijke tekens er in dit type mogelijk zijn:

# tekens	enkel nummers (10)	kleine letters (26)	grote & kleine letters & nummers (62)	eender welk teken (95)
4	0,3 ms	15 ms	490 ms	2,7 s
5	3 ms	400 ms	31 s	4,3 min
6	33 ms	10 s	32 min	6,8 uur
7	330 ms	4,5 min	33 uur	27 dagen
8	3,3 s	1,9 uur	84 dagen	7 jaren

# tekens	enkel nummers (10)	kleine letters (26)	grote & kleine letters & nummers (62)	eender welk teken (95)
9	33 s	2,1 dagen	14 jaren	670 jaren
10	5,6 min	54 dagen	890 jaren	$6,3 * 10^4$ jaren
11	56 min	3,9 jaren	$5,5 * 10^4$ jaren	$6 * 10^6$ jaren
12	9,3 u	100 jaren	$3,4 * 10^6$ jaren	$5,7 * 10^8$ jaren
13	3,9 dagen	$2,6 * 10^3$ jaren	$2,1 * 10^8$ jaren	$5,4 * 10^{10}$ jaren
14	39 dagen	$6,8 * 10^4$ jaren	$1,3 * 10^{10}$ jaren	$5,1 * 10^{12}$ jaren
15	1,1 jaar	$1,8 * 10^6$ jaren	$8,1 * 10^{11}$ jaren	$4,9 * 10^{14}$ jaren
16	11 jaar	$4,6 * 10^7$ jaren	$5 * 10^{13}$ jaren	$4,7 * 10^{16}$ jaren

Tip

Per verdubbeling van het aantal GeForce-kaarten halveert de tijd.

Opmerking

Om bovenstaande gigantische getallen wat te duiden: de leeftijd van ons universum wordt op 13,8 miljard jaar geschat, oftewel $1,38 * 10^{10}$ jaren. Onze mooie blauwe planeet is ongeveer 4,5 miljard jaar oud. De Tyrannosaurus Rex liep ongeveer 70 miljoen jaar geleden rond, oftewel $7 * 10^7$ jaren geleden.

3.4.1.2 Dictionary attack

Wanneer de cryptanalist vermoedt dat de te zoeken sleutel iets anders is dan volledig willekeurige tekens dan kan hij de bruteforce aanval verbeteren (denk aan "administrator2022"). In plaats van alle mogelijke combinaties (permutaties) van de sleutel te testen, zal hij een woordenboek (**dictionary**) gebruiken met daarin alle mogelijke sleutels en woorden die mogelijk de originele sleutel bevatten.

Tools zoals John The Ripper kan je *voeden* met een dergelijk woordenboek en dan vragen om sleutels te testen die gebaseerd zijn op zaken uit dat woordenboek, inclusief bijvoorbeeld door er tekens voor en na te zetten. Als in het woordenboek het woord *god* staat, dan kan John The Ripper bijvoorbeeld ook alle sleutels testen zoals *god1*, *god2*, etc.

Er zijn tal van woordenboeken online te downloaden die gevuld zijn met de meest gebruikte wachtwoorden die cryptanalisten (en dus ook de digitale stropers) kunnen gebruiken om de sleutel sneller te vinden. Het is aangeraden om zeker geen wachtwoorden (of permutaties ervan) te gebruiken die in volgende lijsten voorkomen: https://github.com/danielmiessler/SecLists/tree/master/Passwords/Common-Credentials

Dit waren in 2020 de 10 meest gebruikte wachtwoorden:

123456, password, 12345678, qwerty, 123456789, 12345, 1234, 111111, 1234567, dragon

Dit soort lijsten worden opgesteld door gekende datalekken te analyseren op welke wachtwoorden er in voorkomen.

3.4.2 Soorten cryptanalytische aanvallen

Geregeld zullen we in dit boek bepaalde zwakheden beschrijven die in algoritmes misbruikt kunnen worden door een bepaald type cryptanalytische aanval. Deze aanvallen zijn afhankelijk van de informatie die de cryptanalist bezit:

- Enkel de ciphertext: vanuit het standpunt van de cyberboswachters is dit het beste soort informatie dat de aanvaller bezit. Hij heeft enkel een hoop geëncrypteerde informatie en moet proberen daar de originele plaintext uit te krijgen. Vanuit het standpunt van de cryptanalist is dit dus de minst goede situatie om vanuit te starten.

- Gekende plaintext: de cryptanalist heeft één of meerdere stukken informatie waarvan zowel de ciphertext als de bijhorende plaintext gekend is.
- Gekozen plaintext: de cryptanalist kan zelf plaintext kiezen waarvan de bijhorende ciphertext moet gemaakt worden. Dit zorgt ervoor dat de cryptanalist als het ware kan experimenteren.
- Gekozen ciphertext: het zelfde concept als *gekozen plaintext* maar deze keer kiest de cryptanalist de ciphertext waarvan hij de bijhorende plaintext wil genereren.

Er zijn nog enkele meer gespecialiseerde types, maar voor deze cursus zullen we het bij deze vier basistypes houden.

> **i** Opmerking
>
> Er wordt in deze sectie soms over aanvaller gesproken, alsof de cryptanalist automatisch van kwade wil is. De wetenschap van de cryptanalyse is dat uiteraard verre van: enerzijds zorgt het ervoor dat bestaande en nieuwe cryptografische algoritmes op hun sterkte kunnen getest worden. Anderzijds helpen ze in tijden van oorlog om boodschappen van vijanden te onderscheppen en proberen lezen.

3.4.3 De menselijke factor: de échte beste “cryptanalyse”

Tijd voor een ongemakkelijke waarheid: de meest effectieve aanval op een cryptografisch systeem is in de praktijk *géén* cryptanalyse. Het is véél goedkoper, sneller en betrouwbaarder om de sleutel gewoon **te vragen aan de gebruiker**. Een welgemikte phishingmail, een vervalste helpdeskoproep, een post-it onder het toetsenbord, of een collega die “even snel” z’n scherm ontgrendelt — dát zijn de aanvallen waar cyberboswachters écht wakker van liggen.

There is no patch for human stupidity, is een vaak gehoord cliché in de security-wereld. Hoe sterk je algoritme ook is, hoe lang je sleutel ook, als de gebruiker z’n wachtwoord opschrijft op een briefje of doorgeeft aan wie er “vriendelijk om vraagt”, dan valt heel je cryptografische kaartenhuis in elkaar.

Deze categorie aanvallen noemen we **social engineering** en we behandelen ze uitgebreid in een later hoofdstuk. Onthoud voor nu: cryptografie is een noodzakelijke voorwaarde voor veiligheid, maar zelden voldoende.

3.4.4 En wat met quantum-computers?

Al jaren houdt de crypto-wereld angstvallig de ontwikkelingen in de quantum-computer wereld in het oog. Alhoewel we nog maar in de babyfase van quantum-computers zijn, is het toch best mogelijk dat binnen afzienbare tijd (20, 30 jaar?) we effectief zodanig sterke quantum-computers zullen hebben die alle bestaande cryptografische systemen in een handomdraai kunnen “kraken”.

Daarom hanteren veiligheidsdiensten al vele decennia ook het **store now, decrypt later** principe. Ze gaan ervan uit dat computers steeds krachtiger worden: berichten die in de jaren 60 werden versleuteld, kunnen nu in een handomdraai ontcijferd worden. Quantum-computers zullen dit proces nog veel sneller maken.

Hoe dit zal gebeuren snapt de auteur ook (nog) niet en zal dus niet verder uitgewerkt worden in dit handboek. Besef gewoon dat quantum-computers van de toekomst potentiële bruteforce aanvallen drastisch zullen kunnen versnellen.

Het is om deze reden dat er nu reeds onderzoek wordt gedaan naar cryptografische ciphers die bestand zullen zijn tegen de computers van de toekomst. Dit soort ciphers worden *post-quantum cryptografische ciphers* genoemd en zullen niet in dit boek besproken worden.

> **i** Opmerking
>
> Trouwens, ook andere systemen die gebruik maken van cryptografische concepten zullen in één klap hun nut verliezen. Of zoals dit artikel zegt *“And [as]encryption is everywhere in modern day life, from e-commerce, to online payments, to passwords, everything will be vulnerable!”*
>
> Denk daarbij bijvoorbeeld aan *cryptocurrencies* zoals Ethereum en Bitcoin:
>
> *Cybersecurity specialist Itan Barmes led the vulnerability study of the Bitcoin blockchain. He found the level of exposure that a large enough quantum computer would have on the Bitcoin blockchain presents a systemic risk. “If [4 million] coins are eventually stolen in this way, then trust in the system will be lost and the value of Bitcoin will probably go to zero,” he says.* Bron

3.5 Moderne cryptosystemen

De klassieke ciphers die we tot nu toe zagen — Caesar, Vigenère en scytale — zijn fantastisch om de basisprincipes van substitutie en transpositie uit te leggen, maar in de praktijk vallen ze ondertussen om als een kaartenhuis. Caesar kraak je in maximaal 25 pogingen, Vigenère valt na Kasiski's truc terug op een handvol parallelle Caesars, en de sleutelruimte van een scytale beperkt zich tot het aantal vlakken dat redelijkerwijs op een stok past. Tegen de rekenkracht van vandaag (denk terug aan de bruteforce-tabel hierboven) houdt geen enkel klassiek algoritme nog stand.

Moderne cryptografische systemen zijn dan ook compleet anders opgebouwd:

- **Veel grotere sleutelruimtes** (128, 192, 256 bits en meer) waardoor bruteforce praktisch onhaalbaar wordt.
- **Iteratief**: niet één enkele substitutie, maar tientallen rondes van substitutie én transpositie achter elkaar, waardoor statistische patronen volledig verdwijnen.
- **Publiek ontworpen en getest** door duizenden experts wereldwijd (Kerckhoffs in actie) → géén *security through obscurity*.
- **Wisselende subkeys per ronde**, afgeleid uit de hoofdsleutel. Eigenlijk het idee van Vigenère, maar dan op steroïden.

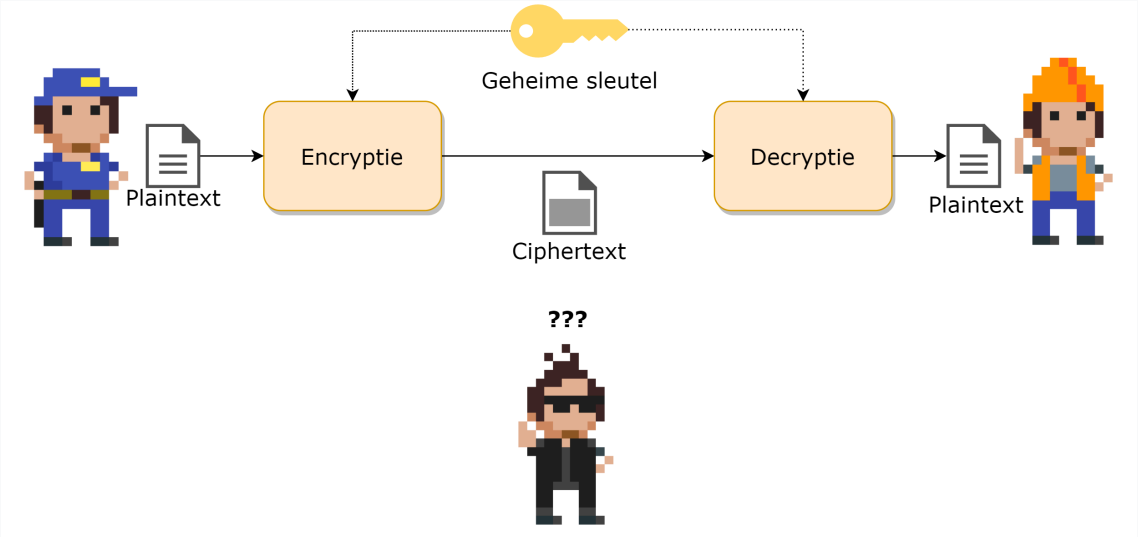
Wanneer we deze moderne systemen klasseren volgens het **aantal sleutels** dat ze gebruiken, vallen ze in twee grote families:

- **Symmetrische** systemen: zender en ontvanger gebruiken **dezelfde** sleutel.
- **Asymmetrische** systemen: er zijn **twee** sleutels — één publiek (om te encrypteren) en één privaat (om te decrypteren).

We starten met de oudste van de twee, de symmetrische cryptosystemen. Verderop in dit hoofdstuk komen de asymmetrische aan bod, en zal blijken dat beide families elkaar in de praktijk perfect aanvullen.

3.6 Symmetrische encryptie

Symmetrische systemen zijn de oudste vorm van encryptie: zowel ontvanger als verzender gebruiken dezelfde sleutel. De term *symmetrisch* verwijst naar het feit dat het algoritme exact hetzelfde doet aan beide zijden. Het enige verschil is dat bij de verzender de plaintext in het systeem wordt gestoken, wat resulteert in een ciphertext. Terwijl de ontvanger de ciphertext in het systeem plaatst om een plaintext te krijgen.



Figuur 3.8: Het basismodel van symmetrische encryptie.

- De symmetrische encryptiesystemen zijn de oudste vorm: alle klassieke algoritmes waren van dit principe. Asymmetrische systemen zijn pas in de 20e eeuw ontwikkeld (circa 1970).
- Voorbeelden van bestaande symmetrische encryptiesystemen zijn: AES, DES, IDEA, RC4, Blowfish, etc.

Dit type encryptie is nog steeds het meest gebruikte en wordt overal gebruikt waar data op een veilige manier (confidentiality) moet bewaard, verstuurd of verwerkt worden.

3.6.1 Sleuteloverdracht

De moeilijkheid bij symmetrische systemen is de sleuteloverdracht. Daar ontvanger en verzender dezelfde sleutel hanteren is het natuurlijk belangrijk dat deze de sleutel op een veilige manier kunnen uitwisselen. Dit probleem wordt niet opgelost door symmetrische cryptosystemen. Afhankelijk van de context kan deze uitwisseling op verschillende manieren gebeuren:

- Via een asymmetrisch encryptiesysteem dat wél sleutels op een veilige manier kan uitwisselen (zie verder).
- Via een ander beveiligd kanaal, in eender welke vorm (bijvoorbeeld fysiek de sleutel aan de andere persoon geven of zeggen, deze opsturen via een reeds opgezet symmetrisch encryptiekanaal, etc.).

3.6.2 Block- en streamciphers

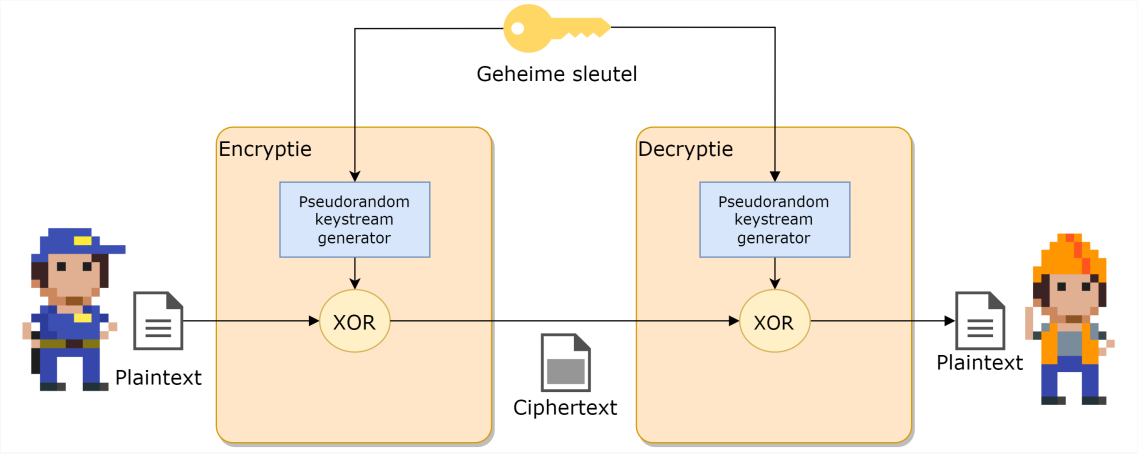
Er zijn twee soorten symmetrische encryptieciphers als we kijken naar de manier waarop ze de te encrypteren data verwerken:

- **Streamciphers**: hierbij wordt de data letterlijk als een stroom (*stream*) van tekens beschouwd. Waarbij teken per teken individueel geëncrypteerd wordt (het bekendste voorbeeld is RC4). Voor ieder teken dat verwerkt wordt zal er exact één geëncrypteerd teken gegenereerd worden. Dit soort algoritmes zijn over het algemeen sneller dan blockciphers.
- **Blockciphers**: de data wordt in blokken (van bijvoorbeeld 128 tekens) verwerkt. Bekendste voorbeelden die we verderop behandelen zijn AES, DES, 3DES, etc.

3.6.3 Streamciphers

De werking van een symmetrisch streamcipher is verrassend eenvoudig en bestaat uit twee delen:

- Een **pseudorandom keystream generator**: deze zal de sleutel als het ware expanderen naar een sleutel met de zelfde lengte als de stream, genaamd een **keystream**. Als je 400 bytes aan data wenst te encrypteren, zal je een keystream van 400 bytes moeten genereren. Daar we met een stream werken zal deze generator teken per teken genereren. Hoe dit gebeurt, leggen we verderop uit.
- De **XOR** of "exclusieve of" functie: deze zal de plaintext naar een ciphertext omzetten door de plaintext met de keystream samen te voegen. Deze stap is de feitelijke encryptie!



Figuur 3.9: Het streamcipher proces.

Aan de ontvangerzijde gebeurt exact hetzelfde. **Enkel indien de ontvanger dezelfde sleutel gebruikt, zal deze dezelfde keystream kunnen genereren, en bijgevolg enkel dan de originele plaintext verkrijgen.**

Het hart van een symmetrisch streamcipher is dus enerzijds de XOR-functie én, belangrijker, de manier waarop de keystream wordt gemaakt.

3.6.3.1 De XOR functie

De waarheidstabel van de XOR-functie is de volgende:

Plaintext input	Keystream input	Ciphertext output
1	0	1
0	1	1
0	0	0
1	1	0

De XOR-functie wordt in schema's aangeduid door een cirkel met een plusje in: $\oplus$

Beeld je in dat we het bericht `1010` willen versleutelen, en we hebben een gegenereerde keystream `1101`. Als we deze *XOR'n* dan geeft dit `0111`. Dit is dus de ciphertext. Als de ontvanger dezelfde keystream kan genereren en deze XOR'd met de verkregen ciphertext, dan krijgt deze terug de originele plaintext.

3.6.3.2 De keystream generator

De keystream generator heeft dus als doel om voor ieder karakter dat moet geëncrypteerd worden een bijhorend keystream karakter te maken. Deze karaktergeneratie moet onvoorspelbaar zijn (*random*) tegenover de sleutel die wordt gebruikt en het voorgaande karakter dat werd gemaakt. Echter, dit moet wel PSEUDO (*schijn*)-willekeurig zijn: dezelfde sleutel als beginpunt (**seed**) moet steeds dezelfde reeks genereren.

De kracht (en zwakte) van een symmetrisch streamcipher ligt in de implementatie van de manier waarom deze keystream generator werkt. Mogelijke zwakheden kunnen bijvoorbeeld zijn dat de gegenereerde stroom informatie van de sleutel "lekt" naar de keystream (wat desastreuze gevolgen bleek te hebben bij de originele wifi-security (WEP), waarover later meer) of een voorspelbare "randomiteit" van de keystream.

Om aan encryptie te kunnen doen, hebben we systemen nodig die onvoorspelbaar zijn. Als de aanvaller kan voorspellen wat de uitvoer van een onderdeel van de encryptie zal zijn, dan kunnen we geen confidentiality en integrity voorzien. Kortom, we hebben algoritmes nodig die willekeurige getallen kunnen generen die 100% onvoorspelbaar zijn. Net zoals het werpen van een dobbelsteen niet voorspeld kan worden, zo ook moeten onze algoritmes een (digitale) dobbelsteen hebben.

Digitale systemen die perfect willekeurige getallen genereren noemt men **random number generators** (RNG). Uiteraard moet een RNG geprogrammeerd kunnen worden: dat behelst dus een algoritme. Een algoritme is per definitie "voorspelbaar". Alles hangt daarom af van de invoer die het algoritme gebruikt om random getallen te beginnen genereren. We spreken dan van een **pseudorandom number generator** (PRNG), pseudo (**schijnbaar**) omdat de uitvoer afhankelijk is van het startgetal, de zogenaamde **seed**. Die seed kan bijvoorbeeld de encryptiesleutel zijn: enkel met dié sleutel zal het algoritme dezelfde reeks getallen generen. Er zijn echter ook systemen die bijvoorbeeld de huidige tijd of de staat van een flipflop als startpunt gebruiken (wanneer je een flipflop aanzet kan je niet voorspellen of deze op 1 of 0 zal staan, daar deze staat beïnvloed wordt door de elektromagnetische straling). Uiteraard is een dergelijke seed voor een keystream generator nutteloos, daar zowel verzender én ontvanger dezelfde reeks getallen moeten kunnen genereren.

3.6.3.3 Een eenvoudig PRNG: middle-square

Een van de eerste PRNG-algoritmes werd in 1946 bedacht door de legendarische wiskundige John von Neumann: de **middle-square methode**. De werking is verbluffend eenvoudig:

1. Neem een startgetal (de **seed**).
2. Kwadrateer het.
3. Neem de middelste cijfers als volgende pseudo-willekeurig getal.
4. Gebruik dat getal opnieuw als invoer voor stap 2, enzovoort.

Laten we dit uitvoeren met seed `1111` (pad tot 8 cijfers zodat we altijd 4 middelste cijfers hebben):

- $1111^2 = 01234321 \rightarrow$ middelste 4 cijfers: `2343`
- $2343^2 = 05489649 \rightarrow$ middelste 4 cijfers: `4896`
- $4896^2 = 23970816 \rightarrow$ middelste 4 cijfers: `9708`
- ...

Onze pseudo-willekeurige reeks wordt dus `2343, 4896, 9708, ...`. Wie met dezelfde seed start, krijgt gegarandeerd dezelfde reeks. En dát is precies wat we willen voor encryptie: zender én ontvanger moeten dezelfde keystream kunnen genereren.

In de praktijk is middle-square ondertussen niet meer bruikbaar: de gegenereerde reeksen vallen snel in korte cycli of landen op `0000` waarna het algoritme vast komt te zitten. Maar het idee - *deterministisch uit een seed een schijnbaar willekeurige reeks maken* - blijft de basis van alle moderne PRNGs zoals die in RC4.

3.6.3.4 PRNG in de praktijk: Random() in C#

Zo goed als iedere moderne programmeertaal heeft een ingebouwde PRNG. In C# gebruik je `new Random(seed)`, in Python `random.seed()`, in JavaScript... tja, daar is het een beetje complexer. Een eenvoudig voorbeeld in C# maakt het principe meteen concreet:

```
int key = 666;

// Zender genereert 10 getallen
Random s = new Random(key);
for (int i = 0; i < 10; i++) Console.Write(s.Next(1, 10));
// Output: 6337963648

// Ontvanger gebruikt dezelfde seed → identieke reeks
Random r = new Random(key);
for (int i = 0; i < 10; i++) Console.Write(r.Next(1, 10));
// Output: 6337963648

// Eve probeert met een andere seed
Random e = new Random(123);
for (int i = 0; i < 10; i++) Console.Write(e.Next(1, 10));
// Output: 9978711226
```

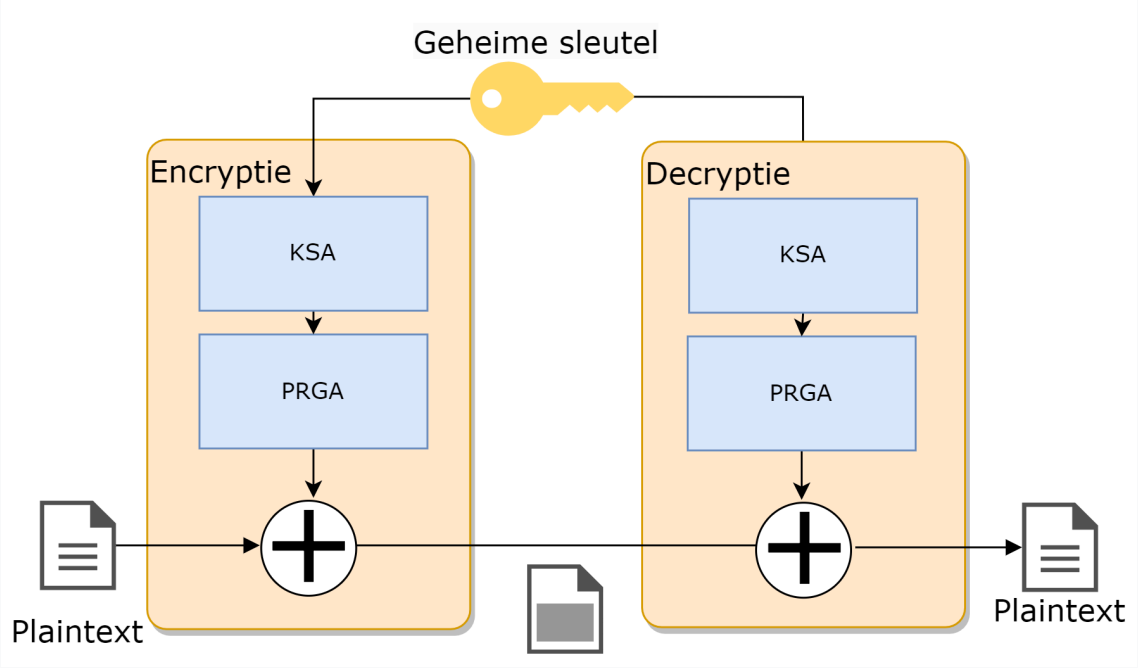
Zender en ontvanger die dezelfde sleutel (seed) gebruiken, krijgen identiek dezelfde reeks — perfect als keystream voor een streamcipher. Zonder de juiste seed krijg je een totaal andere reeks en is de keystream nutteloos voor cryptanalyse.

> ⚠ Waarschuwing
>
> De ingebouwde `Random`-klasse in C# is **niet cryptografisch veilig**. Hij is prima voor games, simulaties of dobbelstenen, maar niet geschikt om data mee te beschermen. Voor écht cryptografisch gebruik neem je de klasse `RandomNumberGenerator` uit `System.Security.Cryptography`. Het basisprincipe (*seed + deterministisch algoritme = reproduceerbare reeks*) blijft wel identiek.

3.6.3.5 RC4 tot op het bot

Laten we eens één van de meest gebruikte streamciphers bekijken, het RC4 cipher. Dit algoritme, ontwikkeld door Ron Rivest (de afkorting staat trouwens voor *Rons Cipher 4*), wordt gebruikt onder andere om een beveiligde SSL-tunnel (zie later) op te zetten en zit in het hart van veel geëncrypteerde communicatiekanalen.

RC4 werkt zoals we eerder verklaarden hoe een streamcipher werkt: het heeft een keystream generator en zal de keystream vervolgens XOR'n met de plaintext. Eerst zal de ingevoerde sleutel (die 40 tot 2048 bits lang mag zijn) omgezet worden naar een compatibele werksleutel met behulp van een **Key scheduling algorithm** (KSA). Deze werksleutel zal dan als seed gebruikt worden om een keystream in het **Pseudo-random generator algorithm** (PRGA) te maken.



Figuur 3.10: RC4 tot op het bot.

KSA

De **KSA** heeft als doel om de ingevoerde 40 tot 2k-bit sleutel om te zetten naar een compatibele sleutel voor de PRGA. Het doet dit volgens een eenvoudig algoritme:

Stap 1: plaats de ingevoerde sleutel in een array (**T**) van 256 karakters. Herhaal de sleutel indien nodig.

Stap 2: maak een sleutelarray (**S**) aan, ook van 256 karakters, en plaats er de waarden 0 tot en met 255 in.

Stap 3: permuteer (*transpositie*) de elementen in de array `S` met behulp van de array `T` als volgt:

```
j = 0
for i = 0 tot 255
{
  j = (j + S[i] + T[i]) % 256
  Verwissel(S[i],S[j])
}
```

Deze stap zal dus de elementen in de sleutelarray `S` naar nieuwe posities in diezelfde array plaatsen en deze ook de hele tijd van plek wisselen. De index `j` is afhankelijk van de originele sleutel uit stap 1 en zorgt er dus voor dat iedere finale sleutel een unieke array `S` zal opleveren.

Uitgewerkt voorbeeld van KSA

Stel dat onze sleutel "ab" is. De decimale ASCII-waarden van a en b zijn 97 en 98 respectievelijk. Onze array T zal dus bestaan uit 128 keer de waarden 97 en 98 na elkaar aan de start:

Index	Waarde
T[0]	97
T[1]	98
T[2]	97
etc.	

Stap twee genereert de tabel `S` die gewoon de waarden 0 tot en met 255 heeft in de 255 plekjes van de array (de waarde is dus in deze fase gewoon ook de index van het element)

Als we dan stap drie toepassen dan krijgen we na de eerste iteratie van de loop (`i=0`):

```
j = (0 + S[0] + T[0]) % 256
```

oftewel

`j= (0 + 0 + 97) % 256  => j wordt 97`
De eerste wissel die in `S` zal plaatsvinden is dan `Verwissel(S[0],S[97])`
`S` ziet er dan als volgt uit na de eerste iteratie:

Index	Waarde
S[0]	97
S[1]	1
S[2]	2
...	
S[97]	0
etc.	

En dit herhalen we nog 255 keer. **Finaal hebben we nu in tabel `S` een sleutel die bestaat uit de getallen 0 tot en met 255 verdeeld over willekeurige plekken in de array.** Indien we een andere initiële sleutel zouden hebben gebruikt dan zou deze tabel er totaal anders uitzien.

PRGA

Nu we een compatibele sleutel `S` hebben kan de keystream generatie van start gaan. Deze bestaat uit een loop die blijft doorgaan telkens een nieuw plaintext karakter binnenkomt, als volgt:

```
i = 0
j = 0
Herhaal telkens keystream karakter nodig is
{
  i = (i+1) % 256
  j = (j +S[i]) % 256
  Verwissel(S[i],S[j])
  k = S[(S[i]+S[j]) % 256]
  output k naar keystream
}
```

Telkens zal het algoritme één specifieke waarde (tussen 0 en 255) uit de array teruggeven als keystream karakter `k`. Zoals je ziet zal de array `S` voorts de hele tijd van "gedaante" blijven veranderen. Telkens we een element `k` uitsturen zal ook de tabel `S` weer wat zijn veranderd daar we de waarden van `S[i]` en `S[j]` onderling verwisselen.

Uitgewerkt voorbeeld van PRGA

Als we verder werken met de tabel `S` van het vorige uitgewerkte KSA voorbeeld en de waarden ervan gebruiken na één iteratie dan krijgen we onderstaande berekeningen (*We veronderstellen even dat we de KSA niet verder hebben uitgevoerd en de tabel `S` dus dezelfde is gebleven als op het einde van het voorbeeld, wat in het echt niet zal zijn.*):

```
i = (0+1) % 256 => 1
j = (0 + S[1]) % 256 => (0+1) % 256 => 1
  (S[1] is nog steeds 1: in de KSA-iteratie hebben we enkel S[0] en S[97] gewisseld)
Verwissel(S[1], S[1]) => geen werkelijke wissel, want i en j vallen samen
k = S[ (S[1] + S[1]) % 256 ] => k = S[2] => 2
we outputten de waarde 2 als eerste keystream-karakter
```

> **i** Opmerking
>
> In dit specifieke voorbeeld vallen `i` en `j` toevallig samen, waardoor er geen wissel gebeurt. In een realistische PRGA-run (na een volledige KSA van 256 iteraties) is de tabel `S` echter grondig dooreengeschud en zal vrijwel iedere PRGA-iteratie wél een wissel opleveren.

Finaal zal de output, de waarde `k`, ge-XOR'd worden met het huidige karakter van de plaintext stream.

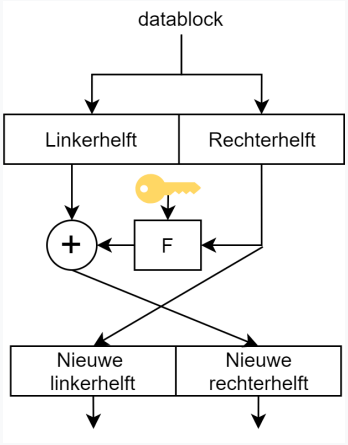
> **i** Opmerking
>
> Zo, dat viel nog mee he? Zoals al gezegd, een belangrijke motivatie van dit boek is aantonen dat je niet bang hoeft te zijn van wat er achter de schermen van de cyberwereld gebeurt. De hoeveelheid wiskunde die we bijvoorbeeld nodig hadden, is beperkt gebleven tot onze trouwe modulo (%)-operator en meer niet. Wanneer we zo meteen een blockcipher gaan uitkleden, zal je ook daar ontdekken dat je best in staat bent schijnbaar complexe technologieën te begrijpen. Hop naar de blockciphers dus!

3.6.4 Blockciphers

Blockciphers, de naam zegt het al, zullen eerst de plaintext in blokken karakters opdelen (bv. 128 bits) en vervolgens blok per blok encrypteren.
Indien het aantal te encrypteren bytes aan data geen exact veelvoud is van de blokgrootte dient er **padding** te gebeuren. Hierbij zal het laatste blok opgevuld (*gepad*) worden met extra bytes tot het blok terug de juiste blokgrootte heeft. De manier waarop de padding gebeurt, is afhankelijk van het cipher dat gehanteerd worden.

3.6.4.1 Feistel-structuren

Ook hier zullen we dezelfde soorten operaties (XOR, substituties en transposities) zien terugkomen. Echter, ook **Feistel**-structuren worden hier gebruikt: in deze operatie zal de data steeds in twee helften worden gesplitst en wordt steeds een specifieke operatie (aangeduid met F van functie in de figuur), zoals een substitutie, op één helft uitgevoerd dat dan wordt ge-XOR'd met de andere helft. Dit wordt meerdere keren herhaald, waarbij de linker (L) en rechterzijde (R) steeds afwisselend door de specifieke encryptie-operatie gaan.



Figuur 3.11: Een enkele Feistel-structuur.

Net zoals bij RC4 zullen we ook vaak met een zogenaamd key scheduling algoritme werken zodat de sleutel niet constant doorheen het hele proces dezelfde is en we met **subkeys** of *round keys* werken.

3.6.4.2 DES tot op het bot

Een van de oudste blockciphers is **DES** oftewel **Data Encryption Standard**. Deze standaard werd in 1977 geboren en vereiste toen blokken van 64 bits data en gebruikte een 56bit sleutel. Ondertussen is dit cipher zeer outdated, maar we gaan hem toch bekijken omdat deze enerzijds erg belangrijk was voor de wereld - grote delen van het bankwezen beschermden er hun financiële transacties mee (en nadien met de opvolger 3DES, zie verder) - en de standaard is erg duidelijk om een goed inzicht in blockciphers te verkrijgen.

Er was wel veel controverse rond deze standaard (gebaseerd op het door IBM ontwikkelde Lucifer cipher) omdat de originele versie (genaamd Lucifer) werkte met een dubbel zo grootte sleutel (128 bit) en bijgevolg dus veiliger was op lange(re) termijn. De Amerikaanse veiligheidsdienst, NSA, was niet zo

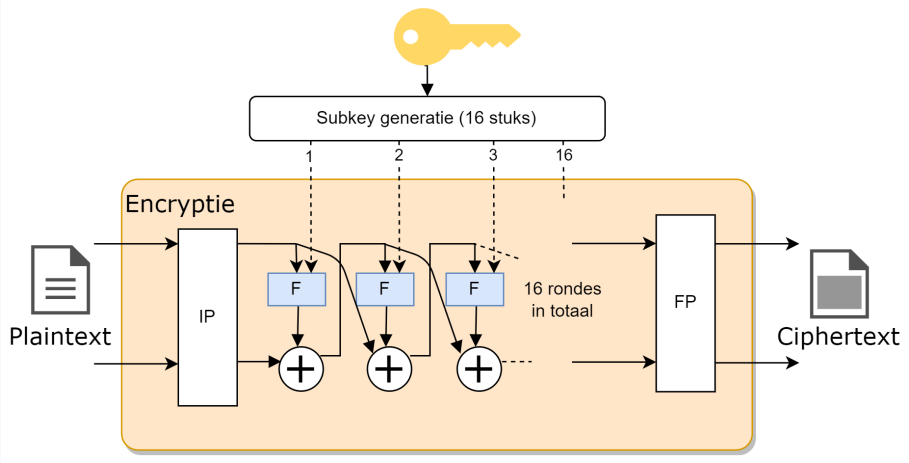
happig om een dergelijk goed beveiligd algoritme commercieel te maken en zorgde er daarom voor dat de sleutellengte gehalveerd werd.

Data met DES encrypteren (en bijgevolg ook decrypteren daar het een symmetrisch cipher is) bestaat uit twee onderdelen:

1. **Versleuteling**: een reeks Feistel-structuren na elkaar (**16 rondes**) die de data blok per blok versleutelen.
2. **Subkeys maken**: een key scheduling algoritme dat 16 subkeys genereert (één voor iedere encryptieronde), gebaseerd op de 56 bit sleutel.

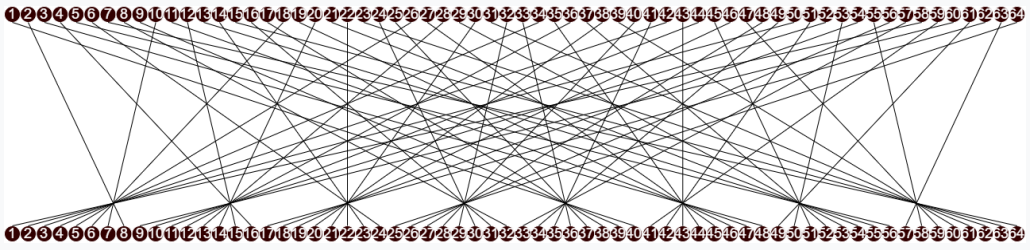
Versleuteling

Volgende schema toont de encryptie bestaande uit 16 rondes:



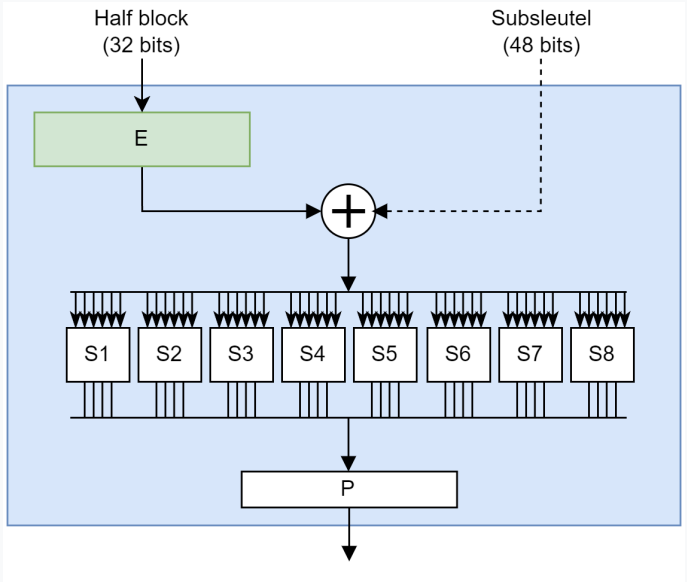
Figuur 3.12: DES encryptie. 16 Feistel-structuren na elkaar.

De data wordt blok per blok doorheen dit gedeelte gestuurd. Eerst gebeurt er een *Initiële Permutatie* (*IP* in de figuur) waarbij iedere bit naar een andere plek wordt gestuurd volgens een vast patroon. Achteraan gebeurt dit nogmaals in een *Finale Permutatie* (*FP*).



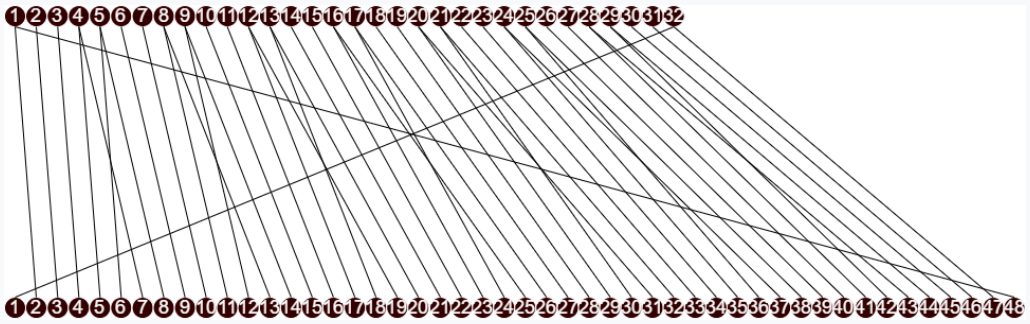
Figuur 3.13: De Initiële Permutatie.

Na de *IP* gaat de data door 16 Feistel-structuren die telkens hetzelfde doen. De data wordt in twee helften gesplitst waarbij de rechterzijde door de F-operatie gaat (die we zo meteen toelichten), het resultaat hiervan wordt ge-XOR'd met de linkerhelft van de data. Het resultaat van deze XOR, een 32 bit blok, wordt nu het rechterblok in de volgende ronde en omgekeerd.



Figuur 3.14: Binnenin de F-operatie.

In het F-blok wordt eerst het 32-bit blok uitgebreid (*E* in de figuur, van expansie) naar een 48 bit blok zodat deze even lang is als de subkey voor deze ronde. De expansie gebeurt, net als de initiële permutatie, volgens een vast patroon:



Figuur 3.15: De expansie van 32 naar 48 bits.

Nu worden deze 48 bits ge-XOR'd met de subkey. Het resultaat wordt in blokjes van 6 bits door een *S*-blok gestuurd (zogenaamde *Selection blocks*). In dit blokje wordt 6 bit omgezet naar 4 bit. In de figuur hieronder zien we bijvoorbeeld hoe de omzetting in blok *S5* gebeurt. Ieder blokje heeft een soortgelijke tabel, maar met andere resultaten. De 6 bits bestaan uit de 2 outer bits, namelijk de eerste en de laatste bit, alsook de 4 innerbits. De **outer bits vormen de rij-index** in de S-box-tabel, de **inner bits de kolom-index**. De figuur toont bijvoorbeeld dat de output `1001` zou zijn indien er `011011` in het blok wordt geplaatst: outer bits `01` wijzen naar rij 1, inner bits `1101` naar kolom 13, en op die positie in de S5-tabel staat de waarde `1001`.

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

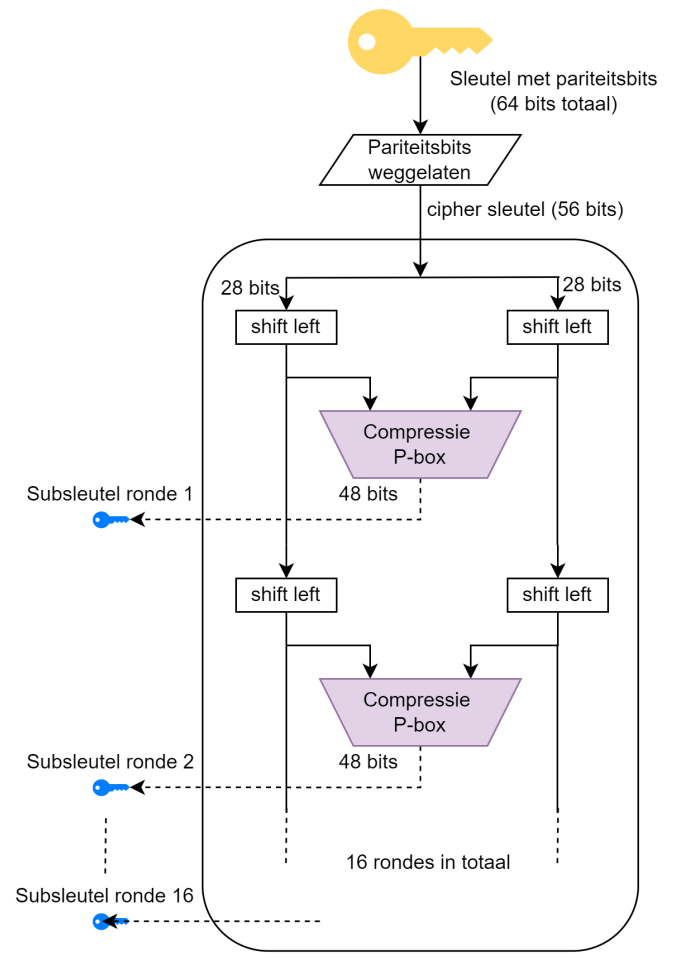
Figuur 3.16: Waarheidstabel van het S5-blok (Bron wikipedia).

Na 16 rondes krijgen we terug een 32 bit datablok dat nog een *Finale permutatie* ondergaat die weer de bits van plaats verandert en de output hiervan is een geëncrypteerd blok data dat kan doorgestuurd worden naar de ontvanger.

> Tip
>
> Je kan de DES standaard op web.archive.org/web/20040410171758/http://www.itl.nist.gov/fipspubs/fip46-2.htm nalezen en ontdekken dat deze niet zo lang is zoals je zou verwachten van een wereldwijd gebruikte standaard.

Subkeys maken

Iedere ronde tijdens de versleuteling vereist een sleutel. Om te voorkomen dat steeds de hoofdsleutel wordt gebruikt (en er zo potentieel dezelfde keystreams worden gemaakt), wordt deze sleutel doorheen een *round-key generator algoritme* gestuurd. Dit algoritme bestaat uit 16 rondes waarbij de 56 bit sleutel (8 van de 64 bits in de originele sleutel zijn zogenaamde pariteits-bits, die dienen om te controleren of de sleutel geen fouten bevat) steeds in twee helften van 28 bits wordt *geknipt*.



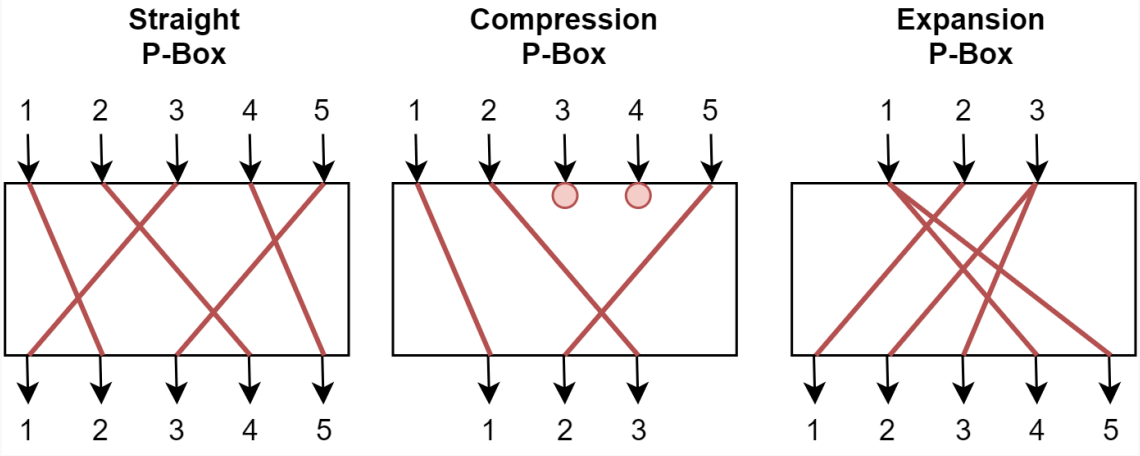
Figuur 3.17: De 16 rondes die telkens 1 subkey maken.

Iedere ronde wordt de helft van de sleutel 2 bits *geshift* (in ronde 1,2, 9 en 16 maar 1 bit). Dat wil zeggen dat alle bits twee plekjes opschuiven en de eerste (of laatste) bits komen dan achteraan (of vooraan) te staan. Deze twee geshifte helften worden dan enerzijds doorgestuurd naar de volgende ronde, anderzijds naar een *compressie P-box* waarvan het resultaat een 48 bits subkey zal zijn van die ronde.

De naam *compressie P-box* doet al vermoeden wat er gebeurt:

- Compressie: een aantal bits zullen wegvallen (er komen 56 bits in, maar we hebben maar 48 bits nodig).

- P-box: een permutatie oftewel transpositie dat alle bits van plek zal veranderen.



Figuur 3.18: Er zijn drie types P-Boxes, afhankelijk van wat ze met de data doen.

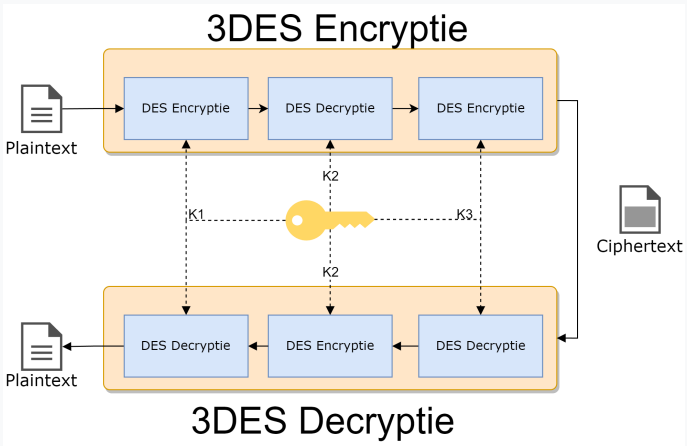
In de Compression P-Box wordt een aantal bits van de sleutel "tegengehouden". Welke bits dat zijn hangt af van de P-Box. Iedere ronde wordt er een andere Compression P-Box gehanteerd.

En zo hebben we het einde van de werking van DES bereikt. Dat viel al bij al nog mee, niet?

3.6.4.3 3DES

Al van bij de start gingen er stemmen op dat de originele sleutellengte voor DES (56 bits, 48 in effectiviteit vanwege de pariteitsbits) redelijk snel zou gebruteforced worden. Om die reden werd 3DES in het leven geroepen in 1995. De oplossing, 3DES, was een mooi staaltje compromisvorming: het bood een verhoogde beveiliging doordat het een lange sleutel had (tot 168 bits lang) maar bleef tegelijkertijd compatibel met de bestaande DES hardware en software.

De werking van 3DES (*triple DES*) is verrassend eenvoudig: ieder blok data wordt drie keer doorheen een DES-cipher gestuurd, telkens met een andere sleutel. Om compatibel te blijven met bestaande DES hardware wordt de data hierbij eerst *Encrypted*, dan *Decrypted*, en tenslotte opnieuw *Encrypted*. Dit patroon wordt daarom het **EDE-schema** (Encrypt-Decrypt-Encrypt) genoemd. Daar we in iedere fase een andere sleutel gebruiken, heeft dit (dankzij de eigenschappen van symmetrische ciphers) als effect dat we effectief drie maal na elkaar encrypteren met steeds een andere sleutel. Aan de ontvangerzijde gebeurt het omgekeerde — decryptie, encryptie, decryptie (DED) — en ook dit met dezelfde DES hardware!



Figuur 3.19: 3DES.

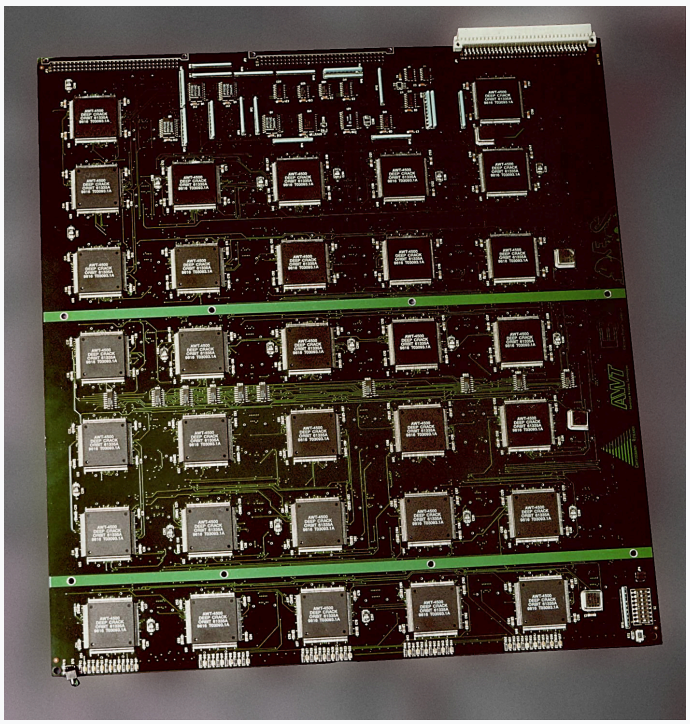
> **Tip**
>
> 3DES laat dus ook (single) DES encryptie toe. Het enige dat je hiervoor moet doen is de subsleutels K2 en K3 gelijkstellen waardoor de tweede en derde fase tijdens de encryptie (en decryptie) eigenlijk niets doen, daar het gewoon de data encrypteert in ronde twee en dan ogenblikkelijk in ronde drie terug decrypteert.

> **Opmerking**
>
> Het bankwezen gebruikt 3DES nog steeds (of varianten die erop gebaseerd) zijn om financiële transacties van onder andere Visa en Mastercard te beveiligen.

3.6.4.4 DES in de praktijk gekraakt: EFF Deep Crack

De zorgen om de korte 56-bit sleutel van DES werden in 1998 op spectaculaire wijze bevestigd. De *Electronic Frontier Foundation* (EFF) bouwde voor ongeveer $250.000 een gespecialiseerde machine, bijgenaamd **Deep Crack**, die bestond uit **1.856 custom chips** die speciaal ontworpen waren om DES-sleutels te testen.

Het resultaat? Deep Crack kon iedere willekeurige DES-sleutel in een **kwestie van dagen** bruteforcen. Een paar jaar later deed het project, in samenwerking met distributed.net, een DES-sleutel zelfs in minder dan 24 uur. Daarmee werd ondubbelzinnig aangetoond dat DES niet langer veilig was voor gevoelige data — exact wat critici al jaren beweerden.



Figuur 3.20: Een Deep Crack circuit board met 64 custom DES-kraakchips. De volledige machine bevatte er 1.856. (Bron: Electronic Frontier Foundation / Matt Crypto, CC-BY 3.0).

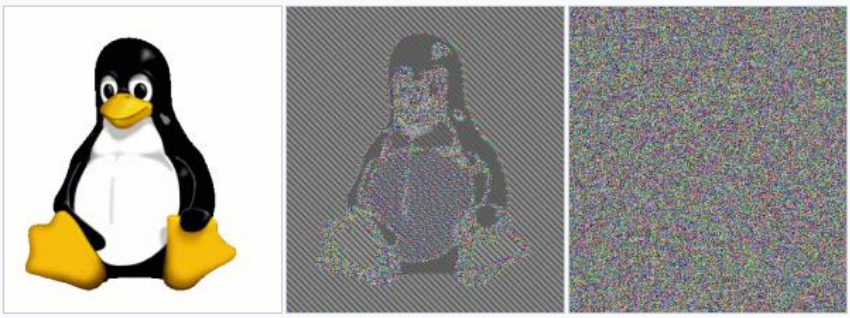
Dit was eigenlijk geen verrassing voor wie had opgelet: de NSA had bij de originele Lucifer-standaard de sleutellengte van 128 bits naar 56 bits *laten* halveren. Een toevalligheid? Critici waren al jaren overtuigd van niet. Deep Crack bevestigde hun vermoeden op de meest publieke manier mogelijk en gaf zo ook retroactief gelijk aan de keuze voor 3DES in 1995.

> **i** Opmerking
>
> Bij de publieke aankondiging werd een versleuteld bericht gekraakt in 56 uur. Het bericht? *"It's time for those 128-, 192-, and 256-bit keys."* Een niet zo subtiele boodschap aan de industrie om over te schakelen naar sterkere algoritmes. In 2001 gebeurde dat effectief, met de komst van AES (zie verder).

3.6.4.5 Block cipher modes

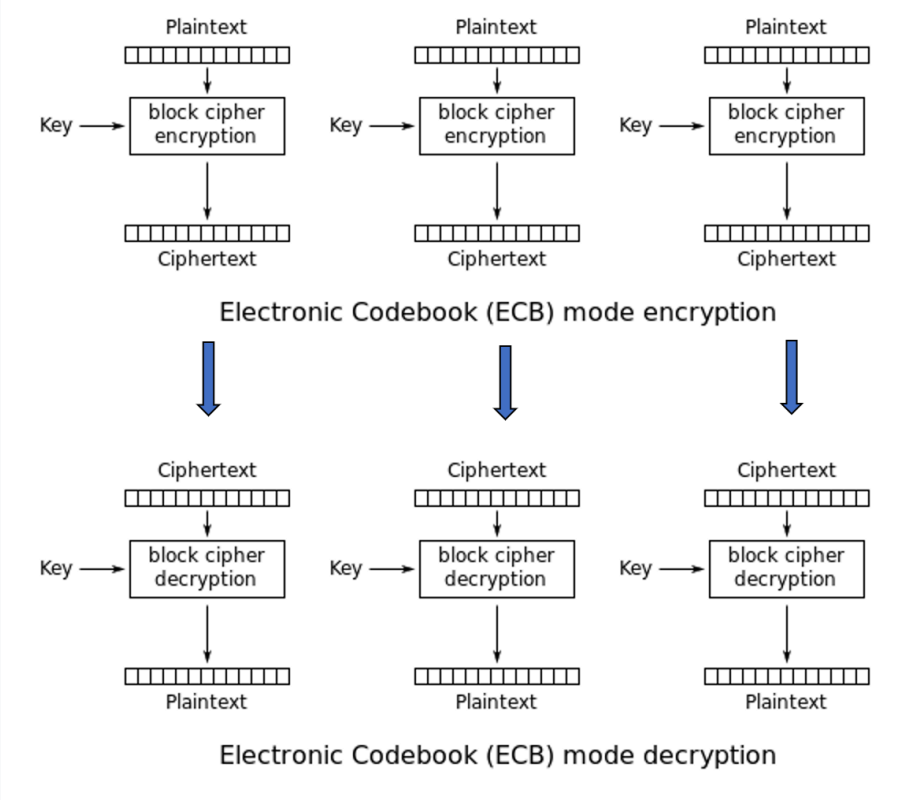
Tot hiertoe gingen we data steeds blok per blok in het encryptiecipher sturen en het resultaat ervan doorsturen. Klaar. Oplettende mensen hebben hier mogelijk al een hiaat in gezien: wat als twee blokken exact dezelfde data bevatten? Beiden zullen dezelfde ciphertext als resultaat genereren, daar we telkens dezelfde sleutel (en dus subkeys) gebruiken. Twee ciphertexts die identiek zijn, willen we vermijden daar het potentiële informatie over de plaintext zichtbaar maakt. Voorts laat dit soort werking ook replay attacks toe: de aanvaller kan een geëncrypteerd pakket bewaren en op een later moment terug opsturen, zonder dat hij moet weten wat de plaintext bevat.

Stel dat we een afbeelding van Tux De Pinguïn opsplitsen in ongeveer 100 bij 100 datablokken. Als we nu ieder blok individueel met een blockcipher encrypten en zouden visualiseren dan krijgen we iets dat mogelijks toch nog wat informatie van Tux *doorlekt* (zie de tweede afbeelding) daar blokken van de afbeelding met exact dezelfde informatie ook dezelfde ciperblock zullen genereren. Vergelijk dit met de derde afbeelding waarin we een andere modus gebruiken (die we zo meteen gaan uitleggen) waarin repetities in de plaintext geen invloed hebben op repetities in de ciphertext.



Figuur 3.21: Het volgend voorbeeld toont een (overdreven) manier waarom ECB minder veilig is dan de modes die we nog gaan behandelen (Bron wikipedia).

Voorgaande modus, waarin we ieder blok onafhankelijk van het vorige encrypteren, noemen we de **Electronic Codebook (ECB)** modus.



Figuur 3.22: ECB mode (Bron wikipedia).

Alhoewel deze modus dus duidelijk een veiligheidsprobleem met zich mee draagt, heeft deze modus ook één voordeel:

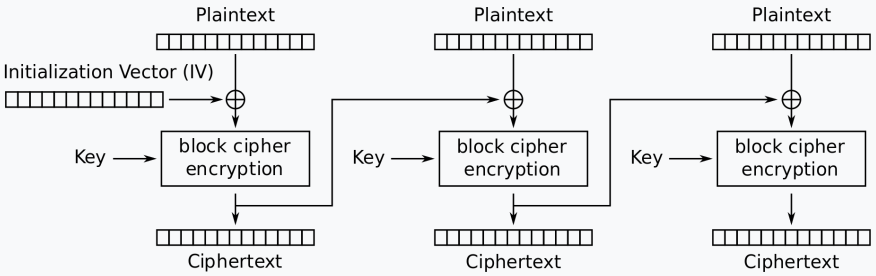
- Ieder blok wordt onafhankelijk van andere blokken gedecrypteerd. Als er dus een blok niet gedecrypteerd kon worden door een fout, dan heeft dat geen invloed op de daaropvolgende blokken. Dit is dus voor streaming-situaties nuttig: beeld je in dat je decryptie faalt halverwege het binnenkrijgen van een film die je aan het bekijken bent. Je zou helemaal opnieuw moeten beginnen.

ECB is een niet zo veilige manier om een blockcipher toe te passen. Veel interessanter (veiliger) wordt het wanneer we extra informatie gebruiken om een blok te encrypteren. **Enkel het huidige blok en dezelfde sleutel gebruiken is namelijk niét veilig.**

Er zijn verschillende modes om veiliger te encrypteren dan ECB. We bespreken hieronder de belangrijkste.

CBC (Cipher Block Chaining)

Bij CBC wordt de output van het vorige blok (de ciphertext) mee als input voor de encryptie van het volgende blok gebruikt. Concreet wordt de ciphertext van het vorige blok ge-XOR'd met het huidige plaintext-blok, vóór de encryptie plaatsvindt. Hierdoor is de encryptie van elk blok afhankelijk van alle voorgaande blokken, wat patronen in de plaintext verbergt.



Figuur 3.23: CBC encryptie (Bron wikipedia).

Laten we dit concreet maken met een minimalistisch voorbeeld. Stel dat ons blockcipher maar 2 bits per blok verwerkt, volgens volgende substitutietabel:

Input	Output
00	01
01	10
10	11
11	00

Onze plaintext is `00 01 10 11` (vier blokken), onze IV is `10`. De CBC-encryptie verloopt dan als volgt:

Blok	Plaintext	XOR met	Na XOR	Door cipher	Ciphertext
1	`00`	IV=`10`	`10`	⟶ `11`	**`11`**
2	`01`	`11`	`10`	⟶ `11`	**`11`**
3	`10`	`11`	`01`	⟶ `10`	**`10`**
4	`11`	`10`	`01`	⟶ `10`	**`10`**

De uiteindelijke ciphertext wordt `11 11 10 10`. Merk op dat onze vier plaintext-blokken allemaal verschillend waren (`00`, `01`, `10`, `11`), maar dat er in de ciphertext toch herhalingen verschijnen (`11 11` en `10 10`). Dat is exact wat we willen: repetities in de ciphertext dragen géén informatie meer over repetities in de plaintext. De link tussen patronen in plaintext en ciphertext is doorgeknipt.

CFB (Cipher Feedback)

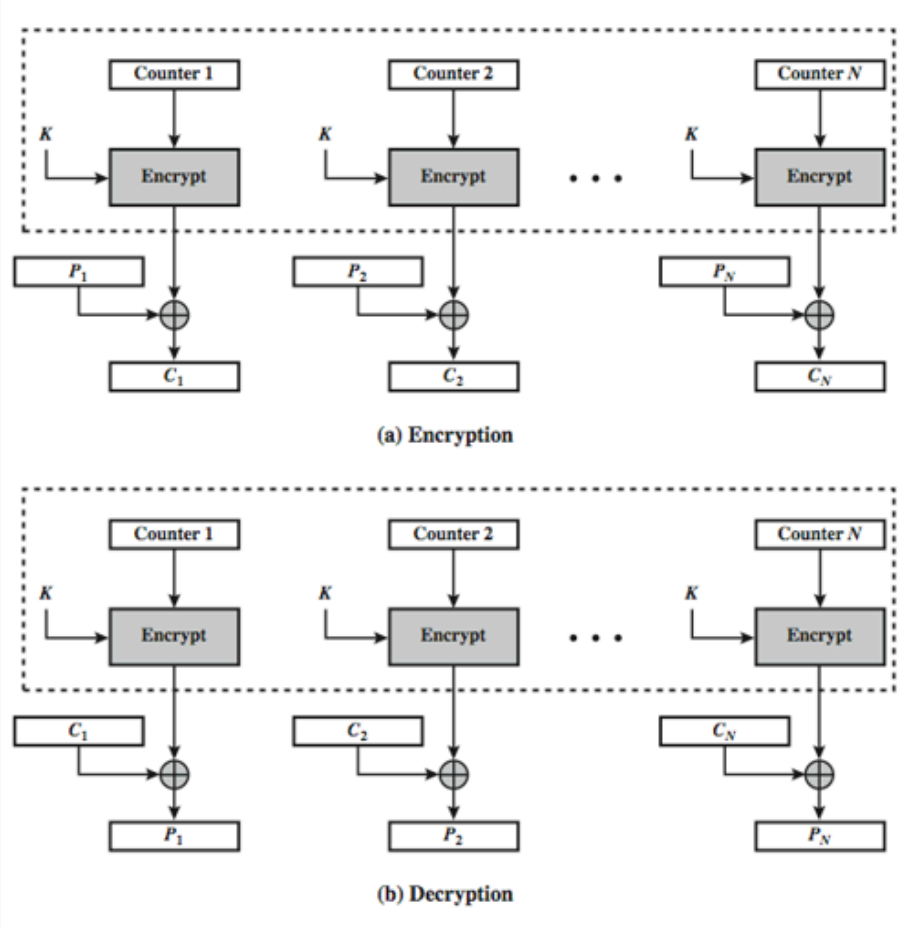
CFB werkt vergelijkbaar met CBC, maar de ciphertext van het vorige blok wordt iets later in het encryptieproces van het volgende blok gebruikt. Het vorige ciphertext-blok wordt eerst door het blockcipher gestuurd en daarna ge-XOR'd met de plaintext.

OFB (Output Feedback)

Bij OFB wordt het blockcipher als een streamcipher gebruikt: de output van het blockcipher wordt telkens opnieuw als input voor het blockcipher gebruikt om een keystream te genereren. Deze keystream wordt vervolgens ge-XOR'd met de plaintext.

CTR (Counter Mode)

CTR is één van de meest gebruikte modes en werkt fundamenteel anders dan CBC of CFB. In plaats van blokken aan elkaar te ketenen, wordt een **teller** (counter) als input voor het blockcipher gebruikt. Deze teller bevat een **Initialisatie Vector (IV)** die bij elk volgend blok met 1 wordt verhoogd.



Figuur 3.24: CTR mode (Bron wikipedia).

De werking is als volgt: het blockcipher encrypteert niet de plaintext zelf, maar de tellerwaarde. Het resultaat hiervan wordt vervolgens ge-XOR'd met het plaintext-blok om de ciphertext te bekomen. Doordat elk blok een unieke tellerwaarde gebruikt, levert dezelfde plaintext in verschillende blokken steeds andere ciphertext op.

Een groot voordeel van CTR is dat blokken **parallel** verwerkt kunnen worden: de encryptie van blok 5 is volledig onafhankelijk van blok 4, aangezien enkel de tellerwaarde en de sleutel nodig zijn. Dit maakt CTR bijzonder geschikt voor toepassingen waar snelheid belangrijk is, zoals bij het streamen van video (denk aan Netflix). CTR wordt onder andere gebruikt in **WPA2** en **IPSEC**.

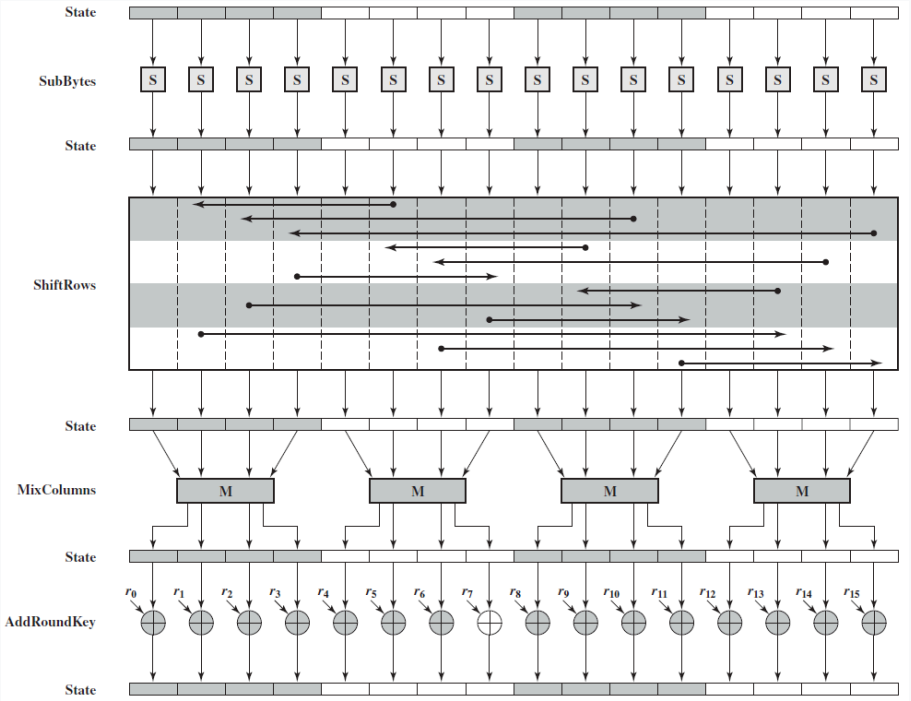
Overige modes

Naast bovenstaande modes bestaan er nog andere, zoals **Propagating CBC (PCBC)**, waarbij bij decryptie van een blok ook alle vorige blokken vereist zijn. Alle modes uit de doeken doen is hier niet aan de orde, maar het moge duidelijk zijn dat ECB de minst veilige mode is en deze best vermeden wordt.

> Tip
>
> Het concept **Initialisatie Vector (IV)** zal je veel zien terugkomen in ciphers. Een IV is een getal dat men als extra seed meegeeft tijdens de encryptie, naast de sleutel. Op deze manier voorkomen we dat steeds enkel de sleutel als seed wordt gebruikt en we dus effectief steeds met een *andere* sleutel werken. Uiteraard zal ook de andere zijde over dezelfde IV moeten beschikken en zal deze dus doorgestuurd moeten worden. Dit gebeurt meestal via de header van het bijhorende pakketje en is ongeëncrypteerd. Dit lijkt contra-intuïtief - de IV onbeveiligd doorsturen - maar is geen probleem. Uiteraard is het belangrijk dat er een goed *IV selectie algoritme* wordt gebruikt dat bepaalt hoe steeds het volgende IV moet worden berekend (bv. steeds met 1 verhogen, een willekeurig, etc.).

3.6.4.6 AES

Alhoewel 3DES een verbetering op DES was, was er toch nood aan een nieuwe encryptie-standaard die langere tijd kon bestaan. In 2001 werd daarom de **Advanced Encryption Standard (AES)** boven het doopvont gehouden als de nieuwe de facto encryptiestandaard wereldwijd. Deze Amerikaanse standaard is gebaseerd op het **Rijndael**-algoritme waar we als Belgen fier op mogen zijn: Rijndael is ontwikkeld door twee Belgische KUL-cryptografen Vincent Rijmen en Joan Daemen.



Figuur 3.25: AES encryptie (Bron wikipedia).

AES is een symmetrisch blockcipher dat data in blokken van 128 bits zal opsplitsen en sleutels tot 256 bits lang toelaat. De volledige werking van AES gaan we hier niet uit de doeken doen, het voldoet te begrijpen dat in grote lijnen hetzelfde soort stappen worden doorlopen als DES en andere symmetrische ciphers:

- **KeyExpansion**: uit de hoofdsleutel worden alle round keys afgeleid (één per ronde, net zoals bij DES).
- De data wordt vervolgens doorheen **10, 12 of 14 rondes** gestuurd, afhankelijk van de sleutellengte (respectievelijk 128, 192 of 256 bits). Het te encrypteren blok wordt daarbij voorgesteld als een 4×4-matrix van bytes, de zogenaamde *state*.
- Iedere ronde bestaat uit vier vaste stappen op die state:
  - **SubBytes** — iedere byte wordt via een vaste S-box vervangen door een andere byte (*substitutie*).
  - **ShiftRows** — de rijen van de state worden cyclisch verschoven (*transpositie*).
  - **MixColumns** — binnen iedere kolom worden de bytes met elkaar gemengd via een lineaire transformatie (zorgt voor *diffusie*). Deze stap valt weg in de allerlaatste ronde.
  - **AddRoundKey** — de state wordt ge-XOR'd met de round key van die ronde. Dít is waar de sleutel opnieuw in de encryptie binnenkomt.

Merk op dat ook hier, in de AddRoundKey-stap, de XOR-functie nog steeds dienst zal doen als de feitelijke encryptie van de data. Zonder deze XOR-functie zou al het voorgaande enkel maar resulteren in data die van plek verandert, volgens een patroon waar de geheime sleutel niet bij van te pas komt.

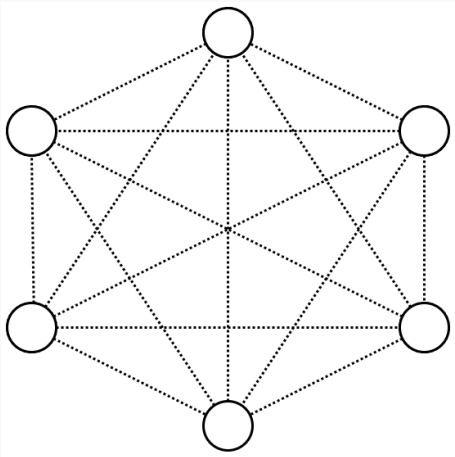
3.7 Asymmetrische encryptie

3.7.1 Het probleem met symmetrische encryptie

Wat als je bij symmetrische encryptie met meerdere mensen wilt communiceren zonder dat iedereen elkaars berichten kan zien? Bob kan onmogelijk dezelfde sleutel gebruiken om met Alfredo te communiceren die hij al gebruikte met Alice. Kortom, je hebt per *eindpunt* een aparte sleutel nodig. Het aantal

sleutels dat je nodig hebt, zeker als ook alle gebruikers onderling nog eens willen communiceren wordt snel erg groot. Je kan het aantal benodigde sleutels berekenen met de formule $n * \frac{(n-1)}{2}$, waarbij n het aantal gebruikers voorstelt:

- 6 gebruikers vereisen 15 sleutels.
- 7 gebruikers vereisen 21 sleutels.
- 10 gebruikers vereisen er al 45.
- 100 gebruikers vereisen er 4950!



Figuur 3.26: Bij 6 gebruikers zijn er al 15 sleutels nodig.

Symmetrische encryptie heeft dus een *key distribution problem* wanneer er encryptie op een grote schaal nodig is. Zeker als we spreken over online communicatie, over het Internet, wordt de schaal ogenblikkelijk gigantisch groot en wordt het *sleutelmanagement* problematisch. Een andere oplossing is dus aan de orde.

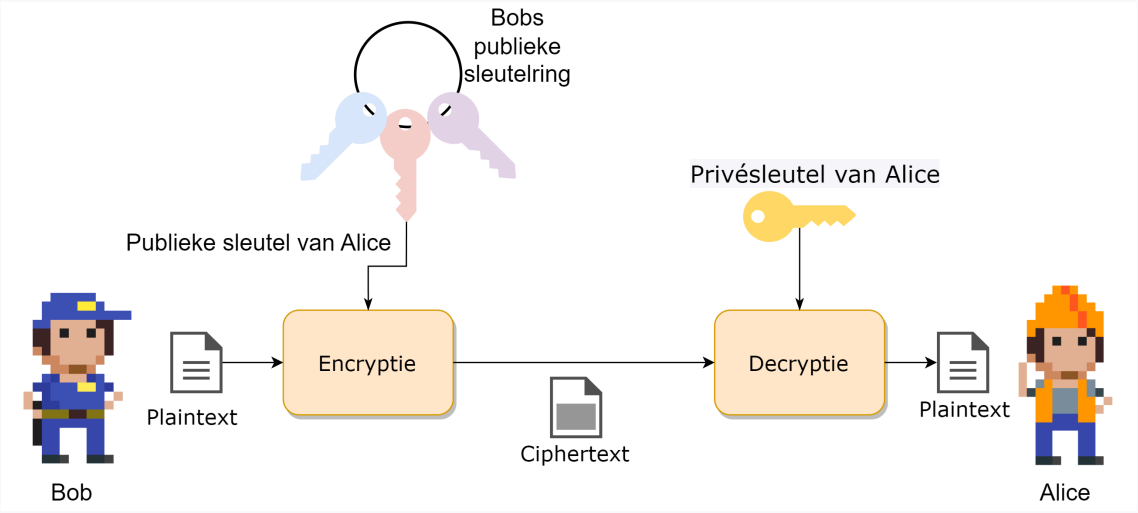
3.7.2 Publieke cryptografie

Publieke crypto oftewel **asymmetrische encryptie** zal het probleem met symmetrische encryptie oplossen doordat er gebruikt wordt gemaakt van twee sleutels:

- Eén publieke sleutel.
- Eén private sleutel.

Iedereen kan de publieke sleutel gebruiken om versleutelde berichten naar de eigenaar te sturen. Enkel de eigenaar van de bijhorende private sleutel kan deze berichten echter decrypteren. Kortom, we lossen nu een deel van het sleutelprobleem op: iedereen heeft z'n eigen private sleutel (**en moet deze geheim houden!**) en kan via z'n publieke sleutel berichten ontvangen.

Het concept van publieke cryptografie werd in 1976 gepubliceerd door Diffie en Hellman en had een grote impact op de manier waarop beveiligde communicatie op het Internet mogelijk werd. Hun eigen artikel beschreef meteen een sleuteluitwisselingsprotocol (Diffie-Hellman, zie verder); het eerste volwaardige asymmetrische encryptie-algoritme (RSA) volgde een jaar later, in 1977.



Figuur 3.27: Publieke crypto: Bob gebruikt de publieke sleutel van Alice om haar een beveiligd bericht te sturen.

> **Tip**
>
> Je kan publieke encryptie beschouwen als volgt: de publieke sleutel, die niet geheim is, is een openstaande kist. Iedereen kan de kist gebruiken om een geheime boodschap in te plaatsen en vervolgens deze op slot te klikken (*encrypteren*). Enkel de eigenaar van deze kist heeft de bijpassende private sleutel die echter deze kist kan opendoen en de originele boodschap kan lezen (*decrypteren*).

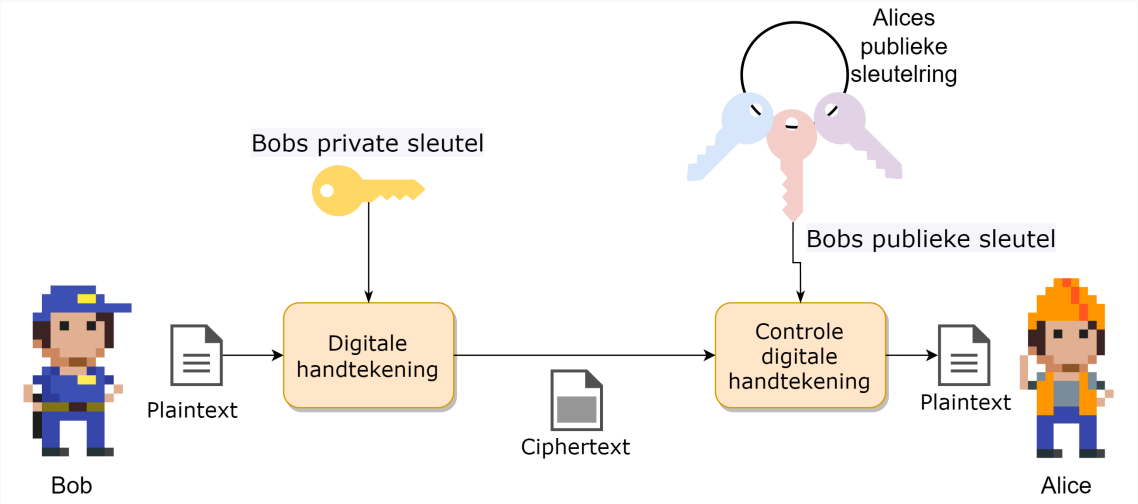
> **Opmerking**
>
> Enkele van de bekendere publieke cryptosystemen zijn onder andere RSA, DSS en El Gamal.

3.7.3 Drie toepassingen van publieke cryptografie

Publieke cryptografie wordt in de praktijk voor drie doeleinden ingezet:

- **Asymmetrische encryptie** (bv. RSA): berichten versleutelen met de publieke sleutel van de ontvanger, die ze enkel met zijn private sleutel kan lezen.
- **Sleuteluitwisseling** (*key exchange*, bv. Diffie-Hellman): twee partijen spreken over een onveilig kanaal een gemeenschappelijke symmetrische sleutel af, die daarna voor snellere symmetrische encryptie wordt gebruikt.
- **Digitale handtekeningen**: met de private sleutel wordt een bericht "getekend", en iedereen kan met de bijhorende publieke sleutel verifiëren dat het bericht effectief van die persoon afkomstig is en onderweg niet werd aangepast.

Die laatste toepassing is een erg nuttige extra eigenschap: daar enkel de ondertekenaar de private sleutel in z'n bezit heeft, levert een geldige handtekening tegelijk **authenticatie** (de verzender is wie hij beweert te zijn) én **integriteit** (het bericht is onderweg niet gewijzigd).



Figuur 3.28: Het ondertekenen van een document met behulp van je private sleutel.

In wat volgt werken we deze drie toepassingen één voor één uit, te beginnen met sleuteluitwisseling.

3.7.4 Diffie-Hellman sleuteluitwisseling

We starten met **sleuteluitwisseling**, omdat dit concept mooi illustreert hoe publieke crypto in de praktijk wordt gebruikt.

Het is namelijk zo dat symmetrische crypto sneller is én dus voor (realtime) communicatie interessanter is. We weten echter dat het sleutelmanagement bij symmetrische crypto een probleem is als we met grote groepen gebruikers zitten. Het Diffie-Hellman sleuteluitwisselingsconcept helpt ons hierbij: het laat toe dat twee gebruikers over een onveilig kanaal op een veilige manier een gemeenschappelijk geheim (*shared secret*) afspreken.

Belangrijk om meteen mee te geven: de waarden die Bob en Alice bij Diffie-Hellman gebruiken zijn **sessie-specifiek** (*ephemeral*) en worden na de uitwisseling weggegooid. Ze zijn dus niet hetzelfde als een langdurig publiek/privaat sleutelpaar zoals bij RSA — DH dient louter om een gemeenschappelijke symmetrische sleutel af te spreken, niet om berichten rechtstreeks te versleutelen.

Voor de eigenlijke uitwisseling start, spreken Alice en Bob eerst twee **publieke parameters** af: een (groot) priemgetal p dat als modulus dient, en een **basis** g (ook wel *generator* genoemd), kleiner dan p. Deze twee waarden mogen over een onveilig kanaal worden uitgewisseld — een eventuele afluisteraar mag ze gerust kennen, dat is geen probleem voor de veiligheid. In het rekenvoorbeeld hieronder gebruiken we $g = 7$ en $p = 11$.

Zowel Bob als Alice kiezen vervolgens elk een **geheime waarde** die ze enkel voor deze sessie gebruiken. Op basis van deze geheime waarde berekenen ze elk een **publieke waarde** (via de formule $g^{geheim} \bmod p$) die ze naar de ander sturen (wat kan over een onbeveiligd kanaal). De ontvanger zal deze publieke waarde combineren met de eigen geheime waarde wat zal resulteren in een *shared secret* dat beiden nu kennen en kunnen gebruiken, bijvoorbeeld als de symmetrische sleutel om verdere communicatie te beveiligen.

De reden dat dit werkt, is met dank aan de modulo operator en de eigenschappen ervan. Een voorbeeld (met $g = 7$ en $p = 11$):

Stap	Alice	Bob
1	Alice kiest een geheim getal A. Ze kiest A= 3	Bob kiest ook een geheim getal B = 6.
2	Alice berekent $7^A \bmod 11 = 343 \bmod 11 = 2$, genaamd X.	Bob berekent $7^B \bmod 11 = 117649 \bmod 11 = 4$, genaamd Y.
3	Alice stuurt X=2 naar Bob	Bob stuurt Y=4 naar Alice.
4	Alice berekent $Y^A \bmod 11 = 4^3 \bmod 11 = 9$.	Bob berekent $X^B \bmod 11 = 2^6 \bmod 11 = 9$.

Zoals je merkt kunnen nu Alice en Bob het berekende getal **9** als gedeeld geheim gebruiken. Enkel zij kennen dit getal.

⚠ Waarschuwing

Uiteraard zullen in de praktijk Bob en Alice véél grotere getallen kiezen dan 3 en 6.

i Opmerking

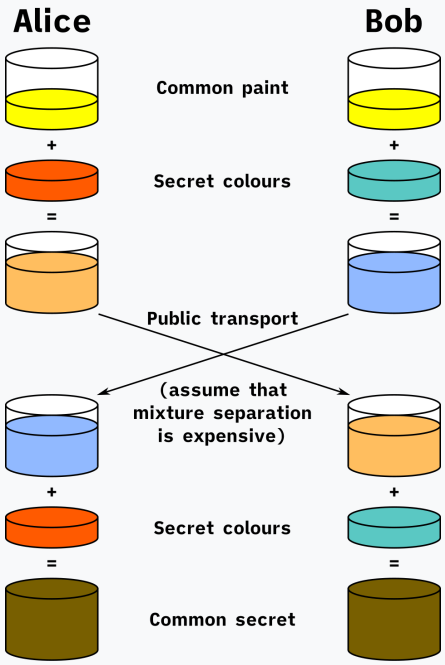
Dat Bob en Alice de waarden X en Y naar elkaar kunnen sturen is dankzij de eigenschappen van de modulo-berekening. X en Y kunnen het resultaat zijn van een gigantische hoeveelheid berekeningen en een stroper zal dus veel rekenwerk nodig hebben om alle mogelijkheden te testen.

Een intuïtieve metafoor: verfkleuren mengen

Je kan Diffie-Hellman visueel begrijpen via verfkleuren:

1. Alice en Bob spreken **publiek** een gemeenschappelijke startkleur af (bijvoorbeeld geel).
2. Elk kiest daarnaast een **geheime** kleur die nooit gedeeld wordt (bijvoorbeeld Alice oranje, Bob turquoise).
3. Beiden mengen hun geheime kleur met de gemeenschappelijke kleur en sturen het mengsel naar de andere partij. Een stroper kan de mengsels onderweg zien, maar kan ze niet ontleden in de originele kleuren.
4. Alice voegt haar geheime kleur toe aan Bobs mengsel; Bob doet hetzelfde met het mengsel van Alice.
5. Beiden komen uit op exact **dezelfde eindkleur**: het gedeeld geheim.

Het principe berust erop dat *mengen* makkelijk is, maar *ontmengen* praktisch onmogelijk. In de digitale versie vervult de modulo-berekening die rol.



Figuur 3.29: De verfkleurenmetafoor voor Diffie-Hellman. Bron: Wikimedia Commons (CC BY-SA).

3.7.5 RSA

Eén van de oudste, maar nog steeds populairste, publieke cryptosystemen is het in 1977 ontwikkelde RSA algoritme. RSA, wat staat voor de achternamen van de drie ontwikkelaars (Rivest, Shamir en Adleman) gebruikt sleutels van 1536 tot 4096 bits lang. Het systeem is vrij traag maar heeft als voordeel dat het veilige

sleuteltransmissie toestaat over een onveilig kanaal: we zien daarom vaak RSA gebruikt worden om eerst sessiesleutels uit te wisselen, vervolgens wordt overgeschakeld op een sneller symmetrisch cipher.

De exacte berekeningen die gebeuren tijdens encryptie en decryptie leiden ons iets te ver, maar volgend voorbeeld toont een vereenvoudigde wijze waarop RSA wordt toegepast:

Data encrypteren met behulp van asymmetrische versleuteling gebeurt op bijna dezelfde wijze als de Diffie-Hellman sleutel uitwisseling. Ook nu zullen beide zijden rekenen op de eigenschappen van de modulo-operator om over een onveilig kanaal veilige communicatie te kunnen doen.

Eerst dient een publieke sleutel aangemaakt te worden:

- Hiertoe dient Bob twee grote priemgetallen, `q` en `p` te kiezen, bijvoorbeeld `p=17` en `q=11`.
- Vervolgens berekent Bob `N` door deze priemgetallen met elkaar te vermenigvuldigen (`p*q` geeft 17 * 11, `N=187`).
- Bob kiest nu nog een priemgetal `e`, bijvoorbeeld `7`.
- Bob kan nu zijn eigen geheime, private sleutel `d` maken, zodat geldt: $e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$. In dit voorbeeld wordt dat $7 \cdot d \equiv 1 \pmod{16 \cdot 10}$, oftewel $7 \cdot d \equiv 1 \pmod{160}$.
- Om `d` te vinden zoeken we dus een getal zodat $7 \cdot d \bmod 160 = 1$. De mogelijke waarden voor $7 \cdot d$ zijn bijgevolg $1, 161, 321, \ldots$. Met $7 \cdot 23 = 161$ klopt het: `d=23`.

> **Tip**
>
> Om d te berekenen maken we gebruik van de zogenaamde *Uitgebreid Euclidisch algoritme*, een eeuwenoud algoritme gebaseerd op het *Algoritme van Euclides* dat we in het lager leerden gebruiken om de grootste gemene deler te berekenen van twee getallen.

Bob heeft dus nu:

- Publieke sleutel bestaande uit $N = 187$ en $e = 7$.
- Private sleutel $d = 23$.

Iedereen die nu naar Bob iets wilt sturen, kan dit via z'n publieke sleutel (`N` en `e`).

Stel dat Alice het ASCII-karakter X naar Bob wil sturen:

- De ASCII-waarde van X is 88.
- De encryptie door Alice gebeurt dan als volgt: $C = \text{data}^e \bmod N$.
- De te versturen ciphertext C wordt dus: $88^7 \bmod 187$ oftewel `C=11`.

Enkel Bob zal deze ciphertext met zijn private sleutel `d` kunnen decrypteren door $C^d \bmod N$ te doen, oftewel $11^{23} \bmod 187$ wat terug de plaintext `88` geeft!

> **Tip**
>
> De sterkte van publieke crypto stoelt dus op het feit dat ontbinden van (grote) getallen in factoren computationeel veel moeilijker is dan de omgekeerde stap, namelijk twee getallen met elkaar vermenigvuldigen.
>
> 15621 in z'n factoren ontbinden is veel moeilijker dan de getallen 123 en 127 vermenigvuldigen (wat dus ook 15621 zal geven).

Zonder in detail te treden hoe cryptocoins en blockchains werken, is het nuttig om te vermelden dat bij cryptocoins ook de public crypto concepten worden gebruikt. Ook hier is je private sleutel uiterst belangrijk: enkel de eigenaar van de private sleutel "bezit" de bijhorende cryptocoins in de *blockchain*. Daarom is het belangrijk dat je **NOOIT** je private sleutel aan derden geeft, want zo geef je hen toegang tot jouw *coins* en kunnen ze vervolgens deze stelen door de private sleutel te vervangen.

3.7.6 Elliptic Curve Cryptografie (ECC)

RSA baseert zich op de moeilijkheid van het ontbinden van grote getallen in priemfactoren. **Elliptic Curve Cryptografie** (ECC) is een modernere vorm van asymmetrische crypto die zich baseert op de wiskundige eigenschappen van **elliptische krommen over eindige velden**. Het onderliggende wiskundige probleem — het *Elliptic Curve Discrete Logarithm Problem* (ECDLP) — is nóg moeilijker op te lossen dan factorisatie, waardoor ECC met veel kleinere sleutels hetzelfde beveiligingsniveau kan bieden als RSA:

ECC sleutellengte	RSA equivalent	Beveiligingsniveau
256 bits	3072 bits	128 bits
384 bits	7680 bits	192 bits

Het **beveiligingsniveau** (*security strength*) drukt uit hoeveel rekenwerk een aanvaller nodig heeft om de encryptie te kraken, uitgedrukt in bits. Een beveiligingsniveau van 128 bits betekent dat een aanvaller 2^{128} bewerkingen moet uitvoeren — evenveel als nodig is om een symmetrische sleutel van 128 bits (zoals AES-128) te bruteforcen. Zo kunnen we de sterkte van verschillende cryptosystemen met elkaar vergelijken: een ECC-sleutel van 256 bits en een RSA-sleutel van 3072 bits zijn dus *even moeilijk te kraken.*

Kleinere sleutels betekenen snellere berekeningen, minder dataverkeer en lager energieverbruik. Dat maakt ECC bijzonder geschikt voor toepassingen waar rekenkracht of bandbreedte beperkt is, zoals **mobiele toestellen** en **IoT-apparaten**.

ECC wordt vandaag breed ingezet. De twee belangrijkste toepassingen zijn:

- **ECDH** (*Elliptic Curve Diffie-Hellman*): een variant van de eerder besproken Diffie-Hellman sleuteluitwisseling, maar dan gebaseerd op elliptische krommen.
- **ECDSA** (*Elliptic Curve Digital Signature Algorithm*): een digitaal handtekening-algoritme dat onder andere door Bitcoin en andere blockchains wordt gebruikt.

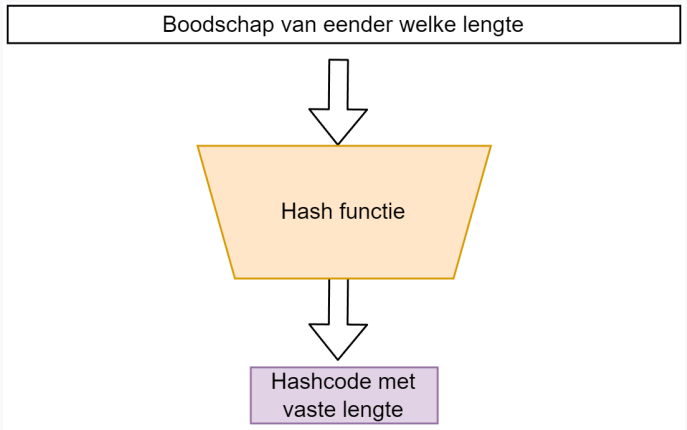
Moderne TLS-verbindingen (en dus HTTPS) gebruiken vrijwel altijd ECC-gebaseerde algoritmes voor de sleuteluitwisseling, omdat ze sneller en veiliger zijn dan klassiek RSA bij vergelijkbare sleutellengtes.

> ⚠ Waarschuwing
>
> Net als RSA is ook ECC kwetsbaar voor toekomstige **quantumcomputers**. Daarom wordt er actief gewerkt aan zogenaamde **post-quantum cryptografie**: nieuwe algoritmes die bestand zijn tegen aanvallen met quantumcomputers. In 2024 publiceerde NIST de eerste standaarden hiervoor.

3.7.7 Intermezzo: Hashes

Voor we de digitale handtekeningen uitwerken, slaan we even een zijtak in om het concept “hash” te bespreken. Een hash is een concept uit de informatica dat we gebruiken om te controleren of een digitaal stuk tekst werd aangepast of niet. Door de tekst in een hashfuntie te steken wordt een hash aangemaakt. Deze hash is een stuk code met een vaste lengte, ongeacht de originele input. Wanneer 1 bit of meer wordt aangepast in de originele boodschap dan zal deze in een totaal andere hash resulteren. Enkel dus wanneer een identiek stuk tekst als invoer (tot op bitniveau identiek) wordt gebruikt, zullen twee hashes gelijk zijn.

Voorgaande is uiteraard onmogelijk: daar een hash meestal veel korter is dan de originele boodschap, is het mathematisch mogelijk dat twee totaal verschillende teksten toch dezelfde hash geven. Het is de opdracht van een goede hashfunctie om dit soort **hash collisions** zo klein mogelijk te houden.



Figuur 3.30: Het hash proces.

Een hashfunctie is niet omkeerbaar: men mag onmogelijk aan de hand van een hash (ook wel *digest* of *hashcode* genoemd) terug de originele tekst kunnen achterhalen. Een hashfunctie is dus een eenrichtingsfunctie, ook wel afbeelding genoemd in wiskundige termen.

Er bestaan veel verschillende hashfuncties. Enkele van de bekendere zijn:

- MD5, oftewel Message Digest 5: deze zal een 128-bit hashwaarde genereren.
- SHA-X, oftewel Secure Hash Algorithms. Zo is er SHA-256 wat een 256 bits hash zal genereren.

De sterkte van een hash algoritme zit hem in de grootte van de kans waarop hash collisions kunnen optreden. Zo zijn er bij MD5 al veel meer collisions gevonden dan bijvoorbeeld bij het recenter gepubliceerde SHA3-512 algoritme.

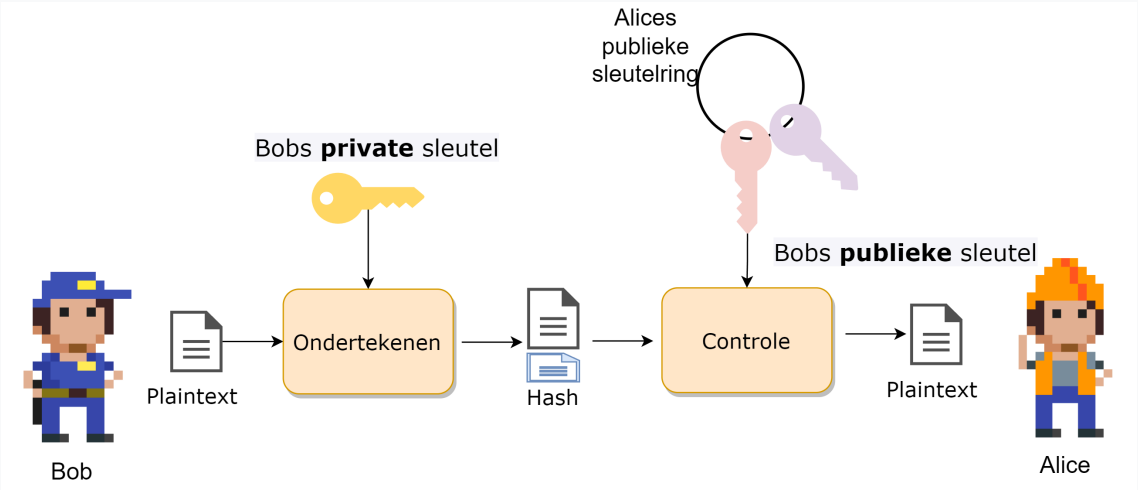
Een digitale hash is dus een ideaal middel om boodschappen digitaal te ondertekenen.

3.7.8 Boodschappen ondertekenen

Een probleem bij online communicatie is dat we geen zekerheid hebben dat het ontvangen bericht wel degelijk van de persoon komt van wie we verwachtten dat deze het bericht had opgesteld. We kunnen daarom het publieke crypto systeem gebruiken om berichten te ondertekenen. Daar enkel Bob de bijhorende private sleutel kan hebben die bij z'n publieke sleutel hoort, is het bezit van deze private sleutel hebben het bewijs dat hij de rechtmatige eigenaar van een bepaalde publieke sleutel is.

Onder andere RSA laat toe om een digitale handtekening (**digital signature**) te plaatsen bij een bericht om zo te bewijzen dat de verzender de rechtmatige eigenaar van een bijhorende publieke sleutel is.

Een digitale handtekening wordt als extra bericht achteraan de te versturen boodschap geplaatst. De signature is als het ware een hash berekend aan de hand van de private sleutel. De ontvanger zal nu met de bijhorende publieke sleutel van de verzender kunnen verifiëren of de bijhorende private sleutel werd gebruikt om de handtekening te genereren.



Figuur 3.31: Boodschappen ondertekenen met je private sleutel.

Om een digitale handtekening te berekenen moeten we eerst een hash van het bericht berekenen. We gebruiken hier bijvoorbeeld MD5 of één van de SHA-algoritmes voor. Deze hash gaan we nu "verpakken" met de private sleutel.

Stel dat de berekende hash van ons bericht de waarde 35 heeft (in werkelijkheid is een hash veel langer, maar voor dit voorbeeld houden we het klein) en we beschikken over volgend sleutelpaar (bron):

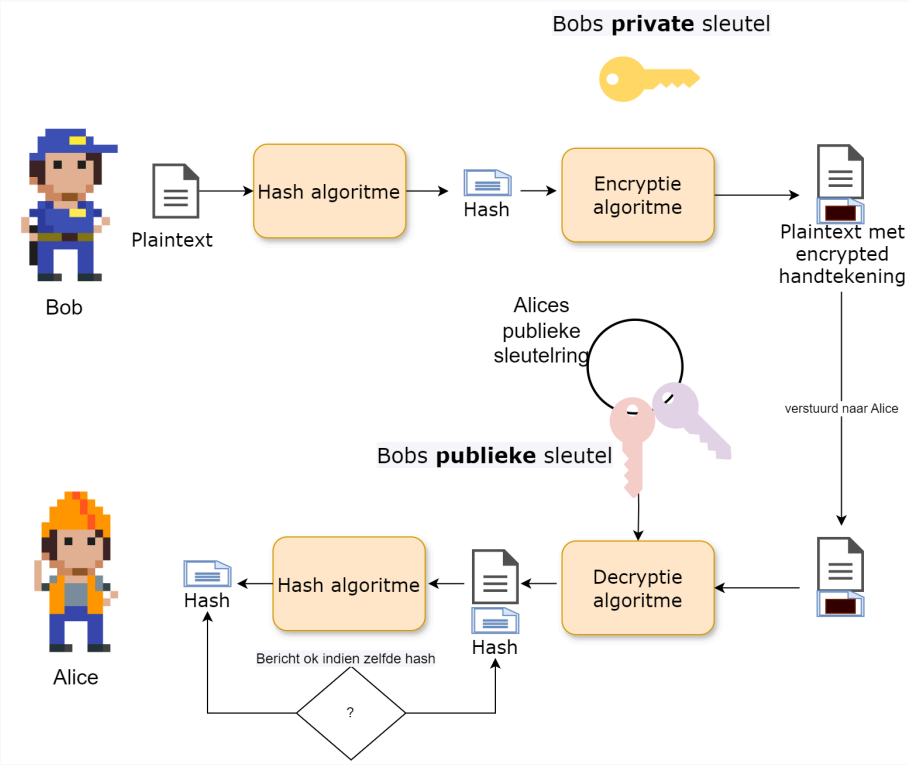
- publieke sleutel: $e = 5$ en $n = 91$.
- private sleutel: $d = 29$.

De handtekening wordt berekend door de hash te versleutelen met de private sleutel: $s = m^d \bmod n$, oftewel $s = 35^{29} \bmod 91 = 42$.

We versturen naar de ontvanger dus de boodschap zelf én de bijhorende handtekening 42.

De ontvanger kan nu controleren of de boodschap klopt. Hij berekent eerst zelf de hash van het ontvangen bericht. Vervolgens "ontsleutelt" hij de handtekening met de publieke sleutel van de verzender door $s^e \bmod n$ te berekenen, oftewel $42^5 \bmod 91 = 35$. Als de zelf berekende hash gelijk is aan dit

resultaat, dan weet de ontvanger dat het bericht afkomstig is van de verwachte verzender én onderweg niet werd aangepast.

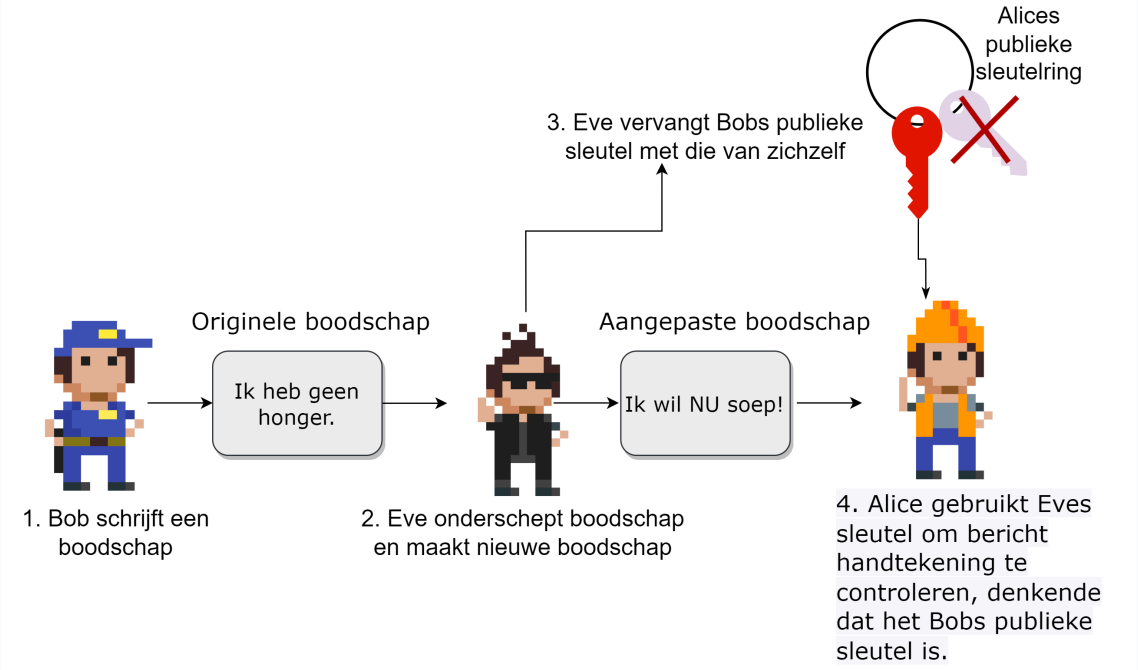


Figuur 3.32: Het volledig proces bij een digitale handtekening.

3.7.8.1 Het probleem met digitale handtekeningen

We hebben echter een probleem. Hoe weet je eigenlijk dat je wel de juiste publieke sleutel gebruikt? Publieke sleutels zijn, wel, publiek. Iedereen kan jou een publieke sleutel geven en zeggen *"Dit is de sleutel van persoon X"* zonder dat jij kan controleren of dat zo is.

We kunnen daarom als kwaadwillig persoon bijvoorbeeld een legaal bericht onderscheppen, aanpassen en dan vervolgens ondertekenen met onze eigen handtekening. Als we vervolgens aan de ontvanger kunnen wijsmaken dat jouw publieke sleutel zogezegd bij de originele verzender hoort, dan zal de ontvanger jouw aangepaste bericht "geloven".



Figuur 3.33: Eve misbruikt het vertrouwen dat zit ingebouwd in het digitale handtekening proces.

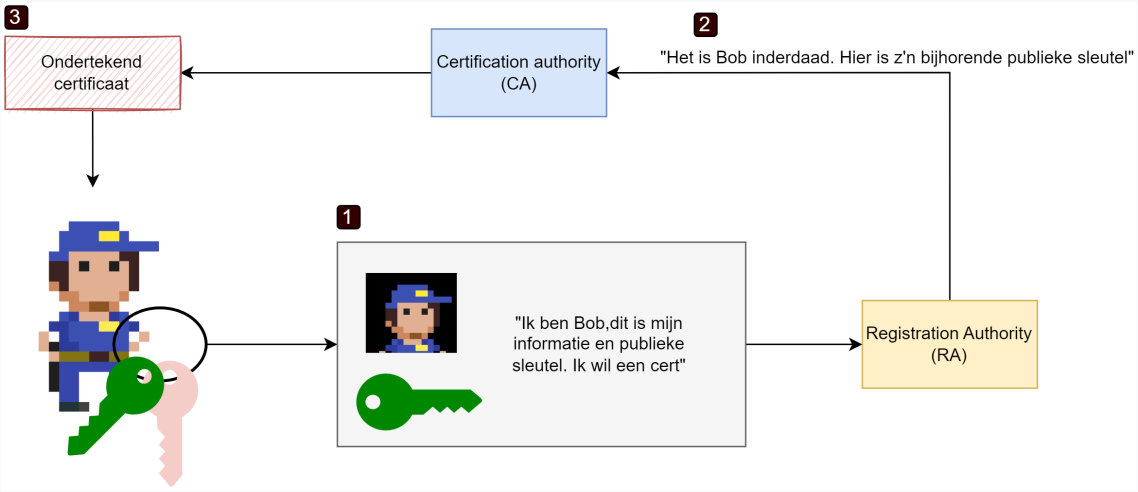
Kortom, we hebben een manier nodig om de **identiteit van de eigenaar** van een publieke sleutel te verifiëren. Kom binnen: **certificaten**.

3.8 Digitale certificaten

Een certificaat is een (digitaal) document dat de identiteit van een gebruiker bindt aan een publieke sleutel. Dit document werd digitaal ondertekend door een vertrouwde derde partij (**trusted third party**) zodat bij twijfel van de echtheid van het certificaat men altijd bij deze derde partij terecht kan. Uiteraard is het belangrijk dat we deze derde partij kunnen vertrouwen, anders kunnen we ook niet het certificaat vertrouwen die zij onderschrijven.

> Tip
>
> Certificaten worden beschreven in de **X.509** standaard.

Om een certificaat aan te maken dient Bob naar een **Registration authority** (RA) te gaan die zijn identiteit zal verifiëren. Dit gebeurt aan de hand van de typische documenten die ook buiten het Internet worden gebruikt om iemands identiteit te bewijzen: identiteitskaart, paspoort, rijbewijs, etc. In sommige gevallen zal de RA zelfs eisen dat Bob zich naar een fysiek kantoor begeeft om daar z'n identiteit *in the flesh* te bewijzen. Indien de RA de identiteit heeft bevestigd zal deze de aanvraag van Bob doorsturen naar een **Certification authority** (CA), inclusief Bobs publieke sleutel, die een certificaat zal aanmaken én ondertekenen.

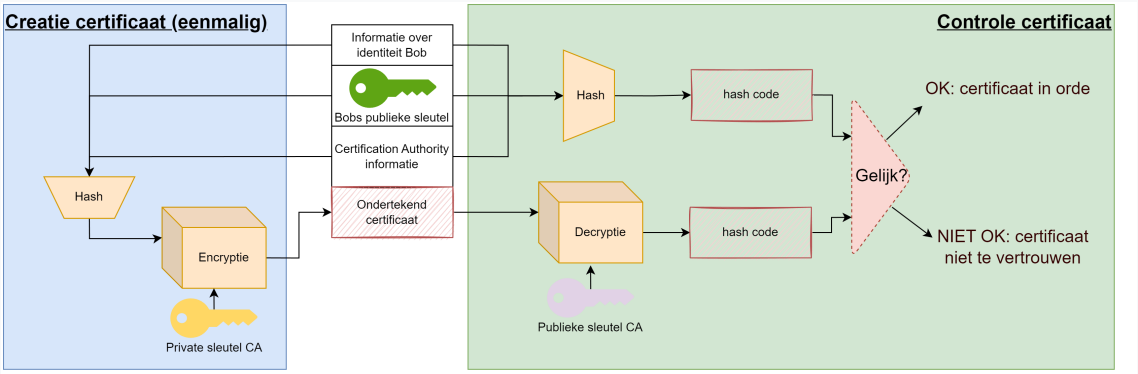


Figuur 3.34: Een certificaat registreren.

> **Tip**
>
> Het gehele systeem van CA's, RA's, etc. dat bestaat om certificaten uit te geven, beheren en bewijzen heet een **Public Key Infrastructure** (**PKI**).

De CA zal deze informatie gebruiken om een certificaat, van een bepaalde levensduur, te genereren. Hierbij zal de echtheid van het certificaat achteraf bewezen kunnen worden door de CA:

- Het certificaat bevat Bobs publieke sleutel en informatie over Bob en de CA in **leesbare vorm**. Daaraan wordt een **digitale handtekening** van de CA toegevoegd: een hash van al deze informatie, versleuteld met de private sleutel van de CA.
- Om later de echtheid van het certificaat te verifiëren, berekent de ontvanger zelf de hash van de inhoud van het certificaat en vergelijkt die met de ontsleutelde handtekening (die met de publieke sleutel van de CA wordt gedecrypteerd). Als beide hashes gelijk zijn, weten we dat het certificaat door de gegeven CA werd ondertekend — enkel de houder van de private sleutel van die CA kan zo'n geldige handtekening produceren.



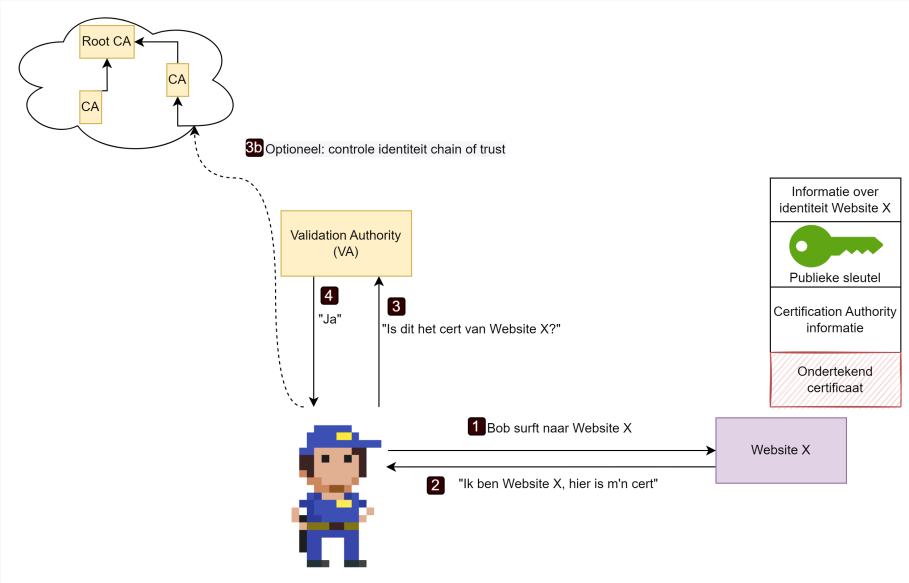
Figuur 3.35: Een certificaat aanmaken.

Een X.509-certificaat bevat minstens volgende velden:

- **Versie** van de X.509-standaard (vandaag doorgaans v3).
- **Serienummer**: uniek binnen de uitgevende CA.
- **Signature algorithm**: welk algoritme de CA gebruikte om te ondertekenen (bv. SHA-256 met RSA).
- **Issuer**: naam van de CA die het certificaat uitgaf.
- **Validity**: begin- en einddatum van geldigheid.

- **Subject**: naam van de eigenaar (domeinnaam of persoon).
- **Public key** van de eigenaar, met het gebruikte algoritme.
- **Extensions**: bijkomende info zoals gebruiksbeperkingen of alternatieve domeinnamen.
- **Signature**: de digitale handtekening van de CA over al het bovenstaande.

Voorgaande proces zal bijvoorbeeld plaatsvinden wanneer je browser via een **HTTPS** verbinding surft naar een website en zo wil controleren of wel degelijk met de website wordt gecommuniceerd en niet met een imposter. Indien de browser (of de gebruiker) twijfelt aan de echtheid van de publieke sleutel van de CA die het certificaat van de website ondertekent, dan zal het voorgaande proces zich herhalen, maar deze keer om het certificaat van de CA te controleren met behulp van een bovenliggende CA. Op die manier kan het dus zijn dat een keten van CA's ontstaan die telkens CA's onder zich bewijzen. Uiteraard zal er steeds bovenaan zo'n ketting een **root CA** staan. Als je die vertrouwt, dan kan je al de CA's er onder dus ook vertrouwen...maar ook vice versa!



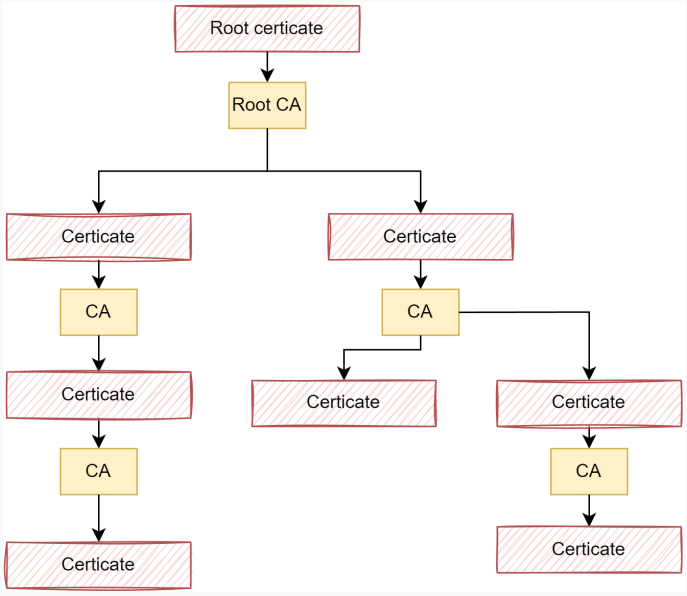
Figuur 3.36: Het certificaat tijdens het surfen.

i Opmerking

Alhoewel **HTTPS** al sinds 1995 bestond, werd het tot voor kort amper door websites aangeboden. Nochtans geeft HTTPS een extra defensielaag tijdens de communicatie van jouw computer met die waar een website op *gehost* staat. HTTPS zal namelijk je communicatie versleutelen zodat enkel zender en ontvanger kunnen lezen wat er gezegd wordt. Met HTTP is dat niet: al je communicatie kan door eender wie gelezen worden die zich tussen jouw computer en je eindbestemming nestelt. Het helpt echter niet dat je data versleuteld wordt als je niet kan bevestigen dat de ontvangende website ook effectief diegene is die je nodig hebt, vandaar dat dus certificaten en HTTPS in tandem werken om gebruikers een veiliger Internet aan te bieden.

Pas in 2017 boden meer dan de helft van de websites wereldwijd HTTPS aan. In 2021 gebruikt ongeveer 70% van alle websites HTTPS als standaard communicatiemiddel aan (vroeger waren er al websites met HTTPS, maar HTTP was de standaard oplossing).

Het ergste dat voor een CA kan voorvallen is dat de betrouwbaarheid van de CA in het gedrang komt. Als een CA bijvoorbeeld weet heeft van een potentiële inbraak op zijn systemen dan bestaat er de kans dat aanvallers de private sleutel van de CA hebben bemachtigd en dus zelf certificaten *op naam van de CA* kunnen genereren, met alle gevolgen van dien! Indien dus deze kans bestaat, is er een *breach of trust* en zullen alle certificaten van deze CA als ongeldig worden bestempeld, inclusief alle certificaten van sub-CA's! Dit kan verregaande gevolgen hebben.



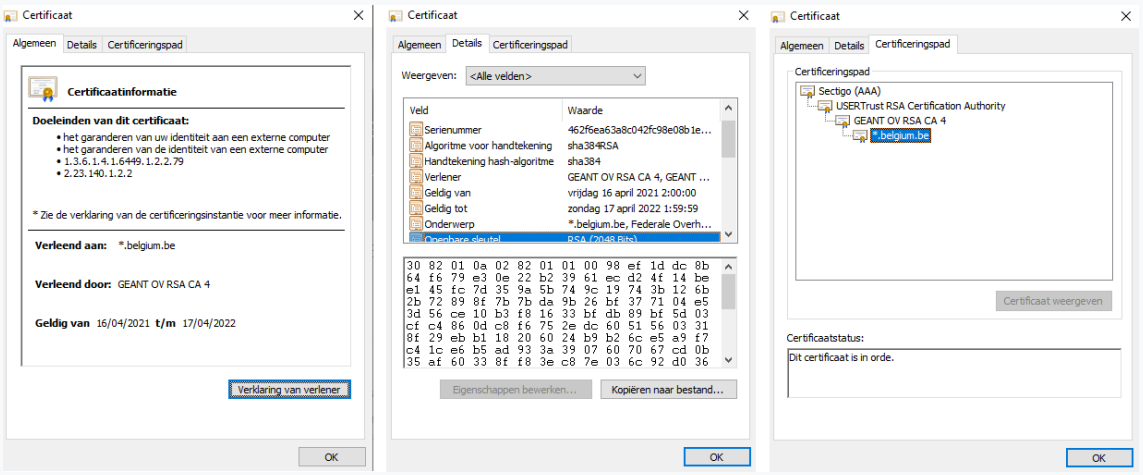
Figuur 3.37: De chain-of-trust: oh zo belangrijk bij digitale certificaten.

i Case: de val van DigiNotar (2011)

In 2011 werd de Nederlandse CA **DigiNotar** gehackt. De aanvallers konden valse certificaten uitgeven voor onder andere `google.com`, die vervolgens werden ingezet om Iraanse Gmail-gebruikers te bespioneren. Zodra de inbraak publiek werd, verwijderden browserfabrikanten DigiNotar uit hun lijst van vertrouwde root-CA's. **Alle** certificaten van DigiNotar werden daarmee in één klap ongeldig — ook die van de Nederlandse overheid, die DigiNotar gebruikte voor DigiD en andere diensten. DigiNotar zelf ging binnen enkele weken failliet. Het incident toont hoe fragiel de chain of trust is: één gecompromitteerde CA kan het vertrouwen voor duizenden sites kapotmaken.

3.8.1 Certificaten bekijken

In iedere moderne browser kan je snel bekijken hoe zo'n certificaat er juist uitziet. Als je via een HTTPS verbinding naar een website surft, dan op het slotje naast de URL in de adresbalk klikt kan je doorklikken om het certificaat te openen. Als je naar *HTTPS://www.belgium.be* surft en dit doet dan krijg je eerst wat samenvattende informatie:

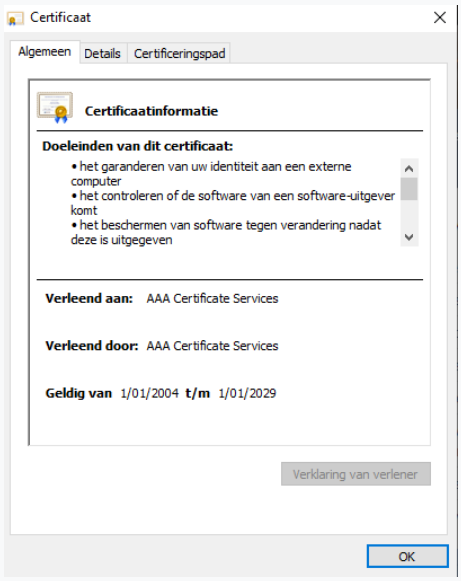


Figuur 3.38: Het certificaat van België.

Zo zien we onder andere de geldigheidsduur, alsook de CA die dit certificaat heeft gegenereerd. Onder details kunnen we onder andere de publieke sleutel zien van de website alsook de gebruikte algoritmes voor de hash, e.d.

En op de laatste tab, Certificeringspad, zien we de chain of trust. We kunnen vervolgens hier de bovenliggende certificaten bekijken.

Het certificaat van Sectigo is uiteraard een **selfsigned certificate**, daar zij "bovenaan de hiërarchie staan". Als we Sectigo niet vertrouwen dan kunnen we ook de communicatie met *belgium.be* niet vertrouwen.

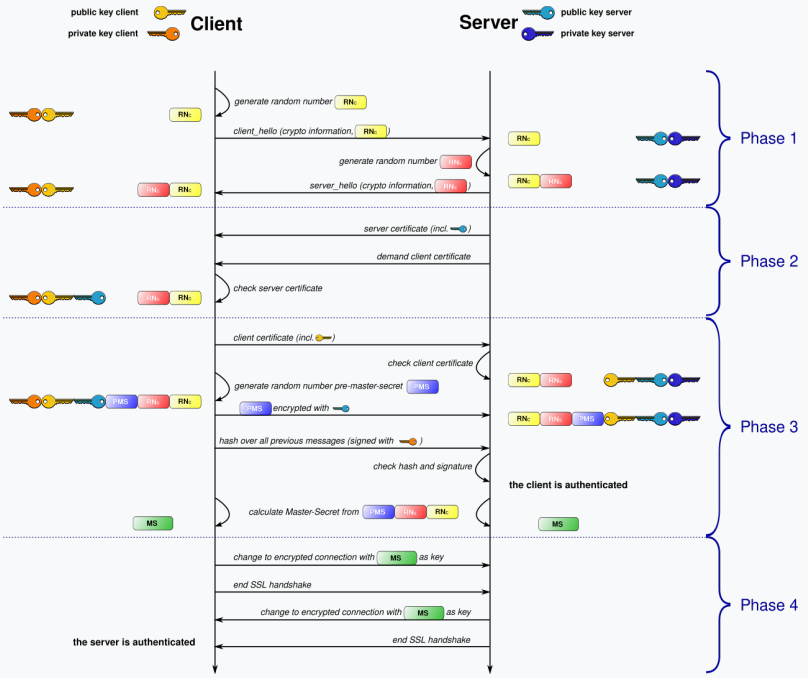


Figuur 3.39: Sectigo heeft een self-signed certificaat wat je herkent aan het feit dat de velden *Verleend aan* en *Verleend door* dezelfde waarde hebben.

3.8.2 Persoonlijke certificaten

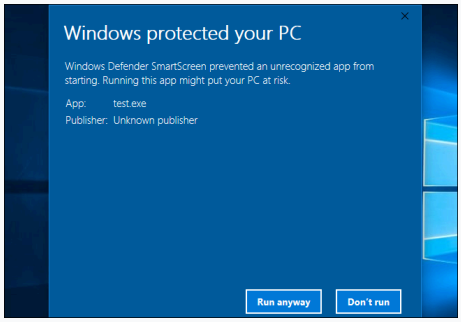
Naast certificaten voor webservers (zogenaamde **SSL certificaten**) kan je ook een persoonlijk certificaat aankopen om je eigen identiteit aan derden te bewijzen tijdens bijvoorbeeld e-mail-communicatie. Voorts heb je ook **code signing** certificaten die de echtheid van een applicatie bewijzen zodat je zeker bent dat je geen malware installeert als je programma X hebt gedownload.

Persoonlijke certificaten worden ook gebruikt voor **mutual TLS** (mTLS), een uitbreiding op gewone HTTPS waarbij niet alleen de server, maar ook de **client** zich met een certificaat authenticeert. Dit wordt vaak gebruikt in bankomgevingen, e-government, bedrijfs-VPN's en communicatie tussen backend-servers, waar de server zeker wil zijn dat de client effectief is wie die beweert te zijn. Bij gewone HTTPS gebeurt dit niet omdat een website in principe elke bezoeker welkom heet.



Figuur 3.40: Bij mutual TLS presenteren zowel server als client een certificaat; beide zijden worden geverifieerd tegen dezelfde CA. Bron: Wikimedia Commons (CC BY 3.0).

Als je in Windows 10 of nieuwer een applicatie of installer probeert uit te voeren dan zal de ingebouwde *SmartScreen* service ogenblikkelijk de echtheid (of ontbreken van) het certificaat controleren, net zoals dit ook in de browser zou gebeuren.

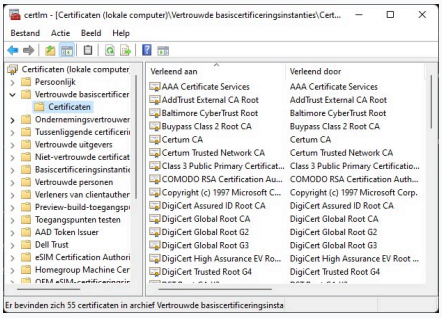


Figuur 3.41: Windows 10 Smartscreen beschermt je van niet digitaal ondertekende software.

Wil dat dan zeggen dat je applicaties niet kunt vertrouwen die door Smart Screen als onveilig worden aangeduid? Neen, dat niet. Je mag niet vergeten dat een certificaat geld kost en dat niet alle software-ontwikkelaars de middelen hebben om een officieel certificaat te kopen. Het loont dus altijd om extra waakzaam te zijn wanneer Smart Screen een waarschuwing geeft, maar het is dus niet zo dat de software automatisch als onveilig moet gehanteerd worden.

Tip

Je kan via de Certification Manager van Windows bekijken welke certificaten je lokaal hebt geïnstalleerd, welke worden vertrouwd, etc. Je kan de GUI-versie van deze tool opstarten door “certlm.msc” uit te voeren.



Figuur 3.42: Let er altijd op welke informatie je deelt via screenshots.

3.8.3 Web of Trust: een alternatief voor PKI

Het PKI-model steunt op **gecentraliseerde** Certificate Authorities die de identiteit van sleuteleigenaars garanderen. Er bestaat echter ook een **gedecentraliseerd** alternatief: het **Web of Trust** (WoT), dat bekend werd door **PGP** (Pretty Good Privacy) en de open-source variant **GPG** (GNU Privacy Guard).

In een Web of Trust zijn er geen centrale autoriteiten. In plaats daarvan ondertekenen gebruikers *elkaars* publieke sleutels. Als Alice de publieke sleutel van Bob persoonlijk heeft geverifieerd (bijvoorbeeld door zijn *key fingerprint* te vergelijken tijdens een ontmoeting), kan zij zijn sleutel ondertekenen met haar eigen private sleutel. Hiermee verklaart Alice: *“Ik bevestig dat deze publieke sleutel effectief van Bob is.”*

Stel nu dat Carol de sleutel van Bob nodig heeft maar hem niet persoonlijk kent. Als Carol wél Alice vertrouwt en ziet dat Alice de sleutel van Bob heeft ondertekend, dan kan Carol via dat **vertrouwenspad** besluiten om ook Bobs sleutel te aanvaarden. Zo ontstaat een netwerk — een *web* — van onderlinge vertrouwensrelaties.

Eigenschap	PKI	Web of Trust
Vertrouwensmodel	Hiërarchisch (top-down via CA’s)	Gedecentraliseerd (peer-to-peer)
Wie valideert?	Certificate Authorities	De gebruikers zelf
Zwak punt	Eén gecompromitteerde CA treft iedereen	Vergt actieve deelname van gebruikers
Typisch gebruik	HTTPS, e-mail (S/MIME), code signing	PGP/GPG e-mailencryptie, softwarepakketten

Opmerking

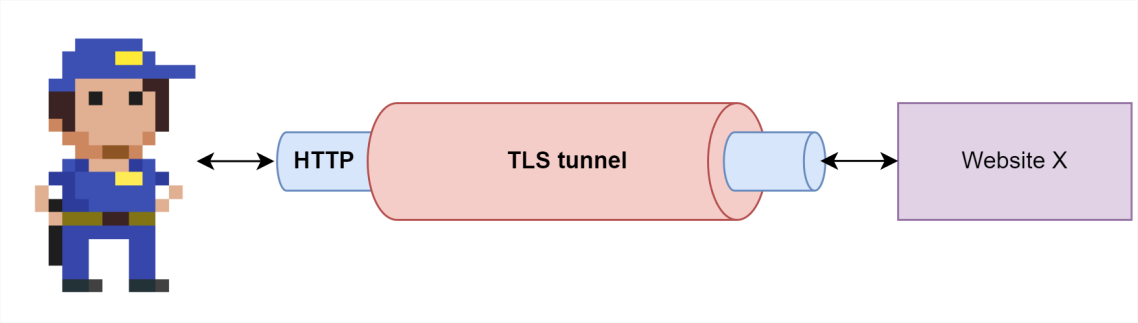
Het Web of Trust werd jarenlang gebruikt door de PGP/GPG-gemeenschap, onder andere via zogenaamde **key signing parties** waar mensen fysiek samenkwamen om elkaars sleutels te verifiëren en te ondertekenen. In de praktijk bleek het model echter moeilijk schaalbaar: het vergt veel moeite van individuele gebruikers en het is lastig om een betrouwbaar vertrouwenspad te vinden naar iemand die je niet kent. Daarom wordt voor de meeste toepassingen op het Internet vandaag het PKI-model met Certificate Authorities gebruikt.

3.9 HTTPS en TLS

Certificaten vormen het hart van een veilige manier van surfen. HTTPS, “HTTP-Secure”, zorgt ervoor dat de communicatie tussen je browser en de website via een beveiligde, geëncrypteerde tunnel gebeurt. Tegenwoordig is HTTPS de default manier om een website te benaderen, maar dat is maar recent. Vroeger gebeurde alles via HTTP, waardoor iedereen die jouw trafiek kon sniffen, kon zien welke informatie je met de website uitwisselde.

HTTPS is een protocol dat een beveiligde encryptietunnel opzet tussen jou en de website waarover vervolgens gewoon HTTP-verkeer kan verlopen (dit gebeurt over poort 443 in plaats van de klassieke poort 80 waarover HTTP verloopt). Deze tunnel wordt opgezet door het **TLS**-protocol, het *Transport Layer Security* protocol, dat de opvolger is van **SSL** (*Secure Sockets Layer*). TLS gebruikt certificaten om te vergewissen dat de website aan de andere zijde wel degelijk de website is die de gebruiker verwacht. Het doet dit door de publieke sleutel van de website eerst te controleren voor het vervolgens deze sleutel gebruikt om een gemeenschappelijke sleutel af te spreken die zal dienst doen als de encryptie-sleutel voor de gemeenschappelijke tunnel. Vanaf dit punt kunnen derden de trafiek van en naar de website niet meer sniffen.

Samengevat: **HTTPS is de combinatie van HTTP en een veilige encryptietunnel die met behulp van het TLS protocol wordt opgezet.**



Figuur 3.43: De erg belangrijke TLS tunnel tijdens het surfen met HTTPS.

Samengevat zal dus TLS twee zaken doen:

1. Door middel van een certificaat (asymmetrische crypto) wordt de identiteit (de publieke sleutel) van de website gecontroleerd.
2. Door middel van een afgesproken algoritme (Diffie-Hellman, Forward Secrecy, Elliptic Curve, etc.) een gemeenschappelijke sleutel(s) afspreken en uitwisselen.

> **Tip**
>
> Zoals reeds eerder vermeld is asymmetrische crypto trager, waardoor het altijd aanbevolen is om de trafiek tussen 2 punten finaal via een symmetrische crypto verbinding te laten plaatsvinden. TLS/HTTPS combineert met andere woorden de sterktes van beide soorten crypto om zo de zwaktes van beiden te neutraliseren.

De manier waarop een TLS-verbinding wordt opgezet is vrij uitgebreid. Volgende briljante website (tls.ulfheim.net/) visualiseert de berichten die server en client uitwisselen om zo’n verbinding te starten, onderhouden en eindigen.

⚠ Waarschuwing

Alhoewel HTTPS onze verbinding een pak veiliger maakt, heeft het voor je ISP (Internet Service Provider, bijvoorbeeld Telenet of Proximus) en de website ook enkele nadelen. Omdat alle informatie geëncrypteerd wordt heeft de ISP geen enkel idee wat voor informatie je aan het uitwisselen bent, waardoor caching ook niet meer mogelijk is. In een normale HTTP-omgeving kan een ISP trafiek over het Internet uitsparen door een reeds bewaarde versie van hetgeen jij nodig hebt uit de cache te halen en naar je te sturen. Ook de website naar waar je surft, ondervindt dit nadeel: het zal met HTTPS veel meer trafiek genereren dan wanneer de tussenliggende ISP een deel van het werk via zijn caching overnemen.

3.9.1 End-to-end encryptie onder druk: Apple en de UK

End-to-end encryptie (E2EE) zorgt ervoor dat enkel de zender en ontvanger de inhoud van berichten of data kunnen lezen — zelfs de dienstverlener (zoals Apple of Google) heeft geen toegang. Dit principe is een directe toepassing van de publieke cryptografie die we in dit hoofdstuk bespraken: data wordt versleuteld met de publieke sleutel van de ontvanger en kan enkel met diens private sleutel worden ontsleuteld.

In 2025 werd Apple door de Britse overheid gedwongen om **Advanced Data Protection** (ADP), de end-to-end encryptie van iCloud-data, uit te schakelen voor alle gebruikers in het Verenigd Koninkrijk. De overheid eiste namelijk een *backdoor*: een manier voor opsporingsdiensten om toegang te krijgen tot versleutelde gegevens. Apple weigerde een backdoor in te bouwen — omdat dit de beveiliging voor **alle** gebruikers zou verzwakken — en koos er in plaats daarvan voor om de E2EE-functionaliteit in het VK volledig te verwijderen. iMessage, FaceTime en iCloud Keychain behielden wel hun end-to-end encryptie.

⚠ Waarschuwing

Dit voorbeeld illustreert een fundamenteel spanningsveld in cryptografie: **een backdoor die enkel voor "de goeden" werkt, bestaat niet**. Zodra er een achterpoortje in een encryptiesysteem zit, is het slechts een kwestie van tijd voordat ook kwaadwillige actoren deze ontdekken of misbruiken. Dit gaat recht in tegen Kerckhoffs principe: de veiligheid van het systeem mag enkel afhangen van de geheimhouding van de sleutel, niet van het verbergen van zwakheden in het systeem zelf.

3.9.1.1 Chat Control: ook in de EU

Dit debat speelt niet enkel in het Verenigd Koninkrijk. De Europese Commissie stelde in 2022 de **Child Sexual Abuse Regulation** voor, beter bekend als **Chat Control**. Dit voorstel zou chatdiensten zoals WhatsApp en Signal verplichten om berichten van alle gebruikers automatisch te scannen op illegale inhoud — ook berichten die end-to-end versleuteld zijn. Critici, waaronder cryptografen en organisaties als de EFF en EDRi, waarschuwen dat dit technisch neerkomt op het inbouwen van een backdoor of het installeren van *client-side scanning* (spyware op het toestel zelf), wat de facto end-to-end encryptie onmogelijk maakt.

Na jarenlange controverse en tegenstand vanuit het Europees Parlement, werd het meest omstreden onderdeel — het verplicht scannen van versleutelde berichten — in 2025 afgezwakt. Het voorstel wordt echter nog steeds onderhandeld en de uiteindelijke impact op E2EE blijft onzeker.

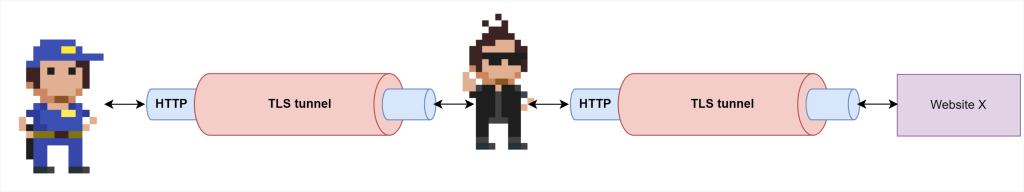
ℹ Opmerking

De kern van het probleem is telkens hetzelfde: je kan niet tegelijk **echte** end-to-end encryptie garanderen én een manier voorzien om berichten te lezen. Of de sleutel is geheim, of hij is het niet — er is geen tussenweg.

3.9.2 Mitmproxy

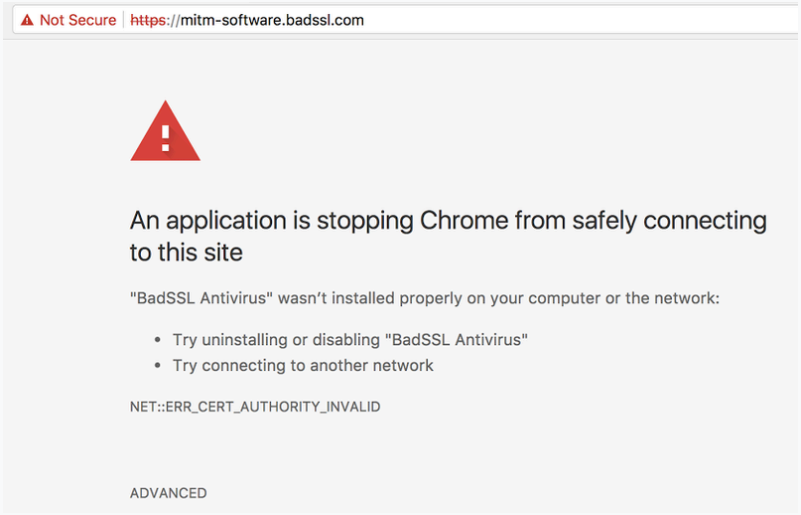
Mitmproxy is een krachtige linux-tool die een man-in-the-middle aanval op HTTPS toelaat. Het zal ervoor zorgen dat een aanvaller zich tussen jou en het Internet kan nestelen en vervolgens doen alsof al je HTTPS-verbindingen veilig blijven. In de praktijk zorgt mitmproxy ervoor dat alle HTTPS-verbindingen

van de client naar de aanvaller gebeuren, die op zijn beurt TLS tunnels zal opzetten met de website waar het slachtoffer naar surft. Hierdoor kan de aanvaller enerzijds alle trafiek lezen, maar bijvoorbeeld ook ongezien aanpassen.



Figuur 3.44: Een man-in-the-middle aanval met TLS.

De aanvaller zal echter nog steeds geen geldige certificaten kunnen genereren waardoor moderne browsers normaal gezien hier een waarschuwing zouden moeten geven.



Figuur 3.45: De waarschuwing die Chrome genereert wanneer het een, potentiële, mitm-aanval detecteert.

3.10 Samenvatting

Cryptografie is de gereedschapskist waarmee we CIA realiseren. De centrale inzichten uit dit hoofdstuk:

- **Kerckhoffs** is de kern: alleen de **sleutel** moet geheim blijven, nooit het algoritme. *Security through obscurity* is een valstrik.
- Oude technieken (Caesar, Vigenère, scytale) combineren **substitutie** en **transpositie** — dezelfde bouwstenen die AES nog steeds gebruikt.
- **Symmetrische encryptie** is snel (AES als de-facto standaard, gebaseerd op het Belgische Rijndael) maar kent het **sleuteloverdrachtsprobleem**.
- De **blockcipher-mode** is even kritiek als de cipher zelf: **ECB lekt patronen**, **CBC/CTR** niet — mits correcte IV/nonce.
- **Asymmetrische encryptie** (RSA, ECC) lost het sleuteloverdrachtsprobleem op via een publiek/privé-sleutelpaar. **Diffie-Hellman** laat twee partijen zonder vooraf gedeeld geheim een gezamenlijke sleutel afspreken.
- **Hashes** zorgen voor *integrity*, **digitale handtekeningen** voegen daar authenticiteit en *non-repudiation* aan toe.
- **Digitale certificaten en PKI** lossen het vertrouwensprobleem op — maar enkel zolang de CA betrouwbaar blijft (cfr. de val van DigiNotar).
- **HTTPS/TLS** combineert al deze bouwstenen, en toch blijft het kwetsbaar zonder goede certificaatvalidatie, zoals *mitmproxy* aantoont.
- **Cryptanalyse** dwingt sleutels steeds langer te maken; **quantum-computers** bedreigen RSA/ECC op lange termijn — *store now, decrypt later* is een realistische dreiging.

In het volgende hoofdstuk zien we hoe authenticatie op deze crypto-bouwstenen steunt.

4. Wifi security

i Leerdoelen

Na dit hoofdstuk kan je:

1. Uitleggen welke fundamentele **beveiligingsproblemen draadloze netwerken oplossen én introduceren** t.o.v. bedrade netwerken.
2. Concreet beschrijven **waarom WEP fundamenteel gebroken is** (RC4-misbruik, te korte IV, zwakke CRC, geen sleutelbeheer).
3. De **evolutie WEP → WPA1/TKIP → WPA2/CCMP → WPA3** chronologisch én technisch motiveren: welk probleem loste elke stap op?
4. Het verschil tussen **personal en enterprise mode** (incl. de rol van **802.1X**) toelichten en een correcte keuze maken voor een gegeven context.
5. De impact van een kwetsbaarheid zoals **KRACK** kaderen en uitleggen waarom **forward secrecy** in WPA3 belangrijk is.

4.1 De problemen van wifi

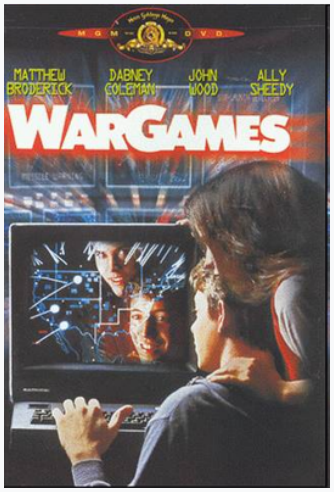
We kunnen draadloze netwerken, specifiek wifi-netwerken, niet meer uit ons leven wegdenken. De opkomst van de IEEE 802.11b standaard (spreek IEEE uit als *"Ai-trippel-i"*) in 1999 veroorzaakte een kleine revolutie in de manier waarop bedrijven en privégebruikers konden werken. Plots kon je met een laptop van overal in het gebouw - en zelfs er buiten - op het netwerk geraken. Die vrijheid voor de gebruikers betekende wel een nachtmerrie voor de cyberboswachters. Een netwerkkabel heeft een intrinsieke extra beveiliging: enkel daar waar de kabel ligt kunnen gebruikers op het netwerk geraken. Zolang je dus geen netwerkkabel naar de publieke parking brengt kan niemand van daar illegaal het netwerk benaderen. Met wifi leek het alsof plotseling het hele bedrijfsnetwerk in een straal van tientallen meters rond het gebouw beschikbaar was, met alle gevolgen van dien.

Figuur 4.1: Een oud voorbeeld van hoe wifi signalen "uit" een gebouw veel verder geraken dan verwacht (bron van de afbeelding: onbekend)

Al gauw werd een nieuwe sport uitgevonden door hobbyist hackers en professionele cybercriminelen: **wardriving**. Het idee is eenvoudig: je rijdt rond in de stad en laat de laptop naast je in de auto scannen

naar alle draadloze netwerken, met extra aandacht voor die netwerken die geen of zwakke beveiliging hebben.

i Opmerking

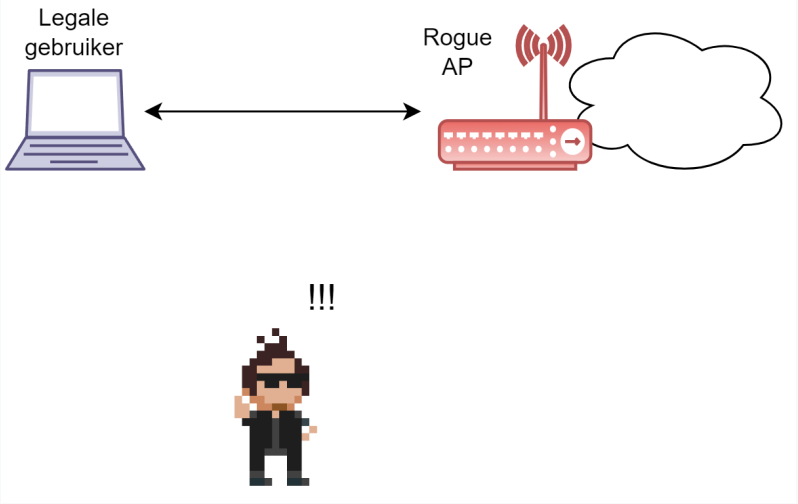


Figuur 4.2: Wargames: een cultklassieker uit 1983.

De term wardriving komt van de term *wardialing* die op zijn beurt gebaseerd is op de klassieke cybercult film "Wargames" uit 1983. Voor de geschiedkundigen onder ons, wardialing was het opbellen van willekeurige telefoonnummers met je modem in de hoop een zogenaamd *bulletin board system* oftewel **BBS** (een pre-Internet forum zeg maar) te vinden.

Draadloze netwerken die gevonden worden hebben dan ook een schare aan problemen:

- **Eavesdropping**: iedereen kan "zien" wat er door de lucht vliegt. Dit heb je niet met een kabel. Bij een bedraad netwerk is de fysieke koperdraad het communicatiemedium, bij wifi is dat eigenlijk letterlijk de lucht.
- **Invasion**: je kan eenvoudig verbinden met het netwerk indien er geen beveiliging voorzien werd. Iets dat je bij een bedraad netwerk enkel kunt als je fysiek toegang hebt tot een netwerkkabel.
- **Man-in-the-middle aanvallen**: hier gaan we zo meteen dieper op in.
- **Backdoor**: een veelvoorkomend probleem zijn zogenaamde **rogue access points** die worden bijgeplaatst op het netwerk door goedbedoelde werknemers die zo het bereik van het netwerk wat willen uitbreiden. Vaak zijn de beveiligingsinstellingen van zo'n (meestal goedkoop) access point niet zo sterk als die van het bedrijf. Bijgevolg is dit de ideale backdoor voor malafide gebruikers die het netwerk aan het surveilleren zijn. Wanneer ze een netwerkscan doen zullen ze tientallen goed beveiligde access points zien en één zwak beveiligd.



Figuur 4.3: Een rogue access point is de ideale manier om binnen te geraken voor hackers.

Defensieminister legde eigen wifi aan in het Pentagon

De Amerikaanse minister van Defensie Pete Hegseth heeft een onbeveiligde internetverbinding laten aanleggen in het Pentagon, melden twee bronnen aan AP. Zo kon hij de beveiliging van het ministerie omzeilen en met zijn persoonlijke laptop gebruikmaken van Signal.

De bronnen beschrijven dat het Pentagon twee beveiligde netwerken heeft om cyberaanvallen tegen te gaan. Een netwerk is bedoeld voor niet-gevoelige informatie, maar voorziet wel in een extra beveiligde internetverbinding. Het andere netwerk is bestemd voor het delen van uiterst gevoelige militaire informatie.

De bronnen zeggen dat Hegseth soms drie laptops bij zich had: een voor persoonlijk gebruik, een voor het delen van gevoelige informatie en een om te communiceren over geheime militaire informatie. Hegseth ging daarmee in tegen de regels van zijn eigen ministerie. Iedereen die het kantoor van de defensieminister binnenkomt, moet namelijk de persoonlijke telefoons en laptops achterlaten in een beveiligde kast.

Een woordvoerder van het Amerikaanse ministerie van Defensie zegt in een reactie aan nieuwszender ABC dat het geen informatie kan geven over de manier waarop Hegseth gebruikmaakte van de communicatiesystemen van het Pentagon. Wel verzekerde de woordvoerder dat de minister van Defensie Signal nooit heeft gebruikt op zijn werklaptop.

Hegseth is een van de mensen die betrokken was bij het Signalschandaal. Nationaal veiligheidsadviseur Mike Waltz besprak in maart met onder meer Hegseth via Signal gevoelige militaire informatie over aanvallen in Jemen, en voegde per ongeluk een journalist toe aan het gesprek. De minister van Defensie zou diezelfde info later ook in een andere Signal-groep hebben besproken met zijn familie. (VK)

De Amerikaanse minister Pete Hegseth kwam al in opspraak door Signal-gate. © PHOTO NEWS

Figuur 4.4: Het kan iedereen overkomen...

- **Denial-of-service op fysiek niveau**: om een gebruiker toegang tot een bedraad netwerk te ontzeggen op fysiek niveau dien je de kabel door te knippen. Bij wifi is dit nog eenvoudiger: de "kabel" bij wifi zijn de frequentiebanden in de lucht waarbinnen de apparaten mogen werken (circa 2.4 Ghz bij de oudere wifi-apparaten, nu meestal rond de 5 Ghz band). De wifi-apparaten kunnen enkel met elkaar communiceren indien zij een signaal naar elkaar over die frequentieband kunnen sturen op een moment dat niemand anders in de buurt die band gebruikt (zie note hierna). Als een malafide gebruiker die

frequentieband continue vult met andere signalen, dan zullen de legale gebruikers nooit iets kunnen uitsturen. Wil je dus een wifi netwerk *DoS'n*, koop dan een signaalgenerator die op de juiste frequentieband de nodige ruist uitzendt en klaar is kees.

> **i** Opmerking
>
> Bedrade netwerktoestellen werken volgens het CSMA/CD (Carrier Sense Multiple Access / Collision Detection) om met elkaar over de draad te communiceren, hierbij detecteren ze wanneer er zich 'botsingen' tussen signalen voordoen en de informatie dus opnieuw moet uitgestuurd worden. Bij draadloze netwerken is het echter onmogelijk om *botsingen in de lucht* te detecteren, daarom werken ze met een variant: **CSMA/CA** , oftewel CSMA/ **Collision Avoidance**. Een wifi-apparaat dat iets wil uitsturen zal eerst controleren of de frequentieband waarop ze werken vrij is. Vervolgens zal het apparaat broadcasten dat het iets gaat versturen, gevolgd door de effectieve informatie.

4.1.1 Onbeveiligde managementframes

Veel keuzes die in de IEEE 802.11 standaard werden gemaakt zijn, zijn vermoedelijk het gevolg van leentje buur spelen bij de reeds bestaande IEEE standaarden voor bedrade netwerken. Echter, bij bedrade netwerken had je niet de inherente onveilige omgeving van het draadloze aspect, waardoor op gebied van beveiliging hier weinig tot geen aandacht aan werd besteed.

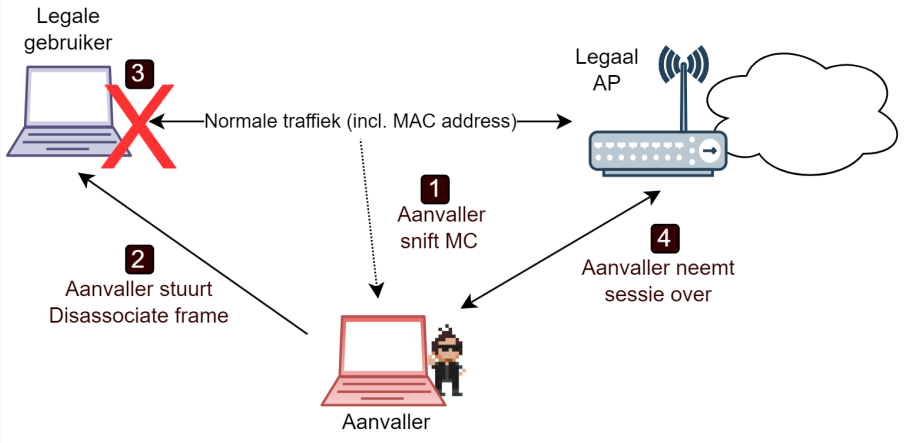
Management frames in wifi zijn frames die ervoor zorgen dat alle aanwezigen op het netwerk op een ordentelijke manier met elkaar kunnen communiceren. Deze frames hebben onder andere als doel:

- Om een client verbinding te laten maken met een netwerk.
- Om SSIDs te broadcasten.
- Clients de opdracht te geven het netwerk te verlaten.
- Om te verbinden met een ander access point van hetzelfde netwerk (*handover*).

Net zoals bij bedrade netwerken, koos men bij de IEEE wifi standaard om deze managementframes **zonder enige vorm van beveiliging** te gebruiken (er werd geen *CIA* voorzien). Dit zorgde ervoor dat niet legale gebruikers zelf management frames konden uitsturen op een netwerk, zonder dat de ontvangers ervan konden controleren of de bron wel een legaal access point of client was.

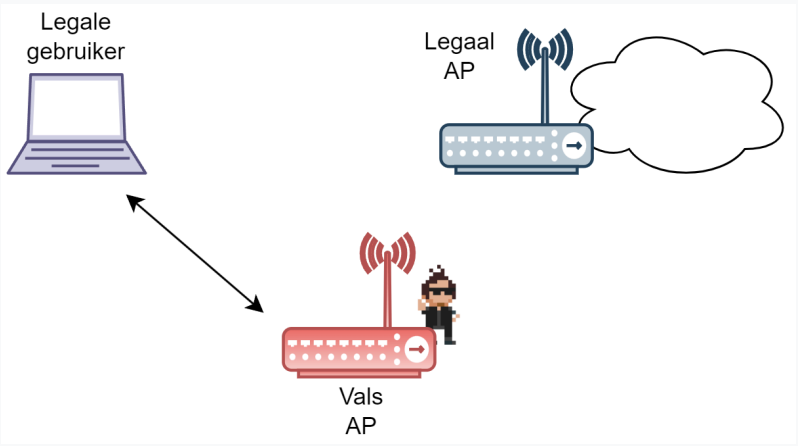
Dit resulteerde in onder andere volgende scenario's:

- **Disassociation flooding**: een hacker kan legale gebruikers DoS'n door constant zogenaamde *disassociation frames* naar hen te sturen. Dit frame, gebruikt door access points, geeft aan clients de opdracht dat ze het netwerk moeten verlaten. De hacker kan zo'n frame uitsturen en daarbij het "source" veld instellen op het MAC-adres van het access points. Deze vorm van spoofing kan ongecontroleerd gebeuren, waardoor legale gebruikers dit frame altijd zullen aanvaarden én vervolgens uitvoeren: de gebruiker kan niet meer verbinden met het netwerk zolang de disassociation frames blijven verstuurd worden.
- **Identity spoofing** was ook eenvoudig daar de hacker eender welk veld in de frames kan aanpassen. Van zodra hij een legale gebruiker met voorgaande techniek van het netwerk heeft geschopt, kan hij vervolgens zichzelf voordoen als deze gebruiker. Hiervoor moet hij gewoon het MAC-adres spoofen van de legale gebruiker tijdens de communicatie met het access point.



Figuur 4.5: Eve gebruikt spoofing om de wifi-sessie van Alice ongemerkt over te nemen.

- En *last but not least* laten de onbeveiligde management frames toe dat we eenvoudig een access point kunnen nabootsen (**impersonation**). Vervolgens kunnen we een **man-in-the-middle aanval** uitvoeren daar een hacker zich kan plaatsen tussen de gebruiker en het Internet en zo informatie kan ontfutselen.



Figuur 4.6: Een fake access point opzetten met dank aan de onbeveiligde managementframes.

Tip

De linux tool *AirSnarf* laat toe om fake hotspots (publiek wifi netwerk) op te zetten. Hierbij maakt het gebruikt van de onbeveiligde management frames. Een scenario in België dat gegarandeerd succes had tot enkele jaren geleden (vanuit het standpunt van de hacker) was een fake Telenet Wi-Free hotspot op te zetten. Hierbij werd eerst de inlogpagina van Telenet Wi-Free door de hacker gecloned en via een lokale webserver aangeboden aan de gebruikers die op het fake access point met de naam "Telenet Wi-Free" verbonden. Vervolgens kon de stroper nu van iedere gebruiker de gebruikersnaam en het wachtwoord stelen telkens deze die informatie op de fake loginpagina invoerde.

Nu dat Telenet Wi-Free is overgeschakeld op WPA-Enterprise (zie verder) kan deze aanval gelukkig niet meer zo eenvoudig uitgevoerd worden.

4.2 De 802.11 standaard qua beveiliging

Om de huidige, en betere, beveiliging van wifi te appreciëren gaan we terug in de tijd om te kijken hoe de originele IEEE wifi standaard de beveiliging beschreef. Het zal een ietwat horror-achtige tocht worden waarin

we gaan ontdekken dat enkele stevige hiaten ervoor gezorgd hebben dat illegale toegang tot bijna ieder wifi netwerk rond de eeuwwisseling binnen enkele minuten kon gebeuren. Lees verder en huiver mee.

De originele "ANSI/IEEE Std. 802.11" die de wifi specificaties beschrijft, werd geschreven in 1999. Het had als doel *"to develop a medium access control (MAC) and physical layer (PHY) specification for wireless connectivity for fixed, portable, and moving stations within a local area."* Hoofdstuk 8 van deze standaard had als titel "Authentication and privacy" en was maar tien pagina's lang, in vergelijking met de totale grootte van het document (528 pagina's) was dit misschien wel een voorbode hoe weinig aandacht er aan beveiliging zou worden gegeven.

Voor we dat kleine hoofdstuk gaan openbreken - en ontdekken hoe WEP in de eerste generatie wifi-apparaten een navenante beveiliging aanbood - zullen we eerst bekijken hoe gebruikers met een draadloos netwerk effectief kunnen verbinden. Zoals reeds vermeld, gebeurt dit gebruik makende van de onbeveiligde management frames.

4.2.1 Verbinden met een netwerk

> Tip
>
> Vanaf nu zullen we geregeld het woord access point afkorten naar **AP**. In deze tekst kan een AP zowel een eenvoudig access point zijn als een complexe draadloze router of modem.

Voor digitale stropers actieve aanvallen kunnen starten op een netwerk, dienen ze verbinding te maken met het netwerk. Dit proces bestaat uit vier stappen die doorlopen moeten worden voor legale en illegale gebruikers effectief gebruik kunnen maken van het netwerk en *higher-layer* datatrafiek kunnen verwerken.

1. **Scannen**: passief of actief zoeken naar de netwerken in de buurt.
2. **Verbinden (*joining*)**: kiezen met welk netwerk zal verbonden worden.
3. **Authenticatie**: bewijzen dat de gebruiker toegang heeft tot het netwerk.
4. **Associatie (*association*)**: bijhouden met welk AP de gebruiker verbonden is van het netwerk. Pas vanaf deze stap kan het netwerk gebruikt en misbruikt worden.

4.2.1.1 Scannen

In deze stap zal de client ontdekken welke netwerken er in de omgeving zijn. Oftewel zoekt de client specifiek naar een netwerk, oftewel wil hij gewoon een oplijsting van alle aanwezige netwerken. Ieder draadloze netwerk heeft een netwerknaam, de **SSID** (*service set identifier*) die bestaat uit een 32 byte ASCII karakter string, en zal deze op geregeld tijdstip broadcasten voor iedereen die hier nood aan heeft. Naast het SSID zal ieder netwerk ook fysieke parameters uitsturen die de client in de volgende stap zal nodig hebben.

Er zijn meerdere kanalen beschikbaar in het spectrum waar binnen een netwerk mag werken. De client zal bij het scannen daarom steeds enkele milliseconden op een bepaald kanaal luisteren om potentiële SSID broadcasts op te vangen, om dan op het volgende kanaal hetzelfde te doen.

> Tip
>
> Sommige gebruikers schakelen het broadcasten van hun netwerknaam (het zogenaamde **SSID**) uit in de hoop dat zo dat buren, voorbijgangers en wardrivers hun netwerk niet kunnen zien. Helaas is dit zogenaamde *snake's oil*: het geeft een vals gevoel van veiligheid. Je kan weliswaar SSID-broadcasting uitzetten (dit is een pakketje dat je access point elke paar seconden uitstuurt om aan iedereen die luistert te zeggen *"Hallo, hier is het netwerk met naam X, en dit zijn de parameters die je nodig hebt om met mij te verbinden"*) maar het SSID wordt ook in een hoop andere pakketjes de lucht in gestuurd. Een beetje wifi hacker kan dus het SSID ogenblikkelijk uit de lucht plukken, ongeacht dat broadcasting werd uitgeschakeld of niet.

4.2.1.2 Verbinden

Wanneer de gebruiker gekozen heeft met welk SSID er moet verbonden worden zal in deze stap de wifi-netwerkkaart ingesteld worden op de juiste fysieke parameters (frequentie kanaal, snelheid, etc.) zodat als het ware AP en client synchroon lopen (en dus op de juiste moment iets tegen elkaar kunnen zeggen

zonder elkaars signalen te storen). In de scan-stap keken we als het ware goed rond, in deze stap richten we onze blik nu op een bepaald netwerk.

4.2.1.3 Authenticatie

> **i** Opmerking
>
> Deze en de voorgaande stappen gebeuren volledig automatisch indien de gebruiker eerder heeft geopteerd om automatisch met een netwerk te verbinden.

In een bedraad netwerk zit authenticatie impliciet vervat in het fysiek aspect: wanneer je de kabel in je laptop kan steken dan is de kans groot dat je een legale gebruiker bent in het gebouw en dus is er geen extra authenticatie nodig (kort door de bocht gezien, weliswaar). Dat is niet zo met draadloze netwerken die voorbije de grenzen van het gebouw hun signaal uitsturen. Er is daarom een extra stap nodig voor we het netwerk kunnen betreden.

In de 802.11 standaard staan twee mogelijke authenticatie-methoden beschreven (die we verderop zullen toelichten):

- **Open-system authentication**
- **Shared-key authentication**

Origineel was deze authenticatie een enkelrichtingstraat. Enkel de client dient zich te authenticeren. Hierdoor bestaat dus de kans dat de gebruiker verbindt met een rogue of fake AP (daar er geen *mutual authentication* bestaat in de originele standaard).

Naast voorgaande twee methoden is er nog een derde, die niet in de standaard staat beschreven maar die wel door veel fabrikanten werd aangeboden in hun wifi-apparaten:

- **MAC Address Authentication gebruik makend van een MAC-ACL**: hierbij kan de beheerder van een AP een *access control list* (ACL) aanleggen waarin alle MAC-adressen staan van toegelaten apparaten.

Open-system authentication

Deze vorm van authenticatie werd eerst gezien als de versie om te zeggen: “er is geen authenticatie nodig op dit netwerk”.

Om op netwerken met deze modus te authenticeren, worden er exact twee frames tussen client en AP uitgewisseld:

1. Eerst vraagt de client aan het AP toegang.
2. Vervolgens stuurt het AP naar de client een “Welkom” frame.

Meer gebeurt er niet in deze modus.

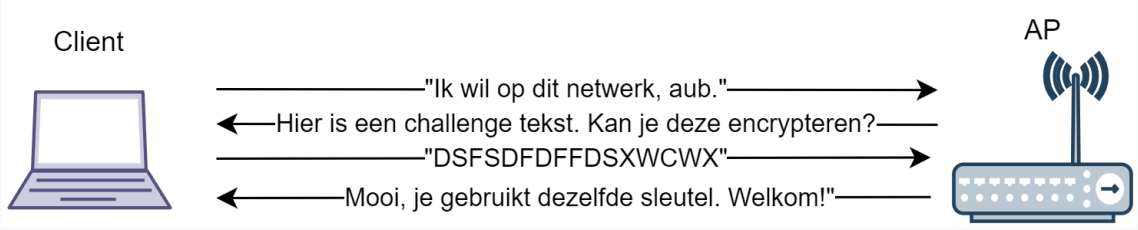
Als er in dit soort netwerk geen encryptie wordt toegepast dan kan dus eender welk apparaat in de omgeving verbinding maken met dit netwerk. Als er wél WEP encryptie wordt gebruikt, dan zal het bezit van de WEP-sleutel als een soort toegangscontrole werken: je kan namelijk wel authenticeren (daar het AP iedereen toelaat in deze authenticatie-fase) maar vervolgens kan je geen data lezen en versturen tenzij je een geldige WEP-sleutel hebt (zie verder).

Verderop zullen we ontdekken dat - oh ironie - deze authenticatie-methode veiliger is dan de shared-key authentication die we zo meteen gaan uitleggen.

Shared-key authentication

In deze modus moet je bewijzen dat je in het bezit bent van een geldige WEP-sleutel voor dit netwerk. Het AP zal daarom een **challenge-response** authenticatie opstarten bestaande uit volgende sequentie van frames:

1. Het AP maakt een random string aan, de *challenge*, en stuurt deze in plaintext naar de client.
2. De client encrypteert deze string met z’n WEP-sleutel en stuurt dit terug naar het AP (de *response*).
3. Het AP zal nu de response decrypteren met z’n eigen WEP-sleutel en het resultaat vergelijken met de challenge-string. Als beiden gelijk zijn weet het AP dat de client een geldige WEP-sleutel heeft en dus toegelaten mag worden op het netwerk.



Figuur 4.7: De shared-key authentication aan de start van de verbinding.

> **Tip**
>
> Merk op dat aanvallers die deze *handshake* sniffen, twee interessante frames zien passeren. Eerst zien ze een plaintext, ogenblikkelijk gevolgd door de bijhorende ciphertext ervan (indien ze een gebruiker sniffen met een geldige WEP-sleutel). Dit heet in cryptanalyse een *gekende plaintext* aanval. Dit zal interessante informatie blijken verderop in dit horrorverhaal, waarin we zullen tonen waarom WEP niet zo veilig bleek te zijn als gehoopt.

4.2.1.4 Associatie

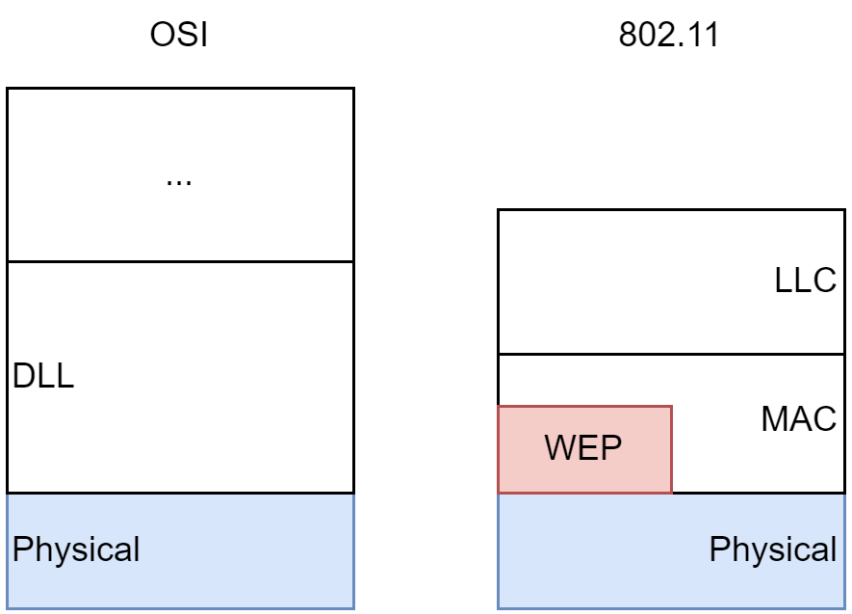
Na een succesvolle authenticatie krijgt de client een *association ID* toegewezen. Dit ID gebruikt het netwerk om te weten waar in het netwerk de client zich bevindt. Veel draadloze netwerken bestaan namelijk uit meerdere AP's en via dit ID weet het netwerk met welk AP de client momenteel verbonden is en zal alle data voor de client dan naar dat AP sturen.

Vanaf dit punt kan de gebruiker dus de bronnen van het netwerk beginnen gebruiken en wordt het tijd om deze communicatie te beveiligen (tenzij het om een publieke hotspot gaat waar iedereen alles van elkaar kan zien).

4.2.2 WEP

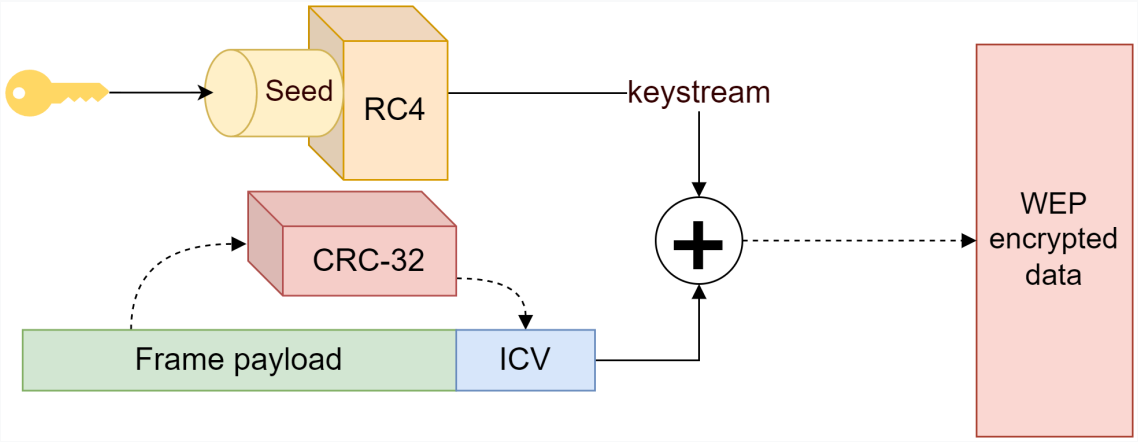
Om potentiële aanvallers ervan te weerhouden dat ze trafiek kunnen sniffen (of zelf op het netwerk zetten) voorziet de 802.11 standaard vanaf de associatie de optie om encryptie te voorzien. Dit gebeurt aan de hand van **WEP**, wat staat voor *wired equivalent privacy*. Een naam die veel beloofde maar niet zo goed was: het idee was dat WEP even veilig zou zijn als een bedraad netwerk.

WEP is optioneel en bevindt zich vlak voor frames naar de fysieke laag (*PHY*) worden gestuurd (die de frames "in de lucht" zal sturen). Het IEEE werkt met lagen die ongeveer overeen komen met de OSI-lagen maar met iets andere namen. In volgende figuur zie je waar WEP, optioneel, zich bevindt.



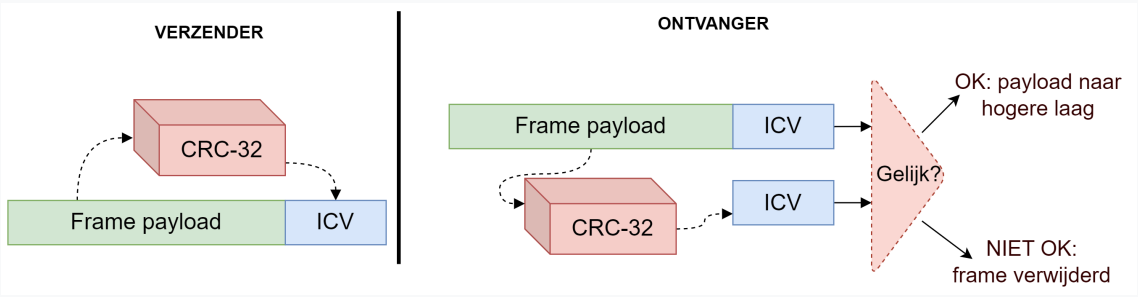
Figuur 4.8: De 802.11 stack (rechts) ten opzichte van de OSI stack.

4.2.2.1 Hoe werkt WEP?



Figuur 4.9: Een vereenvoudigde voorstelling van WEP in z'n geheel.

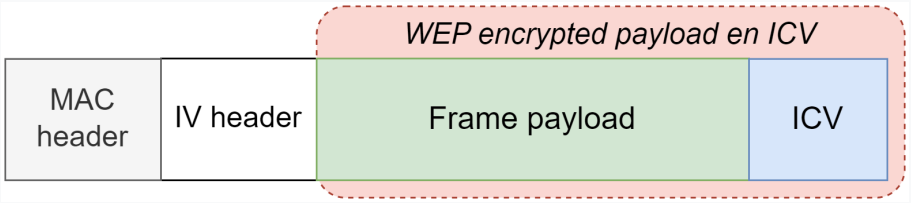
Het hart van WEP is het RC4-algoritme dat we reeds zagen in het crypto-hoofdstuk. De WEP-sleutel zal dienst doen als de seed voor de keystream generatie. Deze keystream zal op zijn beurt ge-XOR'd worden met het te encrypteren frame.



Figuur 4.10: Integrity check.

Confidentiality en integriteit worden tegelijkertijd afgehandeld in WEP. Vlak voor dat het frame via de XOR-operatie wordt geëncrypteerd zal het frame eerst door een *integrity check algoritme* gestuurd worden. Deze integrity check gebeurt met behulp van CRC-32, een oude getrouwe op dit gebied. CRC-32 zal een hash genereren die de ontvanger bij ontvangst kan gebruiken om te zien of het frame werd aangepast na verzenden (bewust door een aanvaller, of door bijvoorbeeld ruis in het netwerk). Deze hash, de **Integrity Check Value (ICV)**, zal mee worden geëncrypteerd door RC4 voor hij verstuurd wordt. Op deze manier kunnen ordinaire aanvallers het frame niet aanpassen zonder dat ze daarmee ook de ICV ongeldig maken.

Finaal verkrijgen we dus volgende WEP-frame dat kan verstuurd worden:



Figuur 4.11: WEP frame lay-out.

> **i** Opmerking
>
> In de originele standaard zat de mogelijkheid om tot vier WEP-sleutels in een netwerk te gebruiken. Via de *keyid* kon een client of AP dan aangeven met welk van de geïnstalleerde sleutels een frame werd geëncrypteerd.

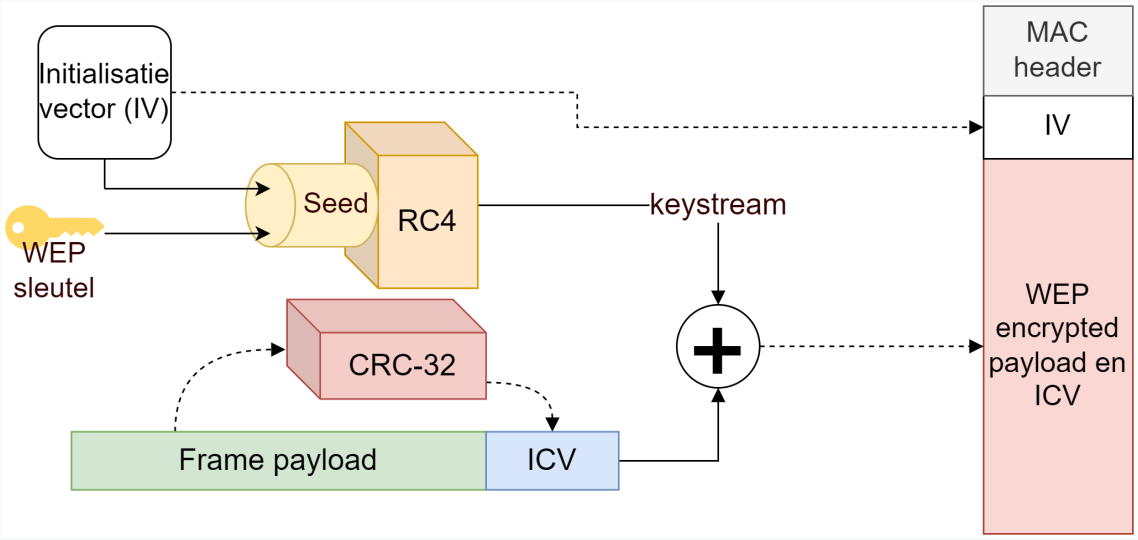
4.2.2.2 Sleutellengte en de IV

Origineel ondersteunde WEP enkel 40-bit WEP sleutels. Ondertussen is dat opgetrokken maar toen de standaard werd geschreven besefte men al dat er sowieso een probleem met de sleutel zou zijn als die zo zou gebruikt worden: ieder frame dat met dezelfde sleutel wordt geëncrypteerd zal dezelfde keystream hebben gebruikt. Dat was natuurlijk geen optie. Om die reden werd gekozen om te werken met een **initialisatie vector (IV)** van 24-bit. Deze IV werd mee als seed aan RC4 gegeven. Door ieder frame een andere IV te kiezen zorgde men er zo voor dat ieder frame een andere keystream gebruikte (**WEP-sleutel+IV werd de nieuwe seed per frame**). In de originele standaard werd echter niet beschreven hoe deze IV moest veranderen, wat nefaste gevolgen zal hebben verderop.

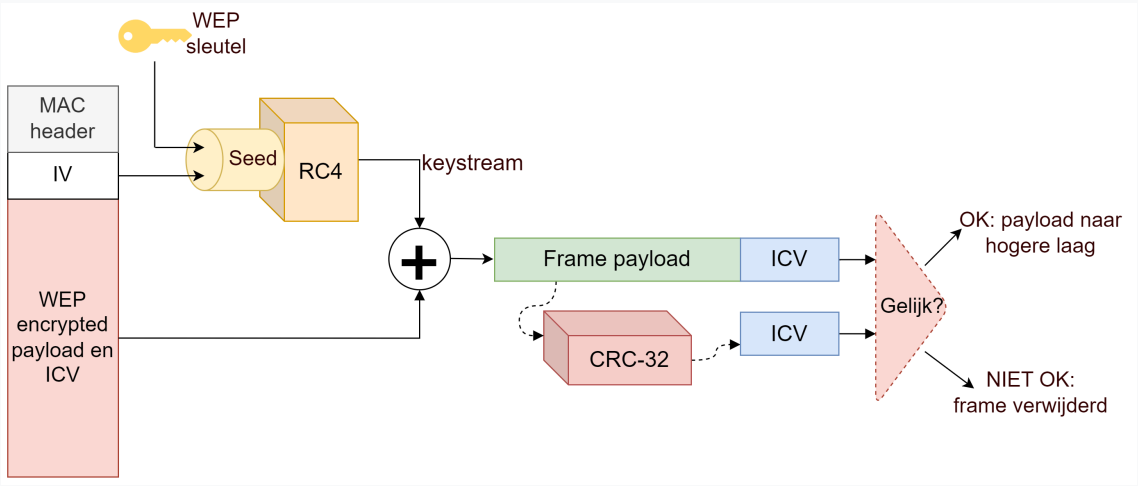
Omdat ook de ontvanger dezelfde keystream moet kunnen genereren tijdens decryptie is het natuurlijk belangrijk dat de IV ook bij de ontvanger gekend is. De enige manier om dit op te lossen is door de IV mee in de header van het frame te plaatsen en door te sturen. Uiteraard moet deze IV als plaintext door het leven gaan.

> **Tip**
>
> Dit concept van een sleutel verlengen met een arbitrair getal heet *salting* en zullen we in het hoofdstuk omtrent wachtwoorden en authenticatie verderop in de cursus nog zien terugkomen.

Finaal krijgen we dus de volgende werking, encryptie en decryptie, als volgt:



Figuur 4.12: WEP encryptie.



Figuur 4.13: WEP decryptie.

4.3 Hoe WEP faalde

Het leek een verbonden vat: hoe populairder wifi werd over de hele wereld, hoe meer papers er verschenen die fouten identificeerden in WEP. Al vrij snel werd duidelijk dat WEP hoegenaamd géén CIA kon aanbieden. De meeste fouten die werden gevonden kunnen gegroepeerd worden in volgende vier zaken:

1. Het **RC4 algoritme** is helemaal niet gemaakt voor een datagramnetwerk zoals wifi.
2. De manier waarop de **IV** in de standaard is beschreven, is ontoereikend en verhoogt de kans op foute implementaties door fabrikanten.
3. **CRC-32** kan omzeild worden en kan dus geen integriteit van frames garanderen.
4. Er is **geen sleutel-management systeem**.

Er is echter nog een vijfde fout die de voorgaande vier als het ware nog versterkt:

5. Er is **geen replay protection**.

Hierdoor heeft de aanvaller dus vrij spel en kan hij een draadloos netwerk als een experimenteertuin gebruiken en duizenden pakketjes te pas en te pas onpas heruitzenden. We zullen nu de eerste vier grote problemen beschrijven. De *replay protection* behandelen we niet apart, maar zullen we bij de andere problemen zien opduiken.

4.3.1 Probleem 1: RC4

De meeste problemen met WEP komen van een verkeerd gebruik van het RC4 algoritme. RC4 wordt in veel moderne beveiligingsapparaten toegepast omdat het een sterk én efficiënt algoritme is (het verbruikt weinig energie omdat er geen dure vermenigvuldiginsoperaties in voorkomen). Echter, streamciphers in het algemeen, RC4 specifiek, zijn eigenlijk geen goede keuze voor datagramnetwerken waarin transmissies onbetrouwbaar zijn.

In een datagramnetwerk worden pakketjes opnieuw verstuurd wanneer er een fout optrad en de ontvanger om een *retransmission* vraagt (dit gebeurt bij ongeveer 20% van de verstuurde data). Dit is volledig normaal gedrag in zowel bedrade als draadloze netwerken, echter voor een streamcipher is dit nefast. Specifiek twee eigenschappen van streamciphers (en dus ook RC4) zorgen voor stevige gebreken in het WEP-protocol inzake confidentiality:

1. RC4 heeft **geen *random access* mogelijkheden**.
2. RC4 staat **geen sleutelhergebruik** toe.

Laten we die twee eigenschappen eens bekijken en welke cascade van problemen ze met zich meebrengen.

4.3.1.1 RC4 heeft geen random access mogelijkheden

Deze eigenschap is niet zo zeer een probleem vanuit beveiligingsperspectief, maar wel vanuit performantieperspectief. RC4 had eigenlijk nooit gekozen mogen worden door het IEEE om in WEP gebruikt te worden. Het verlies van één bit van de datastroom zal er voor zorgen dat alle bits erna met RC4 ook verloren zijn, daar we met een streamcipher werken waarbij de synchronisatie van de stroom bits tussen verzender (encryptie) en ontvanger (decryptie) gelijk moet blijven. Bij het minste dataverlies moeten beide zijden hun *"RC4-motortje"* resetten en opnieuw beginnen.

AES bijvoorbeeld heeft wél die random access mogelijkheid: hierdoor kan steeds herbegonnen worden aan het punt van dataverlies en niet helemaal opnieuw, wat natuurlijk veel efficiënter is (daar we werken in een datagram omgeving waar bitverlies bijna continue voorkomt).

4.3.1.2 RC4 staat geen sleutelhergebruik toe

Stream ciphers hebben een tweede belangrijke eigenschap: **het is uiterst onveilig om een zelfde sleutel twee keer te gebruiken!**

Stel dat je volgende twee plaintext byte sequenties hebt: $p_1, p_2, p_3, \ldots$ en $q_1, q_2, q_3, \ldots$. Beiden worden met dezelfde keystream $k_1, k_2, k_3, \ldots$ geëncrypteerd. Dit geeft ons vervolgens volgende twee ciphertext sequenties:

$p_1 \oplus k_1, p_2 \oplus k_2, p_3 \oplus k_3$
$q_1 \oplus k_1, q_2 \oplus k_2, q_3 \oplus k_3$

Als we nu veronderstellen dat een aanvaller deze twee ciphertexts capteert, wat dan volgt is een grove schending van de confidentialiteit die RC4 moet garanderen (met dank aan de wiskundige eigenschappen van de modulo operator):

$(p_i \oplus k_i) \oplus (q_i \oplus k_i) = p_i \oplus q_i$

Of in andere woorden, wanneer we de beide ciphertexts met elkaar XOR'n krijgen we een sequentie die **niet afhankelijk is van de gebruikte sleutel**! Een stevige hoeveelheid informatie over beide plaintexts wordt zo onthuld. Als één van beide plaintexts gekend is dan volstaat een eenvoudige XOR-operatie om ook de andere plaintext te kennen zonder dat hierbij de gebruikte sleutel moet geweten zijn. Deze fout zullen we verderop misbruiken.

Kortom, streamciphers zijn niet veilig in een datagram omgeving indien er geen vorm van sleutelmanagement bestaat die de sleutels kan vervangen voor ze herbruikt worden. WEP probeert dit gebrek aan sleutelmanagement te omzeilen door met een IV te werken zodat er geen collisions zoals eerder beschreven kunnen optreden…maar ook dat zal een resem problemen met zich meebrengen.

4.3.2 Probleem 2: IV

Het IEEE had dus weet van voorgaand probleem met RC4 en introduceerde daarom de Initialisatie Vector (IV). Vanuit cryptografisch standpunt is dit een solide oplossing. Echter, door het gebrek aan replay protection krijgen we helaas een hoop fouten met de IV.

4.3.2.1 De IV veroorzaakt weak keys

In een paper van 2001 door Scott Fluhrer, Itsik Mantin en Adi Shamir werd aangetoond dat het key scheduling algoritme (KSA) van RC4 een hiaat bevat:

1. Wanneer een deel van de gebruikte RC4 keystream gekend is dan kan een groep *RC4 weak keys* gevonden worden.
2. Wanneer deze weak keys gebruikt worden om een keystream te genereren dan zal er informatie van de gebruikte WEP sleutel in deze keystream gelekt worden.

Of anders gezegd: sommige IV's zorgen voor een sleutel-lekkage naar de keystream, wat natuurlijk nefast is voor eender welk cryptografisch cipher.

Een gevolg van die weak keys is dat, indien de eerste 2 bytes van genoeg keystreams (ongeveer 60) geweten is, men de gebruikte WEP sleutel kan achterhalen door middel van een FMS aanval (de afkorting staat voor de eerste letters van de 3 onderzoekers uit de paper).

De FMS aanval werkt indien:

1. We ongeveer 60 keystreams kunnen capteren waarvan geweten is dat ze *weak* zijn.
2. We de eerste 2 bytes van de plaintext van de bijhorende frames kennen die met deze weak keystreams zijn geëncrypteerd.

Dat tweede is geen probleem, met dank aan de netwerkspecificaties: de payload van een met WEP geëncrypteerd pakket bevat de LLC header (de header van de logical link layer). Volgens de standaard (RFC 2684) moeten *"IP datagram pakketten altijd zichzelf in de header identificeren via de SNAP header"*. En laten de eerste 2 bytes van die header toch wel niet altijd starten met 0xAA. Kortom, quasi alle frames die over een WEP-netwerk vliegen zullen altijd met de hexadecimale waarde AA starten. Nu volstaat het om de bijhorende keystream te achterhalen. En aangezien we de plaintext kennen, kunnen we ook de eerste 2 bytes van die keystream kennen daar we weten dat:

$c_i = k_i \oplus p_i$

(waarbij p_i gelijk is aan 0xAA)

En dus:

$c_i = k_i \oplus p_i \Leftrightarrow k_i = c_i \oplus p_i$

We hebben zo ook deel één van de FMS in onze handen en kunnen nu het algoritme de gebruikte WEP-sleutel laten berekenen (de manier waarop dat gebeurt zou ons te ver brengen).

> **Tip**
>
> De FMS aanval is geïmplementeerd in twee populaire Linux tools: Airsnort & WEPCrack. Beiden kunnen dus gebruikt worden om de WEP-sleutel van een draadloos netwerk te achterhalen.

4.3.2.2 IV collisions treden op

Op zich is de FMS aanval al dramatisch, maar helaas stopt het hier niet. Doordat de IV maar 24 bit groot is, treden er veel sneller collisions op dan intuïtief wordt verwacht. Van zodra twee pakketjes met dezelfde IV zijn verstuurd, treedt er een collision op, en die zijn erg interessant voor aanvallers. Pakketjes met dezelfde IV zijn pakketjes waarvan de payload met dezelfde keystream werd geëncrypteerd:

Een 24-bit IV kan 2^{24} oftewel 16 777 216 mogelijke waarden hebben. Een kleine berekening toont hoe snel collisions optreden:

Gegeven: een eerste generatie AP die aan een miezerige 11 Mbps werkt en continue 1.500-byte frames uitzendt:

- $\frac{11Mbps}{(1500bytes/pakket)*8bits/byte} = 916.67pakketjes/seconde$
- $\frac{16.777.216IVs}{916.67pakketjes/seconde} \approx 18302seconden$

Dat wil dus zeggen dat na ongeveer vijf uur alle IV's opgebruikt zijn en er dan ten laatste collisions optreden.

Deze fout kunnen aanvallers op twee manieren misbruiken: met een passieve of met een actieve aanval.

Passieve IV aanval

Een aanvaller kan passief meeluisteren en stilletjes alle trafiek onderscheppen tot er een IV collision optreedt. Door twee pakketjes met eenzelfde IV te XOR'n verkrijgt de aanvaller een pakket dat bestaat uit de XOR van beide plaintexten van de gecapteerde pakketten. Als dus één van beide plaintexten gekend is, is de inhoud van het andere pakket ook gekend.

IP trafiek is vaak voorspelbaar en bevat aardig wat redundantie (om fouten op te vangen). Hierdoor wordt het makkelijker voor een aanvaller om via cryptanalyse te achterhalen wat de inhoud, of een deel, van het pakket bevat. Een voorbeeld hiervan toonden we bij de FMS aanval waarbij steeds de LLC header gekend was van de meeste pakketten.

Omdat collisions redelijk snel optreden is het voor een aanvaller dus maar een kwestie van lang genoeg te sniffen om zo een grote hoeveelheid pakketten met gelijke IV's op te vangen, waardoor de cryptanalyse ongelooflijk vereenvoudigd wordt.

> **i** Opmerking
>
> In al deze voorbeelden gaan we ervan uit dat de gebruiker geen encryptie toepast op de hogere lagen waar z'n data vandaan komt. Uiteraard wordt cryptanalyse een pak moeilijker als de payload van gecapteerde pakketten geëncrypteerd blijkt te zijn.

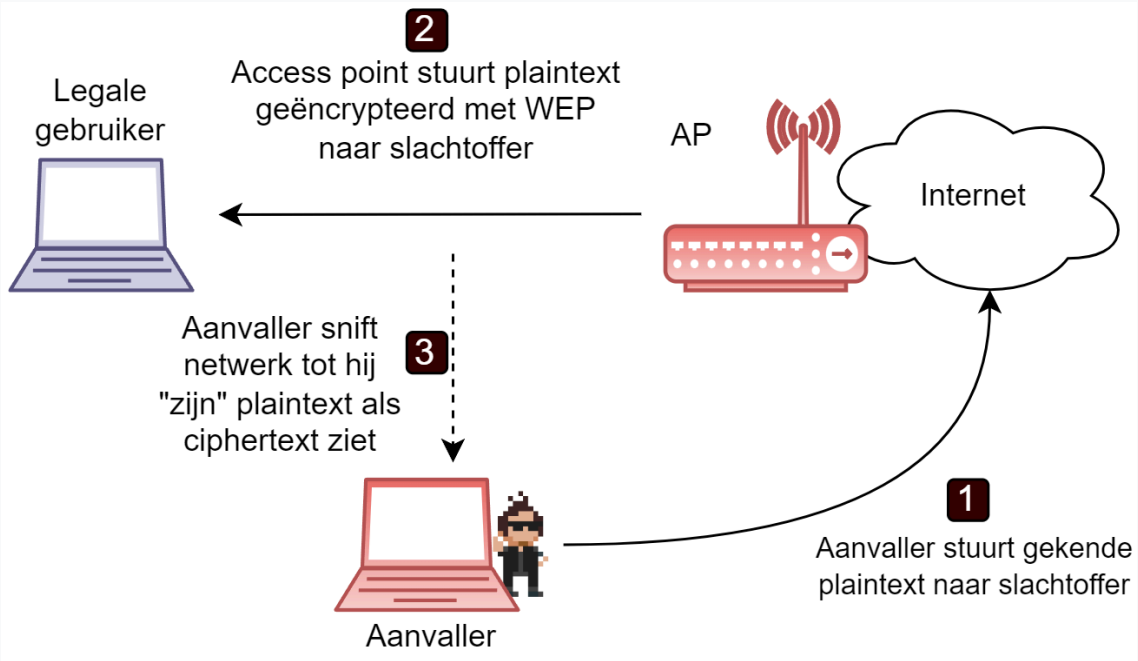
Actieve IV aanval

De passieve aanval heeft als nadeel dat we als aanvaller:

1. moeten wachten op collisions, en dus bijgevolg op trafiek over het netwerk.
2. we enkel door weloverwogen gokken (cryptanalyse) kunnen proberen te weten te komen wat de originele plaintext juist is.

Beide problemen kunnen we als aanvaller echter te niet doen door een actieve rol te gaan spelen. Doordat een AP braaf alle trafiek encrypteert dat het van het bedrade netwerk krijgt om naar een client te sturen, is het voor een aanvaller een kwestie van "gekende" plaintext van buitenuit naar het slachtoffer te sturen. Als volgt:

1. Een gekende plaintext boodschap (bijvoorbeeld een e-mailbericht of ping) wordt naar het AP gestuurd (via het Internet bijvoorbeeld), dat vervolgens door de aanvaller in het oog wordt gehouden. Noot: als de aanvaller enkel het draadloze netwerk ter beschikking heeft (en niet het Internet) dan zal een bitflip-aanval moeten gebruikt worden, wat we verderop zullen uitleggen.
2. De aanvaller blijft sniffen tot hij de ciphertext ziet passeren waarin (vermoedelijk) z'n gestuurde plaintext zit.
3. Vervolgens kan de aanvaller een keystream te pakken krijgen door z'n plaintext te XOR'n met de gecapteerde cipherhtext: $c_i = k_i \oplus p_i \Leftrightarrow k_i = c_i \oplus p_i$.



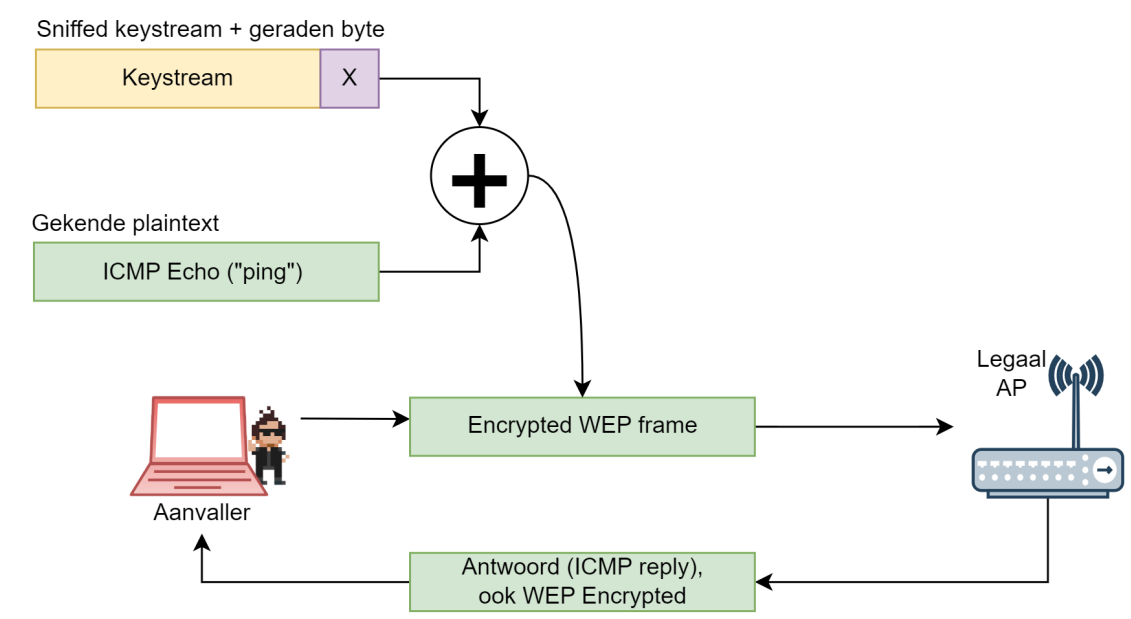
Figuur 4.14: Known plaintext in het wifi-netwerk krijgen.

4.3.2.3 Keystreams groeien

Wanneer een aanvaller met voorgaande IV collisions keystreams kan capteren kan hij in principe data op het netwerk beginnen plaatsen: **aangezien het netwerk ervan uitgaat dat het gebruiken van geldige keystreams, wil zeggen dat de gebruiker geauthenticeerd is omdat hij de bijhorende WEP-sleutel heeft.** De aanvaller kan nu plaintext XOR'n met deze gevonden keystream en op het netwerk zetten. Echter, hij is beperkt tot pakketten die maximum even groot zijn als de keystream die gevangen werd. Het zou véél nuttiger zijn als de aanvaller als het ware een bibliotheekje heeft van geldige keystreams van allerlei lengtes.

Omdat er geen replay protection aanwezig is, kan de aanvaller eenvoudig z'n gecapteerde keystreams doen *groeien* en zo byte per byte een langere keystream genereren. Dit gaat als volgt te werk:

1. De aanvaller maakt een plaintext pakketje aan dat 1 byte langer is dan de keystream die hij al heeft. Het ping-commando (ICMP) kan je met de "-l" optie bijvoorbeeld een ping van eender welke bytesize laten genereren. Het voordeel van het ping-commando gebruiken is ook dat we exact weten wat voor response er kan verwacht worden.
2. De aanvaller plakt nu 1 byte achter de gecapteerde keystream. Hij kiest hierbij een willekeurige waarde en heeft dus 1 kans op 256 om de juiste te kiezen.
3. De aanvaller XOR'd deze keystream met het commando uit stap 1 en stuurt dit op het netwerk.
4. Indien de aanvaller in stap twee de juiste byte gekozen heeft dan zal er een reactie op de ping volgen (daar het pakket werd gedecrypteerd door het AP en dan hoger in de OSI-stack door het netwerk werd gestuurd). Als de keystream fout is zal er geen reactie komen daar het AP het pakketje als foutief heeft weggegooid en dus heeft genegeerd.
5. Als een verkeerde byte werd gekozen in stap twee dan zal de gebruiker dit proces opnieuw starten en nu een andere byte-waarde kiezen. Hij zal dit blijven herhalen tot hij in stap vier reactie krijgt en dus weet dat hij nu z'n keystream met succes heeft doen groeien met 1 byte.



Figuur 4.15: Keystreams byte per byte groeien.

> ⚠ Waarschuwing
>
> We hebben bij deze aanval één belangrijk concept genegeerd waardoor deze aanval op eerste zicht niet mogelijk is: het aanpassen van een payload resulteert ook in een nieuwe CRC. Deze is echter mee geëncrypteerd waardoor het niet duidelijk is hoe we dit kunnen omzeilen. Wacht nog even tot we aan de bitflip aanval komen en alles zal duidelijk worden (dat dit dus geen probleem is).

4.3.2.4 IV selectie

De vierde fout met de Initialisatie Vectoren is de manier waarop de selectie ervan moet gebeuren in de hardware. De 802.11 standaard gaf enkel aan dat de IV *"geregeld moest geüpdatet"* worden. Dat is uiteraard te vaag en heeft ervoor gezorgd dat fabrikanten zelf moesten bepalen welke IV selectie strategie ze in hun hardware zouden implementeren. Hierdoor waren er drie strategieën die hun weg in de verschillende apparaten vonden:

- **Vast IV**: sommige fabrikanten hadden geen flauw benul wat het doel van de IV was vanuit cryptografisch standpunt en kozen daarom zelfs gewoon om alle pakketten steeds met het zelfde IV te versturen. Hierdoor treden er dus collisions op van zodra er een tweede pakketje de lucht wordt ingestuurd.
- **Willekeurig IV**: andere fabrikanten verkozen het om hun hardware bij ieder pakketje een willekeurig IV te laten selecteren. Alhoewel dit uiteraard veel veiliger is dan een "vaste IV"-strategie, treden er toch veel sneller collisions op dan verwacht. Dit valt te verklaren door het zogenaamde **verjaardagenparadox** (zie kader verder) dat verklaart waarom er reeds 50% kans op een collision is na 4823 pakketjes. Dat wil dus zeggen dat al na enkele seconden er meestal collisions optreden.
- **Incrementele IV**: in deze strategie wordt een circulaire teller gebruikt waarbij de IV telkens met 1 wordt verhoogd wanneer een pakket moet worden verstuurd. Meestal begint deze teller op een vaste waarde. Dit zal er dan ook voor zorgen dat er een collisions optreedt van zodra een tweede apparaat zich op het netwerk begeeft en dus begint uit te zenden met de IV gelijk aan de IV van het allereerste pakketje dat het eerste apparaat gebruikte.

Kortom, een 24-bit *salt* is véél te klein in een omgeving met erg hoge data-rates zoals een draadloos netwerk. Dit probleem zou nog beperkt kunnen worden indien de originele WEP geregeld sleutels kon verversen, maar door het gebrek aan enig key management was dat dus uit den boze (want herinner je: de enige reden dat we IV's nodig hadden was omdat anders steeds dezelfde WEP-sleutel als seed werd gebruikt en dus alle keystreams gelijk zouden zijn. Door geregeld een andere sleutel te gebruiken zou onze kleine IV-lengte minder precair zijn, als we maar tijdig de sleutels verversen).

> **i** Opmerking
>
> Volgende tekst uit Wikipedia legt de **verjaardagenparadox** uit: "De verjaardagenparadox is een paradox uit de kansrekening, die een resultaat toont dat tegen de verwachting ingaat. Het gaat om de vraag hoe groot de kans is dat in een groep willekeurig gekozen mensen er (minstens) twee dezelfde verjaardag hebben. Het blijkt dat, onder enkele lichte veronderstellingen, deze kans al meer dan 50% is voor een groep van maar 23 mensen. Bij 57 mensen is de kans zelfs meer dan 99%."
>
> Beeld je nu in dat in plaats van mensen, je honderden pakketten hebt, niet met een verjaardag maar met een eigen IV: de kans op collisions, ook al is de IV 24 bit, wordt dus 50% bij reeds een 5000 tal pakketten.

4.3.3 Probleem 3: CRC

WEP gebruikt een *integrity checksum field* om te voorkomen dat een pakket wordt aangepast tijdens transmissie, namelijk een CRC-32 checksum. Dit was niet zo'n wijze keuze: CRCs zijn in het algemeen vooral bedoeld om random transmissiefouten te detecteren, niet bewuste aanpassingen. Daarnaast heeft de CRC-32 ook nefaste gevolgen in combinatie met RC4 waardoor bitflipaanvallen mogelijk zijn.

CRC-32 is een zogenaamde lineaire functie. Zonder in detail hierover in te gaan volstaat het te begrijpen dat we als gevolg hiervan het volgende hebben: *De CRC van een boodschap ge-XOR'd met de CRC van een andere boodschap is hetzelfde als de CRC nemen nadat beide boodschappen samen werden ge-XOR'd.*

Of beter gezegd:

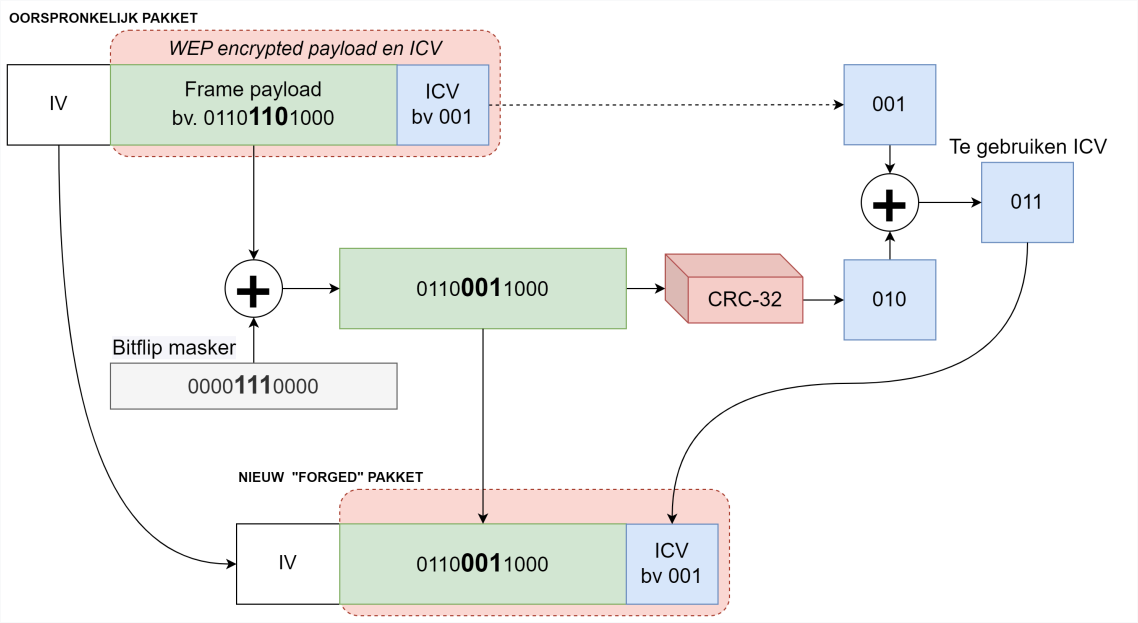
$CRC(boodschap_1) \oplus CRC(boodschap_2) = CRC(boodschap_1 \oplus boodschap_2)$

Door deze eigenschap kunnen we gecontroleerd aanpassingen aan pakketten doen, zonder daarmee de CRC te breken. Welgekome, bitflip aanval.

4.3.3.1 Bitflip aanval

- Om een bitflip aanval te doen heeft de aanvaller enkel **één geldig WEP frame** nodig. Hij hoeft zelfs niet te weten wat de inhoud ervan is, zolang het maar een geldig frame is.
- Vervolgens maakt de aanvaller een **bitflip masker**: dit bestaat uit een reeks 0'n, met enkele bits op 1 (het masker). Dit zijn de bits die geflipt zullen worden in de volgende stap: de XOR nemen het bitflip masker met het payload gedeelte van het oorspronkelijke pakket. Welke bits geflipt worden doet er niet

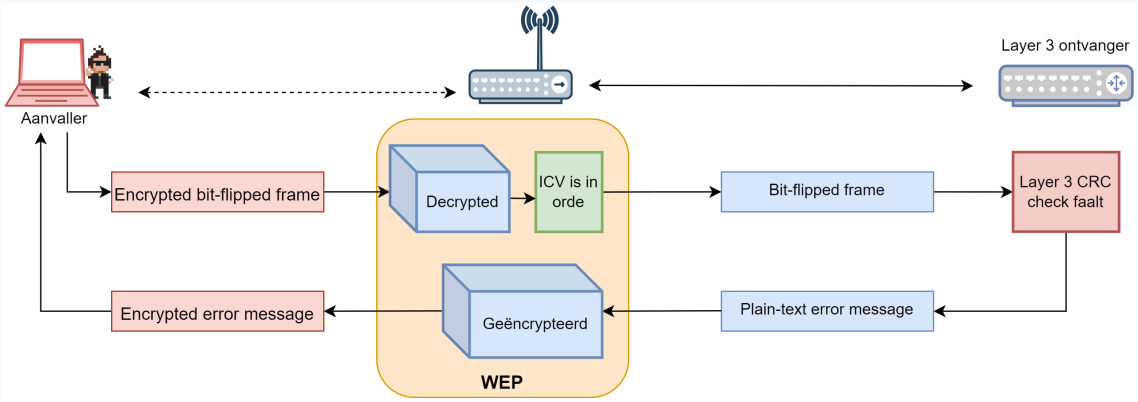
toe, integendeel: we willen net een **geldig WEP-frame maken maar mét foute data als payload**, zoals we zo meteen zullen zien.
- Deze nieuwe payload willen we nu in een geldig frame plaatsen, en dus hebben we een **geldige ICV** nodig. We doen dit door de ICV van de nieuwe payload te berekenen en deze te XOR'n met de, geëncrypteerde originele ICV.
- Het resultaat is een geldige, geëncrypteerde ICV voor de nieuwe payload. Bijgevolg hebben we een geldig frame kunnen maken dat door access points op het netwerk aanvaard zal worden.



Figuur 4.16: De bitflip aanval.

Wanneer een AP dergelijk pakket krijgt, zal het dit pakket braaf naar de volgende laag sturen aangezien het pakket een geldig ICV heeft, ook al bevat de inhoud *rommel*. Op de volgende laag komt nu een pakket aan dat duidelijk stuk is, en deze laag zal bijgevolg een foutboodschap genereren en terugsturen. De aanvaller krijgt deze boodschap in een WEP-frame aan. En alhoewel dit pakket geëncrypteerd is, weet de aanvaller de inhoud van dit pakket (daar hij heeft opgezocht wat de foutboodschap zal bevatten op de volgende laag volgens de standaard van die laag) en kan hij dus een geldige keystream te pakken krijgen:

$$c = k \oplus p \Leftrightarrow k = c \oplus p$$



Figuur 4.17: Layer 3 misbruiken door corrupte, geblitflipte pakketjes, een gekende foutboodschap te laten genereren.

Dankzij de bitflip aanval kan de aanvaller dus zonder problemen keystreams laten groeien zoals eerder uitgelegd.

4.3.4 Probleem 4: Sleutelmanagement

Het moge duidelijk zijn: constant dezelfde WEP-sleutel gebruiken, op meerdere apparaten, gedurende meerdere dagen, is vragen om problemen. Werknemers die ontslagen worden kunnen een potentiële sleutel-lekkage veroorzaken met alle gevolgen van dien. Administrators moeten manueel nieuwe sleutels invoeren bij de werkgevers, etc. Kortom, twee belangrijke oorzaken zorgen voor een nog lagere beveiligingsgraad van WEP dan er al was ten gevolge van de voorbije drie problemen (IV, RC4, CRC):

1. Er is **geen geautomatiseerd sleutel verversmechanisme**: idealiter worden de sleutels binnen één sessie vervangen voordat de verzameling van mogelijke IV's is opgeraakt.
2. Er is **geen gecentraliseerd sleutelmanagementsysteem**.

> **i** Opmerking
>
> Bart Preneel van de KUL/Cosic, één van België's meest vooraanstaande crypto-experts, zei ooit over WEP dat het het perfecte schoolvoorbeeld is van wat er allemaal kan fout lopen wanneer je beslist om zelf een nieuw crypto-algoritme te ontwikkelen.

4.4 Hoe WEP oplossen?

Rond 2001, nog geen twee jaar nadat de eerste wifi-standaard de wereld "het wonder van draadloze netwerken" bracht, werd duidelijk dat er dringend een oplossing moest verschijnen voor de vele veiligheidsproblemen. Wifi was alomtegenwoordig, zowel bij particulieren als in bedrijven, en dus moest een oplossing gezocht worden voor al die honderdduizenden bestaande apparaten die reeds in gebruik waren. Het IEEE kon moeilijk een nieuwe standaard uitschrijven die alle bestaande gebruikers in de kou zette. Er werd daarom besloten om twee pistes uit te werken:

- **WPA1**: een (tijdelijke) oplossing voor bestaande apparaten, rekening houdend met enkele duidelijke beperkingen.
- **WPA2**: de "ultieme oplossing" die een volledig nieuwe suite aan veiligheidsprotocollen zou bevatten voor toekomstige wifi-producten, maar die niet compatibel zou zijn met bestaande apparatuur.

> **Tip**
>
> WPA staat voor WiFi Protected Access standard.

De beide oplossingen zouden ook de IEEE 802.1X standaard omarmen. Deze standaard zou sleutelmanagement en gebruikersauthenticatie voorzien in combinatie met de nieuwe encryptie en integriteitsoplossingen van WPA1 of WPA2.

Omdat sleutelmanagement én de bijhorende 802.1X infrastructuur redelijk overkill konden zijn voor huis-tuin-en-keukengebruikers van wifi, besloot het IEEE om twee versies van zowel WPA1 en WPA2 te maken:

- **Personal**: de thuisversie voor gewone gebruikers waarbij met een gedeelde sleutel of passphrase wordt gewerkt (een zogenaamde **PSK** oftewel *pre-shared key*). Hierbij wordt géén 802.1X gebruikt.
- **Enterprise**: de versie voor (grote) bedrijven waar sleutelmanagement belangrijk is en dus 802.1X wordt gebruikt

We vatten hier alvast de belangrijkste verschillen tussen WEP, WPA1 en WPA2 samen.

Standaard	Encryptie	Integrity	Authenticatie & sleutelmanagement
WEP	RC4	CRC-32	Geen
WPA1-Personal	TKIP	Michael	pre-shared key
WPA1-Enterprise	TKIP	Michael	802.1X
WPA2-Personal	CCMP	CBC-MAC	pre-shared key
WPA2-Enterprise	CCMP	CBC-MAC	802.1X

> **i** Opmerking
>
> Ondertussen bestaat er ook WPA3, die we op het einde van dit hoofdstuk zullen bespreken. WPA1 en WPA2 zijn echter , historisch gezien, gerelateerd aan WEP en daarom bekijken we deze beiden samen in functie van WEP.

4.5 WPA1

WPA1 (de tussentijdse oplossing) werd ontwikkeld waarbij rekening werd gehouden met volgende beperkingen:

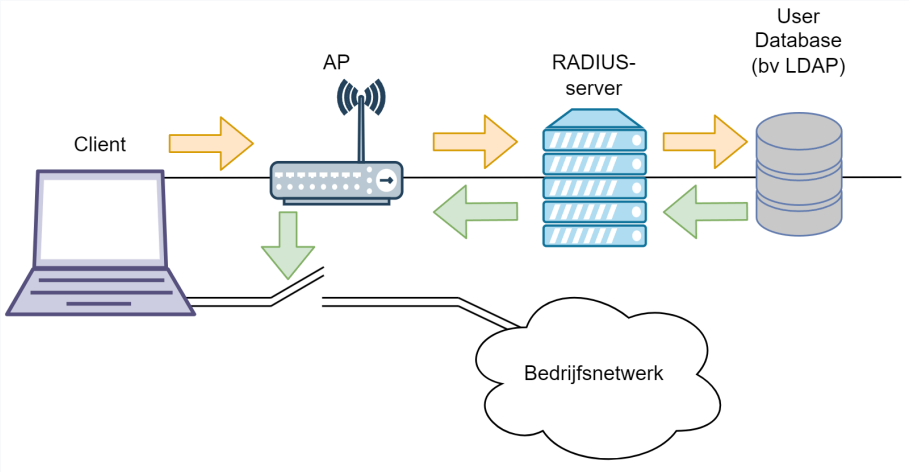
- Zoals verteld, miljoenen WEP-gebaseerde apparaten waren reeds in gebruik. Deze apparaten zouden met behulp van een firmware upgrade gepatcht moeten kunnen worden naar WPA1.
- De meeste AP's werkten met processoren die reeds quasi volcontinue tegen hun maximum capaciteit werkten. De extra algoritmes (van WPA1) die het AP moest draaien mochten maar een beperkte overhead creëren.
- De RC4 encryptie is deels *hardwired* in de hardware van het AP. Hierdoor kunnen bepaalde delen van WEP onmogelijk 'omzeild' worden en hangen we dus inherent vast aan WEP.

4.5.1 802.1X

De IEEE 802.1X standaard is ondertussen een veelgebruikte standaard in veel draadloze én bedrade netwerken. Het voorziet volgende zaken aan WPA1, WPA2 en WPA3 netwerken die in *Enterprise-mode* werken:

- (Mutual) **authentication**.
- Een **centraal user management** systeem.
- Een **veilige manier om geheime sleutels uit te wisselen**.

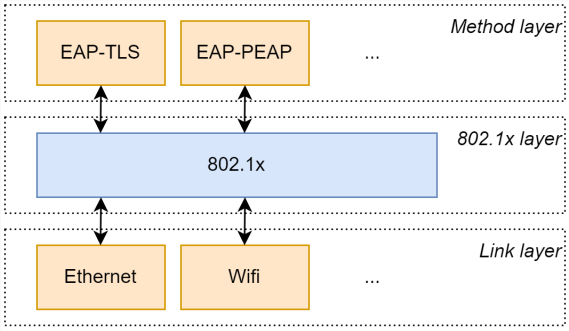
We gaan de volledige werking van de 802.1X standaard hier niet uit de doeken doen, dat zou ons te ver brengen. Het is echter nuttig om te begrijpen dat deze standaard werd gekozen omdat hij ervoor zorgt dat de AP's niet meer zelf de authenticatie moeten doen, maar dat ze gebruik maken van de bestaande (bedrade) authenticatie-infrastructuur van het bedrijf. De AP's zullen gewoon als een doorgeefluik aan de start tussen gebruiker en de authenticatie-server optreden en de boodschappen tussen beiden uitwisselen. Enkel wanneer het AP toestemming krijgt van de authenticatie-server (meestal een RADIUS server) zal het AP de eindgebruiker toegang tot het draadloze netwerk verschaffen.



Figuur 4.18: Port-based authenticatie met 802.1X.

802.1X zelf beschrijft niet hoe de authenticatie moet plaatsvinden: het is geen algoritme. Integendeel: het is een **framework** waar binnen andere algoritmen en standaarden, op maat van het bedrijf, kunnen ingeplugd worden. Hierdoor ontstaat een flexibel concept dat bedrijven (of diehard eindgebruikers) niet

verplicht om een bepaalde manier van authenticatie (en bijhorende soft- en hardware) te omarmen. 802.1X zorgt voor de vertaling van de authenticatie-boodschappen tussen enerzijds het netwerkprotocol (bv. Ethernet, wifi, maar ook Token Ring, etc.) en de *methode laag*. De methode-laag bevat het te gebruiken authenticatie-protocol en dient **EAP**-compatibel zijn.

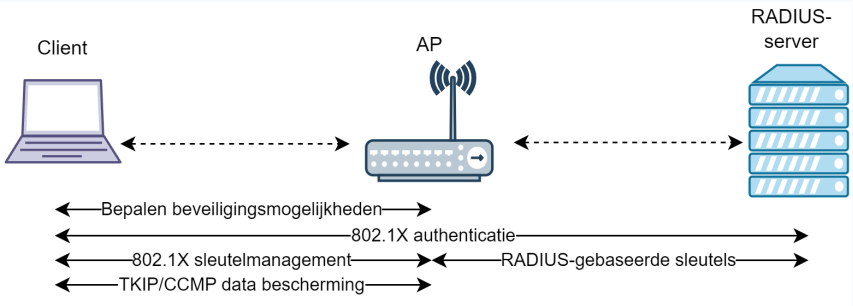


Figuur 4.19: De modulariteit van het 802.1X framework.

EAP oftewel *Extensible Authentication Protocol* is, zoals de naam doet vermoeden, een uitbreidbaar protocol van te gebruiken authenticatie-methoden. Afhankelijk van de keuze van het bedrijf kan voor een bepaald EAP-protocol gekozen worden, het ene is gebruiksvriendelijker en/of veiliger dan het andere. Uiteraard dienen zowel de client als de netwerkinfrastructuur compatibel te zijn met de gekozen EAP-methoden van het netwerk. Koop je dus een AP dat WPA2-Enterprise compatibel is, moet je nog steeds controleren of het AP compatibel is met de gekozen EAP-methode van het bedrijf.

De meest gebruikte EAP-methoden zijn:

- **EAP-TLS**: gebruikt een TLS tunnel om op een beveiligde manier te communiceren (we zagen TLS ook reeds aan het einde van crypto waar het gebruikt werd om HTTPS-trafiek te beveiligen). Hierbij gebeurt een certificaat-gebaseerde authenticatie.
- **EAP-TTLS** (*Tunneled TLS*): omdat niet alle eindgebruikers zich kunnen authenticeren aan de hand van een certificaat, voorziet TTLS authenticatie met behulp van een username/wachtwoord login. Hierbij wordt wel nog steeds een TLS tunnel gebruikt voor veilige communicatie, maar de gebruiker moet geen eigen certificaat bezitten. Ter info: EAP-TTLS is quasi hetzelfde als *Protected EAP* (PEAP), een ander EAP-protocol dat je soms zal zien passeren.



Figuur 4.20: Het authenticatie-proces op een wifi-netwerk met 802.1X (Enterprise-mode).

De voorgaande figuur toont de typische uitwisseling van boodschappen die plaatsvinden wanneer een client voor het eerst verbinding wil maken op een WPA1 of WPA2 Enterprise netwerk:

1. Client en access point zoeken samen welke EAP-methoden zij beiden kunnen gebruiken om de authenticatie te starten. Enkel indien ze samen tot één keuze kunnen komen wordt overgegaan naar stap 2.
2. Vanaf nu zal het AP enkel dienst doen als doorgeefluik tussen de client en de authenticatie-server.
3. Als de server genoeg bewijs heeft om de client te authenticeren, zal de server de nodige sleutels genereren en deze aan het AP geven.
4. Het AP zal vanaf nu instaan voor het verdere sleutelbeheer en wanneer nodig de sleutels verversen tijdens de sessie.

5. Wanneer de sleutels tussen client en AP zijn verwerkt, krijgt de client toegang tot het netwerk.

In wifi-netwerken met 802.1X worden twee sets van sleutels aangemaakt:

- **Sessiesleutels**, ook wel "*pairwise keys*" genoemd: deze zijn uniek per client en zijn enkel gekend door die client en het AP. Zoals de naam doet vermoeden zijn deze sleutels enkel geldig tijdens de net opgezette sessie.
- **Groepssleutels**, ook wel "*group keys*" genoemd: deze worden tussen alle cliënten van hetzelfde AP gedeeld en worden gebruikt voor multicast trafiek.

Indien het *dynamic key exhange protocol* is geconfigureerd dan zal de authenticatie de sessiesleutels eenmalig aanmaken en doorsturen. Daarna zullen specifieke encryptie-sleutels gegenereerd worden bij de client en AP gebaseerd op deze sessiesleutel. De client (en AP) kan dan zelf, automatisch, op gepaste momenten nieuwe encryptie-sleutels genereren.

4.5.2 TKIP

Bij WPA1, wanneer de nodige sleutels zijn uitgewisseld, is het tijd om data te encrypteren. Dit gebeurt door middel van het **Temporal Key Integrity Protocol** (TKIP), wat een suite van algoritmes is dat als een wrapper rond WEP wordt gelegd om zo encryptie aan te bieden en daarbij de fouten van WEP minimaliseert.

TKIP omhult WEP met volgende zaken:

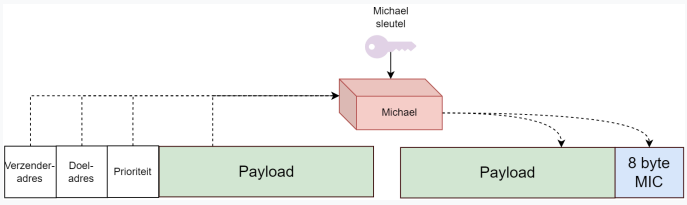
1. Een nieuwe **integrity check genaamd Michael** dat een *message integrity code* (MIC) genereert die wél bestand is tegen bitflip aanvallen.
2. Een nieuwe manier van **IV selectie** die replay aanvallen voorkomt.
3. Een **per-pakket sleutel mixing** algoritme dat het probleem met *weak keys* in RC4 oplost.
4. Een *re-keying* mechanisme dat ongeveer elke 10000 pakketjes een nieuwe sleutel doet genereren.

4.5.2.1 Michael the Mic

Om bewuste aanpassingen aan de payload van een frame te detecteren gebruikt WPA1 een *message integrity check* (**MIC**). Dit is een cryptografisch sterker concept dan de originele CRC checks en wordt verzorgd door het "Michael" algoritme.

> **i** Opmerking
>
> In de literatuur wordt meestal gesproken over "Message authentication codes" of MAC's. Echter, in de IEEE 802 standaarden wordt MAC reeds gebruikt voor *media access control* en werd er dus gekozen voor MIC.

"Michael" berekent de MIC van een payload maar gebruikt hierbij ook de Michael sleutel (een afgeleide van de authenticatie-sleutel), het adres van de verzender én ontvanger. Hierdoor wordt het voor een aanvaller veel moeilijker om een dergelijke MIC na te bootsen, laat staan te *replayen* (vergelijk dit met de originele CRC-32 die enkel de payload gebruikt om de checksum te berekenen).



Figuur 4.21: Het aanmaken van een MIC met Michael.

Wanneer TKIP twee foute MICs na elkaar detecteert, gaat het er van uit dat er een aanval bezig is. Volgende stappen worden dan ogenblikkelijk uitgevoerd door de client:

1. Alle huidige sleutels worden verwijderd.
2. De client verbreekt de verbinding.
3. Er wordt één minuut gewacht voor een nieuwe verbinding (*association*) wordt opgezet.

4.5.2.2 IV selectie verbetering

Om te voorkomen dat fabrikanten weer naïeve oplossingen voor de IV selectie implementeerden, legde WPA1 nu de regels op. Een ontvangen pakket zal pas aanvaard worden indien de IV van het pakket op de IV van het vorige pakket volgt. Uiteraard zit er een kleine marge om hertransmissies toe te staan, maar een pakket met bijvoorbeeld IV 1110 zal nooit aanvaard worden als het AP vlak ervoor een pakket met IV 3789 heeft aangekregen.

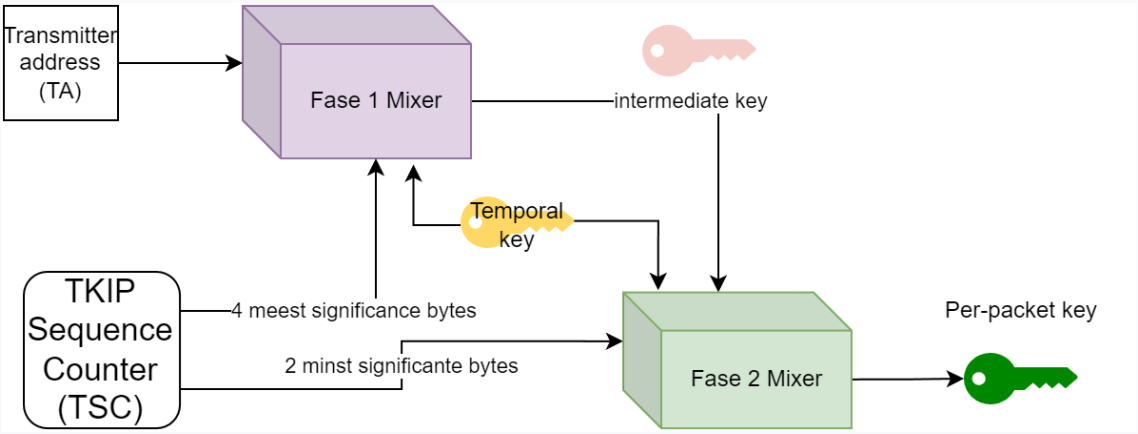
Daarnaast wordt ook de IV lengte gevoelig vergroot. TKIP hanteert namelijk een 48-bit IV, genaamd de *TKIP sequence counter* (TSC). Voor de WEP-encryptie worden de drie minst significante bytes van de TSC gebruikt als IV, waardoor deze compatibel blijft met de oorspronkelijke WEP-specificatie. Door de resterende bits van de TSC te benutten voor replay-bescherming en sleutelmanagement, vermindert TKIP significant het risico op IV-collisions, een grote zwakte van WEP.

4.5.2.3 Key mixing en re-keying

Om weak keys te voorkomen gebruikt een TKIP een *key mixing function* dat zal resulteren in een **temporal of per-pakket sleutel** die dienst zal doen als vervanger voor de WEP-sleutel. Deze sleutel zal geregeld ververst worden (vandaar *temporal* oftewel tijdelijk) en heeft dus een beperkte levensduur voor actieve aanvallen.

Het mixen van de sleutel gebeurt in twee fases, waarbij iedere fase een specifieke zwakte van WEP indijkt:

- Fase 1: zorgt ervoor dat alle clients een eigen sleutel hebben doordat het verzender adres (transmitter address (TA)). wordt toegevoegd aan de temporal key.
- Fase 2: zorgt voor een 'per-pakket' sleutel waarbij kennis van de IV niet meer door de aanvallers kan misbruikt worden.



Figuur 4.22: Het mengen van de verschillende sleutels naar een sleutel die ieder pakketje verandert.

Fase 1 mix

Het **transmitter adres** (het MAC-adres van het apparaat dat het pakket uitzendt) wordt gecombineerd met de **Temporal Key**. Deze sleutel is ofwel afgeleid van een PSK-sleutel in de Personal-modus, ofwel afgeleid van een PMK-sleutel die tijdens de authenticatie via 802.1X is verkregen. Daarnaast worden de vier meest significante bytes van de TKIP Sequence Counter (TSC) toegevoegd. Het combineren van deze inputs gebeurt via een iteratief proces dat hashing-achtige technieken toepast, waaronder XOR-operaties, modulaire optellingen en bitverschuivingen. Het resultaat is een intermediate sleutel die dient als input voor de volgende fase van het sleutelbeheerproces.

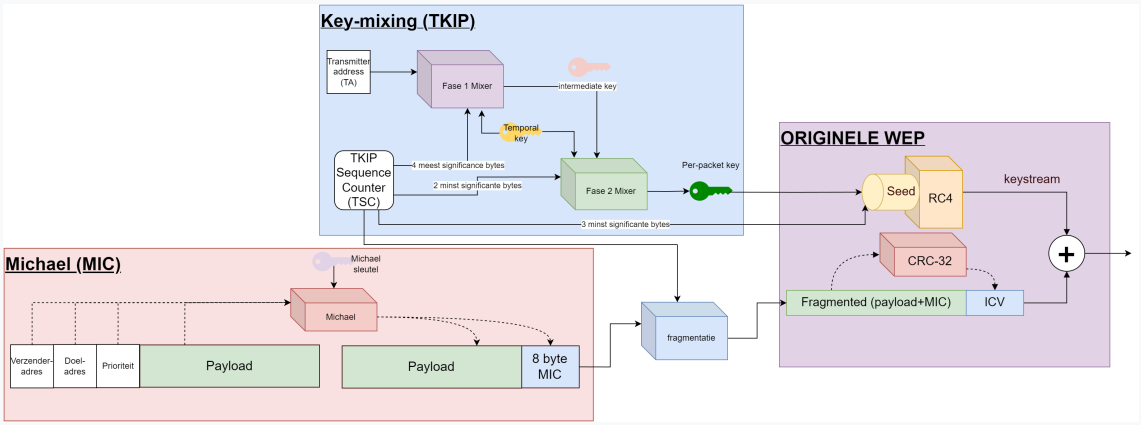
> Tip
>
> Bij Michael spraken we over een destion adres (bron adres) in de afbeelding. Terwijl we nu over een transmitter adres spreken. Dit is geen fout, maar een bewuste keuze:
>
> - TA (Transmitter Address): Verwijst naar het apparaat dat het pakket op dat moment uitzendt.
> - SA (Source Address): Verwijst naar het apparaat dat het pakket oorspronkelijk heeft gemaakt.

Fase 2 mixing

De intermediate sleutel, verkregen uit de Phase 1 Key Mixing, wordt gecombineerd met de twee minst significante bytes van de TSC en de Temporal Key. Dit proces is ontworpen om de tijdelijke sleutel verder te versterken en een unieke encryptiesleutel te genereren voor elk datapakket. De combinatie van deze inputs wordt uitgevoerd via een iteratief proces dat gebruikmaakt van effectieve cryptografische technieken, zoals XOR-operaties, modulaire optellingen en bitverschuivingen. Het resultaat van deze fase is de definitieve RC4-sleutel, ook wel de WEP-seed genoemd. Deze sleutel wordt samen met de drie minst significante bytes van de TSC (die de verbeterde Initialization Vector vormen) ingevoerd in het RC4-algoritme om de keystream te genereren die nodig is voor encryptie in WEP.

4.5.3 Alle blokjes samen

Finaal kunnen we vervolgens WPA1 visualiseren, waarbij duidelijk is dat we vooral een wrapper rond WEP hebben verkregen, maar dat WEP nog steeds het hart van het systeem is.



Figuur 4.23: WPA1 in volle glorie.

> **Opmerking**
>
> Het fragmenteren met de TSC is noodzakelijk vanwege de volgende redenen:
>
> - Wanneer een groot pakket wordt opgesplitst in meerdere fragmenten, moet elk fragment uniek zijn.
> - De TSC fungeert als een unieke identificatie voor elk fragment omdat het incrementeert bij elke transmissie.
> - Dit voorkomt verwarring of overlap tussen pakketten en maakt het mogelijk voor de ontvanger om de fragmenten correct terug samen te stellen.

4.5.4 Are we there yet?!

In 2009 verschenen er al enkele exploits die WPA1-Personal misbruikten waardoor aanvallers de WPA passphrase (de PSK) konden achterhalen door de handshake aan de start van een sessie te capteren. Deze aanvallen waren echter niet zo efficiënt als de WEP-aanvallen en vereisten een aanzienlijke hoeveelheid data om de PSK te achterhalen. Het was echter duidelijk dat WPA1-Personal niet de ultieme oplossing was voor het WEP-probleem.

> **Tip**
>
> Bekijk coWPAtty en Aircrack om WPA1-Personal te *hacken*.

4.6 WPA2

De finale oplossing voor het WEP-veiligheidsprobleem werd gegoten in WPA2 waarvan AES het hart vormt. Toen de standaard werd ontwikkeld was AES zonet verschenen en het bleek de ideale toepassing voor WPA2. Hierbij werden twee onderdelen van AES gebruikt namelijk:

1. AES met *counter mode* voor de encryptie.
2. AES in CBC-MAC mode voor integriteit en authentication.

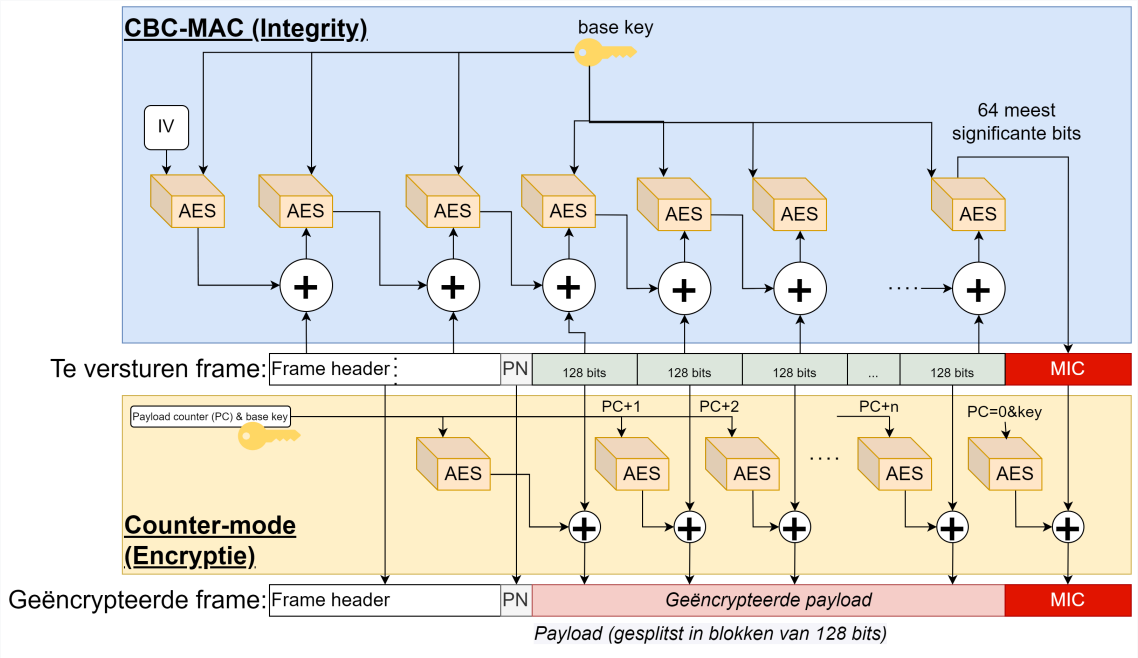
Samen wordt dit encryptieproces **CCMP** genoemd, wat staat voor *"Counter Mode Cipher Block Chaining Message Authentication Code Protocol"*

Voor de sleuteldistributie en management werd, zoals vermeld, gebruik gemaakt van de 802.1X standaard.

> Tip
>
> Toen WPA2 uitkwam ontdekte men dat toch aardig wat bestaande hardware kon gepatcht worden om ook WPA2-compatibel te zijn. Dit was een opsteker voor velen, daar de originele premise van WPA2 net was dat ze geen rekening zouden houden met de bestaande hardware die reeds op de markt was.

4.6.1 CCMP

CCMP gebruikt, net als TKIP, een 48-bit IV die *packet number* (PN) werd genoemd. Deze PN wordt, samen met andere informatie, gebruikt om de AES encryptie van een seed te voorzien. Hierbij wordt het frame (en de header) in blokken van 128 bit verdeeld en zo blok per blok verwerkt. De encryptie gebeurt parallel met het berekenen van de MIC (de ICV in WEP) die finaal achteraan als een laatste blok ook mee wordt geëncrypteerd. Ook nu wordt met een tijdelijke sleutel gewerkt die gebaseerd is op de hoofdsleutel verkregen via 802.1X (enterprise mode) of de passphrase (in personal mode).
Die laatste zin impliceert een ander belangrijk veiligheidsverschil tussen enterprise en personal mode: in personal modus wordt een passphrase sleutel gedeeld met alle gebruikers op het netwerk. Hierdoor kan iemand die in het bezit is van deze passphrase ook de data decrypteren van de andere gebruikers, iets wat onmogelijk is in enterprise modus.



Figuur 4.24: CCMP: confidentiality en integrity in één.

In de figuur zie je duidelijk de elegantie van het systeem en hoe de integriteit in parallel wordt berekend met de encryptie. De MIC wordt berekend met behulp van CBC-MAC, wat staat voor **cipherblock chaining - message authentication code**, een veelzeggende naam. Zoals we uit het hoofdstuk crypto weten, zal er bij CBC een keten van encrypties starten, waarbij de encryptie van het huidige blok ge-XOR'd wordt met het resultaat van de encryptie van het vorige blok. Het finale geëncrypteerde cipherblock zal daarbij dienst doen als MIC. Bij de minste bitfout ergens in de payload of header zal een totaal andere MIC gegenereerd worden.

Om de payload te encrypteren (*merk op dat de header niét geëncrypteerd wordt, het wordt enkel mee betrokken om de MIC te berekenen*) wordt de **counter mode** van AES gebruikt. Hierbij worden de blokken onafhankelijk van de andere blokken geëncrypteerd en zal een *payload counter* (PC) telkens met één verhoogd worden en als extra seed voor de AES-motor van het huidige blok dienen (samen met de te gebruiken sleutel).

4.6.2 Helaas, aan alles komt een einde

Tot 2017 ging alles goed. De AES standaard was al jaren een robuuste standaard gebleken en werd op vele plekken nog steeds gebruikt. En dit zou ook bij wifi zou zijn geweest, waren het niet dat in mei 2017 een Belgische onderzoeker, Mathy Vanhoef, de bevindingen van z'n onderzoek publiceerde. Hij had helaas een belangrijke fout gevonden in de WPA2 standaard. Deze had niets te maken met AES - dat blijft een stevige standaard zijn - maar wel de manier waarop een bepaalde uitwisseling van bepaalde berichten tijdens de initiële handshake gebeuren. Deze aanvallen worden beschreven én gedemonstreerd op krackattacks.com en verplichtten de IEEE om te beginnen werken aan een opvolger voor WPA2.

> Tip
>
> Een lek zoals krackattack vereist natuurlijk een snelle reactie van de vendors. Hoe sneller zij een patch uitbrengen, hoe sneller het lek kan gedicht worden. Maar wat als je een wifi-kaart hebt die al vele jaren oud is en waarvan de fabrikant misschien niet meer bestaat? Dit probleem zien we geregeld opduiken en wordt nog groter wanneer we beseffen dat ook de interne elektronica (de chips) soms gepatcht moeten worden. In het hoofdstuk rond IoT Security gaan we hier dieper op in.

> Tip
>
> De personal mode van WPA1 en WPA2 zal altijd gevoelig zijn voor dictionary aanvallen. Aangezien de gedeelde sleutel (de pre-shared key) een gedeeld geheim is, kunnen andere gebruikers proberen dit te achterhalen. Het is dus belangrijk dat, indien je in personal modus WPA hanteert, je een sleutel kiest die voldoet aan de typische vereisten van een goed wachtwoord.

4.7 WPA 3 ("Wifi 6")

In 2018 kwam de Wifi Alliance uit met de opvolger van WPA2, de titel, je raadt het nooit, was uiteraard WPA3, maar werd ook wel Wifi 6 genoemd. De standaard gaan we hier niet zo gedetailleerd bespreken, het volstaat te begrijpen dat ze

- op veiligheidsniveau state-of-the-art was.
- rekening hield met de noden van 21e-eeuwse draadloze netwerken (denk maar aan Internet-of-Things apparaten, beveiligde publieke hotspots, etc.).

> Tip
>
> De Wifi Alliance is in het leven geroepen als een organisatie die ervoor zorgde dat producten wel officieel konden claimen dat hun producten conform een IEEE standaard waren. Enkel wanneer hun producten de nodige tests van de Wifi Alliance aflegden kon het product het label van "Wifi Alliance compatibel" product dragen.

Ook in WPA3 werden twee modes voorzien: een personal en een enterprise mode. Enkele van de interessantste verbeteringen zijn:

- *Simultaneous Authentication of Equals (SAE)*: een nieuw cryptografisch concept waarbij in de personal mode authenticatie veel veiliger kan plaatsvinden dan voorheen.
- Resistent tegen offline dictionary attacks.
- *Forward secrecy*: zelfs als de aanvaller de wifi-sleutel van oude gecapteerde pakketten vindt zal hij deze toch niet kunnen decrypteren. Het aloude “store now, decrypt later” is dus niet van toepassing op WPA3.
- *Wifi easy connect*: een gebruiksvriendelijke manier om Internet-of-Things apparaten met het netwerk te verbinden.
- *Wifi enhanced open*: publieke hotspots blijven publiek, maar iedere client heeft z’n eigen veilige kanaal met het AP. Gedaan zijn de dagen van je in de Starbucks zetten om zo privé-trafiek van omstaanders te sniffen.
- *Geauthenticeerde encryptie* gebruik makend van *“256-bit Galois/Counter Mode Protocol (GCMP-256)”* een cryptocipher dat we hier niet uit de doeken gaan doen (maar geef toe, met zo’n naam klinkt het toch ogenblikkelijk extra veilig!).
- Gebruikt de meest moderne veilige authenticatie- en sleuteldistributiemethoden mogelijk binnen 802.1X (HMAC, HMAC-SHA384 en ECDH)

4.8 Samenvatting

Draadloze netwerken vervangen de fysieke beveiliging van een kabel door cryptografische beveiliging — en dat is een harde les geweest:

- **WEP** faalde op alle fronten: verkeerd gebruik van RC4, te korte IV (24 bit), zwakke CRC-integriteit en geen sleutelbeheer. Binnen enkele minuten te kraken.
- **WPA1/TKIP** was een softwarepatch op dezelfde hardware: langere IV’s, *key mixing* en **Michael** voor integriteit. Tussenoplossing — niet toekomstvast.
- **WPA2** introduceerde **CCMP** op basis van **AES** en werd jarenlang als veilig beschouwd, tot **KRACK** (Mathy Vanhoef, 2017) een fout in de handshake blootlegde.
- **WPA3** (“Wifi 6”) lost de oude pijnpunten op: **SAE** tegen offline dictionary-aanvallen, **forward secrecy** en **Wifi Enhanced Open** voor veilige publieke hotspots.
- **802.1X** (enterprise-mode) voorziet per-gebruiker-sleutels en centrale authenticatie — een must voor elk bedrijfsnetwerk.

Vier praktische take-aways: **kies WPA3 waar mogelijk**, **vermijd WEP/WPA1 als de pest**, **gebruik enterprise-mode boven personal-mode in bedrijfscontext**, en **hou access points fysiek én softwarematig gepatcht**.

5. Authenticatie

i Leerdoelen

Na dit hoofdstuk kan je:

1. De **drie gouden regels van wachtwoordbeheer** toepassen op een gegeven authenticatie-ontwerp.
2. De werking van **hashing, salting en een rainbow-table-aanval** uitleggen — en tonen waarom salting die aanval onbruikbaar maakt.
3. **CRAM, SCRAM en passkeys** onderling situeren en aangeven welk probleem elk protocol oplost.
4. De vier **MFA-factoren** (weten, hebben, zijn, waar/wanneer) combineren voor een gepaste authenticatiesterkte in een gegeven context.
5. Uitleggen hoe **TOTP** werkt en benoemen waarom **passkeys** veiliger zijn dan TOTP tegen real-time phishing.

Bewijzen wie je bent om toegang te krijgen tot een website of applicatie heet **authenticatie**. Vervolgens, afhankelijk van wie je bent, zal je bepaalde rechten toegewezen krijgen die bepalen wat je wel en niet kunt doen op de website of applicatie, dit heet **autorisatie**. In dit hoofdstuk gaan we ons toespitsen op het eerste deel van dit proces: de authenticatie. Hierbij gaan we vooral kijken hoe we als web- of applicatiebeheerders op een veilige manier moeten omgaan met de login-informatie van gebruikers.

We weten al dat je geheime sleutel een belangrijk onderdeel is in heel veel aspecten van cybersecurity. Het **weten van de geheime sleutel** is een eerste vorm van authenticatie (maar uiteraard niet de beste). Je login gegevens die je gebruikt om toegang te krijgen tot een website bestaan in primaire vorm meestal uit een combinatie van gebruikersnaam en wachtwoord. Het wachtwoord in dit verhaal is kortom gewoon een ander woord voor je geheime sleutel. Als beheerder is het dan ook essentieel dat we uitermate veilig omgaan met de wachtwoorden van gebruiker. We zouden niet willen dat alle login gegevens van onze gebruikers in verkeerde handen vallen.

5.1 Veilige wachtwoorden

Ongeacht de veiligheden die we inbouwen als cyberboswachter, veel blijft afhangen van de manier waarop eindgebruikers omgaan met hun wachtwoorden. Volgende regels worden continu niét gehanteerd, met alle gevolgen van dien:

- Hergebruik nooit een wachtwoord. In principe heb je één wachtwoord pér service (applicatie, website, etc.).
- Gebruik geen wachtwoorden die in *dictionaries* staan. Maar overweeg volledig random gegenereerde wachtwoorden.
- Zorg ervoor dat je wachtwoorden lang genoeg zijn (minimum 16 tekens).
- Zorg ervoor dat je wachtwoorden steeds een combinatie van cijfers, letters (grote én kleine) en leestekens zijn.

Trouwens, herinner je je de McCumber kubus waarin we benadrukten dat technologie maar één aspect is om CIA toe te passen op je data in z'n drie primaire vormen? Het zal je niet verbazen dat cybercriminelen niet altijd gaan proberen databanken aan te vallen om wachtwoorden van gebruikers te pakken te krijgen. Als zij een specifiek doelwit hebben dan gaan ze vaak op andere manieren te werk:

- **Password spraying**: hierbij gaat de hacker een (beperkte) lijst van veelgebruikte wachtwoorden testen op een grote groep *useraccounts* van een bepaalde website, in de hoop een *hit* te hebben (*"spray and pray"*).
- **(Spear) phishing**: bij phishing hanteert de aanvaller de goedgelovigheid of onoplettendheid van de gebruiker om een ogenschijnlijk betrouwbare mail of bericht te sturen met daarin een link naar een pagina die malware installeert of een fake login scherm toont. Bij spear phishing gebruikt de aanvaller

geen massmail, maar gaat hij juist gericht één specifiek doelwit een op maat gemaakte mail of bericht sturen. Spear phishing is heden ten dage één van dé **social engineering** aanvallen bij uitstek.
- **Keyloggers**: als de aanvaller toegang heeft tot de computer (wat uiteraard van over het netwerk kan) dan kan hij een (permanente) keylogger installeren die alle toetsaanslagen op het systeem opneemt. Nadien kan de aanvaller deze logs dan analyseren in de hoop zo ook het wachtwoord of andere gevoelige informatie terug te vinden.

5.2 Hoe wachtwoorden opslaan

De aanvallen die we zonet besproken hebben (spraying, phishing, keyloggers) zijn hoofdzakelijk **online aanvallen**: de aanvaller probeert actief in te loggen of onderschept wachtwoorden bij de bron. Daarnaast bestaan er **offline aanvallen**, waarbij de aanvaller — via een SQL injection, een insider of een datalek — (leestoegang tot) de gebruikersdatabank heeft bemachtigd. Vanaf nu plaatsen we ons in de schoenen van de *beheerder* en gaan we uit van het ergste: **ga er van uit dat jouw databank vroeg of laat zal lekken**. Hoe moet je dan als cyberboswachter de login gegevens van je gebruikers bewaren zodat zo'n lek minimale schade oplevert? We gaan een soort *bottom-up* aanpak hanteren, waarbij we beginnen met de meest naïeve oplossing en telkens verbeteringen zullen aanbrengen.

5.2.1 Paswoorden als plaintext

In het prille begin van het Internet gebeurde dit quasi overal: de login-databank had twee kolommen:

1. gebruikersnaam.
2. gebruikerswachtwoord.

De wachtwoorden in kolom 2 stonden er zoals ze waren. Als een gebruiker wilde inloggen op dit soort websites dan moest hij z'n wachtwoord verzenden en dan ging de *backend* controleren of het ingezonden wachtwoord overeen kwam met het wachtwoord in de database. Het spreekt voor zich dat dit soort databanken van gigantische waarde zijn voor aanvallers: van zodra ze de databank hebben te pakken hebben ze alle wachtwoorden van alle gebruikers! Profit!

Paswoorden mogen nooit in onbeveiligde, leesbare vorm in een databank staan! Wanneer dit wel zo is dan kan je beter ogenblikkelijk je account bij die service deleten. Want alhoewel deze aanpak al lang bestaat en er al bijna even lang van geweten is dat deze erg onveilig is, toch zijn er nog steeds ontelbare websites en applicaties die hieraan zondigen. Als ze dus jouw wachtwoord zo behandelen, dan is de kans reëel dat ook hun andere veiligheidsdiensten niet om over naar huis te schrijven zijn.

Een goede manier om te weten of een service op deze manier werkt is gebruik maken van de *"Ik ben m'n wachtwoord vergeten"*-knop. Als je deze knop gebruikt en je krijgt een e-mail met daarin jouw originele wachtwoord, dan kan je er zeker van zijn dat de service jouw wachtwoord op deze manier bewaart. In principe zou een service NOOIT jouw wachtwoord moeten kunnen zien. We gaan zelfs zien dat **jouw wachtwoord nooit je computer mag verlaten**, laat staan dat deze beschikbaar is als plaintext in een database.

> ⚠ Waarschuwing
>
> **Wie zou nu zo dom zijn?** In 2019 onthulde Facebook dat het jarenlang wachtwoorden van **honderden miljoenen gebruikers** (schattingen tussen 200 en 600 miljoen) gewoon in plaintext had bewaard. De bestanden waren toegankelijk voor zo'n 20.000 Facebook-medewerkers. Zelfs de grootste techbedrijven vallen dus soms in deze klassieke val. Meer lezen: theverge.com.

5.2.2 Paswoord hashing

Door een wachtwoord te hashen kunnen we de wachtwoorden al iets veiliger bewaren. Een gebruiker die zich wenst aan te melden bij een systeem dat met wachtwoord hashes werkt zal nu op zijn lokale systeem z'n hash moeten genereren en dit over het netwerk doorsturen. De service zal deze ontvangen hash vergelijken met de waarde die in de database staat, en indien deze gelijk is dan wordt verondersteld dat de gebruiker het juiste wachtwoord kende.

We versturen dus niet meer het wachtwoord over het netwerk, maar we zijn nu wel vatbaar voor een **pass-the-hash** aanval. Het volstaat om een geldige combinatie van gebruikersnaam en hash te capteren

en deze vervolgens te gebruiken om ergens in te loggen. De aanvaller heeft hierbij geen kennis nodig van het originele wachtwoord.

> Tip
>
> Door een secure hash van een wachtwoord te genereren creëren we een stuk tekst dat niet terug naar het originele wachtwoord kan omgezet worden. In theorie zal ieder wachtwoord een andere hash creëren. Uiteraard kunnen er toch twee zaken zich voordoen:
>
> 1. Twee totaal verschillende wachtwoorden genereren dezelfde hash, een zogenaamde *collision*.
> 2. Twee mensen kiezen hetzelfde wachtwoord en zullen dus ook dezelfde hash genereren.
>
> Dit probleem gaan we verderop oplossen met behulp van *salting*.

> Waarschuwing
>
> Veel websites genereren wel degelijk de hash aan serverzijde. Het verschil hier is echter dat ze eerst een beveiligde TLS tunnel hebben opgezet waarover het wachtwoord werd verstuurd. Het laat echter de website/service toe om meer controle te hebben over de hash-generatie.

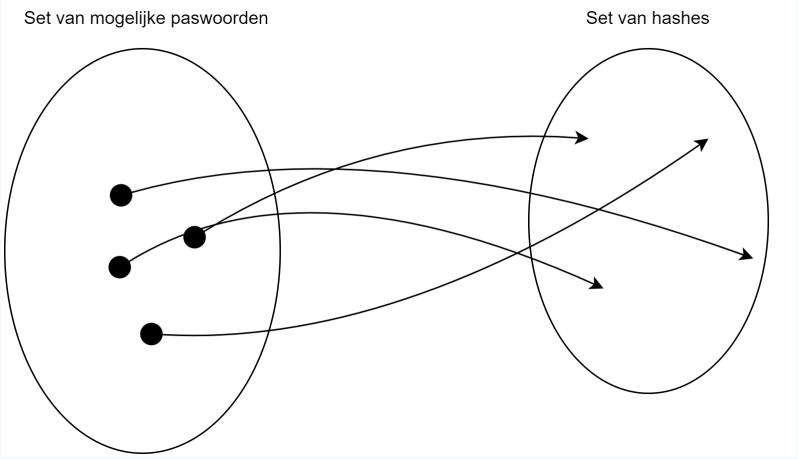
5.2.3 Rainbow table attack

Als de database door aanvallers gestolen wordt dan zitten we ook een tikkeltje veiliger als voorheen (de pass-the-hash aanval zal uiteraard nu zeker werken) indien de aanvaller de wachtwoorden van gebruikers nodig heeft (om bijvoorbeeld vervolgens op een ander systeem te gebruiken). De aanvaller zal een brute-force of dictionary attack moeten uitvoeren om te ontdekken welk wachtwoord resulteert in welke hash. Dit kan een erg tijdrovend proces zijn want enkele veelgebruikte hashing algoritmen (onder andere *scrypt*, *bcrypt* en *pbkdf2*) zijn *by design* zodanig geschreven dat deze erg traag werken. Dit zorgt ervoor dat de tijd om één hash te genereren geen voelbaar verschil geeft, maar wanneer een aanvaller er duizenden per seconden wil kunnen testen, dan zal het algoritme als een stevige ***timebottleneck*** optreden. De aanvaller zou dan in de plaats vooraf alle hashes kunnen *precomputen* als alternatief. Dit heeft dan weer voor gevolg dat zo'n lijst gigantisch groot is en er dus een ***memorybottleneck*** optreedt.

> Tip
>
> **Even concreet.** Stel dat gebruikers enkel wachtwoorden van 8 kleine letters mogen kiezen. Dat geeft $26^8 \approx 2{\cdot}10^{11}$ mogelijke combinaties.
>
> - Bij 1 miljoen hashes per seconde kost het gemiddeld **2,4 dagen** om de volledige ruimte te doorzoeken — nog te overzien, maar met een moderne GPU gaat het veel sneller (commerciële tools haalden in 2011 al **2,8 miljard** wachtwoordpogingen per seconde).
> - De alternatieve piste — álle hashes vooraf precomputen — vraagt **~1,46 TB** opslag, en dat is nog enkel voor deze beperkte set van 8 kleine letters. Ter vergelijking: alle Google-servers tezamen bewaren in de orde van 10^{15} bytes (1 petabyte).
>
> Conclusie: puur bruteforcen *of* puur precomputen loont nauwelijks. Rainbow tables zoeken bewust een **compromis** tussen beide uitersten.

De aanvaller zit dus met het dilemma (tijd versus geheugen) tussen hashen berekenen ter plekke, wat erg traag zal gaan, oftewel alle mogelijke hashes op voorhand berekenen, wat veel geheugenplek vereist. Via een **rainbow table attack** krijgt de aanvaller echter een handig instrument in handen dat een compromis tussen beide bottlenecks aanbiedt.

Een rainbow table is een tabel van precomputed hashes, maar waarvan we ze niet allemaal moeten bewaren om toch over een grotere set te beschikken dan die dat in de tabel bewaard worden. Je zou het kunnen vergelijken met een gecomprimeerde lijst van de getallen van 1 tot en met 101, waarbij we enkel het start (1) en eindgetal (101) bewaren, en dan erbij zeggen dat ieder volgend getal het vorige +2 is.

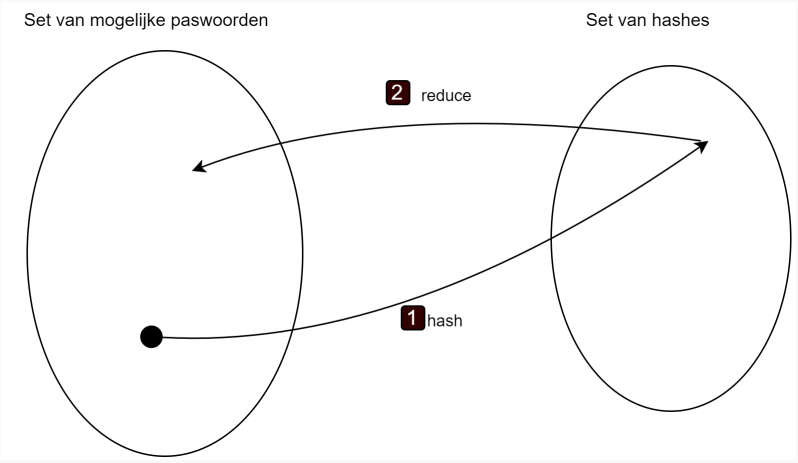


Figuur 5.1: Ieder wachtwoord mapt naar exact één hash.

Een rainbow table stel je als volgt op:

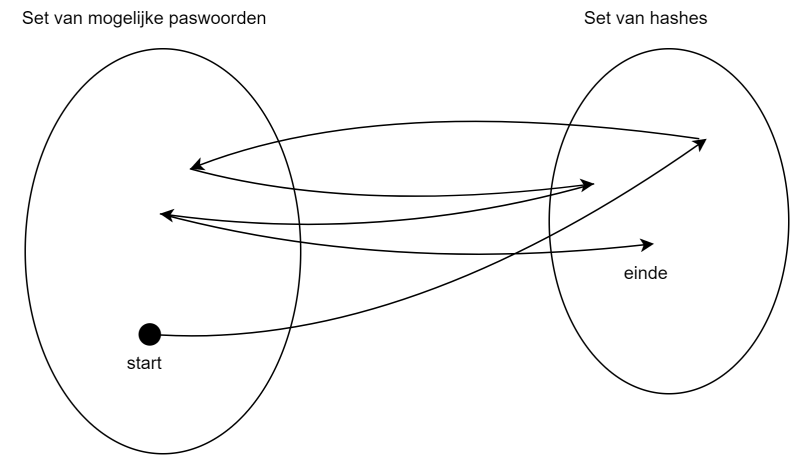
- Je kiest een startpunt, zijnde één van de mogelijk wachtwoorden uit de set van wachtwoorden waarvoor je een rainbow table wilt opstellen (zie figuur hier voor)
- Je genereert de hash van dit gekozen wachtwoord.
- Je past nu op deze hash een *reduction* functie toe. Dit is een zelfgekozen mapping van de verkregen hash terug naar een wachtwoord uit de set van mogelijk wachtwoorden.

Als reductiefunctie zou je bijvoorbeeld kunnen beslissen om de hash om te zetten naar een getal (bv. door de som van de ASCII-waarden van de letters van de hash te nemen) en dit getal dan te gebruiken als index die bepaalt welk wachtwoord je uit de wachtwoordenset gaat kiezen.

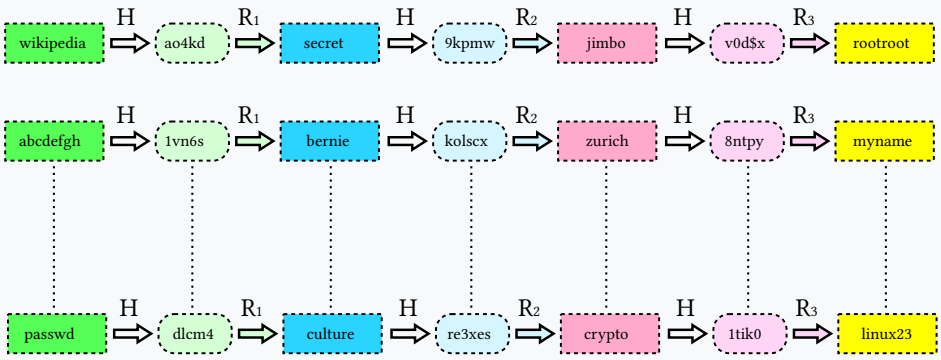


Figuur 5.2: Van de hash via de reductiefunctie terug naar een ander wachtwoord.

- Van dit nieuwe wachtwoord genereer je weer een hash en pas je opnieuw de reduction functie toe.
- Die combinatie hash+reductie blijf je X aantal keer herhalen tot je een lange lijst hebt (een zogenaamde *chain* van bijvoorbeeld 10000 elementen).
- Finaal hou je nu van deze lijst enkel het startpunt bij (het gekozen wachtwoord uit de set) en de allerlaatste gegenereerde hash.



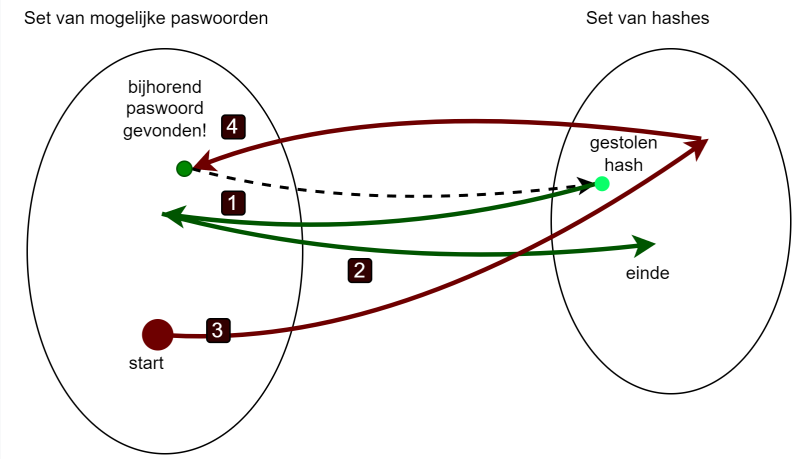
Figuur 5.3: Voorbeeld van een lijst opeenvolgende wachtwoorden en hun hashes.



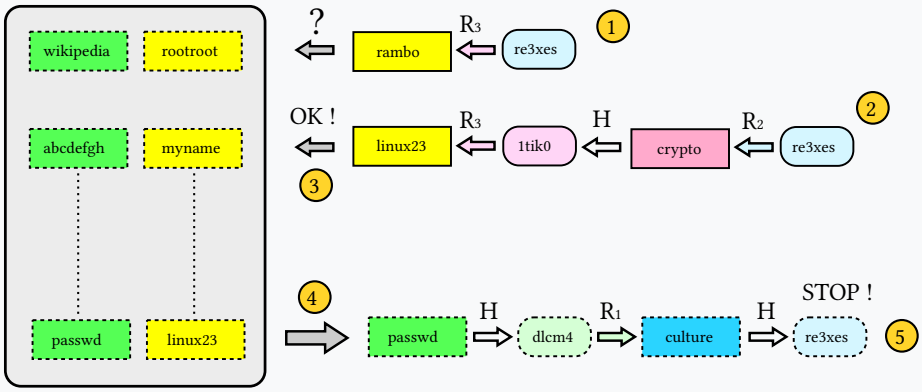
Figuur 5.4: Drie parallelle chains met telkens een andere reductiefunctie (kleur). Bron: Wikimedia Commons, Simple_rainbow_table.svg, CC BY-SA 2.5.

Wanneer de aanvaller nu van een gestolen hash terug het wachtwoord te pakken wil krijgen dan zal hij:

1. Deze hash als startpunt gebruiken en hier telkens weer de combinatie reductie+hash op toepassen.
2. **Totdat** een hash wordt gevonden die als eindpunt voor een van de lijsten werd ingesteld.
3. De aanvaller neemt vervolgens het startpunt van deze lijst (een wachtwoord) en past daarop opnieuw de combinatie van hashing en reductie toe (als het ware opnieuw een rainbow table genereren). Uiteindelijk zal hij weer op de gestolen hash uitkomen.
4. Als hij dan één stapje terug kijkt dan zal hij daar het wachtwoord zien die bij deze gestolen hash hoort.



Figuur 5.5: Ieder wachtwoord mapt naar exact één hash.

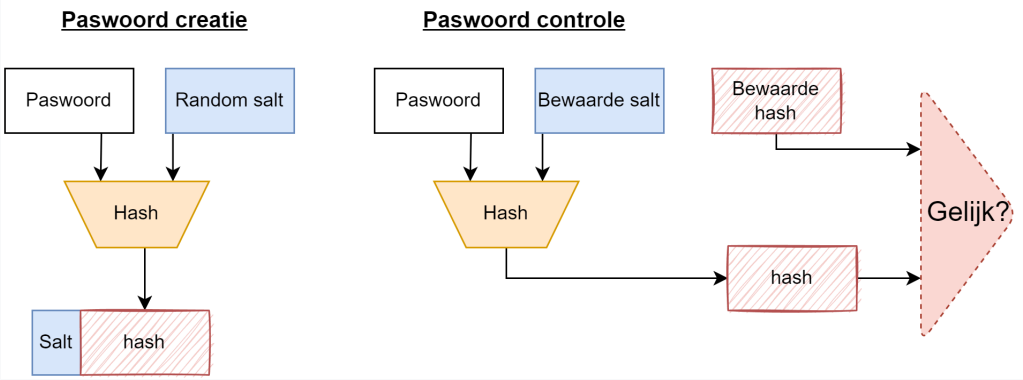


Figuur 5.6: Het volledige zoekproces: begin bij de gestolen hash (`re3xes`), reduceer en hash tot je een bekend eindpunt (`linux23`) bereikt, herbereken die chain vanuit het startpunt (`passwd`), en lees één stap vóór de gestolen hash het bijhorende wachtwoord (`culture`). Bron: Wikimedia Commons, Simple_rainbow_search.svg, CC BY-SA 4.0.

5.2.4 Salting

Om bestand te zijn tegen de rainbow attack dienen we de set van mogelijke wachtwoorden gevoelig te vergroten waardoor het niet meer realistisch is om voor die set rainbow tables te genereren. We kunnen helaas niet verwachten van de eindgebruikers dat zij met véél langere, meer willekeurige, wachtwoorden op de proppen komen en zullen dus een 'oude' truc moeten gebruiken die we ook al bij wifi hebben gezien. Bij wifi hanteerden we een initialisatie vector (IV) om de WEP-sleutel met 24 bits te verlengen zodat zelfs bij dezelfde sleutel, iedere IV eigenlijk zorgt voor een unieke seed.

Wel nu, dit concept kan je ook toepassen bij wachtwoorden en heet **salting**. Een salt is een extra stuk dat je toevoegt aan het wachtwoord **voor je de hash** berekent. Dit extra stukje is een willekeurig getal dat je uiteraard mee zal moeten opslaan in de database. Wanneer twee gebruikers hetzelfde wachtwoord zouden hebben, dan zouden ze (dankzij hun unieke salt) toch beiden totaal verschillende hashes genereren. Niet alleen dat, maar de salt zorgt er dus ook voor dat de set van mogelijk wachtwoorden véél groter wordt.



Figuur 5.7: Het salting proces.

In de database bewaren we nu volgende informatie:

- Gebruikersnaam.
- Gebruikte salt (minimum 32 bits).
- Hash van het wachtwoord en de salt samen.

> ⚠ Waarschuwing
>
> Merk op dat ook nu we nog steeds niet beschermd zijn tegen pass-the-hash aanvallen.

5.2.4.1 Mimikatz

Mimikatz werd origineel ontwikkeld als demo om aan te tonen dat de authenticatieprotocollen van Microsoft onveilig waren. Helaas is de tool totaal erg snel opgenomen in het arsenaal van de digitale stropers. De tool laat toe om *authentication tickets* te tonen en hergebruiken. Zo'n authentication tickets worden door de loginserver aangemaakt na een geslaagde loginfase door de gebruiker. Dit ticket kan de gebruiker dan aan een systeem aanbieden om toegang tot het systeem te krijgen (het is letterlijk een *toegangsticketje*). Wanneer Mimikatz wordt losgelaten op een Microsoft Windows besturingssysteem zal het deze tickets op het systeem zoeken zodat de aanvaller vervolgens zonder login gegevens toch kan inloggen door technieken zoals:

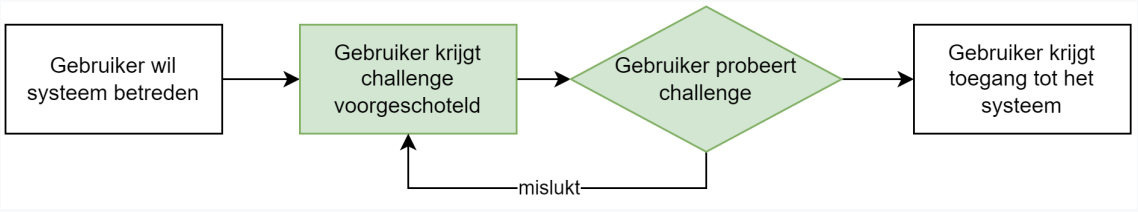
- Pass-the-hash: vroeger werden Windows wachtwoorden als hash (NTLM) bewaard op het systeem waardoor deze techniek erg eenvoudig was.
- Pass-the-ticket: zoals zonet beschreven, maar dan met het Kerberos ticket (zie ook hierna)
- Kerberos Golden Ticket: **Kerberos** is een van de meest gebruikte authenticatieprotocollen. Veel systemen die Kerberos gebruiken hebben echter een verborgen account (*KRBTGT* genaamd) wiens ticket domain admin rechten geeft én dat niet vervalt. Kortom, een gouden ticket!
- Pass-the-cash: identiek aan pass-the-ticket maar deze aanval werkt ook met logindata die zowel op Mac, Unix én Linux kan worden gevonden. Kortom, dit is natuurlijk de *motherload*, daar deze niet meer afhankelijk is van enkel Microsoft Windows besturingssystemen.

Het nadeel van Mimikatz, voor ons als boswachters, is dat de tool erg goed werkt én kan geautomatiseerd worden. In 2017 onderging Oekraïne een stevige ransomware aanval van (zo goed als zeker) Russische makkelij, genaamd **NotPetya** (een variant op de WannaCry ransomware). NotPetya gebruikte een aangepaste versie van Mimikatz zodat de ransomware zichzelf kon verspreiden over het netwerk en op andere systemen in het domein kon inloggen met hashes en tickets dat de Mimikatz variant aantrof.

> 💡 Tip
>
> De oorsprong van Petya en NotPetya werd getraceerd en is vermoedelijk het resultaat van een Russische hackinggroep genaamd *Sandworm* die onder de GRU werken, de Russische militaire inlichtingendienst.

5.3 CRAM en SCRAM

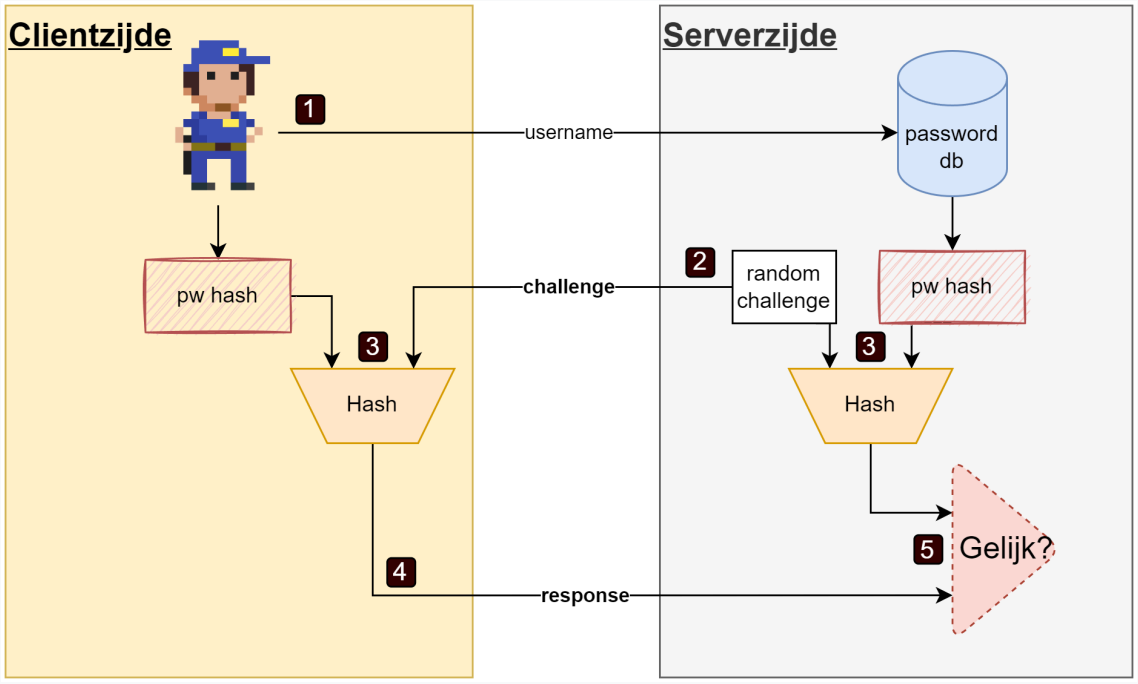
Om iemand te authenticeren spraken we tot nog toe enkel over een username/wachtwoord systeem. Echter, er zijn vele andere manieren om iemand te authenticeren. We spreken over "**Challenge-Response Authentication Mechanism (CRAM)** wanneer de gebruiker een vraag gesteld krijgt (de *challenge*) en hij hierop een geldig antwoord (de *response*) moet geven voor hij wordt toegelaten. Authenticeren met een wachtwoord is dus een vorm van CRAM. Er zijn er echter nog vele andere, denk maar aan de gehekelde CAPTCHA's - de ambetante vraag om te bewijzen dat je geen robot bent door alle boten in een afbeelding aan te duiden - of inloggen met behulp van je irisscan.



Figuur 5.8: CRAM.

Het mechanisme van een CRAM werkt als volgt:

1. De gebruiker stuurt z'n username met de vraag om in te loggen.
2. De server genereert een random challenge en stuurt deze terug.
3. Server en client creëren nu een hash van deze challenge met de hash van het wachtwoord (de server heeft dit bewaard, de client genereert de hash door z'n wachtwoord in te voeren)
4. De client stuurt deze hash, de response, terug naar de server.
5. De server vergelijkt of zijn gegenereerde response hash dezelfde is als die van de gebruiker.

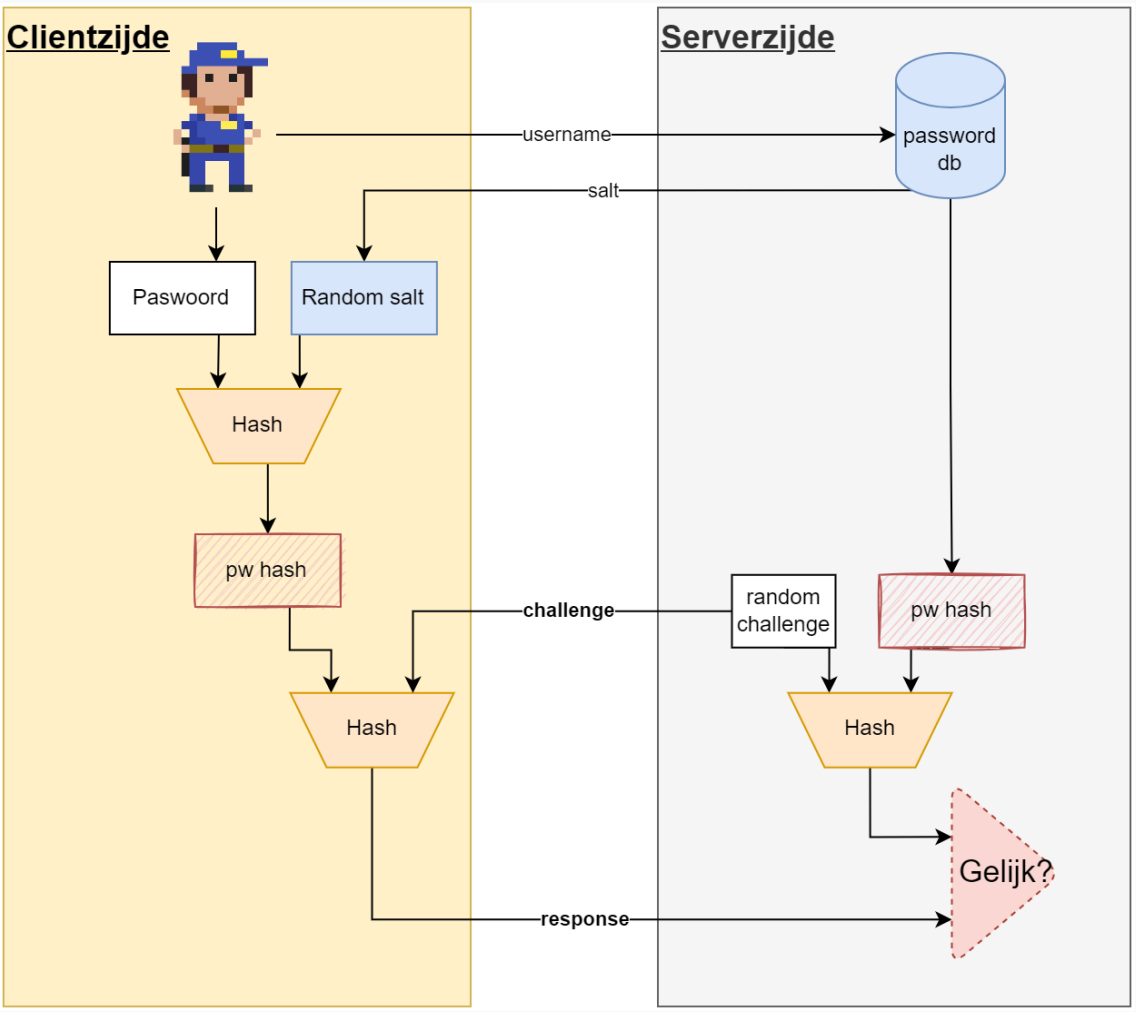


Figuur 5.9: CRAM flow.

Om het probleem van *pass-the-hash* op te lossen kan je gebruiken maken van een **S**CRAM, een **Salted Challenge-Response Authentication Mechanism**. We bespreken een vereenvoudigde versie (een echte SCRAM voorziet ook *mutual authentication*) waarbij we hoofdzakelijk willen uitleggen waarom een SCRAM systeem veiliger is dan een klassieke salted wachtwoord login van daarnet. Met dit systeem zorgen we ervoor dat :

1. De salted hash van de gebruiker NOOIT moet verzonden worden.
2. Geen replay aanval m.b.v. pass-the-hash mogelijk is.

Zoals je in de afbeelding kunt zien zal in dit systeem de server ook de bewaarde salt naar de client sturen, zodat deze geen gebruik kan maken van een bewaarde password hash die hij niet zelf heeft gemaakt.

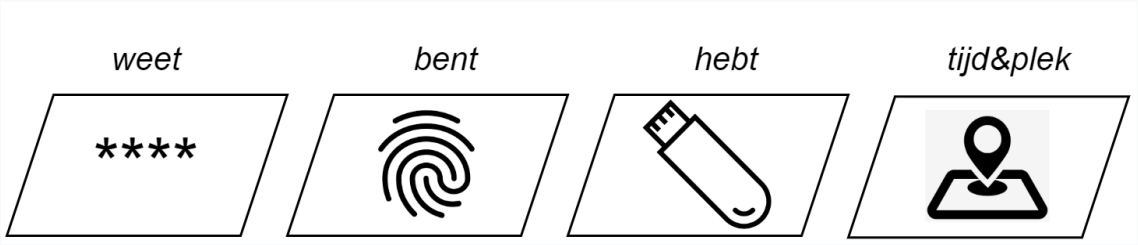


Figuur 5.10: SCRAM.

5.4 Multifactor authentication

We hebben enkel nog maar over wachtwoorden gesproken in dit hoofdstuk, maar uiteraard zijn er ook andere zaken die je kan gebruiken om je te identificeren. Er zijn verschillende **factoren** die kunnen gebruikt worden om te controleren of een persoon wel degelijk toegang mag krijgen tot een systeem:

- Iets wat je **weet**: je wachtwoord, je pincode, je rijksregisternummer, etc.
- Iets wat je **bent**: een eigenschap die uniek is per persoon en onder de noemer "*biometrics*" valt, zoals je vingerafdruk, irisscan, etc.
- Iets wat je **hebt**: een stuk hardware zoals een smartphone, USBkey, etc.
- **Waar** of **wanneer** je bent: je IP-adres , het moment van de dag dat je probeert in te loggen.



Figuur 5.11: Multifactor authenticatie factoren.

We zien meer en meer systemen verschijnen die aan zogenaamde **multifactor authentication (MFA)** doen waarbij het systeem minstens twee factoren (*2FA*) wil controleren voor je toegelaten wordt. Hoe meer verschillende factoren er worden gebruikt bij de authenticatie hoe veiliger het systeem is, maar ook hoe minder gebruiksvriendelijk het wordt. Het blijft dus een afweging tussen die twee eigenschappen om in te schatten wat de ideale hoeveelheid veiligheid en gebruiksvriendelijkheid is die je wenst te hebben.

Systemen die MFA aanbieden doen dit vaak op een gecontroleerde manier: afhankelijk van de gebeurtenissen zal het systeem beslissen of meerdere factoren moeten getest worden of niet. Als je bijvoorbeeld in België woont, maar Google ziet plots dat iemand met jouw wachtwoord probeert in te loggen vanuit een IP-adres op de Azoren, dan zal Google beslissen dat "iets wat je weet" (het wachtwoord) niet genoeg controle is en extra informatie vragen (andere factoren controleren).

5.4.1 Iets wat je weet: Paswoorden

Deze factor hebben we reeds uitvoerig behandeld. Het grote probleem met *dingen weten* is dat:

- Je ze kan vergeten en daardoor niet meer kan inloggen.
- Anderen die informatie kunnen te weten komen en zich vervolgens als jou voordoen.

Kortom, alhoewel deze factor vaak vanuit technologisch standpunt het eenvoudigst te implementeren is, is het ook de minst veilige vanuit een social engineering standpunt. Biometrics en hardware zijn moeilijker door een digitale stroper te stelen dan het wachtwoord en we hoeven niet bij een fingerprint te vrezen dat de gebruiker een "zwakke vingerafdruk" kiest, iets wat bij wachtwoorden vaak hét primaire probleem is.

5.4.2 Iets wat je bent: Biometrics

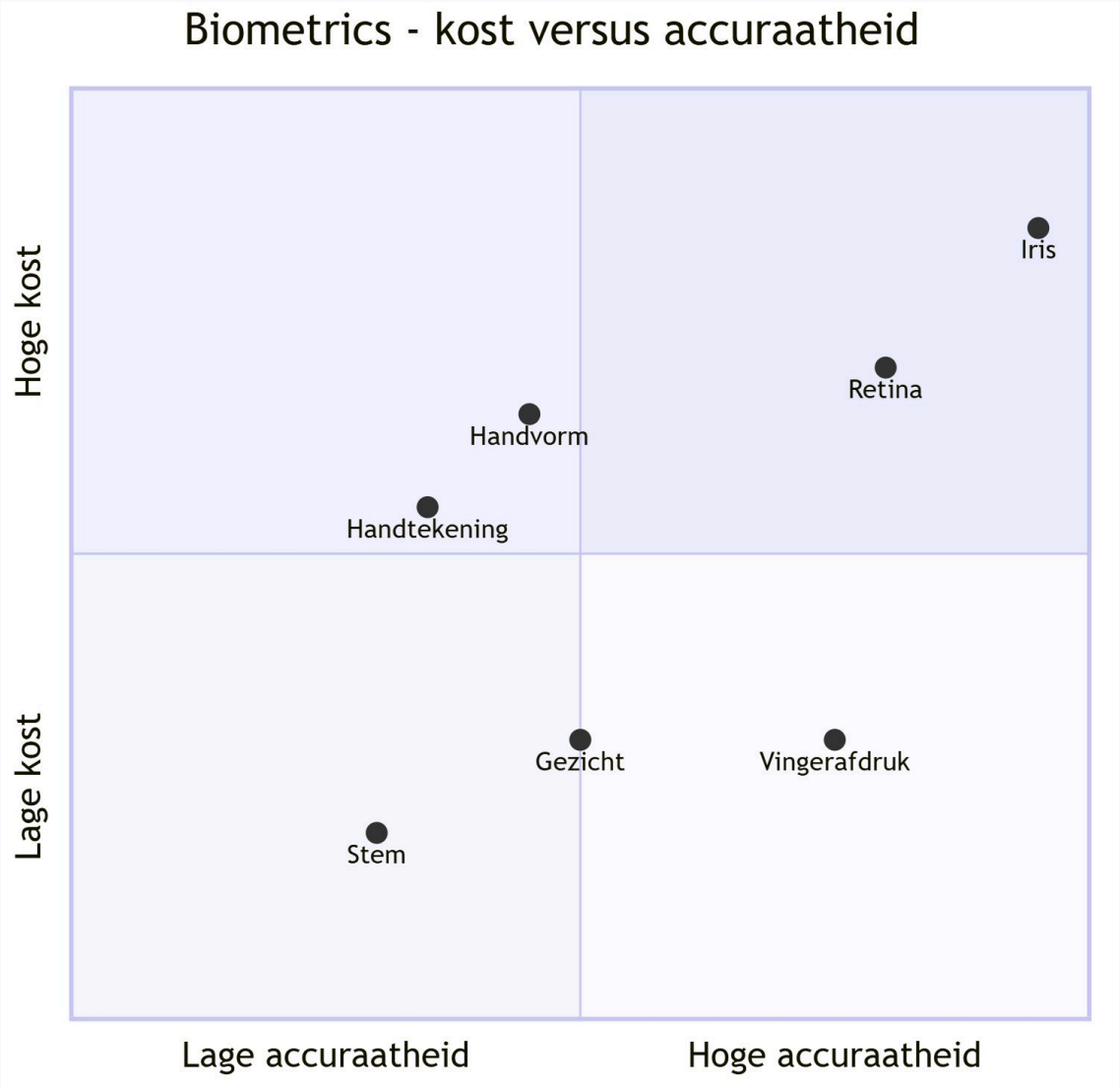
"We zijn allemaal uniek". Iedere mens heeft een hele hoop eigenschappen die uniek zijn per persoon. Zelfs als bepaalde van onze eigenschappen gelijkaardig zijn met andere personen dan zal zeker een combinatie van twee of meerdere eigenschappen dat niet zijn. Door dus menselijke eigenschappen van je te gebruiken als authenticatie hebben we een factor gevonden die zeer moeilijk na te bootsen valt: op voorwaarde dat je een echt unieke eigenschap kiest én deze op deze juiste manier meet.

Enkel veel gebruikte biometrieken als authenticatievorm zijn:

- Vingerafdruk.
- Iris.
- Stem.
- Gezicht (vaak met behulp van "stereo camera" voor 3D beeld).

Maar ook andere metrieken kunnen erg interessant zijn zoals de manier waarop je je wachtwoord invoert, de manier waarop je wandelt (*gait*) etc.

Niet elke biometriek is evenwaardig wat betreft accuraatheid en kost. **Iris**-scans staan bovenaan qua accuraatheid maar vereisen dure, gespecialiseerde hardware. **Retina**- en **vingerafdruk**-scanners zijn redelijk accuraat aan een midden- tot lage kost. Aan de goedkope (maar ook minder accurate) kant vind je bijvoorbeeld **stem**herkenning en **gezicht**sherkenning via een gewone webcam. Je keuze van biometriek hangt dus niet alleen af van technische vereisten, maar ook van budget en use case (een irisscan aan de grens is iets heel anders dan gezichtsherkenning om je telefoon te ontgrendelen).



Figuur 5.12: Indicatieve positionering van verschillende biometrieken op een kost-accuraatheid diagram.

> ⚠ Waarschuwing
>
> Paswoorden van miljoenen mensen opslaan is één ding. De biometrische gegevens is een heel ander verhaal waarbij ook **privacy** plots een erg heikel punt wordt (beeld je even in dat Hitler en zijn trawanten 80 jaar geleden toegang hadden tot biometrische data waarmee met een bepaalde zekerheid kon vastgesteld worden of iemand van Joodse origine was of niet.)
>
> In India is de Aadhaar (Indiaas voor "basis"), hun rijksregisternummer zeg maar, een unieke code die gebaseerd is op *onder andere de irisscan en vingerafdrukken* (alle 10!) van de burger. Deze gigantische database werd in 2018 nog gehackt waardoor mogelijk de informatie van 1.1 miljard geregistreerde burgers werd gestolen.

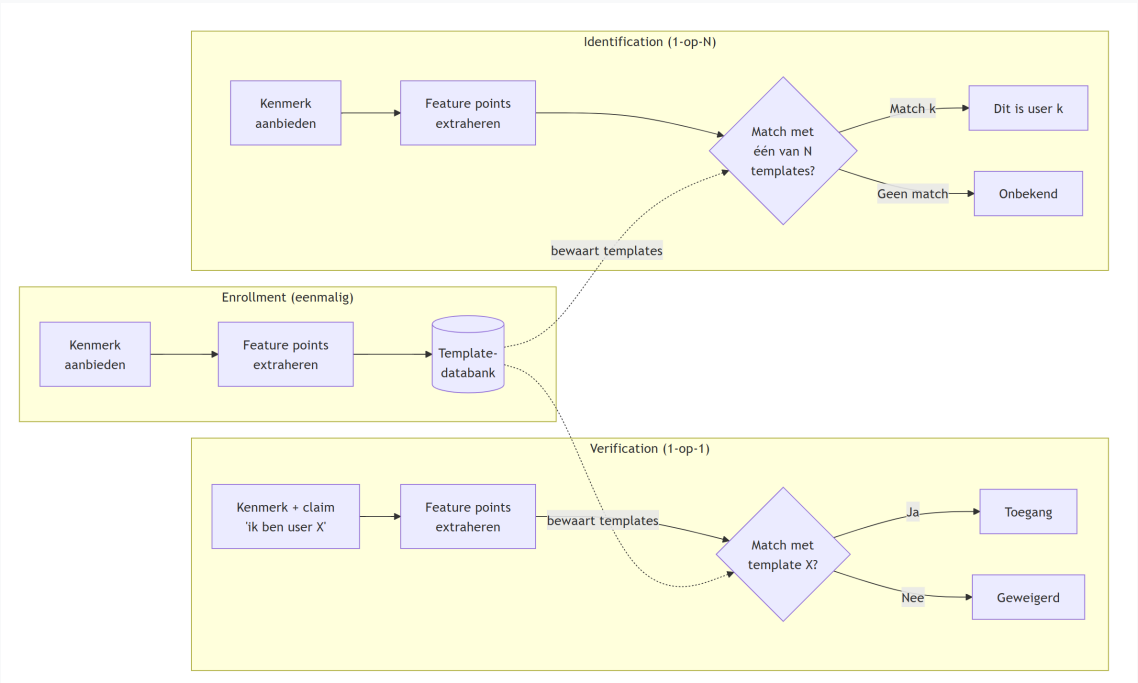
Om een biometriek in de wachtwoord database te bewaren hebben we een manier nodig om deze te digitaliseren op een zodanige manier dat de unieke aspecten ervan bewaard worden. Voorts moet er rekening mee gehouden worden dat het "registreren" van een biometrische eigenschap nooit 100% accuraat kan. Denk maar aan een tijdelijk krasje op je vinger, je baard die anders geschoren is, etc.

De zogenaamde *feature points* van een biometrische eigenschap worden in de database bewaard: dit zijn de unieke waarden waarvan geweten is dat deze per persoon anders zijn. We gaan deze niet per biometrische eigenschap bespreken, het volstaat te begrijpen dat in de gebruikersdatabase meestal een korte sequentie van getallen (of letters, denk maar aan een DNA-sample) wordt bewaard die als het ware

jouw unieke wachtwoord voorstelt voor die specifieke biometrische eigenschap van j. Enkel wanneer je bij het opnieuw inloggen (quasi) dezelfde feature points genereert als bij de registratie zal deze factor aanvaard worden als correct.

Een biometrisch systeem kent drie duidelijk te onderscheiden fasen:

1. **Enrollment (registratie)**: de gebruiker biedt zijn biometrisch kenmerk voor het eerst aan, het systeem extraheert de feature points en bewaart deze in de databank. Dit is het moment waarop de koppeling *user* ↔ *biometrie* wordt gemaakt.
2. **Verification (verificatie)**: *"Ben jij wel degelijk user X?"* — de aangeboden feature points worden vergeleken met één specifieke template in de databank (**1-op-1**). Dit is het klassieke inlog-scenario.
3. **Identification (identificatie)**: *"Wie ben jij?"* — de aangeboden feature points worden vergeleken met álle templates in de databank (**1-op-N**). Dit is een veel zwaardere operatie en wordt typisch ingezet bij bv. grenscontrole of forensisch onderzoek.



Figuur 5.13: De drie fasen van een biometrisch systeem en hun interactie met de templatedatabank.

> **Tip**
>
> Biometrische eigenschappen kunnen niet alleen dienst doen als een extra factor bij het authenticeren, ze zijn uiteraard ook erg handig voor identificatie. In principe kan iemand nog steeds de gebruikersnaam van een ander persoon gebruiken. Als de biometrische eigenschappen als identificatie dienen kunnen aanvallers dat niet meer doen: ze kunnen onmogelijk aan het systeem zeggen *"ik ben persoon x"* terwijl de vingerafdrukscanner duidelijk een vingerafdruk registreert van *persoon y*.

5.4.3 Iets wat je hebt: Hardware

Een fysiek object, zeker als het complex is, kan moeilijk nagemaakt worden en is dus een ideale factor. De elektronica van de 21e eeuw behoort tot de meest complexe dingen ooit die de mensheid heeft kunnen vervaardigen. Het is dan ook logisch dat we deze elektronica gebruiken als extra authenticatiefactor. In essentie zal dit stuk hardware nog steeds gewoon een wachtwoord bevatten, maar dit zal echter ongelooflijk veel langer (en dus sterker) zijn dan het gemiddelde wachtwoord dat een standaard gebruiker kan onthouden.

Er zijn twee grote families van hardware-gebaseerde authenticatie-vormen:

- Een smartphone, met daarop een *authenticator* app.

- Een USB sleutel.

Een nadeel van deze groep is dat het om een fysiek object gaat dat je kan verliezen of dat stuk kan gaan.

5.5 Federation en Single Sign-On (SSO)

Bij cryptografie wordt het ten stelligste afgeraden om zomaar op de *wilde boef* een eigen crypto-algoritme te ontwikkelen. De kans dat je fouten met verstrekkende gevolgen maakt is te groot. Ook bij het omgaan van logindata van gebruikers en hoe je ze authenticeert is het aangeraden om even te bezinnen voor je er zelf aan begint.

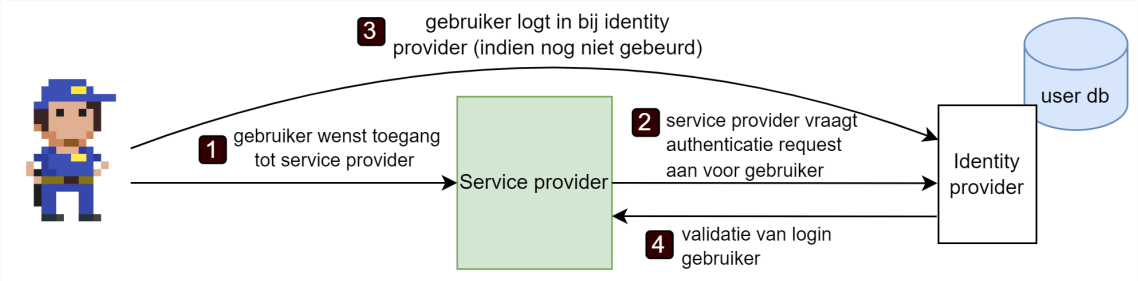
5.5.1 Wat is Single Sign-On (SSO)?

Single Sign-On (SSO) is een authenticatiemethode waarbij een gebruiker zich **éénmaal aanmeldt** en vervolgens automatisch toegang krijgt tot **meerdere applicaties of diensten**, zonder zich bij elk systeem apart te moeten aanmelden. Je kent dit waarschijnlijk al: wanneer je inlogt op je Google-account, heb je meteen ook toegang tot Gmail, YouTube, Google Drive en tientallen andere Google-diensten — zonder telkens opnieuw je wachtwoord in te voeren. Dat is SSO in actie.

Het basisprincipe is eenvoudig: in plaats van dat elke applicatie zelf verantwoordelijk is voor authenticatie, wordt dit uitbesteed aan een centrale **identity provider** (IdP). Die identity provider bevestigt de identiteit van de gebruiker, waarna de applicatie (de **service provider**) de gebruiker vertrouwt op basis van die bevestiging.

5.5.2 Federation

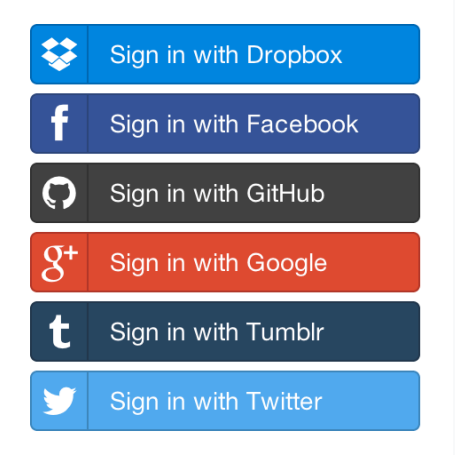
Dankzij het concept **federation** kan SSO ook werken **over de grenzen van organisaties heen**. Gebruikers kunnen inloggen op jouw site of app (de *service provider*) gebruik makend van hun bestaande Google, Facebook of andere accounts. Een *third-party* — die jij en je gebruiker vertrouwen — zorgt dan voor de eigenlijke authenticatie als *identity provider*. Zo hoef je als ontwikkelaar niet wakker te liggen van hoe je gebruikerswachtwoorden gaat opslaan.



Figuur 5.14: Een vereenvoudigd single sign-on proces.

Federation via SSO is een onderdeel van *federated identity management*, een groep technologieën en concepten die ervoor zorgen dat de identiteit van een gebruiker over meerdere, onafhankelijke systemen wordt bewaard en gebruikt. Je zal de termen **delegation** en **federation** soms door elkaar zien tegenkomen wanneer je meer informatie over SSO opzoekt.

Samengevat gaan we bij *delegation* een gebruiker verplichten in te loggen met een bepaalde third-party die dit ondersteunt (bv. inloggen met je Facebook account). Bij *federation* gaat het breder: je website zal éénder welke third-party account aanvaarden, zolang deze maar compatibel is met het authenticatie systeem van je website (een voorbeeld hiervan is OpenID).



Figuur 5.15: Enkele van de vele typische SSO knoppen die je geregeld zal tegenkomen.

> Tip
>
> **OAuth** (*open authorization*) is een gestandaardiseerde manier om aan authenticatie te doen.

Uiteraard moeten we bij federatie benadrukken dat ook hier **privacy** een belangrijk aspect wordt. De vraag is dan ook in hoeverre je een bedrijf zoals Google of Facebook/Meta vertrouwt met jouw (login)data.

5.6 WebAuthn en passkeys

Een nieuwe technologie die in opmars is, is **WebAuthn**. Deze technologie laat toe om in te loggen op websites zonder dat je een wachtwoord moet ingeven. In plaats daarvan gebruik je een *authenticator* die je bij je hebt, zoals een USB-sleutel of je smartphone. Deze authenticator zal een digitale handtekening genereren die de website kan controleren.

Passkeys combineren de kracht van **asymmetrische/public-key crypto en hardware authenticatie**, met als grootste pluspunt dat gebruikers geen complexe wachtwoorden meer moeten onthouden.

Zonder in de ontstaansgeschiedenis te duiken, is het toch nuttig even enkele termen in vet te zetten die je zeker zal tegenkomen als je meer over passkeys wilt leren:

- **FIDO Alliance**: de organisatie die de standaarden voor WebAuthn en U2F beheert. FIDO staat voor *Fast IDentity Online*.
- **WebAuthn** (*Web Authentication API*): een API die websites toelaat om met authenticators te communiceren.
- **FIDO2**: een standaard die WebAuthn ondersteunt, ontwikkel door de FIDO Alliance.
- **U2F** (*Universal 2nd Factor*): een oudere standaard die ook door WebAuthn wordt ondersteund.

> Tip
>
> Op youtu.be/cEhc6vMFTh4 vind je een heel duidelijk overzicht van passkeys, inclusief de technische zijde ervan.

5.6.1 Een Passkey aanmaken: de registratie

Een passkey aanmaken is een eenvoudig proces. Je hebt een authenticator nodig, zoals een USB-sleutel of je smartphone. De authenticator zal een paar sleutels genereren: een *public key* en een *private key*. De public key wordt naar de website gestuurd, terwijl de private key (het wachtwoord met andere woorden) op de authenticator blijft.

Wanneer je dus als gebruiker registreert op een website of app, dan zal de passkey generatie van start gaan, als volgt:

1. Bij het registreren kiest de gebruiker ervoor om een passkey te gebruiken (i.p.v. het klassieke wachtwoord).
2. De gebruiker zal op zijn eigen toestel zichzelf nu eerst moeten identificeren. Dat kan op verschillende manieren: fingerprint scan, een PIN-code, een hardwaresleutel (denk aan bijvoorbeeld aan YubiKey), etc.
3. Het toestel van de gebruiker genereert een sleutelpaar. De private sleutel blijft op het toestel en wordt veilig bewaard, de public sleutel wordt naar de website gestuurd.

Stap 3 gaan we even verder uit de doeken doen: we gaan natuurlijk deze sleutel niet zomaar *over den draad* versturen. We gaan natuurlijk onze kennis van certificaten gebruiken, die ons toelaten om te bewijzen dat de publieke wel degelijk de onze. De gebruiker zal zijn publieke sleutel verpakken in een *attestation object*: de publieke sleutel, samen met een *signed challenge* (zie verder), een *credential ID* en een certificaat. Dit attestation object wordt naar de andere zijde gestuurd, die deze zal bewaren.

De tegenpartij, bijvoorbeeld de website waar je wilt registreren, heeft uiteraard nog bewijs nodig dat het jouw attestation object wel kan vertrouwen. Tijdens stap 1 van de registratie zal de server daarom een challenge sturen, die de gebruiker mee in het attestation object moet plaatsen.

In dit hele proces heeft de website nooit toegang tot de private sleutel van de gebruiker. De website kan enkel de public sleutel zien en gebruiken. Het concept “het wachtwoord verlaat nooit het apparaat” wordt hier dus erg letterlijk genomen.

⚠ Waarschuwing

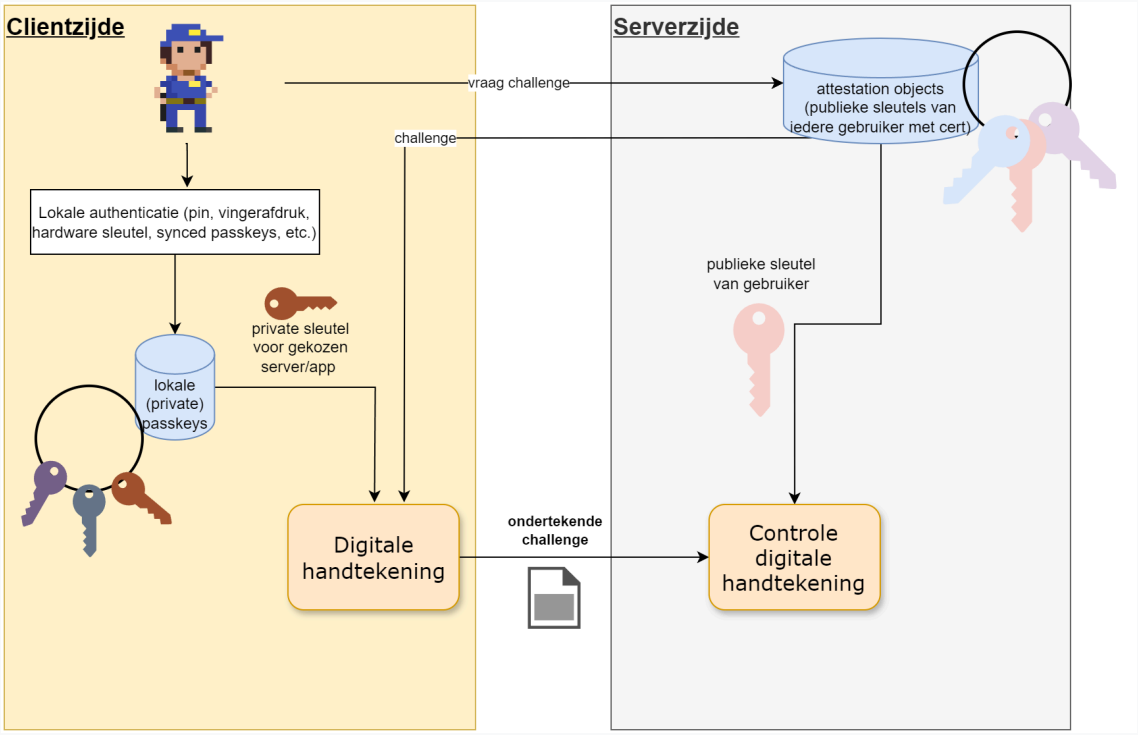
Doordat deze private sleutels niet meer op de website worden bewaard, wordt het gebruik van password managers nog belangrijker. De private sleutels worden immers opgeslagen op de authenticator, en als je die lange, complexe stukken data verliest, ben je al je accounts kwijt.

Een password manager kan je helpen om je accounts te beheren en je private sleutels (je passkeys, in dit geval zijn dit **synced passkeys**, een concept dat ook mee in WebAuthn is ingebouwd) veilig te bewaren én te synchroniseren naar je andere apparaten. I

5.6.2 Inloggen met een Passkey

Het inloggen met een passkey is gebaseerd op wat we weten uit public key crypto: je publieke sleutel kan je aan iedereen geven, enkel de houder van de bijhorende private sleutel zal de data kunnen lezen die met deze publieke sleutel werd geëncrypteerd.

De login-fase is dan ook bijna het zelfde als de registratie. Ook nu zal de gebruiker een challenge krijgen. Deze challenge zal de gebruiker nu encrypteren met z’n private sleutel. Wanneer de server deze geëncrypteerde challenge kan decrypteren met de bewaarde publieke sleutel van de gebruiker, weet deze dat de gebruiker mag toegelaten worden.



Figuur 5.16: Het login proces met een passkey

5.7 Authenticator apps en TOTP

Wanneer je multifactor authenticatie inschakelt op een website, krijg je vaak de keuze om een **authenticator app** te gebruiken, zoals Google Authenticator, Microsoft Authenticator of Authy. Deze apps genereren om de 30 seconden een nieuwe zescijferige code die je moet invoeren naast je wachtwoord. Maar hoe werkt dit? Hoe kan een app op jouw telefoon, die op dat moment geen internetverbinding nodig heeft, dezelfde code genereren als de server verwacht?

Het antwoord is een slim algoritme genaamd **TOTP**: *Time-based One-Time Password*.

5.7.1 De gedeelde geheime sleutel

Het hele systeem steunt, zoals zoveel in cryptografie, op een **gedeelde geheime sleutel** (*shared secret*). Wanneer je een authenticator app koppelt aan een website, gebeurt het volgende:

1. De website genereert een willekeurige geheime sleutel (typisch 160 bits).
2. Deze sleutel wordt aan jou getoond, meestal in de vorm van een **QR-code** die je scant met je authenticator app.
3. Zowel de server als jouw app bewaren nu dezelfde geheime sleutel.

Dit is het enige moment waarop de sleutel wordt uitgewisseld. Vanaf nu hoeven jouw telefoon en de server nooit meer rechtstreeks met elkaar te communiceren om geldige codes te genereren.

> ⚠ Waarschuwing
>
> Omdat de QR-code de volledige geheime sleutel bevat, moet je deze met de nodige voorzichtigheid behandelen. Maak er geen screenshot van die je onbeveiligd bewaart en toon de code niet aan anderen. Wie de geheime sleutel heeft, kan dezelfde codes genereren als jij.

5.7.2 Hoe TOTP werkt

TOTP combineert twee ingrediënten om een code te genereren:

1. **De gedeelde geheime sleutel** (die zowel de app als de server kennen).

2. **De huidige tijd**, afgerond naar blokken van 30 seconden.

Het algoritme werkt als volgt:

1. Neem de huidige Unix-tijd (het aantal seconden sinds 1 januari 1970) en deel deze door 30. Rond af naar beneden. Dit geeft een getal dat we de **tijdstap** (*time step*) noemen. Gedurende diezelfde 30 seconden zullen jouw telefoon én de server exact dezelfde tijdstap berekenen.
2. Gebruik nu een **HMAC** (Hash-based Message Authentication Code) om de geheime sleutel te combineren met deze tijdstap. Het resultaat is een lange hash.
3. Uit deze hash wordt via een vaste procedure (*dynamic truncation*) een **zescijferig getal** geëxtraheerd: de code die je op je scherm ziet.

Omdat zowel de server als jouw app dezelfde geheime sleutel en dezelfde tijd gebruiken, genereren ze onafhankelijk van elkaar dezelfde code. Na 30 seconden verandert de tijdstap en krijg je een volledig nieuwe code.

> Tip
>
> Servers accepteren meestal niet enkel de code van het huidige tijdsblok, maar ook die van het vorige en het volgende blok. Dit geeft een marge van ongeveer 90 seconden en vangt kleine klokverschillen op tussen jouw telefoon en de server.

5.7.3 Waarom is TOTP veilig?

TOTP biedt een aantal belangrijke voordelen:

- **Eenmalig**: iedere code is slechts 30 seconden geldig. Zelfs als een aanvaller je code onderschept, is deze tegen de tijd dat hij deze wil gebruiken waarschijnlijk al vervallen.
- **Geen netwerkverbinding nodig**: de app heeft na de initiële registratie geen internet meer nodig. De tijd is het enige dat de app en server synchroniseert.
- **Niet voorspelbaar**: zonder kennis van de geheime sleutel is het onmogelijk om toekomstige codes te berekenen, zelfs als je vorige codes hebt gezien.

> ⚠ Waarschuwing
>
> TOTP is niet onfeilbaar. Een aanvaller die erin slaagt om tegelijkertijd je wachtwoord én een geldige TOTP-code te bemachtigen (bijvoorbeeld via een real-time phishing aanval die als *proxy* fungeert tussen jou en de echte website) kan nog steeds inloggen. Passkeys (zie eerder) zijn in dat opzicht veiliger omdat ze gebonden zijn aan het specifieke domein van de website.

5.8 Samenvatting: drie gouden regels

We hebben in dit hoofdstuk veel technieken en protocollen behandeld, maar alles valt terug te brengen tot drie kernregels die een goed authenticatiesysteem steeds moet respecteren:

1. Het wachtwoord mag nooit van de **client naar de server** gestuurd worden. *(→ hashing, CRAM, SCRAM)*
2. Het wachtwoord mag nooit van de **server naar de client** gestuurd worden. *(→ denk aan de "Ik ben m'n wachtwoord vergeten"-test)*
3. Het wachtwoord mag nooit **in leesbare vorm op de server** bewaard worden. *(→ hashing + salting)*

Moderne protocollen zoals **passkeys** gaan zelfs een stap verder: het wachtwoord (de private sleutel) verlaat zelfs de *authenticator* niet.

6. IoT Security

i Leerdoelen

Na dit hoofdstuk kan je:

1. De **specifieke uitdagingen van IoT-security** benoemen (low-cost, long-life, fysieke toegang, firmware, default credentials).
2. Aan de hand van de **Mirai-aanval** illustreren waarom zwakke IoT-toestellen een systemisch risico vormen voor het hele Internet.
3. De rol van een **TPM** (of gelijkwaardig) en andere hardware-mitigaties situeren in de IoT-verdediging.
4. De **McCumber-kubus toepassen op een concrete IoT-case**, rekening houdend met data in rust, transport én verwerking.
5. De **wet van *diminishing returns*** toepassen: welke basismaatregelen (KISS, geen defaults, updates) leveren de grootste veiligheidswinst op?

We spraken in het eerste hoofdstuk al over de problemen die inherent zijn bij Internet-of-Things (IoT) apparaten en netwerken. Zo had je in 2016 het Mirai-botnet dat aantoonde hoe krachtig een botnet van IoT-apparaten kan zijn, puur door de grote hoeveelheid potentiële zombies.

Het is niet toevallig dat we dit boek afsluiten met het hoofdstuk omtrent IoT Security: alles (en meer) wat je in de voorgaande hoofdstukken hebt geleerd heb je namelijk nodig om IoT-netwerken te beveiligen. Er zijn zoveel manieren voor digitale stropers om misbruik te maken van een slecht beveiligd IoT-apparaat of netwerk dat het complexe probleem van netwerken beveiligen nog ingewikkelder is geworden, namelijk:

- Van nature werken IoT-apparaten op **low-power**, daar ze vele dagen of weken moeten voortkunnen zonder te veel verbruik. Ingebouwde security protocollen mogen dus geen grote percentages van het vermogen opsouperen, zeker niet als dat ten koste is van de hoofdbestaansreden van het apparaat.
- Meestal, ook weer om een laag energieverbruik te behouden, hebben ze ook een **beperkte bandbreedte** ter beschikking (daarbij: beeld je de miserie op een wifi-netwerk in wanneer honderden kleine sensoren aan hoge bandbreedtes het netwerk mee gebruiken).
- Zowel de fysieke dimensies en bovenstaande twee eigenschappen zorgen er ook voor dat IoT-apparaten meestal maar een **beperkt aantal zaken kunnen**. Het heeft niet altijd de netwerkverbindingsmogelijkheden die je nodig hebt (meestal heeft een apparaat een draadloze - wifi, Zigbee, LoRaWAN,etc. - of bedrade aansluiting, maar zelden beiden) en ook qua opslag en processormogelijkheden zijn de apparaten uiteraard vaak veel beperkter dan de kracht van computers en mobiele telefoons. Het grootste probleem hiervan is echter dat IoT-apparaten *by nature* vaak **niet evident zijn om te updaten**. Sommige apparaten kunnen bijvoorbeeld enkel geüpdatet worden door fysiek met het apparaat te zijn verbonden (en dan door de firmware te flashen bijvoorbeeld). Vanuit een fysieke beveiligingsconcept is dat goed, maar niet als jouw honderden apparaten een kritieke beveiligingslek hebben die dringend gepatcht moet worden.
- Vaak bevinden IoT-apparaten zich op de koop toe op **onbeschermde locaties** die maar moeilijk fysiek te beveiligen zijn. Hierdoor kunnen digitale stropers, zonder schrik om betrapt te worden, ongezien met de apparaten knoeien (*tamperen*).

i Opmerking

We gaan nogal vrij om met de term IoT-apparaat, waarbij we ook negeren dat er grote verschillen zijn tussen “Enterprise IoT” en “huis-tuin-en-keuken IoT”. Die eerste categorie heeft meestal bijvoorbeeld wel zeer doordachte updatestrategieën en dergelijke, terwijl de slimme koelkast in je huis dat vermoedelijk niet zal hebben.

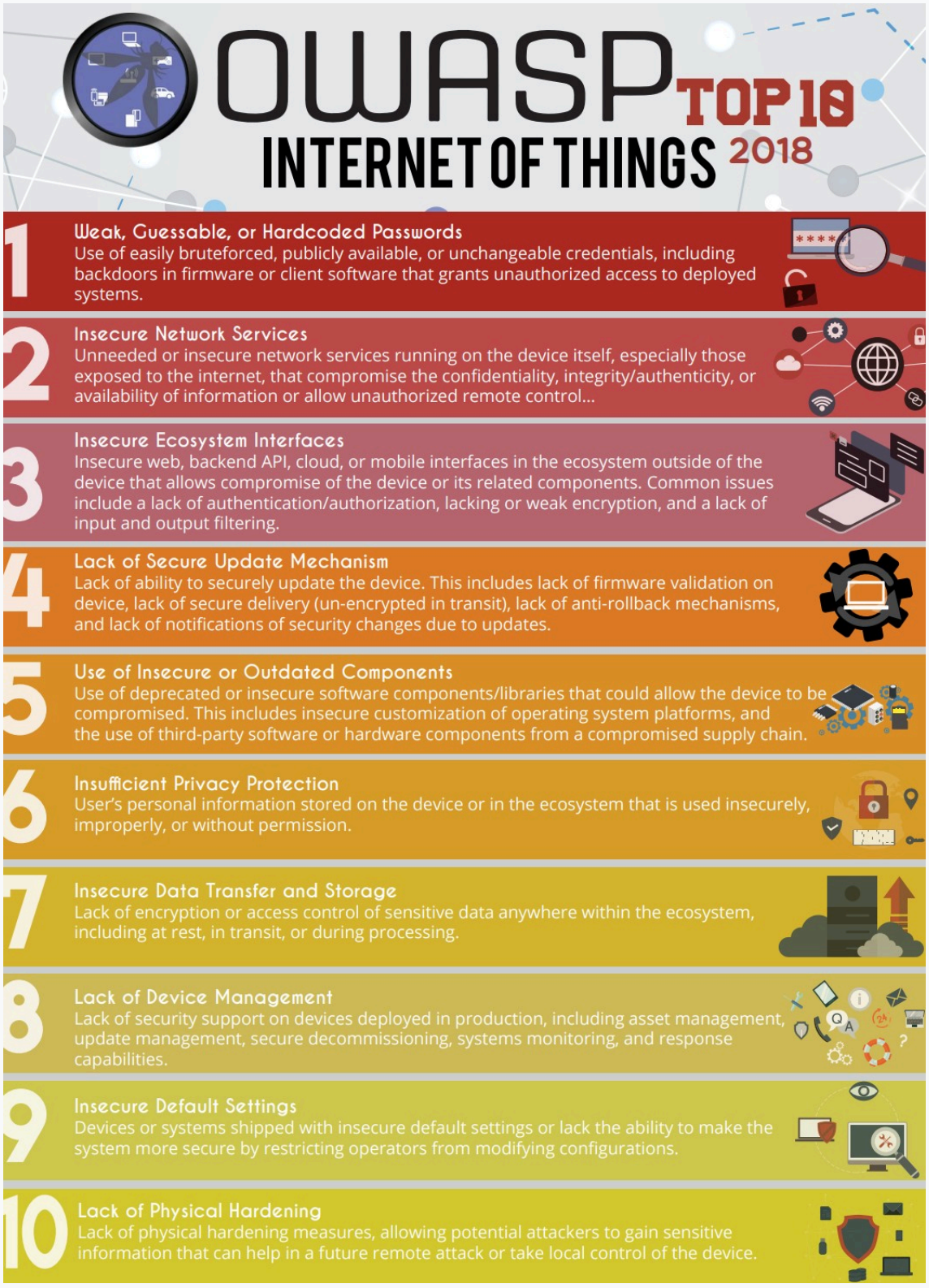
6.1 OWASP IoT Top 10

Het Open Web Application Security Project (**OWASP**) is een open-source-project rond computerbeveiliging. Individuen, scholen en bedrijven delen via dit platform informatie en technieken. Ook de Belgische tak van OWASP is erg actief en een interessante organisatie indien je van plan bent om een carrière binnen de cybersecurity wereld op te bouwen. Ze organiseren geregeld (meestal gratis) workshops en evenementen.

Het OWASP heeft tal van erg interessante "Top 10" overzichten zoals de meest voorkomende webapp zwakheden, etc. Deze projecten bestaan meestal uit een reeks tools, *best practices* en gidsen en zijn dus de ideale manier om je te verdiepen binnen een specifiek cybersecurity domein. Eén van de actieve projecten is het "OWASP Internet of Things" project (beschikbaar via owasp.org/www-project-Internet-of-Things/) waarvan we de Top 10 hier zullen bespreken, gerangschikt van meest voorkomend (en dus belangrijkste om aan te pakken) naar minder.

> **Tip**
>
> Bekijk zeker ook eens het IoT Goat project via github.com/OWASP/IoTGoat/ dat is een *"insecure firmware based on OpenWrt and maintained by OWASP as a platform to educate software developers and security professionals with testing commonly found vulnerabilities in IoT devices. The vulnerability challenges are based on the OWASP IoT Top 10, as well as 'easter eggs' from project contributors."*

> **Waarschuwing**
>
> Om de wel erg algemene term "IoT" wat duidelijker te maken, zullen er hier en daar merknamen en producten worden vermeld. Dit is hoegenaamd geen verdoken reclame (of kritiek) voor deze producten, maar gewoon een poging om de lezer een duidelijkere context te geven.



Figuur 6.1: Versie 2018 (Bron owasp.org).

6.1.1 Paswoorden

Ook bij IoT begint veiligheid bij het hebben van goede, complexe, moeilijk te raden wachtwoorden. Bij IoT-apparaten is er echter een stevige drempel, zeker voor huis-tuin-en-keuken gebruikers: het aanpassen van sommige IoT-apparaten is soms erg omslachtig. Niet alle apparaten hebben een webpagina langs

waar de gebruiker wachtwoorden kan aanpassen en dus moet dit ofwel via omslachtige, vreemde tools, ofwel (gruwel) door fysiek het apparaat te bedienen. Deze apparaten zijn natuurlijk niet gemaakt om complexe wachtwoorden eenvoudig in te voeren. Denk maar aan een digitale weegschaal: die heeft vaak een eenvoudig LED-scherm waarop je gewicht wordt getoond. Sommige weegschalen kan je vervolgens "bedienen" door links of rechts op de schaal te klikken, afhankelijk van wat je wilt doen. De auteur kan je garanderen: je wachtwoord zal niet lang zijn, als je een goed wachtwoord wilt invoeren zal je na enkele minuten in het zweet staan van de tapdans die je moet uitvoeren op je weegschaal.

Omdat het aanpassen van die wachtwoorden vaak omslachtig is, zullen gebruikers meestal opteren om het *default wachtwoord* te gebruiken. Ze redeneren dat **toch niemand dit simpele IoT-apparaat zal willen hacken**. "Wat kan een hacker nu doen met toegang tot mijn digitale thermometer." Wat echter vergeten wordt is dat cyberstropers altijd op zoek gaan naar de *weakest link*, om die vervolgens te misbruiken om tot het eigenlijk doel te geraken. Kortom, ieder IoT-apparaat in je omgeving is een potentiële toegangspoort tot de rest van je netwerk!

6.1.2 Ongebruikte of onveilige netwerkservices

Als je anno 2023 een printer koopt dan zal dit apparaat bijna altijd een IoT-apparaat zijn dat je toelaat om vanop afstand te printen. Echter, dergelijke apparaten hebben vaak tal van netwerkservices draaien waarvan de gewone huis-tuin-en-keuken gebruiker weinig of niets van afweet.

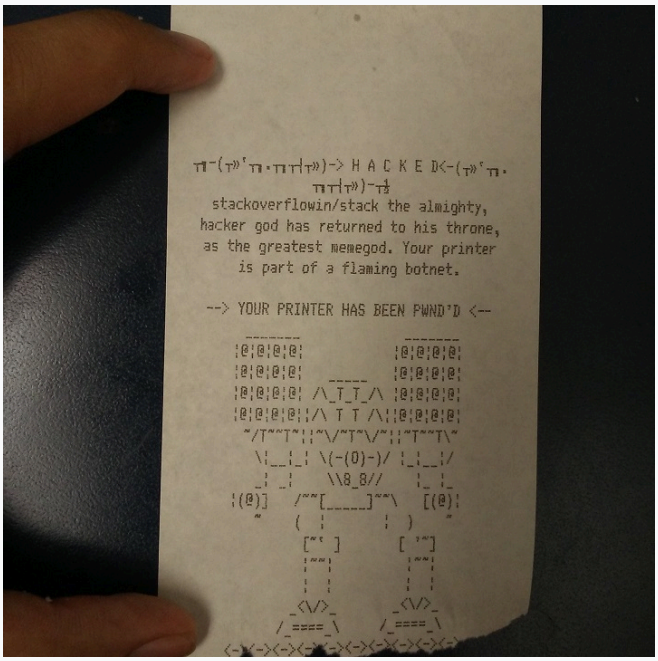
> Tip
>
> Vergelijk dit bijvoorbeeld met je Microsoft Windows besturingssysteem: heb je al eens gekeken welke services allemaal permanent op je systeem draaien (Start -> uitvoeren -> `services.msc`) ?

Dit soort diensten uitzetten behelst ook weer dat de gebruiker weet heeft van een webpagina met administrator-instellingen. Soms zijn die diensten op de koop toe essentieel voor de goede werking van het apparaat en ze uitzetten is dan geen optie. Maar wat als die service op zich inherent onveilig is?! Zo zijn er apparaten die enkel *telnet*-sessies gebruiken om het apparaat in te stellen. Een telnet-sessie geeft je een remote shell die echter **onversleuteld** met het apparaat communiceert (in tegenstelling tot bijvoorbeeld SSH, Secure Shell, dat wél veilige communicatie aanbiedt).

Kortom, het is aangeraden om altijd te controleren welke services nu eigenlijk actief zijn op je apparaat.

i Opmerking

In 2017 was er een grayhat hacker die vermoedelijk 150.000 printers vanop afstand in Nederland kon benaderen. Vervolgens printte hij "ludieke" berichten op de printers om de gebruikers er op te wijzen dat ze een onveilig apparaat gebruikten. In dit geval gebruikte hij poort 9100 van de printer die in veel merken wordt gebruikt om vanop afstand print-opdrachten door te sturen.



Figuur 6.2: Bron afbeelding en artikel: https://www.inktweb.nl/blog/hacker-neemt-150-000-printers-over/

6.1.3 Onveilig ecosysteem

Een IoT-apparaat bestaat niet alleen: meestal maakt het deel uit van een verzameling apparaten en diensten die de gebruiker via online dashboards en apps kan bedienen. Denk maar aan de verschillende Google Nest producten (thermostaat, deurbel, maar ook de Hub, etc.) of lichten van Philips (*Hue*) die ook met behulp van andere toestellen kunnen bediend worden (bijvoorbeeld drukknoppen van Niko of Ikea). Kortom, IoT-apparaten maken meestal deel uit van een heus ecosysteem en dus de mate van veiligheid van die externe zaken bepaalt ook die van het IoT-apparaat. Als bijvoorbeeld de dashboards van fitbit.com zouden gecompromitteerd worden bestaat de kans dat de digitale stropers ook toegang tot je gegevens of apparaten krijgen die gebruik maken van het fitbit-portaal.

Gebruikers vergeten ook met welke externe diensten ze hun apparaten hebben gekoppeld. Het gebeurt vaak dat een gebruiker "even iets wil testen" (denk maar de nuttige website "If this then that" (ifttt.com) dat toelaat om services en apparaten te koppelen die iets moeten doen gegeven zelfgekozen triggers en regels) maar nadien wel vergeet deze koppeling uit te zetten. Dit geldt ook voor de vele third-party websites en services die toegang tot uw Google of Facebook/Meta-accounts wensen. Gebruikers kijken dit zelden na en hebben mogelijk tien jaar geleden al toegang gegeven aan een service die ondertussen al door digitale stropers is gecompromitteerd.

6.1.4 Geen of onveilig updatemechanisme

Dit werd al in de introductie van dit hoofdstuk aangehaald. Het updaten van fysieke apparaten is niet evident, zeker niet als het gaat om kleine, low-power-apparaten die mogelijk op de koop toe op een moeilijk bereikbare plaats hangen en dus enkel praktisch geüpdatet kunnen worden als ze een remote update-mechanisme ondersteunen.

Daarnaast kan het ook gebeuren dat een update, om welke reden dan ook, mislukt waardoor het systeem potentieel *gebrickt* geraakt en de gebruiker of administrator dan alsnog een fysieke rollback moet doen (als

het apparaat die mogelijkheid heeft). Er zijn ondertussen erg dikke boeken geschreven over de *wondere wereld van het updaten van IoT-apparaten*. Weet dus dat het belangrijk is dat je als cyberboswachter van IoT-netwerken goed weet wat de update-strategieën van je IoT-apparaten zijn en wat je moet doen als er hier problemen optreden. Merk op dat die problemen zowel het gevolg van digitale stropers kunnen zijn, maar ook van slechte updates van de fabrikant die op hun beurt mogelijk nieuwe beveiligingsproblemen in je apparaten introduceren.

> Tip
>
> Het is ook essentieel dat je je als cyberboswachter abonneert op de juiste informatiekanalen (CVEs, fabrikant-RSS feeds, etc.)

We zijn afhankelijk van de fabrikanten om beveiligingslekken te dichten in onze software en hardware. Uiteraard zijn er fabrikanten die maar x aantal jaren patches uitbrengen, en er zijn ook al tal van IoT-apparaten op de markt waarvan de fabrikant al jaren niet meer bestaat.

Of wat te denken als er een kritieke bug wordt gevonden in een chip die ontelbare apparaten gebruiken. Wie moet dan de patch voorzien: de fabrikant van het IoT-apparaat, of die van de chip?

> Tip
>
> In september 2021 verscheen een nieuwe Bluetooth kwetsbaarheid, getiteld **BrakTooth** die mogelijk op miljoenen IoT-apparaten kan misbruikt worden. De bug zelf bevindt zich in de Bluetooth-chip van de apparaten. Aardig wat fabrikanten van dergelijke chips hebben al beloofd een patch uit te brengen, maar er zijn er ook die simpelweg zeggen "we zullen enkel patchen als er genoeg vraag naar is", nochtans kan de kwetsbaarheid erg veel schade aanbrengen (DoS-aanvallen en *arbitrary code execution (ACE)*)
>
> 
>
>
> Figuur 6.3: BrakTooth: de zoveelste Bluetooth kwetsbaarheid.

6.1.5 Oude of onveilige componenten

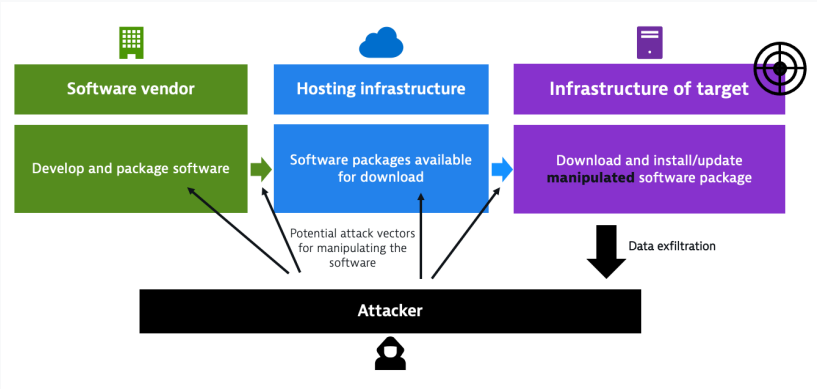
Het voorgaande voorbeeld i.v.m. BrakTooth kan ook in deze sectie gebruikt worden. Een IoT-apparaat is en blijft een combinatie van tientallen, soms honderden hardware-onderdelen. De kans dat al deze onderdelen van de IoT-fabrikant zelf zijn is onbestaande. Gooi eender welke smartphone of digitale horloge open en je zal tal van onderdelen vinden van een andere fabrikant. De IoT-fabrikant moet er dus vanuit gaan dat deze componenten voldoen aan de veiligheidseisen die hij van z'n eigen product verwacht. Dat is geen evidentie.

Een zelfde fenomeen zien we enkele lagen hoger in de OSI-stack: ook op software niveau gebeurt het zelden dat de IoT-ontwikkelaars alle bibliotheken en protocollen *van scratch* hebben ontwikkeld (en dus ook zelf kunnen instaan voor de robuustheid ervan inzake cyberveiligheid). Wanneer er een bug (of bewuste backdoor) wordt ontdekt in een protocol of bibliotheek, dan heeft dit gevolgen voor alles en iedereen die deze dingen in zijn of haar apparaten gebruikt (denk maar aan de Log4J en Heartbleed voorvallen die we in hoofdstuk 1 vernoemden).

Tip

In 2020 was er veel heisa omtrent de *SolarWinds-hack* bij SolarWinds. Duizenden grote bedrijven (en Amerikaanse overheidsinstellingen) zoals Microsoft gebruiken het Orion Platform van dit bedrijf. Dit platform laat toe om enterprise netwerken te monitoren en managen, iets wat niet evident is op de schaal waarmee de klanten van SolarWinds werken. Deze tool moet natuurlijk waterdicht zijn, wat deze niet bleek te zijn: hackers sloegen er in om de FTP-server van SolarWinds te benaderen (het wachtwoord was *solarwinds123...*) en konden zo een backdoor in het Orion platform inbouwen. Vervolgens, wanneer de klanten van SolarWinds, hun Orion-installatie updateten, kregen ze onvrijwillig een versie met een bewust ingebouwde backdoor die de hackers vervolgens konden misbruiken.

Dit soort aanval heet een **supply chain attack**: de aanvallers richten zich op de *weakest link* binnen de supply chain om zo finaal tot bij de klant *achteraan* de keten te geraken.



Figuur 6.4: Bron dynatrace.com

6.1.6 Onvoldoende privacy bescherming

Ook IoT-fabrikanten moeten natuurlijk op een veilige, GDPR-compliant, manier omgaan met de persoonlijke informatie van hun gebruikers. Helaas wordt dit geregeld over het hoofd gezien. Ook hier hebben we weer het probleem van de vele potentiële manieren waarmee een digitale stroper (of de fabrikant zelf!) een IoT-netwerk en apparaat kan misbruiken. De persoonlijke data staat mogelijk wel op een ultra-beveiligde database bij de fabrikant *on-premise*, maar wat baat dat als deze informatie ook als *plaintext* op het apparaat wordt bewaard. Wat ons ook automatisch bij het volgende punt brengt.

6.1.7 Onveilige datatransfer en opslag

Herinner je je de McCumber kubus aan de start van dit handboek? Er werd toen benadrukt dat C.I.A. toepassen op je data geen nut heeft als je geen rekening houdt met alle vormen waarin de data zal voorkomen. Als je enkel encryptie gebruikt tijdens het versturen van data, maar niet tijdens het verwerken en bewaren ervan, dan kan je even goed ook de transmissie beter ongeëncrypteerd doen (zodat we geen *false sense of security* aan de gebruiker geven).

We vallen in herhaling, maar dus zeker bij IoT-systemen die binnen een groot ecosysteem werken, met tal van *third-party* modules en bibliotheken, is het uitermate belangrijk dat ten allen tijde de IoT-data in al z'n vormen, op alle momenten, aan *confidentialiteit, integriteit en beschikbaarheid* doet.

6.1.8 Gebrek aan apparaat management

Spreken over IoT in een enterprise of industriële omgeving is spreken over IoT netwerken met honderden tot duizenden apparaten die bediend moeten worden (denk maar aan remote upgrades, instellingen wijzigen, etc.). Er zijn vele factoren waarom een robuust *IoT device management* systeem vereist is:

- De hoeveelheid apparaten: te veel om manueel te bedienen.
- De locatie: mogelijk bevinden de apparaten zich op moeilijk te bereiken of gevaarlijke plekken.
- Alarmen benodigd: soms wil je op de hoogte gesteld worden wanneer een bepaalde IoT sensor iets meet of niet goed werkt.

- Nuttig gebruik van personeel: personeel enkel naar je IoT-apparaten sturen als dat écht nodig is.
- Deel van *mission critical applicaties*: als je apparaten deel uitmaken van bedrijfskritische systemen is het uitermate belangrijk dat deze apparaten maximaal functioneren, daar iedere downtime of crash in winstverlies of boetes kan resulteren.

Een deftig apparaat management systeem moet natuurlijk zelf ook veilig en robuust zijn. Er zijn meerdere high-end oplossingen beschikbaar maar het spreekt voor zich dat dit soort systemen zelden goedkoop zijn.

6.1.9 Onveilige standaard instellingen

Dit item komt nog steeds voor in alle "10 veiligheidsproblemen die het meeste voorkomen"-lijstjes. Wanneer een apparaat *uit de doos* wordt gehaald is deze ingeladen met een hele hoop standaard instellingen. Instellingen die digitale stropers kunnen opzoeken en misbruiken. Niet alleen het wachtwoord moet ogenblikkelijk aangepast worden, vaak staan ook bepaalde services open zonder dat de gebruiker deze ooit zal gebruiken. Kortom, het is een gezonde gewoonte om, zowel als professional als huis-tuin-en-keuken IoT-gebruiker, steeds alle instellingen van je vers geïnstalleerde apparaat te controleren.

> **i** Opmerking
>
> Er ging lang een *urban legend* de ronde dat je nooit je computer aan het Internet mocht hangen wanneer je Windows XP aan het installeren was. Deed je dat wel dan bestond er de kans dat je computer *gepwnd* werd nog voor je goed en wel de hele installatie had doorlopen.
>
> Ik heb nog geen 100% bewijs hiervan gekregen, maar volgende artikels lijken toch te bevestigen dat het om een effectief probleem gaat:
>
> - **theregister.com/2004/08/19/infected_in20_minutes/**
> - **joeykelly.net/hacks/windows/Windows_XP—Surviving_The_First_Day.pdf**

6.1.10 Gebrek aan fysieke *hardening*

IoT-apparaten liggen meestal verspreid over grote gebieden, ruimtes of gebouwen. De kans dat ze op publiek toegankelijk plekken operationeel zijn is groot, daardoor ook de kans dat digitale stropers toegang tot de apparaten krijgen. Het is daarom belangrijk dat het apparaat ook op fysiek niveau *gehardened* is zodat dit apparaat niet als een open boek uitgelezen of aangepast kan worden. Het is natuurlijk niet evident om dit te voorkomen, net vanwege de aard van het beestje. IoT-apparaten werken dagen, weken, maanden zonder dat er iemand naar om ziet. Computers en telefoons maken deel uit van onze dagelijkse aandacht en daar is het dus veel moeilijker voor aanvallers om ongezien, fysieke, toegang tot te krijgen.

Computers zijn tegenwoordig goed beschermd tegen aanvallers die een laptop stelen en er trachten op in te breken: moderne beschermingen zoals Secure Boot (hoofdzakelijk om rootkits tegen te gaan) en *Trusted Platform Module* (TPM) voorkomen toegang door niet geautoriseerde gebruikers tot de data op het toestel. Dit soort systemen zit zelden in kleine, goedkope, IoT-apparaten waardoor inbreken op dit soort apparaten dus vaak eenvoudig is. Zeker als het apparaat op de koop toe een UART of andere onbeveiligde interface heeft die door de fabrikant worden gebruikt om problemen op te lossen. Ook aanvallers kunnen deze hardware interfaces natuurlijk gebruiken om bijvoorbeeld hun eigen firmware op het apparaat te plaatsen.

> Tip
>
> Er is gelukkig een trend gaande waarbij de TPM-chip (of alternatief) steeds meer z'n weg vindt in nieuwe IoT-apparaten. De initiële kost ervan vergaat meestal in het niets in vergelijking met de schade die kan opgelopen worden wanneer onbeveiligde toestellen (zonder TPM) worden gecompromitteerd.

De Mccumber kubus in gedachte nemende spreekt het natuurlijk voor zich dat ook de data geëncrypteerd op het IoT-apparaat moet bewaard worden.

6.2 Samenvatting

IoT stapelt klassieke security-problemen op een bijzonder ongunstige hardware-context:

- **Low-cost, long-life**: toestellen gaan 10+ jaar mee, update-support vaak een pak korter.

- **Default credentials** en zwakke firmware blijven de #1 instap (cfr. Mirai).
- **Fysieke toegang** betekent bij IoT vaak meteen *game over* — een **TPM-chip** (of alternatief) heft dat grotendeels op.
- **KISS** is een onderschatte verdediging: elke extra feature is extra attack surface.
- De **McCumber kubus** blijft gelden: ook op een sensor moet data beschermd zijn — in rust, tijdens transport én tijdens verwerking.

6.3 Tot slot

En zo hebben we het einde van het prille begin der cyberboswachters bereikt. Zoals je vermoedelijk ook zelf al aanvoelt, hebben we enkel maar het tipje van de ijsberg besproken. Als we er echter gezamenlijk in slagen om dit tipje alvast en zelf toe te passen én aan anderen aan te leren, dan maken we het de digitale stropers al een pak moeilijker. Beschouw dit een beetje als een toepassing van de wet van *diminishing returns.* De primaire, eenvoudige veiligheidstechnieken en gewoonten (goede wachtwoorden, geen default instellingen, KISS, etc.) gaan de grootste impact hebben op je algemene veiligheid. Alle daaropvolgende stappen, hoe belangrijk en nuttig ook, zullen steeds een kleiner stuk van de potentiële problemen voorkomen.

Wees veilig en verspreid het woord! ;)
Tim Dams
2025

Bijlagen

A. GDPR

De General Data Protection Regulation (2018), of in het Nederlands de *Algemene Verordening Gegevensbescherming* is een Europese richtlijn om de privacy van Europese burgers, eender waar in de wereld, te beschermen. Of zoals de EU zelf op haar website beschrijft: “the thoughest privacy and security law in the world” (wat ook effectief zo is: het is de enige wet momenteel die erin slaagt om wereldwijd de regels ervan af te dwingen).

De wet zelf bestaat uit 99 verschillende artikels waarin de rechten, personen en verplichtingen van bedrijven die onder de verordening vallen, worden uitgelegd. Ieder bedrijf, eender waar in de wereld, die data bijhoudt van EU-burgers dient zich aan deze wet te houden op risico van een boete als ze dit niet doet. GDPR gaat dus niet over de verplichtingen van de personen, wel om die van de bedrijven, namelijk ter bescherming van personen.

> **Tip**
>
> We gaan in dit hoofdstuk enkel een high-level overzicht geven van GDPR opdat we niet verzanden in de taal der rechters en advocaten, het *legalese*.

A.1 Wat omvat GDPR?

Het recht om persoonlijke gegevens te wissen of het recht om vergeten te worden. GDPR laat je toe, als EU-burger om eender welke site die ‘jou kent’ te verplichten alle, of specifieke, informatie over je te verwijderen. Als je dus vindt dat Google niet moet weten waar je huisadres is, dan heb je het recht Google te verplichten deze informatie te verwijderen.

> **Opmerking**
>
> Google kreeg een tijd terug een zeer stevige boete omdat het niet inging op de vraag van een EU-burger om *vergeten te worden*. Sindsdien zijn ze een pak behulpzamer in het naleven van de GDPR-wetgeving.

Een ander belangrijk aspect van GDPR is het zogenaamde **recht op overdraagbaarheid van gegevens.** Voor GDPR bestond was het soms een helse opdracht om bijvoorbeeld van Internetprovider te veranderen. Iedere provider werkte met eigen dataformaten waarin de klantengegevens werden bewaard. Bij een overdracht moest dit dan altijd omgezet worden naar het formaat van de nieuwe provider. GDPR verplicht dat dit soort data overdrachten van persoonsgegevens nu volgens een vast formaat moeten gebeuren. En belangrijker: deze overdracht moet zo transparant én zo geautomatiseerd mogelijk gebeuren. Hierdoor kan de data quasi ogenblikkelijk in de database van de ontvanger geïntegreerd worden en zal jij als klant dus veel sneller de overstap kunnen maken zonder alle bijhorende administratieve rompslomp en vertragingen.

> **Opmerking**
>
> De voorganger van de GDPR was de DPD (Data Protection Directive). Dit was geen Europese wet (verordening) maar een aanbeveling en was een stap in de goede richting maar ging nog niet ver genoeg. GDPR heeft ervoor gezorgd dat individuele burgers geen speelbal meer zijn van de grote bedrijven die de data van burgers als het nieuwe goud verzamelen en ermee doen wat ze zelf willen.

Als derde belangrijke aspect voorziet GDPR **de plicht om niet noodzakelijke gegevens te laten wissen**.

A.2 Hoe beschermt GDPR mij als burger?

Zoals je uit voorgaande 3 aspecten al merkt staat de (Europese) burger voorop, niet het bedrijf. GDPR is er om de burger te beschermen tegen de bedrijven die data van hen verzamelen. Het stopt echter niet bij deze 3 aspecten. Volgende rechten worden door GDPR beschermd en zijn minstens even belangrijk:

- Recht op **inzage** (*transparantie*): je hebt het recht om te weten welke data een bedrijf over jou heeft.
- Recht op **rectificatie**: je hebt het recht om foute data te laten corrigeren.
- Recht op **beperking** van verwerking: je hebt het recht om te eisen dat een bedrijf stopt met het verwerken van jouw data. Ook mogen bedrijven je persoonsgegevens niet oneindig lang bewaren. En moeten er duidelijke richtlijnen zijn over hoe lang ze de data mogen bewaren.
- Recht op **verzet**: je hebt het recht om je te verzetten tegen het verwerken van je data. Uiteraard is een bedrijf niet verplicht om jou dan gebruik te laten maken van hun diensten.
- Recht op **geautomatiseerde besluitvorming**: je hebt het recht om te weten wanneer een bedrijf beslissingen neemt op basis van geautomatiseerde processen.
- Recht op **gegevensportabiliteit**: je hebt het recht om je data in een leesbaar formaat te ontvangen.
- Recht op **gegevensminimalisatie**: bedrijven mogen enkel de data bewaren die ze nodig hebben voor hun dienstverlening.

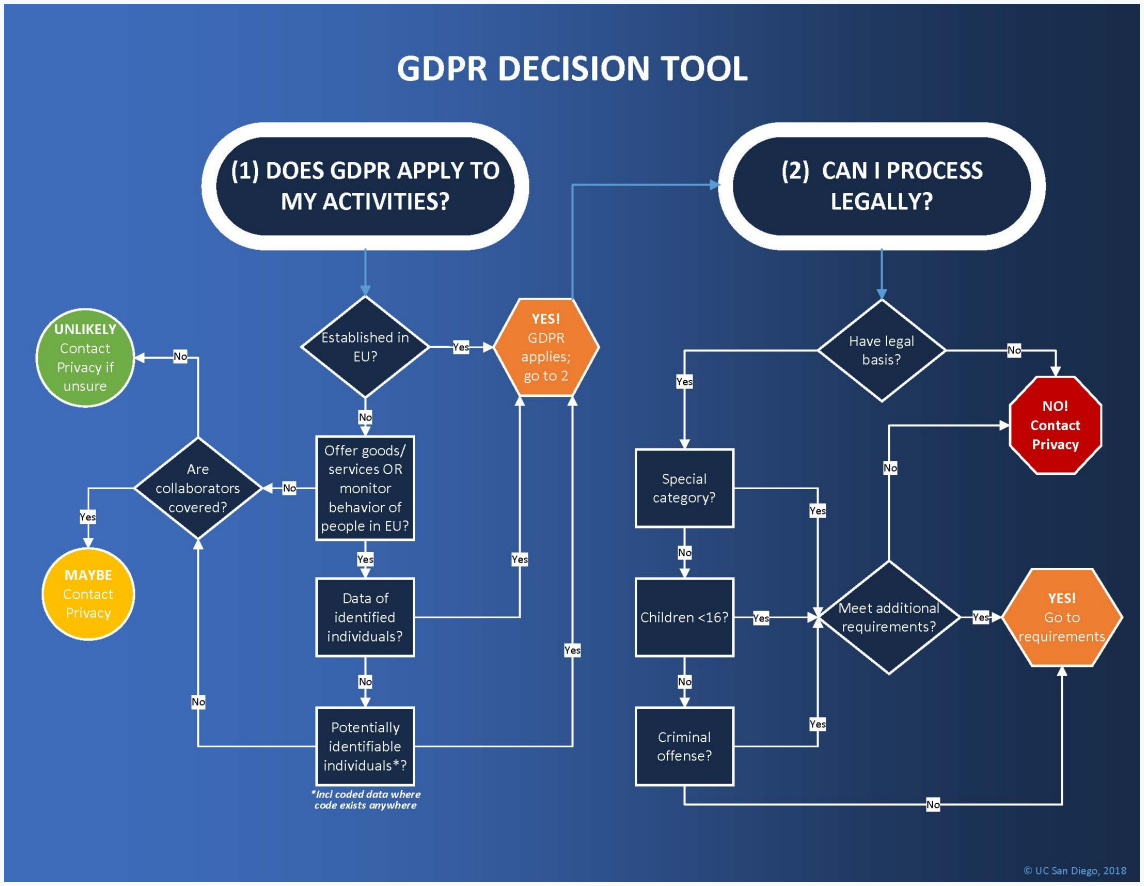
A.3 Wat beschermt GDPR?

De GDPR wet zorgt ervoor dat bedrijven veel bewuster met gebruikersdata omgaan. Ze kunnen namelijk verantwoordelijk gehouden worden indien gebruikersdata gestolen of gelekt wordt. De GDPR is een privacy-wetgeving in de eerste plaats: het wil de privacy van het individu beschermen. Volgende data valt dan ook onder de GDPR-wetgeving:

- Persoonlijke identificeerbare informatie: namen, adressen, geboortedatums en rijksregisternummers.
- Op Internet gebaseerde gegevens: zoals gebruikerslocatie, IP-adres en cookies.
- Gezondheids- en genetische gegevens.
- Biometrische gegevens (denk maar aan je irisscan of vingerafdruk).
- Raciale en/of etnische gegevens.
- Politieke meningen.
- Seksuele geaardheid.

A.4 Persoonlijke data buiten EU

Volgende flowchart toont wat bedrijven, binnen en buiten de EU, met gebruikersdata mogen doen.



Figuur A.1: Bron: privacy.ucsd.edu/gdpr/

A.5 Meldplicht datalekken

Bedrijven die onder de GDPR-wetgeving vallen (alle bedrijven die data van Europese burgers bewaren vallen hier onder, ongeacht hun locatie in de wereld) hebben een meldplicht indien zij een datalek hebben.

> **i** Opmerking
>
> Onder een datalek verstaan we *"Een inbreuk op de beveiliging die per ongeluk, of op onrechtmatige wijze leidt tot de vernietiging, het verlies, de wijziging of de ongeoorloofde verstrekking van of de ongeoorloofde toegang tot doorgezonden, opgeslagen of anderszins verwerkte gegevens."*

Van zodra het bedrijf weet heeft van een (mogelijk) datalek dienen zij dit ogenblikkelijk te melden:

- Aan de **toezichthoudende autoriteit**: ieder land heeft z'n eigen autoriteit en in België is dat de **CBPL**, de *Commissie voor de Bescherming van de Persoonlijke Levenssfeer*. Deze commissie zal iedere datalek *case-by-case* beoordelen om te zien of het bedrijf een GDPR-overtreding heeft begaan of niet.
- Aan **ieder individu waar de data betrekking op had**: als de gelekte data ook maar op één manier kan gelinkt worden aan een individu, dan dient deze van de lek op de hoogte gebracht te worden.
- Dit dient **binnen de 72 uur na ontdekking van het datalek te gebeuren**.

Merk op dat tegenwoordig een datalek zelden zo snel wordt ontdekt, dat beseft de GDPR ook. Daarom is de regel dat 72 uur ingaan vanaf het moment dat het bedrijf weet heeft van het datalek. Het is dus perfect mogelijk dat het datalek reeds maanden eerder plaatsvond maar dat het bedrijf het nu pas ontdekt.

A.5.1 Uitzonderingen meldplicht individu

Bij een datalek zijn er enkele mogelijke uitzonderingen waarom het bedrijf niet noodzakelijk het individu op de hoogte moet stellen van het datalek:

- Indien het bedrijf kan aantonen dat de gestolen data zodanig beveiligd is dat deze onbruikbaar is. Als dus de data bijvoorbeeld geëncrypteerd is dan zou het bedrijf dit als argument kunnen gebruiken om de meldplicht te verzaken. Een kanttekening is hier natuurlijk op z'n plaats: er bestaat altijd nog de mogelijkheid dat de data finaal nog kan gedecrypteerd zal worden en er dus alsnog private gegevens in verkeerde handen komen.
- Stel dat het bedrijf de persoonlijke voorkeuren van de gebruikers in een database heeft staan en de identificeerbare persoonsgegevens in een andere. Als nu blijkt dat enkel de database met persoonlijke voorkeuren werd gehackt, dan kan deze data nooit gelinkt worden aan personen en is er dus ook geen sprake van een privacy schending.

> **Tip**
>
> Het is sowieso altijd een goede gewoonte om het adagio *"don't put all your eggs in one basket"* te hanteren. Het is wijzer om de data over verschillende databases te bewaren. Zo voorkom je dat bij een datalek automatisch steeds alle data wordt gestolen.

> **Waarschuwing**
>
> Indien er een erg grote hoeveelheid mensen moeten ingelicht worden na een datalek, dan mag het bedrijf ook beslissen om in de plaats daarvan een publieke aankondiging te doen. Uiteraard zijn de bedrijven hier minder happig op omdat dit sowieso een PR-nachtmerrie is (iedere aankondiging van een datalek is dat trouwens).

A.6 Boetes

Als er uiteindelijk toch een inbreuk op de GDPR-wetgeving wordt vastgesteld dan zal er dus een boete opgelegd worden afhankelijk van de ernst van de inbreuk:

- Minder ernstige inbreuken: boete tot € 10 miljoen, of 2% van de wereldwijde jaaromzet van het bedrijf uit het voorgaande boekjaar, afhankelijk van welk bedrag hoger is.
- Meer ernstige inbreuken (inbreuken die ingaan tegen de principes van het recht op privacy en het recht om te worden vergeten, die de kern vormen van GDPR): boete van maximaal € 20 miljoen, of 4% van de wereldwijde jaaromzet, afhankelijk van welk bedrag hoger is.

Of er een boete zal zijn en hoe groot deze dan is, is gebaseerd op een aantal criteria:

- Ernst en aard van de overtreding.
- Intentie: is het datalek het gevolg van een slordigheid of opzettelijk gebrek aan beveiliging?
- Schadebeperkende omstandigheden.
- Genomen voorzorgen.
- Geschiedenis van overtredingen.
- Medewerking: in hoeverre werkt het bedrijf mee met de toezichthoudende autoriteiten?
- Gegevenscategorie.
- Kennisgeving: heeft men zich aan de meldplicht gehouden?
- Certificering: heeft het bedrijf in het verleden reeds via een externe audit aangetoond aan de GDPR wetgeving te voldoen?
- Verzwarende of verzachtende factoren.

> **Tip**
>
> Merk dus op dat het schenden van de GDPR wetgeving niet automatisch in een boete resulteert. Als het bedrijf kan aantonen dat ze echt alles in het werk hebben gesteld om de data te bewaren zoals een goede huisvader betaamt, dan kan het zijn dat er geen boete zal volgen.

> **Opmerking**
>
> De voorbije jaren (mei 2018 tot eind 2020) werden reeds 272 miljoen euro aan boetes geïnd ten gevolge van GDPR overtredingen. De wet is zo effectief, dat ze ook ondertussen dient als inspiratie wereldwijd wanneer landen of groeperingen nieuwe privacy-wetgevingen willen construeren.

> **Tip**
>
> In de lente van 2025 kreeg TikTok een boete van 530 miljoen opgelegd voor het overtreden van de GDPR-wetgeving. Het bedrijf kon bijvoorbeeld niet aantonen dat de data van de Europese gebruikers adequaat beveiligd werd op de servers in China. We moeten er wel bij vertellen dat de boete gebaseerd is op de situatie voor 2023. Vermoedelijk én hopelijk heeft TikTok ondertussen zijn beveiliging op orde.

A.7 Conclusie

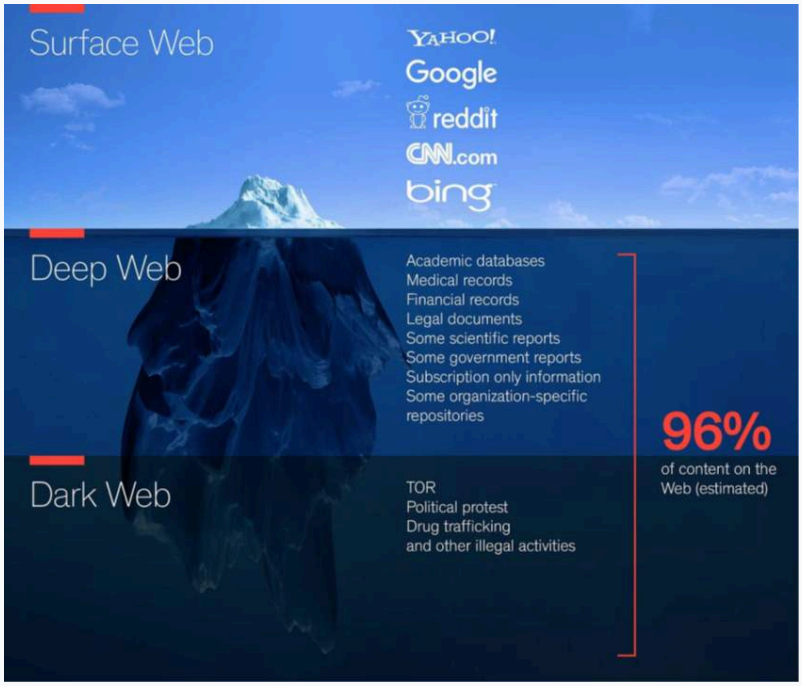
We kunnen dus stellen dat GDPR een zeer mooi waardevol initiatief is, dat ook verder gaat dan “niet toegepaste wetten”. De manier waarop Europese burgers hun data momenteel wordt beschermd is een voorbeeld voor de rest van de wereld. Het is een wet die de privacy van de burger voorop stelt en bedrijven verplicht om hiermee rekening te houden.

Het is nu vooral uitkijken of Europa een soortgelijk voorbeeld kan stellen met de Digital Services Act (DSA) en de Digital Markets Act (DMA) die momenteel in de maak zijn. Deze wetten zullen de macht van de grote techbedrijven zoals Google, Facebook en Amazon moeten inperken en de privacy van de Europese burger nog verder moeten beschermen. Ook de wereld van artificiële intelligentie zal hierdoor grondig veranderen: de AI Act die sinds augustus 2024 gefaseerd wordt ingevoerd zal de manier waarop AI wordt gebruikt in Europa grondig veranderen. Uiteraard, en dat geldt voor alle wetgevingen, is het de uitvoering en de handhaving van de wetten die het verschil zullen maken én wat de impact ervan zal zijn in de concurrentie met de rest van de wereld.

B. Dark web

Het internet is vergelijkbaar met de echte wereld: het heeft goed verlichte plekken die iedereen kan zien, maar ook verborgen achterbuurten. Het Darkweb is zo'n verborgen gedeelte van het internet. Het maakt deel uit van het zogenaamde "Deep Web", dat gedeelte van het internet dat niet via standaard zoekmachines zoals Google te vinden is. Denk hierbij aan databanken, afgeschermde websites en persoonlijke accounts die niet publiek toegankelijk zijn.

Binnen dat Deep Web bestaat een extra verborgen deel, het Darkweb, bekend om anonimiteit en privacy. Dit deel van het internet is alleen toegankelijk via speciale software zoals TOR (The Onion Router).



Figuur B.1: De internet ijsberg

B.1 Kenmerken van het Darkweb

Het Darkweb onderscheidt zich van het reguliere internet door een aantal belangrijke kenmerken:

Anoniem: Gebruikers blijven anoniem doordat hun verbinding via meerdere tussenliggende servers loopt. Hierdoor is het zeer moeilijk te achterhalen wie een bepaalde website bezoekt of een bepaalde boodschap heeft verzonden.

Beveiligd: Communicatie en verkeer op het Darkweb zijn versleuteld met sterke encryptie. Dit voorkomt dat derden kunnen meekijken of informatie onderscheppen.

Gedecentraliseerd: Er is geen centrale autoriteit of punt van controle op het Darkweb. Dit maakt het moeilijk voor overheden en instanties om toezicht te houden, te censureren of sites uit de lucht te halen.

Zoals elke technologie kan ook het Darkweb gebruikt worden voor goede én slechte doeleinden:

- Slecht:
  - Handel in illegale goederen zoals drugs, wapens en gestolen informatie.
  - Verhuur van criminele diensten ("rent-a-hacker").
- Goed:
  - Bescherming van privacy en anonieme communicatie.

– Platforms voor journalisten, klokkenluiders en activisten om veilig te communiceren.

B.2 Waarom bestaat het Darkweb?

Een belangrijk uitgangspunt: **het gewone internet is niet anoniem**. Via je IP-adres, cookies, logs van providers en talloze trackers ben je bijna altijd traceerbaar. Voor de meeste mensen is dat geen probleem — maar voor sommigen wél.

Er zijn immers landen die het hele internet in de gaten houden of censureren, zoals China, Saudi-Arabië, Iran, Egypte, Noord-Korea en Cuba. Dat toezicht kent verschillende vormen:

- **Volledige controle**: de overheid monitort al het verkeer en logt wie wat doet.
- **Blacklisting**: bepaalde sites of diensten (sociale media, nieuwssites, messengers) worden gewoon geblokkeerd.

In die context biedt het Darkweb een veilige ruimte voor journalisten, activisten en klokkenluiders om zich vrij en anoniem te uiten. Denk aan het belang van een **schuilnaam** (*nom-de-plume*): niet iedereen — en zeker niet de overheid — moet weten wat jij in je vrije tijd leest, schrijft of bespreekt. Anonimiteit is zo een cruciaal hulpmiddel voor vrije meningsuiting en democratie.

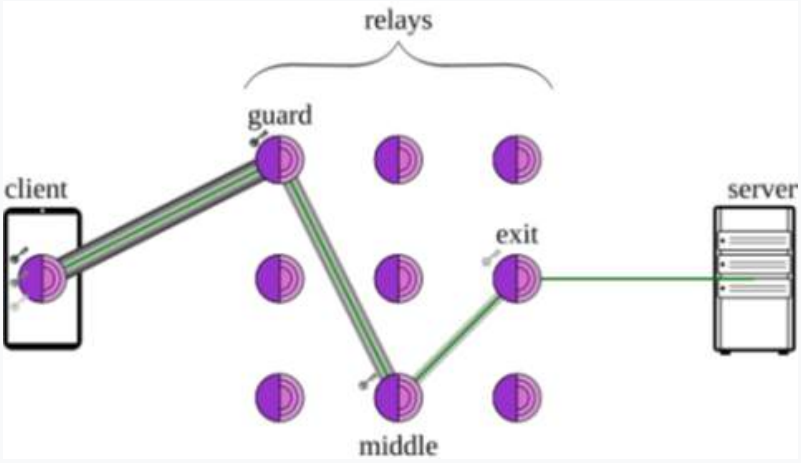
Aan de andere kant trekt anonimiteit ook mensen aan die het internet gebruiken voor illegale activiteiten, zoals handel in drugs, wapens, gehackte gegevens, of gestolen creditcardgegevens. Belangrijk om te onthouden: **anonimiteit is geen vrijgeleide**. Als je iets illegaal doet op het Darkweb, ben je sowieso strafrechterlijk vervolgbaar — politiediensten hebben wel degelijk middelen om criminelen op het Darkweb op te sporen.

Het Darkweb is dus niet per definitie illegaal, maar wat je erop doet kan dat wel zijn. In sommige landen, zoals Rusland en China, is het gebruik van het Darkweb zelf al strafbaar.

Het vinden van een evenwicht tussen privacy en vrijheid enerzijds, en opsporing en veiligheid anderzijds, blijft een **moeilijke afweging** die onze samenleving voortdurend opnieuw moet maken.

B.3 Toegang tot het Darkweb: TOR

De meest voorkomende manier om het Darkweb te bezoeken is via het zogenaamde TOR-netwerk (The Onion Router). Dit netwerk leidt jouw internetverkeer via meerdere relais-servers om jouw identiteit te verhullen. Een website op het Darkweb herken je aan het ".onion"-adres, wat alleen via een TOR-browser te openen is.



Figuur B.2: Er worden letterlijk lagen, zoals een UI, tussen jou en je doel opgebouwd.

B.4 TOR heeft ook nadelen:

Snelheid: Het TOR-netwerk is relatief traag door de vele tussenstations.

Blokkades: Sommige websites blokkeren verkeer vanaf TOR-servers.

Firewall: Veel bedrijfsnetwerken en firewalls blokkeren verkeer van en naar het TOR-netwerk.

Er bestaan ook browsers, zoals Brave Browser, die TOR al geïntegreerd hebben, waardoor het gebruik eenvoudiger wordt.

B.5 Waarom moet je dit nu weten?

Het is belangrijk om inzicht te hebben in de risico's en kansen die het Darkweb biedt. Denk hierbij aan:

- Het begrijpen van criminele fenomenen zoals online drugshandel, cybercrime en mensenhandel.
- Het analyseren van bedreigingen en risico's voor organisaties en individuen.
- Het begrijpen van het belang van privacy, vrije meningsuiting, en de complexiteit van online toezicht en censuur.

Het Darkweb maakt dus deel uit van het bredere veiligheidsvraagstuk binnen de samenleving.

> **i** Opmerking
>
> Vanwege tijdsgebrek werd deze appendix over het Darkweb grotendeels gegenereerd m.b.v. ChatGPT, gebaseerd op de slidedeck.

C. Cybersecurity Awareness creëren

In dit hoofdstuk bekijken we de belangrijke rol die **cybersecurity-awareness** speelt binnen organisaties. Als cybersecrurity-professionals zijn we ons bewust van de risico's, maar hoe zorgen we ervoor dat ook onze collega's deze risico's begrijpen en ernaar handelen? Dit hoofdstuk biedt inzicht in de noodzaak van awareness, de rol van medewerkers, en praktische tips om een cultuur van cybersecurity te bevorderen.

C.1 Waarom awareness?

De voorbije hoofdstukken hebben hopelijk al heel duidelijk gemaakt dat er zich nog te vaak cybersecurity-incidenten voordoen. Veel van die incidenten hadden eenvoudig kunnen worden voorkomen als werknemers (en gevers!) ook maar enige, minimale cybersecurity-awareness-training hadden gehad. Uit onderzoek blijkt dat ongeveer **90% van alle beveiligingsincidenten veroorzaakt wordt door menselijke fouten**, bewust of onbewust. Denk hierbij aan het klikken op phishing-links, het gebruiken van zwakke wachtwoorden, of het verbinden met onveilige wifi-netwerken. Deze menselijke factor maakt awareness onmisbaar binnen een robuust beveiligingsbeleid. Die laatste zin willen we nog even benadrukken: **bewustzijn is onmisbaar binnen een robuust beveiligingsbeleid**. Het is niet voldoende om alleen technische maatregelen te nemen; medewerkers moeten ook begrijpen waarom deze maatregelen belangrijk zijn en hoe ze deze kunnen toepassen in hun dagelijkse werkzaamheden.

C.1.1 Jouw rol binnen een organisatie

Jouw taak als cybersecurity expert gaat verder dan enkel systemen te configureren en onderhouden. Jij vormt ook de brug tussen IT en bedrijfsbeleid. Jij bent als het ware een **bewustzijnsambassadeur** die complexe technische informatie vertaalt naar begrijpelijke taal voor collega's.

Daarnaast heb je ook een signalerende rol:

- Identificeren van potentiële risico's.
- Rapporteren van afwijkingen of verdachte gebeurtenissen, en dit op een snelle en duidelijke manier.

C.1.2 Bescherm de hele keten: McCumber Cube

In hoofdstuk 2 bespraken we reeds de McCumber cube. Deze visualisatie helpt ons om in te zien dat we cybersecurity vanuit al z'n dimensies en facetten moeten bekijken. Hierbij waren de voorbije hoofdstukken vooral gericht op het technologie-aspect in functie van C.I.A en de staat van de data (processing, storage, transmission). In dit hoofdstuk gaan we daarom dieper in op de **menselijke** dimensie van cybersecurity. Dit is een cruciaal aspect dat vaak over het hoofd wordt gezien, maar dat net zo belangrijk is als de technische maatregelen die we nemen. De menselijke factor is immers de zwakste schakel in de beveiligingsketen. Daarom is het essentieel dat we ons richten op het creëren van een cultuur van cybersecurity-awareness binnen onze organisatie.

C.2 Awareness-campagnes

Om de medewerkers in een bedrijf te trainen in cybersecurity, zijn er verschillende methoden en technieken beschikbaar. Het doel van deze trainingen is om medewerkers bewust te maken van de risico's en hen te leren hoe ze veilig kunnen werken. Dit kan door middel van e-learning, workshops, of zelfs gamification. Het is belangrijk dat deze trainingen regelmatig worden herhaald, zodat medewerkers op de hoogte blijven van de laatste ontwikkelingen en trends op het gebied van cybersecurity.

Meestal gaan we dit doen aan de hand van, terugkerende, **awareness-campanges**.

C.2.1 Gebruikersverwarring voorkomen

Losse technische maatregelen zijn niet voldoende. We moeten ook zorgen dat medewerkers begrijpen waarom deze maatregelen belangrijk zijn en hoe ze deze kunnen toepassen in hun dagelijkse werkzaamheden. Dit vereist een combinatie van technische kennis, communicatievaardigheden en leiderschap.

Medewerkers hebben namelijk vaak vragen zoals:

- "Waarom mag ik geen publiek wifi gebruiken?"
- "Waarom moet ik steeds opnieuw inloggen met multifactor-authenticatie (MFA)?"
- "Waarom moet ik mijn laptop vergrendelen?"

Het doel van awareness-campagnes is medewerkers zelfstandig antwoorden op deze vragen te laten geven. Niet door ze regels op te leggen, maar door ze inzicht te geven in de risico's en gevolgen van bepaalde gedragingen. Deze aanpak geldt trouwens eender waar in een omgeving waar er regels, wetten en procedures zijn. Het is niet voldoende om enkel te zeggen "dat mag niet". We moeten ook uitleggen waarom iets niet mag en wat de gevolgen zijn van het negeren van deze regels.

Effectieve awareness-campagnes leggen dus niet enkel uit **wat** je moet doen, maar vooral ook **waarom**:

- Gebruikers leggen zelfstandig de link tussen risicogedrag en de mogelijke gevolgen.
- Ze maken bewust de keuze voor veilige werkmethoden.
- Veilig handelen wordt een routine, intrinsiek gemotiveerd door begrip.

Uiteindelijk willen we dat medewerkers autonoom zijn:

- Ze herkennen en rapporteren nieuwe risico's zelfstandig.
- Ze kunnen geleerde principes toepassen op onbekende situaties.

C.2.2 Succesvolle awareness-methoden

De mate waarin een campagne werkt hangt van vele factoren af. Het is belangrijk dat de campagne dan ook binnen de context van het bedrijf past: er is geen "one size fits all". Hoe beter de campagne is afgestemd op het bedrijf en de medewerkers, hoe effectiever de campagne zal zijn. Gebruik met andere woorden realistische scenario's en voorbeelden die aansluiten bij de dagelijkse werkzaamheden van de medewerkers. Dit maakt het makkelijker voor hen om de informatie te begrijpen en toe te passen in hun eigen werk.

Alhoewel er tal van bedrijven zijn die awareness-campagnes aanbieden, zijn er ook een aantal goede gratis bronnen beschikbaar. Denk hierbij aan de website van de **SANS Institute** of de **Cybersecurity & Infrastructure Security Agency (CISA)**. Deze organisaties bieden gratis materialen en trainingen aan die bedrijven kunnen gebruiken om hun medewerkers bewust te maken van cybersecurity-risico's.

Enkele effectieve methodes voor cybersecurity-awareness zijn:

- **Regelmatige micro-trainingen**: korte, frequente sessies. Dit voorkomt dat medewerkers overweldigd worden door informatie en zorgt ervoor dat ze de informatie beter onthouden.
- **Phishing-simulaties**: directe feedback op acties zorgt voor snelle leerprocessen. Dit kan bijvoorbeeld door medewerkers te testen met nep-phishing-e-mails. Dit helpt hen om verdachte e-mails te herkennen en voorkomt dat ze in de val lopen van echte phishing-aanvallen.
- **Gamification**: quizzen en uitdagingen verhogen motivatie en betrokkenheid. Je zou bijvoorbeeld via een dashboard kunnen tonen welke dienst binnen het bedrijf het beste presteert op het gebied van cybersecurity-awareness. Dit kan medewerkers motiveren om beter op te letten en hun kennis te vergroten.
- **Posters en nieuwsbrieven met impact**: visuele reminders in de werkomgeving. Dit kan bijvoorbeeld door posters op te hangen met tips voor veilig werken of door het gebruik van stickers op laptops en andere apparaten. Dit zorgt ervoor dat medewerkers regelmatig worden herinnerd aan de belangrijkste principes van cybersecurity. Als deze posters op de koop toe grappig, of zelfs uitdagend zijn, zullen medewerkers ze sneller opmerken en onthouden.

Figuur C.1: Awareness posters hoeven geen ellenlange teksten te bevatten. Less is more.

- **Leiderschap**: het management moet het goede voorbeeld geven. Dit kan bijvoorbeeld door zelf ook regelmatig deel te nemen aan trainingen of door openlijk te praten over cybersecurity-risico's. Dit laat zien dat cybersecurity een prioriteit is binnen de organisatie en dat iedereen verantwoordelijk is voor het waarborgen van de veiligheid.

Gedragsverandering is vaak de moeilijkste uitdaging binnen cybersecurity. Positieve gedragsverandering bereik je niet door medewerkers te straffen of te bedreigen, maar door ze te motiveren en te inspireren.

C.3 Praktische best practices

Doorheen het handboek hebben we al tal van praktische "best practices" aangereikt. We vatten ze hier nog eens samen, daar zij de *backbone* zullen vormen van de inhoud van je awareness-campagnes. Dit zijn de basisprincipes die elke medewerker moet kennen en toepassen in zijn of haar dagelijkse werkzaamheden. Deze principes zijn niet alleen van toepassing op cybersecurity, maar ook op andere gebieden binnen een organisatie. Ze helpen medewerkers om veilig en efficiënt te werken, ongeacht hun functie of verantwoordelijkheden.

Wachtwoordhygiëne

- Gebruik sterke, unieke wachtwoorden.
- Maak gebruik van een wachtwoordmanager.
- Zet altijd multifactor-authenticatie aan.

Secuur apparaatbeheer

- Installeer automatische updates.
- Gebruik antivirussoftware.
- Vergrendel je laptop wanneer je weggaat.

Veilig werken met documenten en netwerken

- Vermijd onbekende USB-sticks en kabels.
- Gebruik enkel vertrouwde wifi-netwerken.
- Maak regelmatig backups van documenten, zowel fysiek als digitaal.

Blijf alert en leer verder

- Deel kennis actief binnen je team.
- Blijf bij met nieuwe ontwikkelingen op het gebied van cybersecurity.

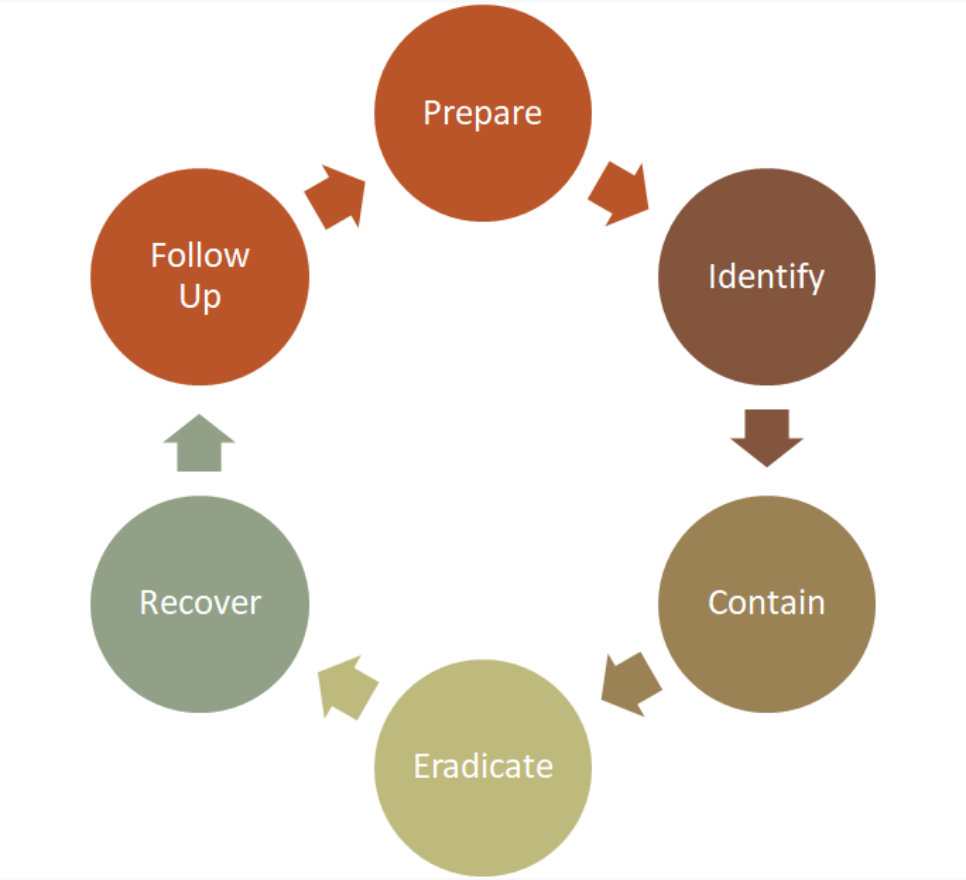
C.4 Incident response

Awareness is ook cruciaal in incidentbeheer. Indien je medewerkers bewust zijn van de risico's en weten hoe ze moeten reageren op incidenten, kunnen ze sneller en effectiever handelen. Dit kan de impact van een incident aanzienlijk verminderen.

Incident response is een gestructureerde aanpak voor het afhandelen van beveiligingsincidenten. Het doel is om de schade te minimaliseren, de oorzaak van het incident te begrijpen, en herhaling in de

toekomst te voorkomen. Een goed **incident response-plan** omvat duidelijke procedures, verantwoordelijkheden, en communicatiekanalen. Dergelijk plan bestaat uit verschillende stappen, die vaak worden weergegeven in een cyclus. De stappen zijn als volgt:

1. **Prepare**: Voorbereiden op incidenten vóór ze gebeuren. In deze fase worden alle procedures en verantwoordelijkheden gedefinieerd. Dit omvat ook het trainen van medewerkers in hoe ze moeten reageren op incidenten. Dit is waar awareness-campagnes een belangrijke rol spelen. Het incident response-plan moet meer zijn dan een document dat in de kast ligt. Het moet een levend document zijn dat regelmatig wordt bijgewerkt en waar medewerkers actief mee aan de slag gaan.
2. **Identify**: Incidenten snel herkennen. Dit kan door middel van monitoring, logging, en rapportage. Het is belangrijk dat medewerkers weten hoe ze verdachte activiteiten kunnen herkennen en rapporteren. Dit kan bijvoorbeeld door het gebruik van een meldingsformulier of een speciaal meldpunt binnen de organisatie.
3. **Contain**: Beperk de schade direct na detectie. Dit kan door het isoleren van getroffen systemen of het blokkeren van verdachte accounts. Het is belangrijk dat medewerkers weten hoe ze snel kunnen handelen om verdere schade te voorkomen.
4. **Eradicate**: Verwijder het incident volledig. Vanaf deze, of de vorige stap zal meestal de IT-afdeling de verantwoordelijkheid nemen. Het is belangrijk dat medewerkers weten dat ze niet alleen verantwoordelijk zijn voor het melden van incidenten, maar ook voor het helpen bij het oplossen ervan. Dit kan bijvoorbeeld door het verzamelen van informatie over het incident of door het uitvoeren van bepaalde taken binnen de organisatie.
5. **Recover**: Herstel systemen en data.
6. **Follow Up**: Analyseer wat er is gebeurd en verbeter processen.



Figuur C.2: Vaak wordt een incident response plan als een lineaire sequentie getoond, zonder te visualiseren dat er altijd na de followup informatie moet doorstromen naar de prepare fase. Op die manier leren we uit onze fouten, in de hoop dat deze volgende keer niet meer zullen voorkomen.

Awareness helpt vooral bij de **Identify** stap:

- Goede meldprocedures zijn dan ook cruciaal (wie, wat, hoe snel).
- Maak meldkanalen toegankelijk en laagdrempelig.
- Snelheid van identificatie bepaalt vaak de impact van een incident.

Het heeft geen nut om uiterst complexe incident response-plannen te maken, als medewerkers niet weten hoe ze deze moeten toepassen. Daarom is het belangrijk dat medewerkers goed op de hoogte zijn van de procedures en verantwoordelijkheden binnen het incident response-plan.
Het internet staat vol van nuttige hulpmiddelen om awareness campagnes uit te voeren. Gaande van toffe posters, tot *fullblown* phishing simulators (kijk zeker eens naar **GoPhish**). Met dank aan de immer krachtiger wordende mogelijkheden van ChatGPT en friends wordt het ook steeds eenvoudiger om succesvolle campagnes uit te werken en te visualiseren.

C.5 Social engineering en phishing

De eenvoudigste manier om medewerkers te hacken is door middel van social engineering. De aanvaller heeft hier namelijk geen tools of technieken voor nodig, enkel een goed verhaal. Social engineering is het manipuleren van mensen om vertrouwelijke informatie te verkrijgen of toegang te krijgen tot systemen. Dit kan bijvoorbeeld door middel van phishing-e-mails, waarbij de aanvaller zich voordoet als een vertrouwde bron om medewerkers te misleiden. Het is belangrijk dat medewerkers zich bewust zijn van deze risico's en weten hoe ze verdachte e-mails kunnen herkennen en rapporteren.

Omdat het leuk is om lijstjes te maken, hebben we hier een lijstje met de meest voorkomende social engineering technieken. Meestal zullen de digitale stropers een combinatie van deze technieken gebruiken om hun doel te bereiken.

- **Pretexting**: De aanvaller doet zich voor als iemand anders, bijvoorbeeld een collega of een IT-medewerker, om vertrouwelijke informatie te verkrijgen. Dit kan bijvoorbeeld door middel van een telefoongesprek of een e-mail. Meestal kan je dit herkennen aan de vraag om gevoelige informatie, zoals wachtwoorden of persoonlijke gegevens. Door een **challenge** te stellen, kan je de aanvaller vaak ontmaskeren. Vraag bijvoorbeeld om een bevestiging van de identiteit van de persoon die je aan de lijn hebt. Dit kan bijvoorbeeld door middel van een terugbelprocedure of door het controleren van hun e-mailadres.
- **Tailgating**: De aanvaller volgt een medewerker naar binnen in een beveiligd gebied. Dit kan bijvoorbeeld door zich voor te doen als een medewerker of door simpelweg achter iemand aan te lopen. Dit kan je voorkomen door altijd je toegangspas te gebruiken en nooit iemand anders binnen te laten zonder hun identiteit te verifiëren. Het voelt soms lastig aan om een wildvreemde te vragen om zich te identificeren, maar het is beter om voorzichtig te zijn dan om een beveiligingsrisico te creëren.
- **Vishing**: Voice phishing, waarbij de aanvaller zich voordoet als iemand anders via de telefoon. Dit kan bijvoorbeeld door zich voor te doen als een IT-medewerker of een bankmedewerker. Dit kan je herkennen aan verdachte vragen of verzoeken om persoonlijke informatie. Wees altijd voorzichtig met het delen van gevoelige informatie via de telefoon en verifieer altijd de identiteit van de persoon die je aan de lijn hebt. Doordat A.I. steeds beter wordt, is het ook mogelijk dat de aanvaller je stem kan imiteren. Dit maakt het nog moeilijker om te bepalen of je met een echte persoon of een nep persoon aan de lijn hebt. Meestal kan je de aanvaller ontmaskeren door ook nu weer een **challenge** te stellen. Je zou bijvoorbeeld een vraag uit het verleden kunnen stellen die enkel jij en de andere persoon kan weten. Als de persoon per ongeluk het juiste antwoord geeft dan kan je nog steeds doen alsof dit niet juist is om te zien of de aanvaller z'n correcte antwoord aanpast ("Uw bank is KBC", "Neem hoor (ook al is het dat wel)", "Oh excuses, ik bedoelde Argenta".) Deze vorm van challenge wordt ook wel **reverse social engineering" genoemd.
- **Baiting**: Deze techniek zal het slachtoffer lokken met iets dat hij echt wilt. Denk maar aan een USB-stick van het bedrijf, gratis een maand toegang tot Spotify Premium, etc. Van zodra je de *bait* aanvaardt, zal via deze weg meestal de aanval verder gezet worden (denk maar een virus dat op de USB-stick staat, of de link naar de gratis Spotify blijkt een link naar een malware server te zijn.) Het paard van Troje is een mooi voorbeeld van social engineering met behulp van *baiting*.
- **Quid Pro Quo**: 'Voor wat hoort wat'. De aanvaller zal een iets aanbieden (*bait*) maar dat enkel geven in ruil voor iets anders. Om bijvoorbeeld de bait te krijgen moet je eerst een vragenlijst invullen, maar in de vragenlijst staan ook vragen die je gevoelige informatie zullen ontfutselen ("Hoe heette je eerste huisdier"). Dit kan ook door bijvoorbeeld een gratis softwarepakket aan te bieden in ruil voor je e-mailadres of andere persoonlijke informatie.

Het is niet altijd eenvoudig om social engineering-aanvallen te herkennen, maar er zijn een aantal signalen waar je op kunt letten. Dit zijn enkele tips om verdachte e-mails of telefoontjes te herkennen: *

Controleer altijd het e-mailadres of telefoonnummer van de afzender. Dit kan je doen door het e-mailadres of telefoonnummer te vergelijken met eerdere communicatie of door het op te zoeken op de website van het bedrijf. * Let op verdachte links of bijlagen in e-mails. Dit kan je doen door de muisaanwijzer boven de link te houden zonder erop te klikken. Dit toont je waar de link naartoe leidt. Als de link er verdacht uitziet, klik er dan niet op. * Wees voorzichtig met het delen van persoonlijke informatie. Dit kan je doen door altijd te vragen waarom de informatie nodig is en hoe deze zal worden gebruikt. Als je twijfelt, deel de informatie dan niet. * Let op verdachte verzoeken of vragen. Dit kan je doen door altijd te vragen waarom de informatie nodig is en hoe deze zal worden gebruikt. Als je twijfelt, deel de informatie dan niet.

Bij twijfel kan je altijd een challenge stellen. En als je nog steeds twijfelt, doe dan een extra challenge, tot je overtuigd bent. Je zal verrassend vaak merken dat de aanvaller niet voorbereid is op deze extra vragen. En als blijkt dat de tegenpartij wel degelijk legitiem is, dan kan je altijd nog excuses aanbieden. Dit is een kleine prijs om te betalen voor de veiligheid van jezelf en je organisatie.

D. Meer weten

De smaak te pakken? Volgende online bronnen beveel ik aan om te volgen zodat je op de hoogte blijft van wat er allemaal gebeurt in de wereld van digitale stropers en boswachters.

D.1 Nieuws en blogartikels

- DarkReading.com
- Fox It: zeer in-depth en technisch.
- Graham Cluley blog
- Krebs on Security
- Troy Hunt: de blog van "Have I been pwnd"-oprichter Troy Hunt.

D.2 Podcasts

- Darknet Diaries: waargebeurde "horror-verhalen" over het darknet, met boeiende, in-depth interviews met slachtoffers én aanvallers. Ik kan deze podcast niet genoeg aanbevelen!

D.3 Tutorials

- Hacksplaining
- HackTricks book

E. Bronnen

- 802.11 Wireless Network (first edition), Gast, O'Reilly, ISBN: 0-596-0018-3
- Meerdere boeken van William Stallings (Computer Security, Cybersecurity, etc.)
- Cursus Software Security van M. Boeynaems.
- Afbeelding " Worm Propagation Model" komt uit: Fink, Glenn & Ball, Robert & North, Chris & Jawalkar, Nipun & Correa, Ricardo. (2004). Network Eye: End-to-End Computer Security Visualization.
- De uitleg en tekeningen i.v.m. rainbow tables is gebaseerd op de uitstekende uitleg op kestas.kuliukas.com/RainbowTables/).

F. Slides

Hieronder vind je een overzicht van de beschikbare presentaties bij het handboek.

F.1 Beschikbare presentaties

Hoofdstuk	Presentatie	PDF
🕒 H1: Wordt het erger?	Bekijk slides	Download PDF
🕒 H2: Cybersec Fundamenten	Bekijk slides	Download PDF
✅ H3: Cryptografie: deel 1	Bekijk slides	Download PDF
✅ H3: Cryptografie: deel 2	Bekijk slides	Download PDF
🕒 H4: Wifi Security	Bekijk slides	Download PDF
🕒 H5: Authenticatie	Bekijk slides	Download PDF
🕒 H6: IoT Security	Bekijk slides	Download PDF
🕒 Appendix A: GDPR	Bekijk slides	Download PDF
✅ Appendix B: Dark web	Bekijk slides	Download PDF
🕒 Appendix C: Awareness creëren	Bekijk slides	Download PDF

Legende:

- ❌: Slides nog niet beschikbaar
- ✅: Slides beschikbaar
- 🕒: Slides in ontwikkeling (mogen niet als leerstof gebruikt worden)

Bibliografie