CYBERSECURITY EN CYBERWARFARE

Diederik de Vries

jul. 09, 2023

Inhoudsopgave:

1

Over deze reader

Deze reader is bedoeld als naslagwerk voor de module “Cybersecurity en Cyberwarfare”, gegeven aan de Rotterdam Academy. Op onze opleiding hebben we 7 weken les. In deze cursus willen we weekopdrachten geven. Deze reader wil de ondersteuning bieden die studenten nodig kunnen hebben bij het maken van de opdrachten.

2

Bijdragen of fouten melden

Wil je iets bijdragen aan deze reader? Heb je een fout gevonden?

1) Maak een clone van deze repository
2) Maak een branch aan, bij voorkeur 1 waarbij je kan zien wát je aangepast hebt
3) Maak de verbetering aan
4) Push je eigen change naar je eigen repository
5) **Maak een *pull request* aan**.

Ik ga er vanuit dat ik het eens ben met je pull-request. Zodra ik de change verwerkt heb kan het tot maximaal twaalf uur ’snachts duren om jóuw wijziging doorgevoerd te zien worden. Als je je naam en mailadres doorstuurd in je pull-request, dan zet ik je bij de auteurs :)

3 Week 1 - Basis

Week 1 wil een *level playing field* geven voor alle studenten. Na het doornemen van week 1 weet je de grondbeginselen van information security:

- Gevoeligheid in data
- Wat is een **vulnerability**?
- Wat is een **exploit**?
- Wat is een hacker eigenlijk? En zijn er wel hackers?
- Niet benoemd in de Powerpoint-Slides, maar wel belangrijk: CIA: **Confidentiality**, **Integrity** en **Availability**

3.1 CIA - Confidentiality, Integrity and Availability

Confidentiality, Integrity en Availability, vertaald in het Nederlands Vertrouwelijkheid, Integriteit en Beschikbaarheid (de “BIV-driehoek”) is één van de grondbeginselen van informatiebeveiliging In de basis heb je te kijken naar alle drie van de begrippen. Je kan ook niet één van de onderdelen weglaten, want dan zou je te kort doen aan het geheel van informatiebeveiliging.

3.1.1 Vertrouwelijkheid

Vertrouwelijkheid is het begrip dat **alleen de mensen die bij de informatie kunnen die er bij moeten kunnen**. Als je bijvoorbeeld een bank neemt, dan is het fijn om te weten dat alleen geschikte/gescreende bankmedewerkers bij jouw bankinformatie kunnen.

3.1.2 Integriteit

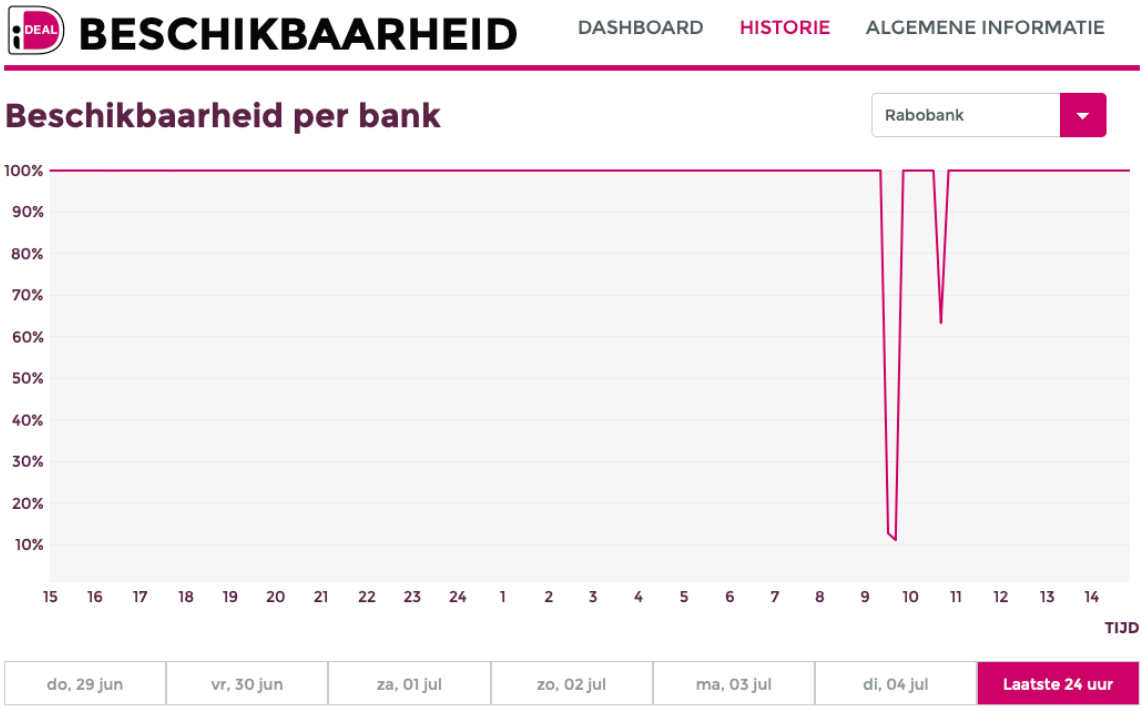
Integriteit is een iets lastiger te begrijpen onderdeel van informatiebeveiliging. Waarschijnlijk omdat *wij* alleen maar bezig zijn met hackers.

Integriteit geeft aan dat **de data die je hebt ook daadwerkelijk klopt**. Bij de bank zou het fijn zijn dat als je €110.000 op je rekening hebt staan, dat de app niet aangeeft dat je €10.- op je rekening hebt staan. Een misschien extremer voorbeeld is dat het bedrag op je rekening *aangepast* wordt naar iets wat niet klopt. Stel je voor dat de bank zelf óók de overtuiging heeft dat die €110.000 *verschwunden* (Duits: verdwenen, sorry, zal het nooit meer doen) is. Dan wordt de situatie toch wel heel erg vervelend.

3.1.3 Beschikbaarheid

Beschikbaarheid is een makkelijker begrip. Beschikbaarheid geeft aan dat wij **bij de informatie kunnen wanneer wij dat willen, met de snelheid dat wij willen**.

Wederom het bankvoorbeeld: leuk als ik weet dat ik €110.000 op de bank heb staan, maar als ik er niet bij kan vanwege een *dDOS* op de bank: wat heb ik er aan?



3.1.4 Non-repudiation

Een laatste kernbegrip wat in dit rijtje hoort is het begrip *Non-repudiation*. In Nederlands is dit te vertalen als *Onweerlegbaarheid*.

In het bankenvoorbeeld kan je het voorstellen dat jij een bedrag naar iemand overmaakt dat jij aan kan tonen dat je dat gedaan hebt. Daarmee wordt het onweerlegbaar. Het zou vervelend worden als jij een bedrag overmaakt naar iemand van Marktplaats, en nadat je dat hebt gedaan, hij beweerd dat je dat *niet* gedaan hebt. Jij kan onweerlegbaar bewijzen dat je het bedrag overgemaakt hebt.

4

Gevoeligheden

Een gevoeligheid in een applicatie ontstaat omdat een programmeur programmeerfouten maakt. Een programmeerfout is zo gemaakt. Interessant leesvoer in deze is het werk van Edsger Dijkstra [#]1[#]_ , een Nederlandse programmeur die grondlegger van de moderne informatica is geweest, en alles vanuit wiskunde bekeek. De vraag blijft: kan je wiskundig bewijzen of een applicatie foutloos is? Zover ik weet is dat nog niet mogelijk.

Dus, vanuit een programmeerfout wordt een *gevoeligheid* gegenereerd.

5 Vulnerability

Niet elke gevoeligheid zorgt voor de mogelijkheid tot misbruik. Als dit wel ontstaat, dan heb je het over een *vulnerability*. Een vulnerability kán bestaan in software zonder dat iemand daar vanaf weet. Zodra dit bekend wordt bij de maker van de software, kan deze gaan werken aan een *patch*. Een patch heeft als doel om de vulnerability weg te doen nemen. Veelal zal de versie van de software dan opgehoogd worden, en zullen de gebruikers dan de software moeten *patchen* of *updaten*.

Dit patchen alleen al zorgt voor risico's. Zodra er een patch uitkomt ("Installeer versie x.y.z NU"), dan wordt het voor de aanvaller een kwestie van het vinden van die vulnerability.

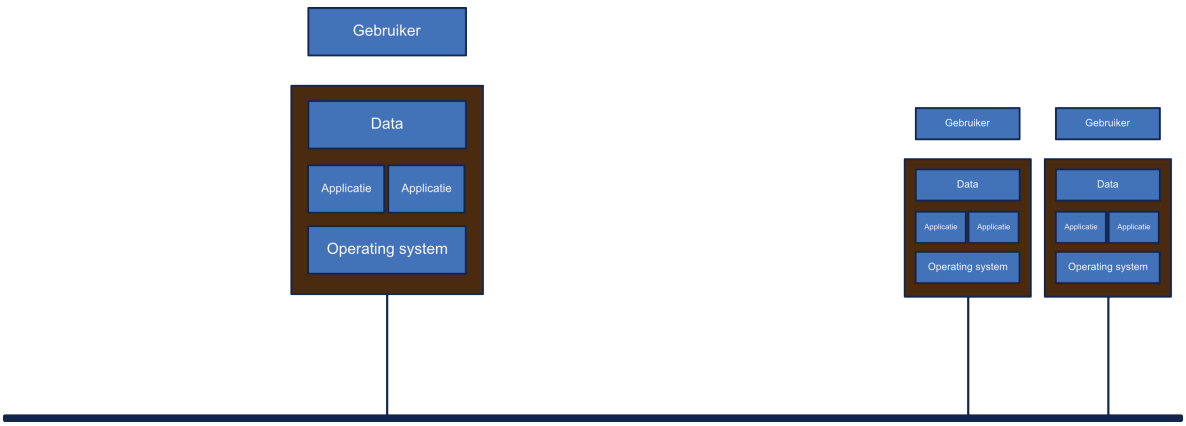
6 Exploit

De aanvaller kan dan gaan kijken hoe hij de vulnerability kan gaan *exploiten*, oftwel: misbruiken. Een **exploit is dus een stuk programma om misbruik te maken van een vulnerability**. De exploit heeft vaak als doel om verder toegang te krijgen tot het operating system van het systeem waar het oorspronkelijke programma op draait. De hacker krijgt dan de mogelijkheid om *lateraal te bewegen* door het netwerk, oftwel het verkrijgen van toegang op het netwerk met dezelfde rechten als degene die de software draait. Op het moment dat de hacker probeert om meer rechten te krijgen dan hebben we het over *privilege escalation*. In dit geval probeert de hacker via de gebruiker bijvoorbeeld de rol van **Administrator** of **root** te krijgen.

Bijzonder interessant leesvoer in deze is de **Cyber Kill Chain** van *Lockheed Martin*.

6.1 The Stack

We zullen veel van de gevoeligheden bekijken aan iets wat ik een *stack* ben gaan noemen. In deze is de stack een verzameling objecten wat allemaal kapot kan.



6.1.1 Netwerk

Elke computer is tegenwoordig verbonden met elkaar. Een applicatie heeft een *socket* nodig op het operating system om te communiceren tussen applicaties en applicaties die geïnstalleerd zijn op een andere computer. Als die socket niet goed geprogrammeerd is, dan is daar een aanvalsvector beschikbaar.

6.1.2 Hardware

Hardware zorgt voor het aan kunnen sluiten van een computer aan het netwerk, het invoeren en het uitvoeren van data. Alleen: ook hardware is niet onfeilbaar. Het gaat íets ver om op in te gaan, maar neem in ieder geval mee:

1) Als je bij de data kan die op hardware staat, dan ben je de eigenaar van die data. Als ik bijvoorbeeld in een ziekenhuis een laptop mee zou nemen, dan zou ik bij die data kunnen door bijvoorbeeld de harde schijf uit te bouwen. Een oplossing hier is *encryptie*. Hier gaan we later naar kijken.
2) Mocht je dit vak écht leuk vinden, dan kan je eens zoeken naar de begrippen *side-channel analyse*. Ook **Meltdown** en **Spectre** is leuk (maar pittig!) leesvoer.

6.1.3 Operating system

Je operating system verbind je netwerk, hardware, software in de vorm van applicaties en data aan de gebruiker. Hier komt eigenlijk alles samen.

Je hebt bij programming essentials, en bij Operating Systems uitgebreid geleerd over hoe een Operating System gebruikers scheidt, en hoe het OS er voor zorgt dat de applicaties de resources krijgen zoals systeembeheer ze voor ogen had. Je kan je dan ook voorstellen wat de mogelijke schade kan zijn als er een fout ontstaat in een OS! Op het moment dat een OS een zwakheid heeft, dan is het mogelijk dat de hacker van de ene applicatie overspringt naar de andere applicatie. Hij kan er voor zorgen dat hij *persistent* blijft op het operating system. Persistence is dat de hacker toegang blijft houden zolang de PC aan staat. Je ziet met de stack ook dat hij opeens de mogelijkheid krijgt om andere PC’s en servers in het netwerk aan te vallen.

6.1.4 Applicaties

Een applicatie die gehackt is zorgt voor een springplank. Er vanuit gaande dat de hacker nog niet in staat is gebleken om *privilege escalation* toe te passen, kan hij wél bij de data van de gebruiker. Ook kan hij mogelijk bij de servers met de zelfde rechten als dat de gebruiker heeft.

6.1.5 Data

Data is in deze redelijk statisch. Tenzij ik mijn applicatie slecht schrijf, en vía de data extra rechten verkrijg.

6.1.6 Gebruiker

De gebruiker is altijd de zwakste schakel. Het probleem is: we kunnen tamelijk niet zonder ze :)

7 Week 2 - OWASP, CVE, CWE en meer

In week 1 hebben we het gehad over het ontstaan van exploits.

7.1 Doelen van deze week

Vorige week hebben we gekeken naar diverse onderwerpen met als doel om een basis te leveren voor security. In deze week gaan we kijken naar een aantal afkortingen.

- CVE
- CWE

Ook gaan we kijken naar een aantal organisaties die bijdragen aan een veiliger wereld.

- OWASP
- SANS

Deze week

7.1.1 OWASP - The Open Web Application Security Project

De organisatie **The Open Web Application Security Project** (OWASP) <https://www.owasp.org>_ is een organisatie die als doel heeft om software meer secure te maken. Ze zijn bekend om een aantal zaken:

- Ze hebben een aantal fantastische tools gemaakt om security te testen of om te leren: * OWASP Zed Attack Proxy (ZAP) is een Proxy die je kan gebruiken om webapplicaties te testen * OWASP Juice Shop is een Applicatie die doelbewust zwak is gemaakt. Je kan hier uiteraard OWASP-ZAP voor gebruiken, maar je kan ook zelf los gaan op een veilige omgeving.
- Nee

Naast tools houden ze een aantal lijsten bij voor lering en vermaek. De meest bekende hierbij is de **OWASP TOP 10**. Ook noteworthy zijn de **cheatsheets** <"https://cheatsheetseries.owasp.org">_ op zekere onderwerpen. REST, NodeJS en meer zaken worden perfect behandeld!

Naast dit alles is OWASP ook een community. Voor niet heel veel kosten (€20/j) kan je gratis naar meetings toe.

7.1.2 OWASP Top 10

Het lijkt mij niet handig om alles wat op internet staat hier te repliceren. Ga dus vooral kijken op <"https://owasp.org/Top10/"> Ik heb vanuit de Top10 zaken die ik veel fout zie gaan op de werkplaats opgenomen.

A01:2021 - Broken Access Control (<https://owasp.org/Top10/A01_2021-Broken_Access_Control/>_)

- Geen scheiding van rechten
- ACL's voorbij komen door het aanpassen van de URL
- Toegang verkrijgen tot de API door het missen van ACL's op GET, POST, PUT en DELETE
- Elevation of privilege. Je kunnen gedragen als een gebruiker zonder ingelogged te zijn. Of taken als admin kunnen doen terwijl je een reguliere gebruiker bent.

A02:2021 - Cryptographic Failures (<https://owasp.org/Top10/A02_2021-Cryptographic_Failures/>_)

Je wil alle data (at rest en in transit) zoveel mogelijk versleutelen. Op te letten op:

- Wordt er data uberhaupt plaintext verstuurd?
- Heb je je crypto-engines up-to-date? Maak je gebruik van oude cyphers?
- Heb je default keys in gebruik (vaste sleutels die overal hergebruikt worden)
- Gebruik je oude hashingmethoden?

Op te lossen:

- Zorg er voor dat je weet waar je data hebt. Check dit met DFD en STRIDE-analyse.
- Móet je data opslaan?
- Versleutel data als het gevoelige data is
- Gebruik geen plain-text protocollen als FTP en SMTP

A01:2021 – Broken Access Control

A01:2021 – Broken Access Control

A01:2021 – Broken Access Control

A01:2021 – Broken Access Control

A01:2021 – Broken Access Control

A01:2021 – Broken Access Control

8 Licentie

De reader is geplaatst in het Open Source Domein. Ik ben nog aan het zoeken naar de juiste licentie voor deze reader.

Hoofdlijnen:

- Gebruik en aanpassen van deze reader is **GRATIS**. Ik wil je wel vragen om een link terug te geven naar de auteur van de reader
- Gebruik is zonder garantie van kwaliteit.