**Basic Security – PE Opdracht**

**Algemeen**

De opdracht is een **open opdracht**: er wordt opgegeven wat de minimale vereisten zijn van de opdracht, maar er worden ook uitbreidingen voorgesteld om een hogere score te behalen. De studenten mogen zelf ook uitbreidingen voorstellen en implementeren, in overeenstemming met de begeleidende docent.

De opdracht is een **team opdracht**: er wordt in een team gewerkt van 3-4 studenten. De docent zal regelmatig de samenwerking en vordering van het project bij de teams nagaan. Een groepsopdracht betekent ook een groepscijfer, tenzij de studenten tijdens deze feedback momenten zelf aangeven dat er een eventuele individuele aanpassing gewenst is.

Er zal tijdens de lessen steeds tijd voorzien worden om in team aan de opdracht verder te werken. De **verdediging** van de opdracht zal tijdens de laatste lessen van het OLOD plaatsvinden.

De opdracht zelf bestaat uit **3 aspecten**. Ieder team wordt verondersteld de minimale requirements van hun gekozen track te behalen + de basis requirements van ten minste 1 andere track. Aangezien het een open opdracht is, kunnen de studenten ook kiezen om meer dan de basis of minimale requirements van een track te doen.

Voor het **AppDev-aspect**: voor het maken van het Crypto-programma, mag er vertrokken worden van bestaande code op het internet, mits duidelijke referentie in de commentaar + studenten moeten de code wel goed begrijpen. Het onderling sharen van code tussen de teams is wel niet toegestaan.

De bedoeling van het AppDev-aspect is immers niet puur het programmeren, maar wel het toepassen en begrijpen van security aspecten.

Voor het **Systems&Networks aspect**: hier wordt het best gestart met een VM met daarop het Kali OS. De Nessus (OpenVAS) scan wordt dan uitgevoerd vanuit Kali naar een andere VM met het target OS op. Om een Nessus scan te doen, wordt gebruikt gemaakt van de gratis Nessus Home Feed plugin, als vulnerability database.

***LET OP****: het uitvoeren van een vulnerability scan mag enkel op uw eigen hosts/netwerk! Wees er dus zeker van dat je de scan niet doet over het PXL netwerk, of op een host waarvoor je de toestemming niet hebt = cybercrime!*

Voor het **IT Management aspect**: hier mag gestart worden van templates over InfoSec Policy’s die te vinden zijn op het internet, maar dit is enkel voor inspiratie. Het is niet de bedoeling om een juridische tekst te schrijven. Wel moet er duidelijk aangegeven worden welke policies jullie zouden toepassen (en vooral waarom) indien je medeverantwoordelijk zou zijn voor het security management van de PXL infrastructuur (netwerk, printers, hosts,…).

**Opdracht Basic Security – AppDev aspect:**

**Crypto Program**

**Basis:**

De bedoeling is om een programma te maken, in een programmeeromgeving naar keuze,  om een file veilig van A naar B te sturen (volgens de principes van hybride crypto), zodat:

* De ontvanger zeker is dat hij de enige is die de file terug kan decrypteren
* De ontvanger zeker is dat de file van A afkomstig is
* De ontvanger zeker is dat er niets veranderd is aan de file tijdens transport
* De zender niet meer kan ontkennen dat hij de file gestuurd heeft

*Note: In principe heb je voor het bovenstaande certificaten nodig om de linking te kunnen maken tussen een persoon A en zijn public key. Dit is voor de basisuitwerking geen vereiste. De public key van een persoon mag gewoon als string opgeslagen worden.*

De basisvereisten (evt puur command line based, GUI is uitbreiding) van het programma zijn stapsgewijs:

* Programma vraagt keuze te maken tussen encrypteren of decrypteren
* Bij encrypteren:
* Programma genereert 1 symmetrische key (DES of AES)
* Programma genereert 1 private en 1 public RSA key voor een gebruiker Alice
* Private key wordt gesaved in een file als Private\_A
* Public key wordt gesaved in een file als Public\_A
* Programma genereert 1 private en 1 public RSA key voor een gebruiker Bob
* Private key wordt gesaved in een file als Private\_B
* Public key wordt gesaved in een file als Public\_B
* Programma vraagt input aan Alice (bv. een boodschap om door te sturen)
* Programma gebruikt de symmetric key om de boodschap te encrypteren, en saved het resultaat in een file (File\_1)
* Programma encrypteert de symmetric key met de public key van Bob, en saved het resultaat in een file (File\_2)
* Programma maakt een hash van de oorspronkelijke boodschap
* Programma encrypteert die hash met de private key van Alice, en saved het resultaat in een file (File\_3)
* Bij decrypteren:
* Programma vraagt File\_1, File\_2, File\_3, Public\_A en Private\_B
* Programma gebruikt Private\_B om File\_2 te decrypteren, en zo de symmetrische key te verkrijgen
* Programma gebruikt die symmetrische key om File\_1 te decrypteren om terug de verstuurde boodschap te verkrijgen, en toont die boodschap dan ook op het scherm
* Programma berekent zelf de hash van die boodschap
* Programma gebruikt Public\_A om File\_3 te decrypteren, en zo terug de hash te verkrijgen van de originele file
* Programma controleert of de gedecrypteerde hash, en de zelf berekende hash overeenkomen, en geeft een boodschap indien ok of niet

**Minimaal voor AppDev studenten:**

Bovenstaande programma mag in een command-line vorm, als enkel de basis van die opdracht wordt gedaan. Voor de AppDev studenten wordt dit minimaal uitgebreid naar een GUI-vorm.

Vooraleer te starten met de GUI, worden er eerst **mockups** gemaakt voor het programma, zodat er voldoende aandacht kan besteed worden aan de verschillende schermen, plaatsen knoppen, e.d.

Ook de input en output van het programma moet gebruiksvriendelijker gemaakt worden bij de AppDev studenten: gebruik een **filepicker** om een invoerbestand te selecteren (dus niet meer gewoon een tekst ingeven, maar een willekeurige file om te encrypteren), filepicker voor key-management te doen, e.d. meer.

***Mogelijke uitbreidingen:***

1. Steganografie: (aangeraden uitbreiding)

In plaats van de encrypted message door te geven aan de ontvanger, kan je die encrypted message ook eerst gaan verbergen via steganografie in bv een foto. Dit is dus niet het onleesbaar maken van de message, maar het verbergen ervan. Via Google vind je enorm veel tutorials hiervoor. Maak gebruik van lossy images (geen informatie verlies bij compressie). Maak ook een check om te kijken of je message wel in de image past.

Uitbreiding hierop: je kan je message ook verbergen in sound of movie files. Dit is al wel een zware toepassing van steganografie.

1. Gebruik van Belgische eID certificaten

Je kan gebruik maken van certificaten, maar nog een mogelijke uitbreiding hierop is het certificaat te gebruiken via een eID (dezelfde als je voor je elektronische belastingbrief gebruikt). Je hebt hiervoor wel een eID lezer nodig.

Tip: [https://www.eidtoolkit.be/nl/home-eid.arcx](https://www.google.com/url?q=https://www.eidtoolkit.be/nl/home-eid.arcx&sa=D&ust=1489656646020000&usg=AFQjCNGv6Z4HSsCO0FqnwSmjHHpR3ImnqA)

1. Webbased maken met https connectie

In plaats van een standalone applicatie (minimale vereiste) maak je een webbased applicatie, of toch met een webbased front end. Indien je dit doet, moet je ook de browser connectie beveiligen via SSL. Je moet geen betalend certificaat hiervoor gebruiken, een self-signed is goed genoeg.

1. (Web based) vulnerability scan

De S&B opdracht (zie hieronder) is het maken van een vulnerability scan op een OS. Er bestaan ook tools om programma code te laten scannen op mogelijke vulnerabilities (XSS, SQL injection,…). Probeer eens zo een scanner voor uw programmeeromgeving, deel je resultaten mee, en je oplossingen voor mogelijke issues.

Er zijn ook nog tal van andere, extra, uitbreidingen mogelijk voor deze opdracht. Communiceer goed met de docent, indien je een andere uitbreiding overweegt. Het is immers een open opdracht.

Nog wat voorbeelden: gebruik van threading om de snelheid van het programma te optimaliseren, gebruik van meerdere programmeeromgevingen en die met elkaar laten communiceren (bv een front-end in .NET en een backend in Java), …

**Opdracht Basic Security – Systems&Networks aspect: Vulnerability Analysis**

**Basis:**

Vooraleer een server live beschikbaar is via het internet, moet deze gecontroleerd worden op mogelijke vulnerabilities. Indien er kwetsbaarheden gevonden zijn, moeten deze eerst gepatched worden, zodat er geen exploits meer mogelijk zijn.

De meest gebruikte vulnerability scanners zijn: Nessus en OpenVAS. OpenVAS is open source, en Nessus is betalend, tenzij je de Home Feed gebruikt voor persoonlijk gebruik / educational doeleinde.

De basisopdracht van dit aspect is:

* Doe Nessus scan (Home Feed) op een Windows OS (best om een iets oudere versie van Windows te nemen, zoals bv Windows XP). Zowel scannen met, als zonder firewall.
* Geef een omschrijving van de vulnerabilities die je vindt. Geef dus een verstaanbare omschrijving van het gevaar van die vulnerabilities, waar de service kwetsbaar is.
* Patch dan die vulnerabilities en doe de scan opnieuw. Toon aan dat alles nu safe is.
* Doe een Nessus scan op Metasploitable OS, een server waarin expres veel kwetsbaarheden zijn geïntroduceerd, zodat studenten hun skills kunnen testen. In principe zou je ook metasploit kunnen gebruiken, om zo’n gevonden vulnerability te gaan exploiten.
* Geef een omschrijving van minstens 3 vulnerabilities van Metasploitable, en hoe je die zou oplossen (patchen gaat niet goed lukken op Metasploitable OS, omdat dit echt een oefenserver is)

**Minimaal voor S&N studenten:**

* OpenVAS
* System&Networks studenten gebruiken niet alleen Nessus voor de opdracht van hierboven, maar ook OpenVAS. Studenten kunnen dan een vergelijking maken van de 2 systemen.
* Armitage
* Gebruik Armitage op Windows OS van de basis opdracht (evt zonder firewall/patches), en doe een exploit van de vulnerabilities.
* Welke exploits lukken allemaal?
* Welke vulnerabilities misbruik je?
* Patch de vulnerabilities en probeer opnieuw te exploiten. Lukt het nog steeds?
* Metasploitable
* Armitage gebruiken op Metasploitable OS.
* Geef een omschrijving van de gevaren van ten minste 5 vulnerabilities (dus 2 meer dan de basis opdracht)

S&N studenten gebruiken best Kali OS om hun Nessus / OpenVAS / Armitage op te draaien.

*Mogelijke extra uitbreidingen:*

* Opzoeken van specifieke CVE’s om te gebruiken in uw vulnerability scans.
* Extra OS’en scannen en analyse doen van vulnerabilities
* Bv Windows Servers, Linux Servers, Mac OS X, mobile (Android?),…
* De resultaten van de Nessus scan opslagen, en gebruiken in het metasploit-framework (ingebouwd in Kali). Daarna wordt een manueel exploit gedaan (dus niet met Armitage)
* …

**Opdracht Basic Security – IT Management aspect:**

**InfoSec Policy**

**Basis:**

Security is een big issue in het bedrijfsleven vandaag. Daarom nemen de meeste bedrijven ‘information security policies’ op in hun contracten met de werknemers. Deze policies leggen in detail uit hoe de werknemers moeten omgaan met security in allerlei soorten situaties, of welke regels ze hiervoor moeten volgen. Denk hierbij bv aan een ‘Acceptable Use Policy’ (wat bv mag of niet mag op het bedrijfsnetwerk – torrents?), of een ‘Password Policy’ (hoe op te stellen, hoe dikwijls vernieuwen,…).

Omdat deze policies een onderdeel vormen van het contract met de werknemers, zijn ze meestal ook opgesteld op een juridische manier, conform de wetten van het thuisland.

Voor deze opdracht is het niet de bedoeling om zo’n wettelijk document op te stellen (dat is een taak voor juristen), maar wel een document met de krijtlijnen van zo’n policy, samen met **de argumentatie** waarom juist deze richtlijnen in de policy zouden moeten komen.

Als voorbeeldbedrijf, waarvoor jullie een InfoSec Policy gaan uitwerken, nemen jullie de Hogeschool PXL, met zijn verschillende soorten van ‘werknemers’: docenten, studenten, IT dienst, management,…

Er bestaan voldoende policy templates op het internet, die jullie kunnen gebruiken ter inspiratie. Denk er aan: het is niet de bedoeling om een wettelijke tekst te schrijven, maar eerder: welke richtlijnen zouden jullie in een InfoSec Policy plaatsen, en waarom? Probeer dit zo objectief mogelijk te benaderen, en de policy niet op te stellen vanuit de specifieke noden van een IT student, maar eerder om de security van de Hogeschool als geheel te verbeteren.

De basis topics die in de InfoSec Policy moeten behandeld worden, zijn:

* Acceptable Infrastructure Use Policy
* Netwerk: Wired – Wireless: wat mag (niet), waarom?
* Printers
* PingPing
* Password Policy

Denk er aan: duidelijk verwoorden waarom je die richtlijn zou gebruiken, voor welke type gebruiker, onder welke omstandigheden, e.d. !

**Minimaal voor IT Management studenten:**

IT Management studenten worden verwacht om meerder topcis te behandelen voor de InfoSec Policy.

De minimale topics, bovenop de basis topics:

* Email Policy
* Teleworker Policy (Remote Access)
* BYOD Policy
* BYOA Policy (Bring Your Own App)

Deze minimale topics kunnen dan nog verder uitgebreid worden met andere topics, waarvan jullie vinden dat die absoluut in een InfoSec Policy voor de Hogeschool PXL thuishoren.

Denk echt aan de verschillende soorten gebruikers: een student moet niet dezelfde stappen ondernemen om zijn data veilig te houden als bv een docent, of de algemeen directeur.

Denk ook goed aan de relatie tussen usability en security. Hoe hoger de security, hoe lager de usability, en visa versa. Je kan vragen om dagelijks een nieuw paswoord te gebruiken van 100 chars lang (heel secure), maar daardoor gaat het gebruiksgemak wel zeer laag zijn. Dus goed argumenteren waarom je een bepaalde richtlijn zou uitschrijven, om een aanvaardbaar compromis te hebben tussen gebruiksgemak en security.