RADBOUD UNIVERSITEIT NIJMEGEN

BACHELORSCRIPTIE INFORMATICA

Veiliger authenticeren met de Yubikey

Auteur:
Kevin Reintjes (0814954)
K.Reintjes@student.ru.nl

Begeleider:
Prof. dr. B.P.F. Jacobs
Bart@cs.ru.nl

26 juni 2011

Een veel gebruikt authenticatiemiddel bij (web)applicaties is het wachtwoord. Dit kent echter allerlei beveiligings- en praktische bezwaren (bijvoorbeeld gevoelig voor replay-aanvallen en vaak lastig te onthouden). Er zijn dan ook veel alternatieven en aanvullingen bedacht voor het wachtwoord. De Yubikey is daar één van. Het is een klein apparaat om te authenticeren met behulp van One Time Passwords (OTP's). In dit onderzoek wordt de werking van de Yubikey bekeken en een beveiligingsanalyse gedaan om te bepalen of de Yubikey beveiligingsgebreken kent (en zo ja, welke). Zo zal duidelijk worden of de Yubikey een goed alternatief voor (of aanvulling op) het wachtwoord is of niet.

Inhoudsopgave

Hoofdstuk 1

Inleiding

1.1 Probleemstelling

Authenticeren bij een bepaalde computerapplicatie/website gebeurt vaak door middel van een wachtwoord. Deze methode kent echter allerlei beveiligingsrisico's en praktische problemen. Er wordt bijvoorbeeld aangeraden om sterke wachtwoorden te nemen (minimaal acht tekens met letters, hoofdletters en cijfers door elkaar), het liefst voor elke applicatie een ander. Aangezien een gemiddelde computergebruiker toch al vaak te maken krijgt met tientallen verschillende applicaties en/of websites, is het vrijwel onmogelijk al deze wachtwoorden te onthouden. Vaak nemen mensen simpelere wachtwoorden of gebruiken ze hetzelfde wachtwoord voor meerdere applicaties, zonder stil te staan bij alle gevaren die dan optreden. Zelfs unieke en sterke wachtwoorden zijn niet geheel veilig. Ze zijn namelijk gevoelig voor allerlei aanvallen, zoals een replay aanval. Onderschep het wachtwoord eenmalig en gebruik het daarna wanneer je maar wilt om (onrechtmatig) te authenticeren.

Om deze redenen is de Yubikey bedacht. De Yubikey is een authenticatiemiddel dat authenticeren makkelijker en veiliger zou moeten maken. Het is een klein USB product in de vorm van een sleutel (zie figuur 1.1). Authenticeren is simpel, plaats het apparaat in een USB poort en druk op de knop. Meer hoef je (meestal) niet te doen.

Figuur 1.1: Twee Yubikeys

Authenticatiemiddelen zijn vaak in drie hoofdcategorieën te onderscheiden: 'something you have', 'something you know' en 'something you are'. De Yubikey valt in de eerste categorie. Om te authenticeren, moet je iets hebben, namelijk de (juiste) Yubikey. Om aan de applicatie te bewijzen dat je hem echt hebt, maakt het apparaat gebruik van een zogenaamd One Time Password (OTP). Eventueel valt de Yubikey ook te combineren met een standaard wachtwoord. Om te authenticeren heb je dan zowel het wachtwoord (something you know) en de Yubikey (something you have) nodig. Dit (ook wel two-factor authenticatie genoemd) maakt het nog veiliger, maar wel iets omslachtiger. Op het eerste gezicht lijkt de Yubikey een veiligere authenticatiemethode dan het standaard wachtwoord, zeker als het gecombineerd wordt met een wachtwoord. De vraag is echter hoe veilig precies. Een Yubikey kost namelijk geld in de aanschaf en het kost tijd en moeite om een applicatie/website er geschikt voor te maken. Voordat men overstapt naar de Yubikey wil men dus wel echt zeker weten dat deze veilig is. Misschien kent de Yubikey enkele gebreken waardoor deze niet zo veilig is als op het eerste gezicht lijkt. De Yubikey zal daarom aan een kritische analyse onderworpen worden om te bepalen hoe veilig het apparaat nu eigenlijk is.

In dit onderzoek richting we ons hoofdzakelijk op de beveiliging en niet tot nauwelijks op het gebruiksgemak. Dit omdat het gebruiksgemak van de Yubikey al aanbod komt in veel andere artikelen van Yubico zelf of andere partijen (zie bijvoorbeeld [14]), terwijl de beveiliging niet tot nauwelijks besproken wordt (zie ook de verantwoording, sectie 1.2). Bovendien is een beoordeling van het gebruiksgemak ook erg persoons- en situatieafhankelijk. Dit onderzoek kan dus gebruikt worden om een idee te krijgen van hoe veilig de Yubikey is, terwijl andere artikelen (of eigen inschattingen) gebruikt kunnen worden voor een beeld van het gebruiksgemak.

Onderzoeksvraag

Heeft de Yubikey beveiligingsgebreken, zo ja welke en hoe risicovol zijn deze?

Product/Antwoord

Het product zal een analyse van de beveiliging van de Yubikey zijn. Hierbij zal de werking, de onderliggende protocollen en structuur van gebruikte berichten geanalyseerd worden. Uit deze analyse zal blijken hoe veilig de Yubikey is, of het beveiligingsgebreken kent en indien ja, hoe serieus deze zijn. Het antwoord zal een overzicht zijn van de mogelijke beveiligingsgebreken en een inschatting van het risico. Als er geen enkel mogelijk gebrek gevonden wordt, dan zal het antwoord simpelweg nee zijn met een verklaring waarom de beveiliging zo goed op orde is.

1.2 Verantwoording

Zoals al aangegeven is, kent het veel gebruikte standaard wachtwoord (voor authenticatie) enkele beveiligingsproblemen. Zo is het gevoelig voor replay-aanvallen. Verder kent het ook praktische problemen omdat het vrijwel niet meer mogelijk is om voor alle applicaties een uniek en sterk wachtwoord te bedenken en te onthouden. Dit probleem wordt waarschijnlijk alleen maar groter, omdat er meer en meer (web)applicaties bijkomen waarvoor authenticeren vereist is. Het standaard wachtwoord biedt dus eigenlijk geen goede beveiliging en is (als je het juist doet) niet erg praktisch. Het is daarom belangrijk dat er een alternatief of aanvulling op het standaard wachtwoord komt. Het is echter nog belangrijker dat dit alternatief of deze aanvulling wel voldoende beveiliging biedt.

De Yubikey is een voorbeeld van een alternatief of aanvulling op het standaard wachtwoord (het is als beiden te gebruiken). Het is ook een populair voorbeeld. Het product is namelijk al bijna een miljoen keer verkocht aan meer dan 11.000 klanten in 85 verschillende landen. Het is dus belangrijk dat de Yubikey veilig is en geen serieuze gebreken kent. Immers als dit het geval zou zijn, dan zouden heel veel mensen momenteel een beveiligingsprobleem hebben en in de toekomst wellicht nog meer. Bovendien kost de Yubikey geld en kost het ook tijd en moeite om de Yubikey in een applicatie te integreren. Het is dus ook belangrijk om te weten dat het apparaat inderdaad een goede beveiliging biedt voordat je overgaat op aanschaf of integratie.

In de wetenschappelijke literatuur is echter (vrijwel) niets over de Yubikey te vinden. Er lijkt op dit moment geen grondige beveilig

Dit onderzoek kan voor veel mensen interessant zijn, met name mensen die al in het bezit zijn van een Yubikey of deze ooit willen aanschaffen. Mocht blijken dat de Yubikey een serieus beveilig

1.3 Methode

Voor dit onderzoek zal een beveiligingsanalyse van de Yubikey gedaan worden. De Yubikey kent erg veel verschillende mogelijkheden en configuraties (hierover later meer). De analyse zal zich echter naar alleen op de standaard mogelijkheid richten (de configuratie van de Yubikey zoals deze uit de fabriek geleverd wordt).

De beveiligingsanalyse gebeurt in eerste plaats door op globaal niveau te bepalen aan welke beveiligingseisen en -aspecten de Yubikey moet voldoen. Hierbij zullen we ook de situatie bekijken waarbij de Yubikey gebruikt wordt in een (denkbeeldig) groter systeem. Daarna zullen we de Yubikey zelf analyseren. Dit gebeurd in eerste instantie door achtergrond informatie over de Yubikey en gebruikte technieken door te nemen (met name door informatie van Yubico zelf te bestuderen). Daarna zal gekeken worden in welke onderdelen de Yubikey opgesplitst kan worden (voorbeelden: het genereren van de OTP en de verbinding met de validatieserver). Hierbij zullen we (per onderdeel) het protocol van de Yubikey in kaart brengen (de werking ervan) inclusief de structuur van de gebruikte berichten. Dan zullen we de beveiligingsanalyse uitvoeren. Dit doen we in eerste plaats door te kijken welke beveiligingsaspecten van toepassingen zijn per onderdeel. Vervolgens zullen we opzoek gaan naar specifieke beveiligingsgebreken voor elk onderdeel. Dit gebeurt door de beveiliging theoretisch te analyseren (waar zitten de zwakheden) en door aanvallen in de praktijk uit te proberen (bijvoorbeeld door het manipuleren van berichten). Als laatste zullen we een overzicht van de gevonden gebreken geven, samen met een inschatting van het risico van elk gebrek.

Bij het onderzoek zal gebruik worden gemaakt van documenten van Yubico zelf, verschillende gerelateerde open-source implementaties beschikbaar gesteld door Yubico, praktijktests en artikelen over achterliggende protocollen en technieken (zoals AES). Voor de praktijktests zal gebruik gemaakt worden van een Yubikey versie 2.2 in combinatie met diverse software waaronder de Yubikey Configuration Utility en een (aangepaste versie van) de Yubico PHP Web API & Client [1]. Hiermee zullen we de werking van de Yubikey achterhalen en vervolgens een nauwkeurige analyse uitvoeren, waarbij we op zoek gaan naar beveiligingsgebreken.

1.4 Document

Het document is een verslag van het onderzoek, waarin de werking van de Yubikey besproken wordt en de resultaten van de beveiligingsanalyse. Het begint eerst globaal met een inleiding op het onderzoek en de Yubikey (wat het is en hoe het werkt). Hierbij zal de Yubikey worden opgesplitst in een aantal onderdelen. Daarna zullen algemene beveiligingsaspecten aanbod komen welke later gebruikt worden voor de beveiligingsanalyse. Vervolgens wordt elk onderdeel in detail besproken (wat het doet, hoe het werkt en zal op elk onderdeel een beveiligingsanalyse gedaan worden). Als laatste worden alle gevonden gebreken samengevat en wordt er een conclusie gegeven. De opbouw van het document volgt min of meer de methode van het onderzoek, waarbij eerst op globaal niveau gekeken wordt, daarna voor elk onderdeel specifiek en op het einde volgt een overzicht en conclusie.

Hoofdstuk 2

De Yubikey

De Yubikey is het apparaat waar dit onderzoek om draait. Voordat we de beveiliging van de Yubikey kunnen analyseren is het belangrijk om eerst een indruk te krijgen van wat de Yubikey nu eigenlijk is. We zullen hier daarom een beschrijving van de Yubikey geven, waarbij de volgende aspecten aan bod komen: wat is de Yubikey (waarvoor dient het, door wie wordt het gebruikt) en hoe werkt het (globaal).

2.1 Wat is de Yubikey?

De Yubikey is een apparaat om mee te authenticeren. In dit geval gaat het om bij een computer (web)applicatie te bewijzen dat je echt bent, wie je beweert te zijn. De Yubikey kan gebruikt worden als vervanging op het standaard wachtwoord (of ander authenticatiemiddel) of als uitbreiding hierop. Oorspronkelijk is de Yubikey met name bedoelt voor online authenticatie (bij websites of webapplicaties). Een belangrijk aspect hierbij is dat dezelfde Yubikey gebruikt kan worden voor authenticatie bij meerdere (onafhankelijke) applicaties. Hiervoor gebruikt de Yubikey een centraal validatiesysteem opgezet en onderhouden door Yubico (dit systeem controleert de authenticatiepogingen met de Yubikey, maar hierover later meer). De volgende twee voorbeelden zijn typische gevallen van hoe de Yubikey gebruikt kan worden:

- De applicatie is in dit geval webmail. Om in te loggen gebruikte men voorheen een gebruikersnaam en wachtwoord (authenticatiemiddel gebaseerd op 'something you know'). De applicatie besluit een ander authenticatiemiddel te gebruiken, namelijk de Yubikey. Vanaf nu vult men enkel zijn gebruikersnaam in, doet de Yubikey in een USB poort en drukt op de knop.
- De applicatie is in dit geval internetbankieren. Om in te loggen gebruikte men voorheen enkel een gebruikersnaam en wachtwoord. De bank vindt dit niet meer veilig genoeg en voegt de Yubikey toe. Vanaf nu vult men zijn gebruikersnaam en wachtwoord in, doet de Yubikey in de USB poort en drukt op de knop. Dit noemt men two-factor authenticatie, aangezien er twee factoren nodig zijn om te authenticeren (het wachtwoord, gebaseerd op 'something you know' en de Yubikey, gebaseerd op 'something you have').

Ondanks dat de Yubikey oorspronkelijk bedoelt is voor online authenticatie bij meerdere onafhankelijk applicaties, kan de Yubikey ook gebruikt worden voor lokale authenticatie (denk bijvoorbeeld aan toegang verkrijgen tot een bepaalde PC op het werk). Dit kan in combinatie met het centrale Yubico validatiesysteem of met een eigen systeem (alle software hiervoor is vrij beschikbaar). In dit onderzoek richten we ons met name op de situatie van online authenticatie met het centrale validatiesysteem (met af en toe kleine verwijzingen naar hoe het zou zijn in een andere situatie). Dit omdat de Yubikey hier oorspronkelijk voor bedoelt is en dit bovendien (hoogstwaarschijnlijk) de meest onveilige situatie is. Hierbij moet namelijk de meeste communicatie plaatsvinden en dat (vrijwel) allemaal over het publieke internet. Als de Yubikey dus veilig te gebruiken is in deze situatie, dan geldt dat hoogstwaarschijnlijk ook voor andere situaties (uitzonderingen daargelaten, bijvoorbeeld als het eigen validatiesysteem implementatiespecifieke beveiligingsgebreken kent).

Om te authenticeren moet de applicatie op de een of andere manier kunnen controleren dat de gebruiker

in het bezit is van de juiste Yubikey. De Yubikey kent hier eigenlijk diverse mogelijkheden (modussen) toe. Elke modus kent weer diverse instellingen. Deze modussen en instellingen kunnen op de Yubikey geprogrammeerd worden met behulp van de Yubikey Configuration Utility. De beschikbare modussen zijn:

- Yubico OTP modus
- OATH-HOTP modus
- Statisch wachtwoord modus
- Challenge-response modus (aangepaste versie van Yubico OTP modus of OATH-HOTP modus)

De Yubikey is oorspronkelijk bedoelt voor Yubico OTP modus. De Yubikey wordt standaard ook voorgeprogrammeerd geleverd in deze modus. Bij dit onderzoek zullen we ons ook enkel beperken tot deze modus. Wanneer de Yubikey is ingesteld in deze modus wordt gebruik gemaakt van het Yubico OTP-protocol voor het genereren van een One Time Password (eenmalig wachtwoord). Hiermee bewijst de Yubikey aan de applicatie dat de gebruiker in het bezit is van de juiste Yubikey. Hoe dit precies in zijn werk gaat, zullen we later bespreken.

Er zijn ook nog een aantal verschillende varianten van de Yubikey, zoals de Symantec VIP of de RFID variant. We zullen deze verschillende varianten niet bespreken en ons enkel richten op de standaard Yubikey. Om precies te zijn beperken we ons bij dit onderzoek tot de standaard Yubikey firmware versie 2.2 (in Yubico OTP modus, zoals aangeleverd door de fabrikant).

2.2 Hoe werkt het?

De Yubikey maakt gebruik van een publieke identiteit en een One Time Password (OTP) om bij een applicatie te authenticeren. De publieke identiteit is uniek voor elke Yubikey en wordt gebruikt om de toegang tot (een bepaalde account van) de applicatie aan een bepaalde Yubikey te koppelen. Het idee is nu dat alleen met de juiste Yubikey een geldige OTP gegenereerd behorende tot die publieke identiteit (en dus die Yubikey). Als een gebruiker de juiste publieke identiteit en een bijbehorende geldige OTP kan tonen, dan is hij in het bezit van de juiste Yubikey (en dus is de gebruiker wie hij beweerd te zijn). Om te authenticeren moet de Yubikey dus de publieke identiteit gevolgd door een OTP uitvoeren. De Yubikey doet dit zodra je op de knop drukt. Enkele voorbeelden van uitvoeren van de Yubikey zijn:

fifjgjgkhchb**irdrfdnlnghhfgrtnnlgedjlftrbdeut**
fifjgjgkhchb**gefdkbbditfjrlniggevfhenublfnrev**
fifjgjgkhchb**lechfkfhiiuunbtnvgihdfiktncvlhck**

Het eerste deel is de publieke identiteit welke altijd gelijk blijft (voor dezelfde Yubikey). Het tweede deel is de gegenereerde OTP welke dynamisch is (en in principe maar één keer voor kan komen bij een bepaalde Yubikey). In hoofdstuk 4 komt aan bod hoe de OTP precies gegenereerd wordt.

Om een OTP te genereren gebruikt de Yubikey encryptie in combinatie met een privé-identiteit en een sleutel (en nog een aantal andere velden). Een OTP is dus eigenlijk een versleutelde boodschap (ciphertext) over een aantal velden volgens een bepaald protocol. Het is belangrijk dat de privé-identiteit en sleutel geheim blijven, zodoende kan alleen met de juiste Yubikey de juiste OTP gegenereerd worden. Het Yubico OTP-protocol kent drie specifieke instellingen voor het produceren van publieke identiteit en OTP, namelijk:

- Publieke identiteit: Een unieke, willekeurige, één tot zestien bytes grote publieke identiteit (standaard zes bytes groot).
- privé-identiteit: Een willekeurige zes bytes grote privé-identiteit.
- Sleutel: Een 128 bit (= 16 bytes) grote sleutel voor de encryptie.

Deze gegevens staan opgeslagen op de Yubikey. Dit is echter niet de enige plek. De applicatie moet namelijk kunnen controleren of een gegenereerde OTP geldig is (en bij de desbetreffende publieke identiteit hoort). Om dit te doen gebruikt de applicatie een validatieserver. Het is mogelijk een eigen validatieserver te gebruiken (alleen werkt de Yubikey dan alleen met jouw applicaties), maar gebruikelijker is het om de centrale validatieserver(s) van Yubico te gebruiken.

De validatieserver is verantwoordelijk voor het valideren van OTP's (en zegt tegen de applicatie of een OTP geldig is of niet). Om dit te kunnen doen moet de validatieserver in het bezit zijn van de publieke identiteit met bijbehorende privé-identiteit en sleutel. Op die manier kan de validatieserver de volledige OTP ontsleutelen (en de plaintext achterhalen) en vervolgens controleren of een bepaalde publieke identiteit en (ontsleutelde) OTP bij elkaar passen. Dit gebeurt door te controleren of de privé-identiteit in de OTP overeenkomt met de opgeslagen privé-identiteit bij de desbetreffende publieke identiteit (hierover later meer in hoofdstuk 7). Als dit het geval is, dan is de gebruiker echt in het bezit van de Yubikey waarvan hij beweerd in het bezit te zijn.

2.2.1 Voorbereiden applicatie

Om de Yubikey te gebruiken plaats je het apparaat in een USB poort van de computer en druk je op de knop. De Yubikey genereert dan een OTP en stuurt deze naar de computer samen met de publieke identiteit (dit gebeurd door het emuleren van een toetsenbord, hierover meer in hoofdstuk 5). Een Yubikey is echter niet zomaar met elke willekeurige applicatie te gebruiken. Hiervoor moeten er een aantal zaken gebeuren.

Allereerst moet de applicatie ondersteuning bieden voor de Yubikey (in Yubico OTP modus). Als dit het geval is, dan dient (een account op) de applicatie gekoppeld te worden met de Yubikey van de gebruiker. De applicatie heeft hier meestal een functie (knop, pagina, link) voor. Dit is applicatie specifiek en kan in principe voor elke applicatie verschillen. Hier moet een gegenereerde OTP samen met de publieke identiteit worden opgegeven (door op de knop van de Yubikey te drukken). De applicatie controleert of de OTP geldig is en bij de publieke identiteit hoort. Als dit het geval is, dan bewaart de applicatie de publieke identiteit en onthoud dat alleen met de Yubikey met die publieke identiteit toegang tot (een bepaalde account op) de applicatie verkregen mag worden. Daarna is de applicatie klaar voor authenticatie met de Yubikey.

2.2.2 Authenticeren

Nadat de applicatie is voorbereid, is authenticatie met de Yubikey mogelijk. We leggen uit hoe dit gaat door middel van een fictief voorbeeld. In deze situatie wil een student van de Radboud Universiteit zijn cijfers raadplegen in het studenten portal (dit is een webapplicatie). Hij gaat naar het webadres van de portal. De portal vraagt vervolgens om zijn gegevens. We gaan er in deze situatie vanuit dat de portal het studentnummer wil weten (de identiteit van de gebruiker) en een OTP gegenereerd met de Yubikey (om te bewijzen dat hij ook echt deze gebruiker is). De applicatie is al voorbereid, oftewel de portal weet welke publieke identiteit van de Yubikey hoort bij het studentnummer. De validatieserver die gebruikt wordt is de standaard Yubico validatieserver en deze is op de hoogte van de publieke identiteit, privé-identiteit en sleutel van de Yubikey.

De gebruiker kan zich dus gaan authenticeren bij de portal. Het proces dat dan plaatsvindt is te verdelen in een aantal stappen.

De eerste stap is de het genereren van de OTP. In deze stap plaatst de gebruiker zijn Yubikey in een USB-poort. Hij drukt vervolgens op de knop en de Yubikey genereert een OTP en voert deze uit door de USB interface. Behalve de gegenereerde OTP wordt ook de (unieke) publieke identiteit uitgevoerd, aangezien de applicatie en validatieserver deze nodig hebben.

In de tweede stap wordt de OTP gecommuniceerd met de (web)applicatie. De Yubikey doet dit door het emuleren van een toetsenbord en zodoende wordt de OTP dus automatisch ingevoerd in een tekstveld van de applicatie waarop de focus ligt. Daarna wordt de OTP (in dit specifieke geval) gecommuniceerd met de webserver van de portal over een (al dan wel of niet beveiligde) HTTP verbinding. Zodra het studentnummer en de publieke identiteit binnen zijn controleert de applicatie of deze bij elkaar horen (volgens de instellingen opgeslagen bij het voorbereiden van de applicatie). Als dit niet het geval is, dan

wordt de authenticatiepoging gelijk afgekeurd. Als dit wel het geval is, dan gaat het proces door met stap drie.

In de derde stap wil de applicatie de ontvangen OTP gaan valideren. Dit kan de applicatie niet zelf, omdat deze niet de beschikking heeft tot de geheime sleutel en andere instellingen waarmee de OTP gegenereerd is (het zou onveilig zijn als elke willekeurige applicatie hier zomaar toegang tot heeft). Vandaar dat de OTP wordt gevalideerd met behulp van een validatieserver. De applicatie stuurt hiertoe een verzoek naar de validatieserver (waarop de validatieserver later zal antwoorden).

In de vierde stap valideert de validatieserver de binnengekregen OTP. Dit houdt in dat de validatieserver controleert of de OTP past bij de meegestuurde publieke identiteit. Hiervoor gebruikt de validatieserver de van tevoren opgeslagen instellingen bij de publieke identiteit, welke ook gebruikt worden voor het genereren van de OTP (privé-identiteit en sleutel). Als de OTP past bij de publieke identiteit dan wordt dit over dezelfde verbinding als bij stap drie teruggestuurd. Na deze stap weet de webportal (of andere applicatie) of de authenticatiepoging rechtmatig was en is de authenticatiepoging afgerond. De webportal laat nu een cijferoverzicht (of andere geheime pagina) zien zoals gebruikelijk. Wat er verder gebeurt valt in ieder geval niet meer onder de verantwoordelijkheid van de Yubikey en dus buiten dit onderzoek.

Dit zijn de vier stappen welke worden doorlopen bij het Yubico OTP-protocol. We zullen de opsplitsing in deze vier stappen gebruiken als leidraad bij het maken van de opsplitsing van de Yubikey in de onderdelen (om te analyseren).

2.2.3 Onderdelen

De vier bovengenoemde stappen beschrijven het proces wat plaatsvindt als een gebruiker wenst te authenticeren. De verschillende stappen hebben allen te maken met de Yubikey in combinatie met het Yubico OTP-protocol. De vier stappen zouden ook gezien kunnen worden als delen waarin de (werking van de) Yubikey/het Yubico OTP authenticatieprotocol gesplitst kan worden. We hebben deze opsplitsing gemaakt om het onderzoek overzichtelijker te maken. We zullen de beveiliging van de Yubikey en het protocol dan ook bespreken door de beveiliging van elk van deze onderdelen apart te analyseren (een soort verdeel-en-heers techniek).

We hebben de Yubikey in de volgende vier onderdelen opgesplitst:

1. Genereren OTP
2. Communiceren OTP met (web)applicatie
3. Communicatie tussen (web)applicatie en validatieserver
4. Valideren OTP

Een aantal van deze onderdelen bestaan zelf weer uit subonderdelen of maken zelf weer gebruik van (deel)protocollen. Een voorbeeld hiervan is het valideren van de OTP, waarbij gebruik wordt gemaakt van het subonderdeel de Yubikey KSM (maar hierover later meer).

We hebben er voor gekozen om van de applicatie zelf niet een apart onderdeel te maken. Dit komt doordat de applicatie niet veel bijzonders doet, behalve het doorsturen van de OTP naar de validatieserver (via onderdeel drie) en het interpreteren van het resultaat. Verder controleert de applicatie of de publieke identiteit van de Yubikey hoort bij de vooraf opgeslagen instellingen (tijdens het voorbereiden van de applicatie), maar dit is ook niet bijzonder spannend. Bovendien kan elke applicatie dit op zijn eigen manier doen en valt deze stap in principe buiten het Yubico OTP-protocol (binnen het protocol staat alleen gespecificeerd dat het moet gebeuren, maar niet hoe). We gaan er bij het onderzoek dus vanuit dat dit (correct) plaatsvindt.

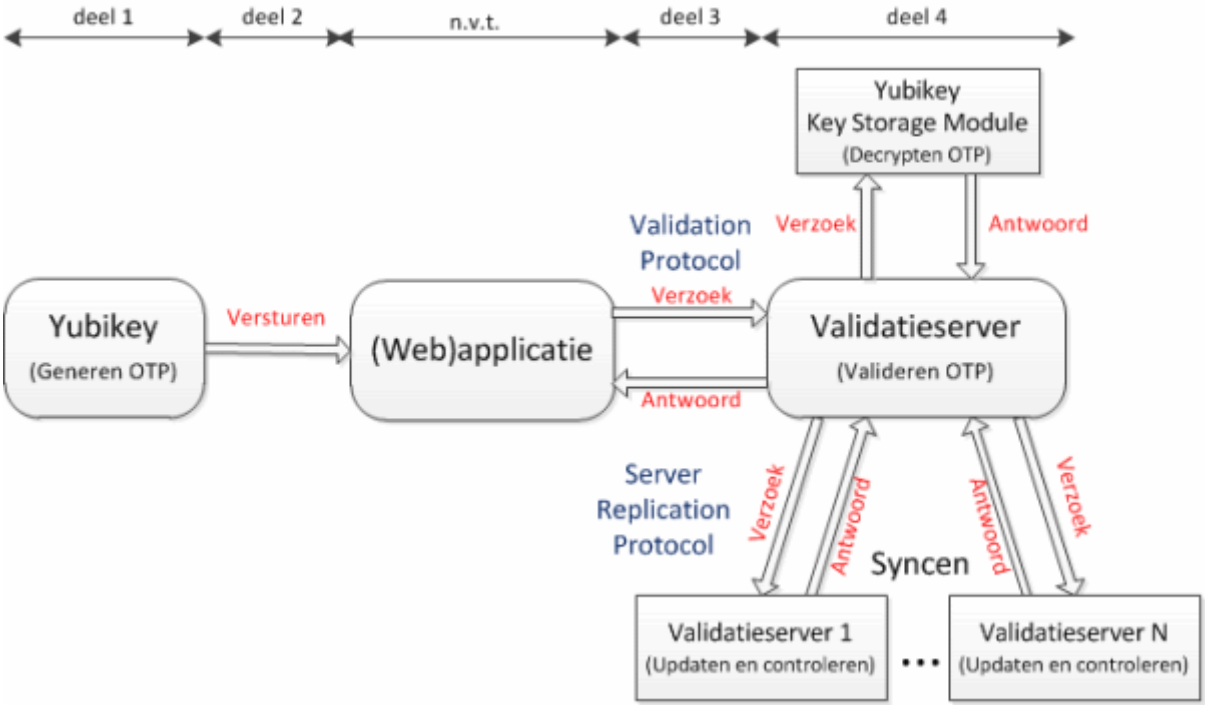
Aan de hand van deze vier onderdelen zullen we de beveiliging van de Yubikey bespreken. We hebben daarom dit onderzoek verderop opgesplitst in de verschillende delen (een hoofdstuk voor elk onderdeel). Voordat we echter ingaan op de beveiliging en gebreken van elk onderdeel, bespreken we eerst de werking en het doel. We doen dit zodat we daarna de beveiliging beter kunnen analyseren (als je niet weet hoe iets precies werkt, is het lastig uitspraken over de beveiliging te doen). De zaken die we zullen bespreken van elk onderdeel zijn:

- De werking: Hoe werkt het onderdeel precies en welke berichten komen eraan te pas?
- Beveiligingsaspecten: Tot welke eisen en beveiligingsaspecten heeft het onderdeel betrekking?
- Beveiligingsanalyse: Welke aanvallen zijn er wellicht mogelijk op het onderdeel, hoe werken ze, hoe is er tegen beveiligd, wat zijn mogelijke oplossingen?

Voordat we overgaan op de specifieke bespreking van elk onderdeel geven we eerst nog een schema waarin de globale werking van de Yubikey duidelijk wordt (welke onderdelen er zijn en hoe ze samenwerken). Daarna zullen we ook nog algemene beveiligingsaspecten noemen welke we daarna per onderdeel zullen behandelen in de daarop volgende hoofdstukken.

2.2.4 Overzicht

De gehele werking van de Yubikey valt samen te vatten in een schema, waarin de verschillende onderdelen en de communicatie hier tussen is aangegeven. Zie figuur 2.1 voor een schema van de totale werking van de Yubikey.



Figuur 2.1: Schema werking Yubikey

Uit dit schema blijkt hoe het totale Yubico protocol in elkaar steekt en hoe elk deel hierin past. Verder hebben we ook een aantal namen genoemd van deelprotocollen (zoals het Validation Protocol), dit zijn protocollen die gebruikt worden door bepaalde onderdelen en welke later nog uitgebreider besproken zullen worden. Hetzelfde geldt voor de Yubikey Key Storage Module en de verschillende validatieservers. Het is op dit moment wellicht nog niet helemaal duidelijk waarvoor deze dienen en hoe deze in het proces passen, maar we wilden ze volledigheidshalve niet weglaten in dit schema. Na het bespreken van de onderdelen in detail, zal het nut van deze subonderdelen en protocollen duidelijk zijn.

Hoofdstuk 3

Algemene beveiligingsaspecten

In de hierop volgende hoofdstukken zullen we diverse zaken met betrekking tot de verschillende onderdelen van de Yubikey gaan bespreken. Hierbij zullen ook diverse eisen aan de Yubikey (en de onderdelen) en beveiligingsaspecten aanbod komen. Voordat we dit doen geven we eerst hier een overzicht van de standaard beveiligingsaspecten (welke vaak binnen de computerbeveiliging gebruikt worden). Daarna zullen we de beveiligingseisen van de Yubikey als geheel noemen, welke we gebruiken om (verderop) de specifieke onderdelen te bespreken. Verder zullen we een voorbeeld bespreken van de Yubikey binnen een groter systeem en welke beveiligingsaspecten de Yubikey binnen dat systeem helpt te beschermen. Als laatste noemen we een aantal randprocessen met betrekking tot de Yubikey waar we ons in dit onderzoek niet expliciet op zullen richten, maar volledigheidshalve niet achterwege willen laten.

3.1 Aspecten

De beveiliging van systemen wordt vaak besproken aan de hand van vijf beveiligingsaspecten. Dit zullen we ook voor de Yubikey doen. De aspecten welke we zullen gebruiken om de beveiliging van de Yubikey te bespreken zijn:

- Integriteit (integrity) - Beschermde data kan niet (ongedetecteerd) worden aangepast.
- Vertrouwelijkheid (confidentiality) - Beschermde data kan niet worden ingezien zonder toegang/toestemming.
- Authenticiteit (authenticity) - Partijen zijn wie ze beweren te zijn.
- Beschikbaarheid (availability) - Het systeem is beschikbaar als nodig (het kan niet door een kwaadwillende worden platgelegd).
- Onweerlegbaarheid (non-repudiation) - Het valt niet te ontkennen dat bepaalde acties zijn uitgevoerd (door een bepaalde partij).

3.2 Eisen

De Yubikey is bedoelt als authenticatiemiddel. Voor authenticatiemiddelen gelden vaak een aantal eisen, zo ook voor de Yubikey. Bij deze eisen komen een aantal beveiligingsaspecten kijken en nog belangrijker, hoe de Yubikey garandeert dat de beveiliging ervan in orde is. Er zijn drie hoofdeisen waaraan de Yubikey moet voldoen om als goed authenticatiemiddel te kunnen functioneren. Het gaat om de volgende eisen:

1. authenticiteiteis: iemand zonder (de juiste) Yubikey mag zich niet (als een ander persoon) kunnen authenticeren. Stel bijvoorbeeld dat Alice in het bezit is van een Yubikey en zich daarmee normaal gesproken authenticeert bij Bob. We willen niet dat Eve zich ook kan authenticeren bij Bob als Alice, terwijl Eve niet in het bezit is van de juiste Yubikey (of helemaal niet in bezit van een Yubikey). Dit zou wellicht kunnen gebeuren als Eve berichten onderschept tussen Alice en Bob (man-in-the-middle aanval). Als Eve zich op deze manier kan authenticeren, dan noemen we dit onrechtmatige authenticatie. De beveiliging van de Yubikey dient er voor te zorgen dat dit niet

mogelijk is. Zoals de naam al doet vermoeden heeft deze eis veel te maken met het beveiligingsaspect authenticiteit. Echter spelen ook vertrouwelijkheid en integriteit een grote rol, zoals later duidelijk zal worden bij de beveiligingsanalyse van de verschillende onderdelen.

2. beschikbaarheideis: authenticatiepogingen moeten altijd gecontroleerd kunnen worden. Dit lijkt voor de hand liggend, maar het probleem zit hem in het woord 'altijd'. Het is niet wenselijk als authenticatiepogingen niet meer gecontroleerd kunnen worden (en dus veel mensen plotseling niet kunnen authenticeren) doordat er toevallig of met opzet iets mis is. Stel bijvoorbeeld dat Alice zich wil authenticeren bij Bob, dan moet Bob de poging gewoon kunnen controleren (ook al is er bijvoorbeeld een validatieserver offline). Het is al helemaal niet wenselijk als een derde persoon (Eve) met opzet (op grote schaal) de controle van authenticatiepogingen kan blokkeren. Wat dus onder andere voorkomen moet worden is dat het complete systeem onbeschikbaar wordt omdat Eve een validatieserver onbereikbaar heeft gemaakt (bijvoorbeeld door middel van een DDoS aanval). Deze eis heeft (uiteraard) veel te maken met het beveiligingsaspect beschikbaarheid.

3. robuustheideis: rechtmatige authenticatiepogingen mogen niet (onterecht) als onrechtmatig bestempeld worden. In principe lijkt dit voor de hand liggend, je gaat er immers vanuit dat dit door het gebruikte protocol al standaard geregeld wordt. We willen echter voorkomen dat een kwaadwillende (ongedetecteerd) rechtmatige authenticatiepogingen kan laten mislukken. Het is niet wenselijk als een rechtmatige authenticatiepoging als onrechtmatig bestempeld wordt (doordat een kwaadwillende iets gedaan heeft). Het is echter heel lastig om dit geheel te voorkomen. We nemen er daarom over het algemeen genoegen mee als het niet ongedetecteerd mogelijk is. Om terug te komen op het vorige voorbeeld: stel Alice wil zich authenticeren bij Bob met de Yubikey, dan is het niet wenselijk als Alice authenticatiepoging als onrechtmatig wordt afgedaan doordat Eve ongedetecteerd berichten van de validatieserver heeft aangepast. Deze eis heeft voornamelijk te maken met de beveiligingsaspecten integriteit en beschikbaarheid. Verder speelt ook vertrouwelijkheid een (kleine) rol.

We zullen deze eisen in een later stadium gebruiken als basis voor de beveiligingsanalyse. We geven nu eerst een beschrijving van de Yubikey binnen een groter geheel en welke aspecten hierbij aan bod komen.

3.3 De Yubikey in een groter geheel

De Yubikey is niet bedoeld om op zichzelf te gebruiken (en dit zou ook van weinig nut zijn). Het apparaat dient gebruikt te worden als authenticatiemiddel binnen een groter systeem. Een voorbeeld van een dergelijk systeem is internetbankieren. De Yubikey zou gebruikt kunnen worden om gebruikers zich te laten authenticeren bij (de website van) hun bank zodat zij hun rekening kunnen inzien en betaalopdrachten kunnen uitvoeren. Het is interessant om te weten hoe dergelijke systemen in gevaar kunnen komen als de beveiliging van de Yubikey niet op orde is. We zullen daarom (voor het algemene geval) kort bespreken welke beveiligingsaspecten van het grotere geheel allemaal in het gedrang komen als de Yubikey niet aan één van de hoofdeisen zou voldoen.

3.3.1 authenticiteiteis

De eis dat onrechtmatige authenticatie niet mogelijk mag zijn, is waarschijnlijk de belangrijkste eis bij de beveiliging van de Yubikey. Immers als dit mogelijk zou zijn en een kwaadwillende gebruiker zich als iemand anders kan authenticeren, dan zijn zowel de integriteit, de vertrouwelijkheid, de authenticiteit en de onweerlegbaarheid in gevaar. Immers een kwaadwillend persoon kan zich dan authenticeren als een gebruiker met hogere rechten en zodoende vertrouwelijke informatie inzien (vertrouwelijkheid in gevaar) of wijzigen (integriteit in gevaar). Verder kan de kwaadwillende acties uitvoeren terwijl hij zich voordoet als iemand anders. Als dit bekend wordt is de onweerlegbaarheid in gevaar (je kan niet meer met volledige zekerheid zeggen dat een bepaalde actie ook echt door een bepaalde persoon is uitgevoerd). Waarom de authenticiteit in gevaar komt is logisch, een kwaadwillende kan zich voordoen bij het systeem als iemand anders en schaadt daarmee de authenticiteit. Er zijn zelfs situaties te bedenken waarbij de beschikbaarheid in gevaar komt. Denk bijvoorbeeld aan een systeem waarbij de gebruiker maar één keer ingelogd kan worden (en anders geweigerd wordt). Als een aanvaller zich al heeft ingelogd als die gebruiker, dan heeft de rechtmatige gebruiker geen toegang meer tot het systeem.

3.3.2 beschikbaarheideis

Deze eis heeft voornamelijk te maken met de beschikbaarheid van een systeem. Als een applicatie een authenticatiepoging niet kan controleren, dan komt de beschikbaarheid van het systeem in gevaar. De applicatie zal dan dus (hoogstwaarschijnlijk) de gebruiker weigeren, die dan dus niet gebruik kan maken van het systeem. Stel bijvoorbeeld dat de applicatie de internetbankieren website van een bank is. Als een kwaadwillende door de validatieservers van Yubico aan te vallen de mogelijkheid op controle kan blokkeren, dan heeft de bank een probleem. De bank kan moeilijk de gebruikers maar toelaten tot het systeem zonder (deugdelijke) authenticatie. Gebruikers tijdelijk weigeren tot het systeem zal in eerste instantie nog niet zo'n probleem zijn, maar wel als dit uren of dagen lang gaat duren. De beschikbaarheid van de Yubikey en alles wat daarbij hoort kan dus van grote invloed zijn op de beschikbaarheid van het gehele systeem. Dat controle altijd mogelijk is (in ieder geval niet gemakkelijk voor langere tijd te blokkeren), is dus een belangrijke eis. Er zijn geen andere beveiligingsaspecten (behalve beschikbaarheid) die in gevaar komen als controle niet meer mogelijk is (mits de applicatie de gebruiker toegang tot het systeem ontzegt en niet zonder controle toegang verleent).

3.3.3 robuustheideis

Ook de eis dat rechtmatige authenticatie niet geweigerd mag worden heeft invloed op de beschikbaarheid van het systeem. Het zou bijvoorbeeld onwenselijk zijn als gebruikers continu (onterecht) geweigerd worden omdat een kwaadwillende bijvoorbeeld berichten van de validatieserver aanpast. In dit geval zouden de gebruikers geen toegang krijgen tot het systeem en is de beschikbaarheid in gevaar. Het wordt zelfs nog een groter probleem als een gebruiker pertinent of voor een lange periode toegang tot het systeem wordt ontzegt doordat er een klein aantal onrechtmatige authenticatiepogingen geweest zijn. Zo kan een kwaadwillende een gebruiker voor langere tijd uitsluiten door maar enkele rechtmatige authenticatiepogingen te laten mislukken. De beschikbaarheid van het systeem is in principe het enige beveiligingsaspect dat in gevaar komt als de Yubikey niet aan deze eis voldoet. De andere aspecten komen niet in gevaar.

Aangezien de authenticiteiteis te maken heeft met vrijwel alle beveiligingsaspecten en het grotere systeem (waarbinnen de Yubikey gebruikt wordt) op meerdere punten kan comprimeren, beschouwen we dit als de belangrijkste beveiligingseis van de Yubikey. De andere eisen hebben slechts met één aspect te maken (namelijk beschikbaarheid), maar zijn zeker niet te verwaarlozen. De schending van deze eisen kan ook grote gevolgen hebben, zoals geïllustreerd bij het voorbeeld met het internetbankieren.

3.4 Randprocessen

In dit onderzoek richten we ons voornamelijk op de kern van de Yubikey, namelijk het Yubico OTP-protocol. Dit houdt in het genereren, communiceren en valideren (controleren) van OTP's. De Yubikey beperkt zich hier echter niet toe. Om het protocol goed te laten werken, zijn er allerlei randzaken (processen) welke ook van invloed zijn op de beveiliging. We zullen deze zaken vanwege de volledigheid kort behandelen. We gaan er hierbij niet te diep op in (we willen ons immers voornamelijk op het Yubico OTP-protocol richten), maar zullen de zaken noemen, kort uitleggen wat het is en hoe het werkt en eventueel hoe het de beveiliging beïnvloedt. Bij een aantal van deze randzaken spelen toch een redelijk aantal mogelijke beveiligingsproblemen een rol. Dit kan een basis zijn voor een eventueel vervolgonderzoek of uitbreiding op dit onderzoek.

3.4.1 Configuratie uploaden

Het is mogelijk een Yubikey opnieuw te configureren. Dit kan bijvoorbeeld wenselijk zijn wanneer een Yubikey eerst was ingesteld op een andere modus (bijvoorbeeld OATH-HOTP), maar later besloten wordt toch weer terug te gaan naar Yubico OTP modus. Wanneer een Yubikey opnieuw geconfigureerd wordt, dan dient deze configuratie (identiteiten en sleutel) ook met de Yubico validatieservers gecommuniceerd te worden. Dit gebeurt door middel van een webpagina, waar een gebruiker zijn e-mail adres, Yubikey serial nummer (optioneel), nieuwe publieke identiteit, nieuwe privé-identiteit, nieuwe sleutel en een OTP (van de nieuwe configuratie) moet opgeven. Yubico communiceert deze gegevens vervolgens naar de validatieservers (het is helaas onbekend hoe dit precies gebeurt).

Belangrijk gegeven hierbij is dat alleen een nieuwe configuratie geüpload kan worden en het dus niet mogelijk is een oude configuratie te wijzigen. Dit is belangrijk voor het waarborgen van de robuustheideis anders is het gemakkelijk een Yubikey configuratie stuk te maken (upload gewijzigde sleutel en/of privé-identiteit voor een bestaande publieke identiteit). Dit is nu niet mogelijk doordat alleen nieuwe configuraties geüpload kunnen worden (elke publieke identiteit is uniek en als een al bestaande publieke identiteit wordt gebruikt, dan geeft de service een foutmelding terug). Een mogelijk beveiligingsgebrek wat hierdoor ontstaat is het expres laten opraken van de beschikbare publieke identiteiten. De service is hiertegen beschermd door middel van een captcha. Bij normaal gebruik zullen de beschikbare publieke identiteiten niet zomaar opraken (een publieke identiteit is standaard 48 bits groot en dat geeft dus $2^{48} = 2.8 \times 10^{14}$ mogelijkheden).

Een ander mogelijk beveiligingsgebrek is het onderscheppen van de nieuwe privé-identiteit en sleutel (deze worden immers als plaintext opgegeven) en hierdoor de authenticiteiteis te ondermijnen. De beveiliging hiertegen is het gebruik van een HTTPS verbinding (dus de vertrouwelijkheid van de gegevens is gegarandeerd). Het mogelijk eerder onderscheppen van de data (met behulp van spyware op de computer van de gebruiker, zoals een keylogger) blijft wel een probleem.

3.4.2 Revocatie service

Yubico biedt ook een service voor de revocatie van Yubikeys (dit is ook een website). Dit kan handig zijn als een Yubikey gestolen is en je wilt voorkomen dat de dief zich met behulp van jouw Yubikey toegang kan verschaffen tot allerlei zaken. Om gebruik te maken van de revocatie service dient men zich eerst te registreren door het opgeven van een e-mail adres, kiezen van een gebruikersnaam en wachtwoord en een gegenereerde OTP van de (master-)Yubikey. Onder de nieuw aangemaakt account kan dan één of meerdere Yubikeys worden gehangen welke (indien eenmaal ingelogd in het systeem) in- en uitgeschakeld kunnen worden. Een Yubikey (configuratie) wordt bij revocatie dus nooit volledig verwijderd en kan later weer hersteld (ingeschakeld) worden (indien gewenst).

Om in te loggen wordt standaard de gebruikersnaam, wachtwoord en een OTP van de master-Yubikey gevraagd (de bij registratie gebruikte Yubikey is de master-Yubikey). Indien de master-Yubikey kwijt is, dan kan worden ingelogd door eerst een speciale URL op te vragen (welke één keer werkt) die verzonden wordt naar het e-mail adres. Er is dus altijd sprake van two-factor authenticatie, op basis van wachtwoord/OTP of wachtwoord/e-mail toegang. Alle communicatie gaat over een HTTPS verbinding (dus de gebruikersnaam en wachtwoord zijn niet makkelijk te onderscheppen). De beveiliging van de toegang tot het systeem lijkt dus redelijk op orde, maar niet waterdicht (met een keylogger kan bijvoorbeeld zowel het wachtwoord voor de revocatie service als het e-mail wachtwoord achterhaald worden). Het is via de revocatie service in principe mogelijk de robuustheideis te ondermijnen (verschaf toegang tot het systeem en schakel de Yubikeys uit), maar de schade is niet groot aangezien dit waarschijnlijk snel opvalt en de Yubikeys later altijd weer hersteld kunnen worden.

Een ander mogelijk probleem is het koppelen van Yubikeys die niet van jou zijn. Dit is in principe niet mogelijk omdat van elke Yubikey een OTP opgegeven dient te worden. Zelfs als een OTP verkregen kan worden, zal dit geen zin hebben aangezien de OTP hoogstwaarschijnlijk al gebruikt is en dus niet meer geldig is. Wel kan het een probleem worden als een OTP onderschept en vervolgens niet doorgestuurd wordt naar de validatieservers (dit wordt uitgebreider besproken in secties 5.3.3, 6.3.4 en 7.3.2).

3.4.3 Yubikey productie

Na het produceren van de Yubikey wordt deze standaard voorgeladen met een Yubico OTP configuratie. De instellingen van deze configuratie (publieke identiteit, privé-identiteit en sleutel) dienen uiteraard met de validatieservers gecommuniceerd te worden. Dit moet veilig gebeuren, net zoals het configureren veilig moet gebeuren (niemand mag de instellingen weten). Yubico heeft hier diverse procedures en protocollen voor opgesteld welke een veilige communicatie zouden moeten garanderen. We gaan verder niet op deze procedures in aangezien het erg lastig is te beoordelen en te controleren hoe veilig deze zijn (dit is afhankelijk van veel (menselijke) factoren). Voor meer informatie over deze procedures zie [18].

Hoofdstuk 4

Genereren OTP

Het genereren van de OTP is het eerste onderdeel van het Yubico OTP authenticatieprotocol en tevens de eerste stap in het authenticatieproces. Dit deel is verantwoordelijk voor het maken van de OTP welke gebruikt wordt als bewijs dat de gebruiker is wie hij beweert te zijn. De OTP moet aan een aantal eisen voldoen (opgebouwd zijn uit bepaalde velden op een voorgeschreven manier), zodat de OTP in een later stadium gecontroleerd kan worden.

We zullen de exacte werking van dit onderdeel uitgebreid behandelen. Daarna als duidelijk is hoe het onderdeel precies werkt, zullen we beginnen met de beveiligingsanalyse. We zullen onderzoeken welke beveiligingseisen en -aspecten een rol spelen bij dit onderdeel. Vervolgens doen we een analyse van de beveiliging met betrekking tot dit onderdeel. Hierbij zullen mogelijke aanvallen aan bod komen en wordt besproken hoe hier juist wel of juist niet tegen beschermd is.

4.1 Werking

Zodra er op de knop van de Yubikey gedrukt wordt, genereert deze een OTP. Deze OTP is een representatie van een versleuteld datablok, bestaande uit een aantal velden (data). Welke velden, representatie en versleuteling gebruikt worden staat beschreven in een deel van het Yubico OTP-protocol. Het genereren van een OTP gebeurt dus altijd op dezelfde manier en zodoende is de validatieserver in staat een OTP te valideren (hoe dit gaat is ook beschreven in het Yubico OTP-protocol, maar hierover meer in hoofdstuk 7).

In deze sectie richten we ons op het deel van het Yubico OTP-protocol waarin beschreven staat hoe een OTP gegenereerd dient te worden. Hierbij komen de verschillende velden waaruit de OTP bestaat aan bod (en lichten we kort toe wat de functie is) en zullen we bespreken hoe de OTP wordt gegenereerd uit deze velden (welke versleutelingen en representaties gebruikt worden). Als laatste zal er een korte introductie zijn van hoe de OTP vervolgens gecommuniceerd wordt met de computer (het volgende hoofdstuk en onderdeel).

Het genereren van de OTP speelt zich geheel binnen de Yubikey (het fysieke apparaat) zelf af. De andere onderdelen van het Yubico OTP-protocol spelen zich elders af (bijvoorbeeld op de validatieservers). Dit hoofdstuk (en onderdeel) richt zich dus op het beschrijven van het proces dat zich binnen de Yubikey zelf afspeelt (als een Yubico OTP gegenereerd wordt).

4.1.1 Velden voor genereren OTP

De OTP wordt gegenereerd door een bepaalde versleutelin toe te passen op een zestien bytes (= 128 bits) groot datablok. Dit datablok bestaat uit verschillende velden. De OTP is opgebouwd uit deze velden en bij het valideren worden (een aantal van) deze velden ook gebruikt. De velden waaruit de OTP bestaat zijn (in volgorde):

1. Zes bytes privé-identiteit (UID)
2. Twee bytes sessieteller
3. Drie bytes timestamp

4. Één byte sessiegebruikteller
5. Twee bytes willekeurige waarde
6. Twee bytes checksum

We zullen de inhoud van elk veld kort toelichten:

privé-identiteit

De privé-identiteit is door middel van de configuratietool (Yubikey Configuration Utility) ingesteld [20]. Dit is ofwel door de gebruiker zelf gedaan ofwel van tevoren door Yubico. De privé-identiteit is zestien bytes groot en wordt opgeslagen in niet-vluchtig geheugen. Het is meestal een willekeurig gekozen waarde (tenzij het voor interne doeleinden handig is hiervan af te wijken). Yubico stelt de privé-identiteit in ieder geval standaard in op een (pseudo-)willekeurige waarde uniek voor elk apparaat.

Het doel van de privé-identiteit is met name om te controleren of de OTP bij de publieke identiteit hoort. Hierover meer in hoofdstuk 7.

sessieteller

De sessieteller houdt bij hoeveel sessies er al geweest zijn. Met een sessie wordt de periode bedoeld waarbij de Yubikey in de USB-poort zit (stroom krijgt). Iedere keer dat de Yubikey opnieuw in de USB-poort wordt gestoken, wordt de sessieteller met één verhoogd. Om de waarde te onthouden, ook als de Yubikey geen stroom krijgt, wordt deze opgeslagen in niet-vluchtig geheugen.

De teller is twee bytes groot, waarbij de eerste bit niet gebruikt wordt (altijd nul is). Dit is vanwege een oude mogelijkheid in de Yubikey versie 1 welke niet meer gebruikt wordt. Zodoende heeft de teller ruimte voor 15 bit grote getallen en kan dus niet groter dan het getal 32.767 worden. Als de teller dit getal heeft bereikt, dan kan de Yubikey niet meer gebruikt worden met de huidige identiteiten en dienen deze opnieuw geconfigureerd te worden (met nieuwe waarden).

Het doel van de sessieteller is het tegengaan van replay-aanvallen. Hierover meer in hoofdstuk 7.

Timestamp

De timestamp is een soort klok, maar werkt iets anders dan een normale klok. De drie bytes timestamp wordt opgeslagen in vluchtig geheugen en de waarde blijft dus niet bewaard als de Yubikey geen stroom meer krijgt. Zodra de Yubikey in een USB-poort gestoken wordt, zal de timestamp op een willekeurige waarde worden ingesteld. Daarna wordt deze verhoogd met een 8 Hz klok. Bij een overflow zal de timestamp weer op nul gezet worden (en vervolgens weer verder oplopen).

De timestamp wordt standaard alleen gebruikt om de OTP meer willekeurigheid te geven en (momenteel) niet bij het valideren. Echter kan de timestamp ook nog een ander doel bieden, namelijk om phishing aanvallen tegen te gaan. Dit laten we echter buiten beschouwing omdat binnen het Yubico OTP-protocol niet gespecificeerd staat hoe dit zou moeten gebeuren (het wordt overgelaten aan de applicatie om hier wel of niet iets mee te doen).

sessiegebruikteller

De sessiegebruikteller houdt bij hoe vaak de Yubikey gebruikt is binnen de huidige sessie. Aangezien deze teller alleen onthouden hoeft te worden binnen de sessie (zolang de Yubikey stroom krijgt), wordt deze opgeslagen in vluchtig geheugen. Bij een overflow wordt de sessieteller verhoogd en de sessiegebruikteller weer op nul ingesteld.

Het doel van de sessiegebruikteller is om samen met de sessieteller replay-aanvallen tegen te gaan. Verder wordt de teller gebruikt om de levensduur van de Yubikey te vergroten. Dankzij de sessiegebruikteller hoeft de sessieteller immers minder vaak verhoogd te worden.

Willekeurige waarde

De willekeurige waarde is twee bytes groot en wordt opgeslagen in vluchtig geheugen. De waarde wordt gegenereerd door middel van een Linear Feedback Shift Register (LFSR) gevoed met data afkomstig van USB-activiteit en de drukknop.

Het doel van de willekeurige waarde is extra beveiliging bieden door het datablok groter en willekeuriger te maken. Zodoende lijken de gegenereerde OTP's ook willekeuriger en is het moeilijker de gebruikte cryptografie (meer hierover in sectie 4.3) te kraken.

checksum

De checksum is twee bytes groot en wordt berekend door het CRC-16 algoritme (ISO13239 standaard) toe te passen op de andere velden (veertien bytes).

De CRC-16 checksum wordt niet gebruikt bij het bepalen of de OTP geldig is of niet. Het wordt enkel gebruikt om te controleren of er geen datacorruptie is opgetreden. Zonder deze controle zou een OTP onterecht kunnen worden afgekeurd of wellicht zelfs een Yubikey onbruikbaar worden gemaakt. Hierover meer in hoofdstuk 7.

4.1.2 Algoritme voor genereren OTP

Nu is duidelijk uit welke velden de OTP opgebouwd wordt. De inhoud van deze velden wordt uiteraard niet zomaar als plaintext verstuurd (dat zou zeer onveilig zijn). Er wordt daarom encryptie toegepast.

Allereerst worden alle velden achter elkaar gezet (in dezelfde volgorde als waarin ze genoemd zijn). Zo ontstaat een zestien bytes (= 128 bits) groot datablok. Op dit datablok wordt de Advanced Encryption Standard (AES) encryptie toegepast. Dit gebeurt met een 128 bits grote AES-sleutel (welke vooraf in de Yubikey geprogrammeerd is met de Yubikey Configuration Utility [20]). De versleuteling vindt plaats in ECB-modus. Dit houdt in dat de datablokken allen apart versleuteld worden en geen feedback van vorige versleutelingen wordt gebruikt (zoals bij andere modussen wel vaak gebeurt). ECB is een logische keuze, aangezien er in het geval van de Yubico OTP slechts één datablok is (van precies 128 bits lang) en deze modus zodoende dus eigenlijk de enige keus is.

Na het versleutelen met AES hebben we een 128 bits OTP (dit is de ontstane ciphertext van de AES-versleuteling toegepast op de eerder genoemde velden). Dit wordt niet direct verstuurd, maar hier wordt eerst nog ModHex codering op toegepast. Hierbij wordt de OTP hexadecimaal gerepresenteerd en vervolgens worden diverse karakters vervangen door anderen. Dit is niet om de OTP veiliger te maken, maar enkel om compatibiliteit te vergroten[1]. Als resultaat ontstaat er uiteindelijk een 32 tekens lange string opgebouwd uit zestien verschillende karakters. Deze string is de OTP die vervolgens met de applicatie gecommuniceerd wordt. Zie figuur 4.1 voor een schema van het complete proces.

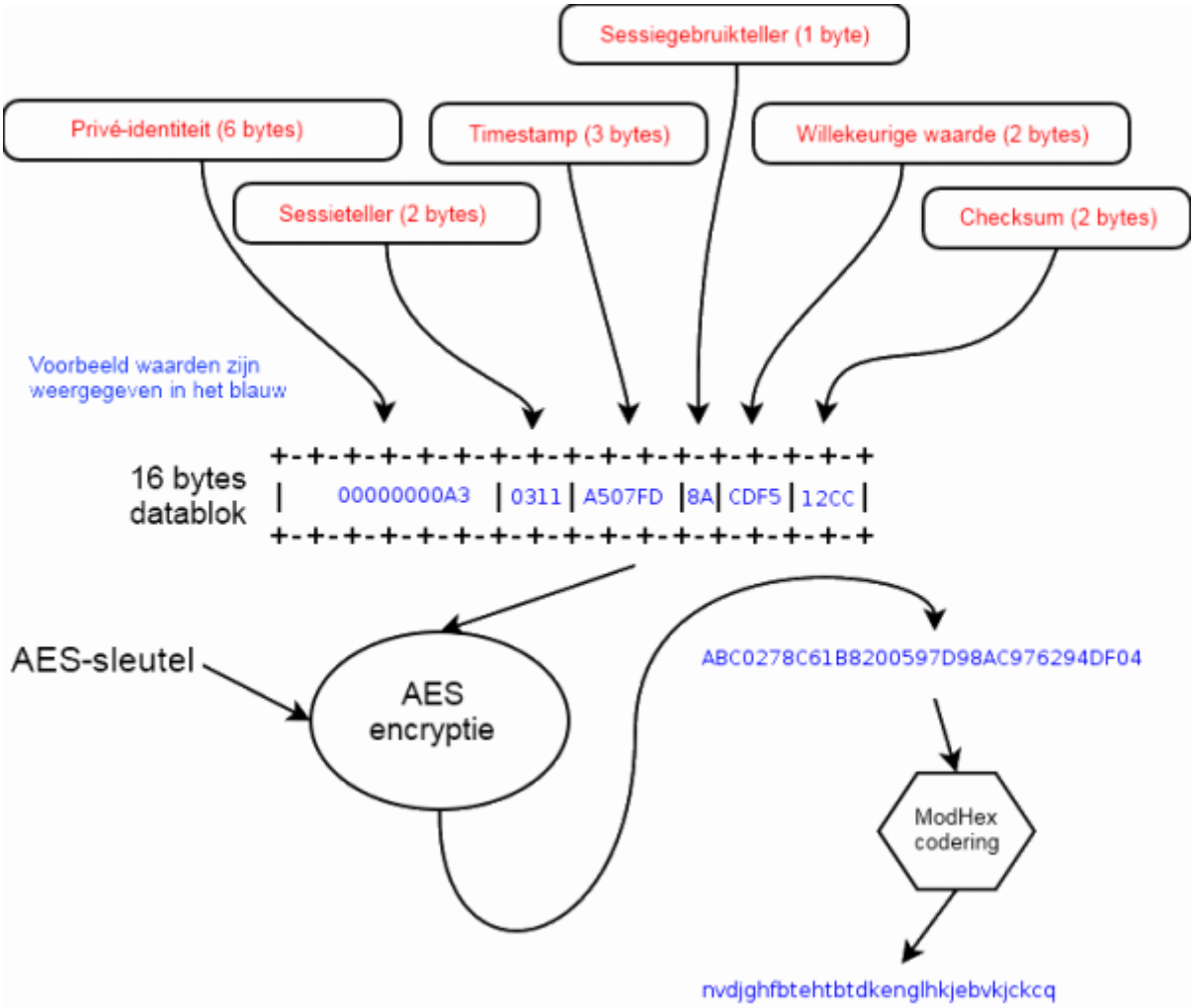
4.1.3 Instellingen

Zoals al aangegeven is een aantal velden voor het genereren van de OTP vooraf op de Yubikey ingesteld (door de gebruiker of door de fabrikant). Dit wordt gedaan met de Yubikey Configuration Utility. Hiermee kunnen gegevens op de Yubikey geschreven (geplaatst, gewijzigd of verwijderd) worden. Het gaat in dit geval om de publieke identiteit, de privé-identiteit en de AES-sleutel. Soms moet er als extra beveiliging eerst een wachtwoord opgegeven worden voordat de instellingen geschreven kunnen worden. Hierover in sectie 4.3 meer. De gegevens kunnen niet direct uitgelezen worden. De Yubikey kan enkel een OTP uitvoeren, maar niet de opgeslagen instellingen (ook hierover in sectie 4.3 meer).

4.2 Beveiligingsaspecten

De belangrijkste beveilingseis waar dit onderdeel betrekking op heeft is de authenticiteiteis (onrechtmatige authenticatie voorkomen). Het genereren van de OTP wordt op zo'n manier gedaan dat alleen met de juiste Yubikey een OTP gegenereerd kan worden die past bij de publieke identiteit van die Yubikey.

[1] Om de OTP te versturen wordt namelijk een toetsenbord geëmuleerd. Om problemen met taalinstellingen te voorkomen worden enkel karakters gebruikt die op elke soort toetsenbord op dezelfde plaats zitten en dus altijd hetzelfde resultaat opleveren, ongeacht de taalinstellingen [21].



Figuur 4.1: Schema genereren OTP

Het algoritme zou dus op een dergelijke manier in elkaar moeten zitten dat het onmogelijk is een OTP te genereren behorende tot die Yubikey zonder die Yubikey (of in ieder geval de informatie die er op staat) in bezit te hebben.

Behalve op de authenticiteiteis heeft dit onderdeel ook nog invloed op de robuustheideis (geen rechtmatige authenticatie weigeren). Het moet voorkomen worden dat het apparaat zomaar kan worden aangepast waardoor de rechtmatige eigenaar zich niet meer kan authenticeren. Er zijn mogelijkheden bij dit onderdeel om dit te doen, vandaar dat we deze eis ook zullen bespreken bij de beveiligingsanalyse.

Wat betreft de aspecten is voornamelijk de vertrouwelijkheid erg belangrijk bij dit onderdeel. Het is essentieel (om de authenticiteiteis te kunnen garanderen) dat informatie (bijvoorbeeld de privé-identiteit) niet zomaar van de Yubikey kan worden uitgelezen. De Yubikey zou dus het liefst op een dergelijke manier beschermd moeten zijn dat opgeslagen geheime informatie onmogelijk valt uit te lezen. Verder is ook het aspect authenticiteit erg belangrijk. De gebruikte cryptografie wordt in dit geval namelijk gebruikt om de authenticiteit te kunnen vaststellen. Er wordt immers van uitgegaan dat alleen met de juiste AES-sleutel de juiste OTP gegenereerd kan worden. Dit speelt een grote rol bij het verifiëren van de authenticiteit.

De aspecten beschikbaarheid en integriteit zijn voor dit onderdeel met name voor de robuustheideis van belang. Als de Yubikey niet beschikbaar is, kan deze geen OTP's genereren en kunnen gebruikers zich niet authenticeren. De meeste zaken met betrekking tot beschikbaarheid vallen echter buiten dit

onderzoek, omdat ze niet zozeer met computerbeveiliging te maken hebben, maar meer met fysieke beveiliging (denk aan Yubikey gestolen, kwijt of kapot). Wat nog wel binnen dit onderzoek valt is de integriteit van het onderdeel. Als een gebruiker de informatie op de Yubikey zomaar kan wijzigen, kan de Yubikey onbruikbaar worden (schrijf bijvoorbeeld de privé-identiteit over met onzinwaarden). Hierdoor zou de beschikbaarheid in het geding komen en dit zou dus niet zomaar moeten kunnen.

Als laatste is er nog het aspect onweerlegbaarheid. Dit aspect is niet belangrijk voor dit onderdeel, omdat op dit onderdeel specifiek geen acties kunnen worden uitgevoerd (of iets gedaan kan worden) waarvan het belangrijk is zeker te weten wie dit gedaan heeft (er worden ook geen logs bijgehouden).

4.3 Beveiligingsanalyse

Hierboven is besproken welke beveiligingseisen en -aspecten belangrijk zijn voor dit onderdeel. We zullen nu een beveiligingsanalyse geven op dit onderdeel. Dit doen we door eerst kort een bespreking te geven van de belangrijkste beveiligingsmaatregelen binnen het onderdeel, gebaseerd op de (van toepassing zijnde) beveiligingseisen. Hieruit zullen mogelijke aanvallen en beveiligingen worden afgeleid welke daarna stuk voor stuk besproken worden. Bij deze bespreking zal de mogelijke aanval worden uitgelegd, de beveiliging worden behandeld (als deze er is), het risico worden ingeschat en de van toepassing zijnde beveiligingsaspecten genoemd worden. We zullen dit voor de volgende onderdelen op dezelfde manier doen.

Om onrechtmatige authenticatie te voorkomen moet de Yubikey zorgen dat het genereren van de juiste OTP in principe alleen mogelijk is met de juiste Yubikey. Om hiervoor te zorgen maakt de Yubikey gebruik van encryptie en geheimen. De beveiliging van de Yubikey staat of valt met de sterkte van de encryptie en hoe goed de geheimen verborgen blijven.

Om precies te zijn kent de Yubikey twee geheimen. De 128 bits AES-sleutel en de zes bytes (= 48 bits) privé-identiteit. Met behulp van deze twee gegevens kan de juiste OTP gegenereerd worden. Om te zorgen dat de privé-identiteit in de OTP geheim blijft, wordt deze (samen met andere data) versleuteld met behulp van AES. Het is dus belangrijk dat AES cryptografisch sterk (genoeg) is en dat de geheimen ook daadwerkelijk geheim blijven. Dit kan gezien worden als twee mogelijke aanvalspunten op dit onderdeel, namelijk het cryptografisch kraken van de Yubikey of het proberen te achterhalen van de gebruikte geheimen.

Om te zorgen dat rechtmatige authenticatie altijd mogelijk is (en blijft), is het belangrijk dat de Yubikey en de data er op intact blijven. Als dit niet het geval is, kan een rechtmatig persoon zich plotseling niet meer authenticeren. Een mogelijke aanval is dus om de Yubikey en/of de data die hier op staat proberen te verwijderen of te wijzigen (en hiermee de robuustheideis in het gedrang te brengen). Dit levert twee (andere) mogelijke aanvallen op, namelijk de Yubikey fysiek aanvallen (stelen of vernietigen) en de data op de Yubikey proberen te benaderen (wijzigen of wissen).

4.3.1 Cryptografie

De beveiliging van de Yubikey berust voor een groot deel op de gebruikte cryptografie, in dit geval AES. Bij het Yubico OTP-protocol kan de (gegenereerde) OTP als de ciphertext gezien worden. Dan is de 128 bits AES-sleutel de encryptie key (sleutel) en het zestien bytes grote datablok met onder andere de privé-identiteit (zoals besproken in sectie 4.1) de plaintext.

Het gebruik van cryptografie in de Yubikey heeft twee doelen:

1. Zorgen dat alleen met de juiste Yubikey een geldige OTP (voor die Yubikey) gegenereerd kan worden. Hiertoe is het belangrijk dat de AES-sleutel en privé-identiteit geheim blijven. Dit doel heeft voornamelijk met de beveiligingseis authenticiteit te maken en berust op de beveiligingsaspecten vertrouwelijkheid en authenticiteit.

2. In combinatie met de gebruikte CRC-checksum voorkomen dat de meegestuurde tellerwaarden in de OTP makkelijk aangepast (verhoogd) kunnen worden. Dit onder andere om replay-aanvallen (zie hoofdstuk 5 en 7) en het onbruikbaar maken van de Yubikey (zie hoofdstuk 5) te voorkomen.

Dit doel heeft zodoende te maken met de beveiligingseisen authenticiteit en robuustheid en berust voornamelijk op het beveiligingsaspect integriteit.

Een mogelijke aanval op de Yubikey zou dus zijn het (proberen te) kraken van de encryptie om zodoende één van de doelen en daarmee een beveiligingseis te ondermijnen.

Om te beginnen met doel één. De Yubikey kent twee geheimen, de 128 bits AES-sleutel en de zes bytes (= 48 bits) privé-identiteit. Deze twee gegevens zijn de basis om te zorgen dat een geldige OTP alleen met de juiste Yubikey gegenereerd kan worden. Het is dus essentieel ervoor te zorgen dat deze gegevens niet kunnen worden afgeleid uit een OTP, want anders kan een aanvaller in de toekomst zelf geldige OTP's genereren. De gebruikte cryptografie (AES) moet voorkomen dat dit kan gebeuren.

In principe geldt dat als de AES-sleutel achterhaald is, de privé-identiteit ook zeer makkelijk achterhaald kan worden (ontsleutel namelijk een willekeurige OTP en je kan de privé-identiteit zo uitlezen). Overigens zou het ook vreemd zijn als zonder AES-sleutel uit de OTP (de ciphertext) zomaar de privé-identiteit achterhaald kan worden (een deel van de plaintext). Dat zou namelijk betekenen dat AES een zeer ongewenste eigenschap heeft, namelijk dat de vertrouwelijkheid niet gewaarborgd is (het hoofddoel van AES-cryptografie [2]). We zullen ons om deze redenen beperken tot het proberen te achterhalen van de gebruikte AES-sleutel.

Wat betreft doel twee geldt in principe hetzelfde. De AES-sleutel dient achterhaalt te worden om de meegestuurde tellerwaarden in de OTP succesvol te kunnen aanpassen. AES zorgt er (in combinatie met de CRC-checksum) voor dat dit niet zomaar kan. Een aanvaller weet immers niet welk deel van de ciphertext hij moet bewerken om de tellerwaarden aan te passen en ook niet hoe hij dit deel zou moeten bewerken (AES gebruikt zowel transpositie als substitutie [2] [15]). Als iemand wel in staat is om (alleen) de tellerwaarden te wijzigen in de waarde die hij wilt, dan zou AES een ongewenste eigenschap bevatten (je moet daarvoor immers van ciphertext naar plaintext kunnen, iets wat zeer onwenselijk is bij encryptie).

Zelfs als een aanvaller de tellerwaarden van de OTP (toevallig) weet op te hogen, zal dit nog steeds niet werken omdat de meegestuurde CRC-checksum niet meer overeenkomt. De aanvaller kan de CRC-checksum niet opnieuw berekenen, omdat hij niet de waarde van (alle) velden kent (gaan we tenminste vanuit als de encryptie goed is). Bovendien weet de aanvaller ook niet wat en hoe van de ciphertext aan te passen om de CRC-checksum te wijzigen.

Beide doelen realiseren komt in dit geval dus neer op het geheim houden (niet zomaar kunnen afleiden) van de AES-sleutel. We zullen daarom bij de verdere bespreking van de cryptografie ons dan ook richten op eventuele mogelijkheden om de sleutel te achterhalen.

Brute-force

Een manier om de gebruikte sleutel te achterhalen zou zijn om deze te brute-forcen. Dit houdt in om "met brute kracht" gewoon alle mogelijke combinaties te proberen, net zolang tot de juiste gevonden wordt. Een belangrijk punt hierbij is weten wanneer de juiste sleutel gevonden is. Over het algemeen zijn hier een aantal mogelijkheden toe, bijvoorbeeld door een zelfgekozen plaintext in te voeren, deze te laten versleutelen en dan de ciphertext te onderscheppen. In dat geval zijn de ciphertext en plaintext bekend en komt het kraken van de key neer op het net zolang proberen van de key om de ciphertext te ontsleutelen (of de plaintext te versleutelen) en te vergelijken met de plaintext (of ciphertext). Als deze twee overeenkomen, dan is de juiste key gevonden. Deze methode is echter bij de Yubikey niet mogelijk, aangezien de plaintext binnenin de Yubikey bepaald wordt en geen velden van buitenaf kent (zie ook sectie 4.3.1).

Er zijn echter wel andere methoden om een brute-force aanval uit te voeren op de Yubikey (om te controleren of een gevonden key de juiste is). Een mogelijkheid is het proberen te ontsleutelen van de ciphertext met een willekeurige key en dan gewoon aan te nemen dat het gelukt is. Probeer dan met deze sleutel en de privé-identiteit, achterhaald uit de ontsleutelde ciphertext, zelf een nieuwe OTP te genereren. Laat deze OTP valideren door de validatieserver. Als hij geaccepteerd wordt heb je blijkbaar de juiste key en plaintext gevonden. Deze methode geeft echter zeer veel werk door het continu moeten valideren van (zelf gegenereerde) OTP's. Bovendien zal het versturen van zoveel (mislukte) validatieverzoeken op den duur waarschijnlijk opvallen waardoor Yubico maatregelen kan nemen.

Een andere, waarschijnlijkere methode is (vreemd genoeg) ontstaan na invoeren van versie twee van het Validation Server Protocol (hierover meer in hoofdstuk 6). Sinds deze versie is het mogelijk de vali-

datieserver delen van de plaintext terug te laten geven bij een OTP (namelijk de waarden van de teller en de timestamp). Deze informatie kan bij een brute-force aanval gebruikt worden om de juiste key te herkennen.

Zoals hierboven aangegeven is het dus mogelijk om de plaintext te herkennen bij een brute-force aanval en dus ook om deze methode te gebruiken om de key te achterhalen. De gebruikte key is 128 bits groot. Dit betekent dat er $2^{128} \approx 3 * 10^{38}$ mogelijke sleutels zijn. Zelfs als het mogelijk zou zijn om één miljard AES-ontsleutelingen per seconde uit te voeren, dan zou het gemiddeld nog $\frac{2^{128}}{2*10^9*86400*365} \approx 5 * 10^{21}$ jaar duren. Dit valt ver buiten de scope van een realistische aanval. Echter neemt computer rekenkracht wel steeds meer toe. Ondanks dat deze aanval nu nog geen groot risico met zich meebrengt, zou dit in de toekomst wellicht problemen kunnen opleveren.

Choosen plaintext aanval

Bij een choosen plaintext aanval is een aanvaller in staat elke zelf gekozen plaintext te laten versleutelen tot ciphertext met het algoritme (AES) en de gebruikte key (de AES-sleutel in de Yubikey). De resultaten hiervan kan de aanvaller vervolgens gebruiken om de key te achterhalen. Deze aanval is bij de Yubikey niet mogelijk, aangezien er geen mogelijkheid is tot handmatige invoer van gegevens om te laten versleutelen bij de Yubikey [2]. Alle velden van de Yubikey welke gebruikt worden om de OTP te genereren staan namelijk intern in de Yubikey. Verder is de gebruike encryptie (AES) in principe ongevoelig voor dit soort aanvallen (zelfs als de mogelijkheid tot een choosen plaintext aanval bestaat, dan zou het nog computationeel zeer veel werk en tijd vergen om de sleutel te achterhalen).

Known plaintext aanval

Zelfde principe als de hiervoor besproken aanval, behalve dat de aanvaller niet elke willekeurige plaintext kan laten versleutelen, maar een aantal plaintext en bijbehorende ciphertext combinaties tot zijn beschikking heeft. In principe is ook deze aanval zeer lastig uit te voeren, aangezien de volledige plaintext niet bekend is. Zo bevat de plaintext de privé-identiteit welke geheim is en ook deels een willekeurige waarde (welke als het goed is niet bekend kan zijn). Wel zijn de waarden van de tellers en de timestamp van elke OTP te achterhalen. Echter zijn er zelfs dan slechts zes van de zestien bytes beschikbaar. Bovendien geldt ook in dit geval dat AES redelijk ongevoelig is voor choosen of known plaintext aanvallen en het computationeel gezien zeer veel tijd zou kosten een dergelijke aanval uit te voeren.

replay-aanvallen

AES in ECB modus gebruikt voor authenticiteit is (net zoals vele andere encryptie-algoritmen) gevoelig voor replay-aanvallen. Dit probleem is elders binnen het Yubico OTP-protocol opgelost door het gebruik van tellerwaarden (en aan de hand hiervan controleren of een OTP nieuw is). Meer hierover in hoofdstuk 5 en 7.

Conclusie cryptografie

Er lijken geen redelijkerwijs uit te voeren aanvallen op de cryptografie van de Yubikey te bestaan (ervan uitgaande dat AES geen ongewenste eigenschappen heeft). Er zijn wel een aantal mogelijke aanvallen welke niet tot nauwelijks te voorkomen zijn, zoals een brute-force aanval. Deze aanvallen zijn echter bij de gekozen sleutellengte op het moment niet binnen afzienbare tijd mogelijk. De rekenkracht van computers neemt echter wel gestaag toe, dus dit kan toch iets zijn om rekening mee te houden in de toekomst.

4.3.2 Geheimen achterhalen

Niet alleen de cryptografie moet goed genoeg zijn om de gebruikte geheimen te beveiligen, ook moet het niet mogelijk zijn de geheimen op een andere manier te achterhalen. De gebruikte cryptografie kan immers nog zo sterk zijn, het is zinloos als de key (op een andere manier) gewoon achterhaald

[2] Dit is overigens wel deels mogelijk bij de Yubikey in challenge-response mode [21], maar we hebben besloten deze modus buiten beschouwing te laten in dit onderzoek

kan worden. Het moet dus onmogelijk zijn de privé-identiteit en/of AES-sleutel van een Yubikey te achterhalen, oftewel de vertrouwelijkheid moet gewaarborgd zijn. We zullen ons nu beperken tot het extraheren van de geheimen uit de Yubikey zelf. Het verkrijgen van de geheimen uit andere onderdelen (bijvoorbeeld de validatieserver), zal later besproken worden.

De Yubikey kent geen mogelijkheden om, behalve dan een gegenereerde OTP, data uit te voeren via de (USB) interface. Zodoende is het dus niet mogelijk om de privé-identiteit en/of AES-sleutel van een Yubikey uit te lezen. Deze velden zijn, net zoals de andere velden in de Yubikey, namelijk alleen te beschrijven en niet uit te lezen.

Het is eventueel wel een mogelijkheid om de Yubikey open te maken en te proberen de data rechtstreeks uit de (geheugen)chips te lezen. Hiervoor is echter fysieke toegang tot het apparaat vereist en bovendien veel specialistische kennis en wellicht dure apparatuur nodig. Zodoende is dit niet echt een optie. Bovendien zal de gebruiker hoogstwaarschijnlijk merken dat zijn Yubikey weg of geopend is (beschadigingen). We zullen ons verder niet tot deze mogelijkheid richten, omdat dit buiten de scope van het onderzoek valt.

Zodoende kunnen we dus stellen dat het achterhalen van de geheimen uit de Yubikey zelf geen serieuze mogelijkheid is (door de beveiliging dat het uitlezen van data simpelweg niet mogelijk is).

4.3.3 Yubikey fysiek aanvallen

De Yubikey fysiek aanvallen is een optie om te zorgen dat rechtmatige authenticatiepogingen mislukken. De Yubikey kan namelijk gestolen of kapot gemaakt worden. Als dat gebeurt is (ook) rechtmatige authenticatie niet meer mogelijk.

In dit onderzoek zullen we ons hier niet op richten. Dit is namelijk een voor de hand liggend probleem, waar geen gemakkelijke oplossing voor te bedenken valt en waar de beveiliging van de Yubikey simpelweg geen rekening mee houdt. Bovendien is voor deze soort aanval fysieke toegang tot de Yubikey nodig en zal het de eigenaar waarschijnlijk snel opvallen waardoor hij tegenmaatregelen kan nemen.

Desalniettemin vormt dit wel een serieus probleem, wat de robuustheideis en het aspect beschikbaarheid kan ondermijnen. Mogelijke oplossingen hiertegen bevinden zich op applicatieniveau. Zo is het een optie om binnen een applicatie een mogelijkheid in te bouwen om in speciale gevallen toch te kunnen authenticeren zonder Yubikey (bijvoorbeeld door het beantwoorden van een geheime vraag of verificatie met behulp van e-mail, zoals ook vaak gebruikt wordt als het wachtwoord verloren is). Dit beperkt de beveiliging van het systeem enigszins (met name met betrekking tot de authenticiteiteis), maar kan desondanks toch een goed idee zijn. Anders zou namelijk een gebruiker de toegang tot het systeem volledig en onherroepelijk kwijt kunnen raken als de Yubikey verloren gaat. Dit kan zelfs per ongeluk gebeuren (de gebruiker kan de Yubikey kwijtraken of de Yubikey kan “spontaan” kapot gaan).

4.3.4 Data wijzigen

Een andere aanval om een Yubikey onbruikbaar te maken (en dus de robuustheideis en de beschikbaarheid te ondermijnen) is door de data op de Yubikey te wijzigen. Deze mogelijke aanval zullen we wel in detail bespreken, omdat dit onopgemerkt kan gebeuren en bovendien wellicht ook zonder fysieke toegang tot de Yubikey (en de aanval dus waarschijnlijker is). Als data op de Yubikey wordt aangepast (bijvoorbeeld de privé-identiteit wordt gewijzigd of de AES-sleutel wordt gewist), dan is authenticatie met de Yubikey niet meer mogelijk. Er is een aantal manieren waarop dit zou kunnen gebeuren:

- Een aanvaller leent de Yubikey tijdelijk en gebruikt de Yubikey Configuration Utility om data op de Yubikey te wijzigen of te wissen. Dit kan snel en onopgemerkt gebeuren.
- Een aanvaller besmet het systeem waarop de Yubikey gebruikt wordt met een virus (host aanval) en laat dit virus de data op de Yubikey aanpassen of wissen. Deze aanval zou kunnen gebeuren zonder fysieke toegang tot de Yubikey en de computer. De aanval is ook mogelijk, aangezien de configuration utility ook in staat is data op de Yubikey te wijzigen en er in principe geen reden denkbaar is waarom een ander programma (virus) dit niet zou kunnen.
- Een aanvaller haalt de gebruiker over de Yubikey op zijn systeem te gebruiken waarop een programma draait om de data aan te passen. Ongeveer dezelfde situatie als hiervoor, behalve dat de

aanvaller niet een ander systeem hoeft te besmetten, maar gewoon zijn eigen systeem gebruikt (wat vele malen makkelijker is).

De Yubikey kent tegen deze aanvallen een beveiliging. Een configuratie op de Yubikey kan namelijk beveiligd worden tegen schrijven met behulp van een zes bytes (= 48 bits) groot wachtwoord. Als het juiste wachtwoord niet wordt meegegeven, dan wordt de schrijfactie niet uitgevoerd. Dit biedt een zekere bescherming, maar is niet optimaal. Allereerst is een 48 bits wachtwoord relatief klein, welke wellicht nu al of in de nabije toekomst makkelijk gekraakt kan worden (met een simpele brute-force aanval). Ten tweede wordt het wachtwoord iedere keer als platte tekst naar de Yubikey doorgegeven, wat betekent dat deze in principe onderschept zou kunnen worden. Zodoende is deze beveiliging gevoelig voor replay- en man-in-the-middle aanvallen. Ten derde (en waarschijnlijk het belangrijkst) is deze beveiliging met het wachtwoord niet standaard. Standaard zijn de configuraties namelijk onbeveiligd. Dit maakt de bovengenoemde aanvallen in veel gevallen erg waarschijnlijk (de meeste gebruikers zullen de beveiliging immers niet aan hebben staan als deze niet standaard is).

Er is ook nog een andere mogelijkheid om de data op de Yubikey te wijzigen, namelijk de Yubikey fysiek aanvallen. Deze aanval laten we buiten beschouwing om dezelfde redenen eerder genoemd (zie sectie 4.3.3).

Hoofdstuk 5

Communiceren OTP met (web)applicatie

Nadat de Yubikey de OTP heeft gegenereerd, dient deze te worden gecommuniceerd met de applicatie. Dit onderdeel is verantwoordelijk voor deze communicatie en moet zorgen dat dit (veilig genoeg) gebeurt.

5.1 Werking

Als de Yubikey de OTP eenmaal gegenereerd heeft, dan moet deze uiteraard naar de applicatie gestuurd worden. De Yubikey doet dit door het emuleren van een toetsenbord. De OTP wordt door de USB interface van de Yubikey verstuurd in de vorm van zogenaamde toetsenbord scan codes. Deze codes geven aan welke toets is ingedrukt en het besturingssysteem vertaalt dit vervolgens naar karakters (een belangrijk punt hierbij is het gebruik van ModHex codering, een door Yubico ontwikkelde codering om problemen met taalinstellingen te ondervangen). Deze karakters worden in de applicatie ingevoerd, meestal door middel van een standaard tekstveld. Het idee is dat zodra de OTP gegenereerd moet worden, de huidige applicatie en het tekstveld waar de OTP in moet komen actief zijn (de focus hebben). Het besturingssysteem zorgt dan namelijk automatisch dat de OTP op de juiste plek terechtkomt (net zoals wanneer je een tekstveld selecteert en gaat typen de letters in dat tekstveld komen).

Dit is de standaard situatie. Er zijn natuurlijk ook andere situaties denkbaar, waarbij de gebruiker bijvoorbeeld de OTP in een tekstverwerker laat genereren en deze vervolgens kopieert. Dit is echter omslachtig en er is geen echte reden om dit te doen. Het is niet mogelijk dat de applicatie automatisch een OTP laat genereren. De gebruiker moet dit proces starten door op de knop van de Yubikey te drukken[1] (hiervoor is gekozen uit veiligheidsoverwegingen). Ook is het niet mogelijk om de OTP via een andere interface dan over de USB interface te communiceren[2]. Dit kan enkel gebeuren door het emuleren van een toetsenbord. Het emuleren van het toetsenbord is enigszins omslachtig (hierdoor is de ModHex codering noodzakelijk), maar vergroot de compatibiliteit met allerlei systemen zonder dat er aparte drivers of software nodig is.

Bij lokale applicaties is de stap van het communiceren met de applicatie nu voltooid. In het geval van een webapplicatie moet de OTP echter nog met de webserver gecommuniceerd worden. We gaan ervan uit dat de OTP op dit moment in een HTML-formulier staat in de browser van de gebruiker (de standaard situatie). Nu moet deze OTP (en eventueel de andere inhoud van het formulier) met de webserver gecommuniceerd worden. Dit gebeurt (zoals in de meeste gevallen) over een HTTP of HTTPS verbinding. Beiden zijn binnen in het Yubico OTP-protocol toegestaan. Qua beveiliging is het HTTPS protocol uiteraard te prefereren (en wordt ook door Yubico aangeraden), maar het is geen vereiste.

[1] Dit is overigens wel mogelijk in de nieuwe challenge-response mode [21]. We laten deze modus echter buiten dit onderzoek zoals in de focus van het onderzoek in hoofdstuk 2 besproken is. Het is niet mogelijk in de standaard Yubico OTP modus.

[2] Ook dit is mogelijk in de challege-response mode en in bepaalde varianten van de Yubikey [21], maar laten we verder buiten beschouwing.

5.2 Beveiligingsaspecten

Op dit onderdeel is voornamelijk de robuustheideis (geen rechtmatige authenticatie weigeren) van invloed. Het algoritme voor het genereren van de OTP en van de validatieserver zorgen er immers voor dat de OTP geaccepteerd wordt als deze aankomt zoals deze gegenereerd was. Het is dus belangrijk dat dit onderdeel ervoor zorgt dat de OTP exact hetzelfde aankomt bij de applicatie als hoe deze de Yubikey verliet.

Ook enigszins belangrijk is de beschikbaarheideis (controle altijd mogelijk). Dit valt echter buiten de scope van het onderzoek, aangezien dit meer van invloed is op of de applicatie in zijn geheel beschikbaar is, dan op het Yubico protocol. Als de applicatie niet beschikbaar is, dan zal de controle ook niet mogelijk zijn, maar dit ligt dan aan de applicatie zelf en niet aan het protocol van de Yubikey. We gaan er voor het gemak vanuit dat als de applicatie beschikbaar is (en communicatie hiermee mogelijk), dat dan de OTP in principe ook met de applicatie gecommuniceerd kan worden en controle dus mogelijk is. Als de applicatie niet beschikbaar is of communicatie hiermee niet mogelijk, dan valt dit buiten dit onderdeel (en het Yubico protocol in zijn geheel) en dus buiten dit onderzoek.

De authenticiteiteis (geen onrechtmatige authenticatie) is ook van toepassing op dit onderdeel. Dit wordt deels al afgevangen bij het genereren en valideren van de OTP. Zelfs als iemand elke mogelijke waarde als OTP kan opsturen naar de applicatie (wat ook prima mogelijk is), dan zou hij zich nog niet (zomaar) als iemand anders moeten kunnen authenticeren. Belangrijk bij dit onderdeel is het eventueel kunnen onderscheppen van OTP's. Iemand zou bijvoorbeeld een OTP kunnen onderscheppen wanneer deze met de applicatie gecommuniceerd wordt en zich zo onrechtmatig als die persoon authenticeren. Hier moet rekening mee worden gehouden.

De belangrijkste beveiligingsaspecten zijn vertrouwelijkheid en integriteit. De vertrouwelijkheid is belangrijk omdat je in principe niet wil dat iemand zomaar OTP's kan onderscheppen. Deze zouden namelijk gebruikt kunnen worden om de cryptografie te kraken (en dus de geheime velden waarmee de OTP gegenereerd is te achterhalen) of om direct te gebruiken om onrechtmatig te authenticeren. Dit heeft dus alles te maken met de authenticiteiteis. De integriteit is belangrijk vanwege de robuustheideis. De OTP's moeten niet zomaar kunnen worden aangepast, dit vanwege de redenen genoemd bij het bespreken van deze eis.

Ook enigszins belangrijk is het aspect beschikbaarheid. Dit vanwege de beschikbaarheideis (en de redenen genoemd bij deze eis). Als het systeem niet beschikbaar is (bijvoorbeeld communicatie tussen Yubikey en applicatie is niet mogelijk), is controle dus ook niet mogelijk en dit is in strijd met de beschikbaarheideis. Echter valt het beschikbaar zijn van de applicatie en de communicatiemogelijkheden in principe buiten de Yubikey en dus buiten dit onderzoek.

De aspecten authenticiteit en onweerlegbaarheid zijn in dit geval nauwelijks belangrijk. De authenticiteit doet in dit geval niet ter zake en hoeft hier niet gecontroleerd te worden. Het hele Yubikey protocol is immers bedoeld om de authenticiteit te controleren (om te kunnen authenticeren) en dit gebeurt in een later stadium (bij het valideren van de OTP). Onweerlegbaarheid is niet belangrijk omdat er geen belangrijke acties worden uitgevoerd waarvan je zeker wilt weten dat een bepaald persoon ze gedaan heeft. Iedereen mag in principe een OTP opsturen en precies achterhalen wie dit gedaan heeft (voordat de authenticiteit gecontroleerd is), doet niet ter zake (en is ook lastig aangezien de authenticiteit op dit punt nog niet vaststaat).

5.3 Beveiligingsanalyse

Dit onderdeel is vanuit de Yubikey qua beveiliging eigenlijk niet echt interessant. Het Yubico OTP-protocol heeft op dit onderdeel nauwelijks tot geen specifieke beveiligingsmaatregelen en dwingt ook niets af. Toch zijn er wel een aantal mogelijke aanvallen op dit onderdeel en zijn er (dus) ook een aantal beveiligingseisen en -aspecten van toepassing. Tegen deze mogelijke aanvallen is uiteraard (voor een groot deel) wel beveiligd in het Yubico OTP-protocol, maar deze beveiliging vindt in andere onderdelen (met name de onderdelen genereren OTP en valideren OTP) plaats. Omdat de beveiligingsmaatregelen (deels) bedoeld zijn om te beschermen tegen mogelijk aanvallen op dit onderdeel, zullen we ze wel hier bespreken. Daarbij lichten we toe hoe ze beschermen tegen de mogelijke aanvallen op dit onderdeel.

Bij het bespreken van dit onderdeel zullen we ervan uitgaan dat de applicatie een webapplicatie is en

de communicatie dus (deels) over het internet verloopt. We doen dit omdat de Yubikey hoofdzakelijk bedoelt is om te gebruiken binnen (verschillende) webapplicaties. Bovendien kunnen we stellen dat als de Yubikey veilig gebruikt kan worden voor webapplicaties, deze zonder meer ook veilig gebruikt kan worden voor lokale applicaties of applicaties binnen het interne netwerk, aangezien de communicatie dan (hoogstwaarschijnlijk) veiliger zal zijn, dan wanneer dit over het publieke internet gaat. Verder kan bij het bespreken van de communicatie met een webapplicatie ook de enige beveiligingsmaatregel van het Yubico OTP-protocol specifiek voor dit onderdeel aan bod komen, namelijk het aanraden om van een met SSL beveiligde verbinding gebruik te maken (via het HTTPS protocol).

Om de belangrijkste eis, de robuustheideis, op dit onderdeel te garanderen is het belangrijk dat de verzonden OTP ongewijzigd van Yubikey naar webapplicatie gaat. Als een aanvaller de OTP (ongedetecteerd) kan wijzigen, dan kan deze ervoor zorgen dat een rechtmatige authenticatiepoging mislukt. Dit is een vorm van een man-in-the-middle aanval welke we dan ook zullen bespreken.

Zoals al vermeld, zullen we de beschikbaarheideis op dit onderdeel buiten beschouwing laten, aangezien dit buiten de scope van dit onderzoek valt. We zullen dus ook geen mogelijke aanvallen op dit onderdeel bespreken. Wel zullen we de authenticiteiteis bespreken. Op dit onderdeel is een aantal aanvallen mogelijk om de authenticiteiteis in het geding te brengen. De mogelijke aanvallen zijn een reflection aanval, een replay aanval en diverse man-in-the-middle aanvallen.

5.3.1 Reflection aanval

Bij een reflection aanval maakt een aanvaller gebruik van meerdere sessies om zich onrechtmatig te kunnen authenticeren (en dus de authenticiteiteis te ondermijnen). Dit soort aanvallen is in principe mogelijk wanneer gebruik wordt gemaakt van symmetrische encryptie om te authenticeren en met name wanneer beide partijen zich authenticeren (two-way authenticatie).

Reflection aanvallen zijn door de aard van het Yubico OTP-protocol niet mogelijk. Dit komt voornamelijk doordat de startende partij (de partij die zich wil authenticeren) zich als eerste (eventueel zelfs als enige) authenticeert. Het doel van het protocol is immers de gebruiker laten authenticeren bij de webapplicatie en niet andersom. Hierdoor heeft het starten van twee sessies geen zin, omdat authenticatie gelijk moet plaatsvinden en de andere partij anders niet verder gaat (en ook geen gevoelige informatie weggeeft).

5.3.2 Replay aanval

Een replay aanval is het eenmalig onderscheppen van informatie en deze informatie vervolgens gebruiken (herhalen) om te authenticeren. Het standaard wachtwoord is hier erg gevoelig voor. Onderschep namelijk eenmalig het wachtwoord en gebruik het daarna net zo vaak als je wilt door het simpelweg te herhalen (totdat het veranderd wordt uiteraard). Het Yubico OTP-protocol is bedoeld als vervanging of uitbreiding op het standaard wachtwoord en een belangrijk aspect hierbij is het voorkomen van replay-aanvallen (waar het standaard wachtwoord wel heel gevoelig voor is).

Een OTP kan dus onderschept worden tijdens de communicatie (bijvoorbeeld met een keylogger), maar niet succesvol gebruikt worden. Als een OTP herhaald wordt, dan zal de validatieserver melden dat het om een replay aanval gaat en zal de authenticatiepoging mislukken. Dit komt niet door bepaalde beveiligingsmaatregelen specifiek voor dit onderdeel (maar door beveiliging bij het genereren en valideren van de OTP), maar wordt toch hier besproken omdat de beveiliging wel bedoeld is om een aanval op dit onderdeel tegen te gaan. Om deze aanvallen tegen te gaan maakt de Yubikey gebruik van tellerwaarden (sessieteller en sessiegebruikteller) welke iedere keer verhoogd worden. De validatieserver houdt de laatst geziene tellerwaarden bij. Als er een OTP wordt ingediend met lagere waarden voor de tellers, dan wordt deze afgekeurd. Zodoende zijn replay-aanvallen bij het Yubico OTP-protocol in principe dus niet mogelijk.

Belangrijk aspect hierbij is de integriteit van de OTP, met name dat de tellerwaarden in de OTP niet zomaar aangepast (verhoogd) kunnen worden. De gebruikte cryptografie bij het genereren van de OTP (in combinatie met de CRC-checksum) zorgt dat dit niet mogelijk is (zoals besproken in hoofdstuk 7). Het is om deze aanval te voorkomen dus niet noodzakelijk dat de applicatie gebruik maakt van een beveiligde HTTPS verbinding. Het zou echter wel nog veiliger zijn (aangezien het dan lastiger wordt de OTP eenmalig te onderscheppen en het bovendien kan helpen voorkomen dat andere gegevens, zoals gebruikersnaam en standaard wachtwoord, onderschept worden).

5.3.3 Man-in-the-middle aanvallen

Bij een man-in-the-middle aanval gaan we ervan uit dat een aanvaller communicatie tussen twee partijen kan onderscheppen, kan lezen en/of kan wijzigen. Dit zorgt voor meerdere mogelijke aanvallen op de diverse beveiligingseisen. We zullen de mogelijke aanvallen opsplitsen en bespreken. Om het bespreken makkelijker te maken zullen we gebruik maken van voorbeelden. Bij deze voorbeelden spelen drie partijen een rol: Alice (de persoon die zich wenst te authenticeren met de Yubikey), Bob (de webapplicatie) en Eve (de aanvaller). Bij het bespreken zullen we eerst de situatie bespreken waarbij de verbinding onbeveiligd is en daarna kort toelichten hoe de situatie verandert (de beveiliging beter wordt, of beter gemaakt kan worden) als een beveiligde verbinding gebruikt wordt (zoals ook aangeraden wordt bij het Yubico OTP-protocol).

Wijzigen OTP

Als een aanvaller de OTP bij het versturen kan wijzigen, dan is het mogelijk een rechtmatige authenticatiepoging te blokkeren (en dus de robuustheideis te ondermijnen). Een manier om dit te doen is door het uitvoeren van een man-in-the-middle aanval. Voorbeeld: Eve onderschept de communicatie (en dus de OTP) van Alice naar Bob en wijzigt de OTP voordat deze wordt doorgestuurd. In dit geval krijgt Bob een gewijzigde OTP binnen welke bij het ontsleutelen andere waarden oplevert. Als Bob deze OTP laat valideren, dan komt terug dat de OTP niet geldig is en wordt Alice (en Eve) de toegang geweigerd.

Deze aanval is zeer gemakkelijk uit te voeren. Het Yubico OTP-protocol biedt hier geen bescherming tegen. Bob kan in principe ook niet direct detecteren dat de OTP onderweg gewijzigd is (het protocol biedt bij dit onderdeel geen beveiliging om de integriteit te garanderen). Wel valt bij het valideren van de OTP (hoogst waarschijnlijk) op dat de checksum over de velden niet klopt en dus dat er ergens iets mis is gegaan. Het is dan echter nog steeds niet mogelijk om te controleren of dit komt doordat de OTP is gegenereerd door een aanvaller (dus dat Alice de aanvaller zou zijn) of dat dit komt door een toevallige communicatiefout of dat er sprake is van een man-in-the-middle aanval. Het is voor Eve in principe niet mogelijk ook de checksum in de juiste waarde te veranderen door de gebruikte cryptografie (zie sectie 4.3.1). Eve kan Alice niet voor al te lange tijd blijven blokkeren, omdat het dan uiteindelijk (door het gebruik van de checksum) wel zal opvallen dat er iets mis is en er waarschijnlijk sprake is van een aanval.

Indien gebruik wordt gemaakt van een beveiligde (HTTPS) verbinding bij het versturen van deze OTP is de aanval lastiger uit te voeren. Het is immers lastiger te bepalen welke data veranderd moeten worden om de OTP te wijzigen (doordat alle data versleuteld zijn). Ook biedt HTTPS integriteit en Bob merkt dus gelijk als iemand de OTP bij de communicatie gewijzigd heeft. De aanval kan dus in principe wel uitgevoerd worden, maar dit zal lastiger zijn en zal eerder (gelijk) gedetecteerd worden.

Samenvattend kan worden gesteld dat het volledig tegen gaan van deze aanval niet mogelijk is. Het valt immers niet te voorkomen dat de OTP onderschept en gewijzigd wordt. Wel kunnen er maatregelen genomen worden om deze aanval (sneller) te kunnen detecteren, zodat er achteraf tegen ingegrepen kan worden.

Een nog veel schadelijkere aanval zou zijn om op een gerichte manier de OTP te wijzigen. In dit geval gaat het om het wijzigen van de sessieteller in een hogere waarde. De validatieserver zal de hoogst geziene waarde voor deze teller onthouden en alle OTP's met een lagere waarde afkeuren (zie hoofdstuk 7). Als een aanvaller een OTP indient met een hele hoge waarde als sessieteller, dan kan de Yubikey daarna tijdelijk of zelfs permanent onbruikbaar worden (en dus de robuustheideis schenden). Deze aanval is echter om dezelfde reden als hiervoor niet mogelijk. Door het gebruik van AES-versleuteling in combinatie met de checksum, kan de aanvaller de OTP niet op een gerichte manier wijzigen. Als een aanvaller dit toch probeert (als hij überhaupt al weet wat precies te wijzigen om de sessieteller te verhogen), dan valt dit gelijk op door de afwijkende checksum en zal de validatieserver de hogere waarden negeren.

Onrechtmatig authenticeren

Een andere mogelijke aanval is het onrechtmatig authenticeren en dus het schenden van de authenticiteiteis. Een voorbeeld van deze aanval gaat als volgt: Alice wil zich bij Bob authenticeren en verstuurt de OTP. Eve onderschept deze OTP en stuurt deze zelf door naar Bob (en zorgt ervoor dat het bericht

van Alice niet aankomt bij Bob). Zodoende heeft Eve zich geauthenticeerd als Alice. Het Yubico OTP-protocol biedt hier geen bescherming tegen. De vraag is echter of de aanval echt waarschijnlijk is. Eve kan namelijk het proces niet starten en moet dus maar wachten tot Alice zich een keer wil authenticeren bij Bob. Als Alice dat dan een keer doet, dan moet Eve gelijk het bericht onderscheppen en tevens blokkeren (anders authenticeert Alice zich zelf al met de OTP en kan Eve deze OTP niet meer gebruiken, het is immers een One Time Password). Er komt dus wel wat bij kijken, maar het is zeker niet onmogelijk en dus iets om rekening mee te houden. Het Yubico OTP-protocol is kwetsbaar voor dit type aanvallen.

Een belangrijk aspect bij deze aanval is de vertrouwelijkheid. Het Yubico OTP biedt geen vertrouwelijkheid bij het sturen van de OTP. Wel raadt het protocol aan om gebruik te maken van een beveiligde verbinding en waarschijnlijk vanwege deze reden. Als er gebruik wordt gemaakt van een beveiligde verbinding is de aanval niet meer mogelijk. Eve kan immers de OTP die Alice naar Bob verstuurt niet meer lezen (wel onderscheppen, maar ze heeft geen idee van de inhoud). Ook wordt het blokkeren van de OTP lastiger omdat Eve niet precies weet wat de inhoud is van elk bericht (een optie is natuurlijk wel om gewoon simpelweg alle berichten te onderscheppen en blokkeren, maar dat gaat waarschijnlijk opvallen). Indien het Yubico OTP-protocol gebruikt wordt in combinatie met het HTTPS protocol (of een andere manier om voor vertrouwelijkheid te zorgen), dan is deze aanval niet mogelijk.

Een zeer belangrijk probleem wat zich nu nog wel voordoet is wanneer een andere applicatie niet gebruik maakt van een beveiligde verbinding. Stel bijvoorbeeld dat Alice gebruik maakt van twee applicaties, Bob en Charlie welke beiden gebruik maken van het Yubico OTP-protocol. Met het Yubico OTP-protocol is het in principe mogelijk veilig te authenticeren bij meerdere applicaties met dezelfde Yubikey. Stel nou dat Bob zich tegen deze aanval wil beschermen en dus gebruik maakt van een beveiligde verbinding. Helaas is het dan nog steeds mogelijk de aanval op Bob uit te voeren, gebruik makend van communicatie tussen Alice en Charlie. Een specifiek voorbeeld: Eve onderschept en blokkeert een OTP verstuurd van Alice naar Charlie. Eve heeft geen interesse in Charlie, maar gebruikt de OTP nu om zich te authenticeren bij Bob als Alice. Bob heeft er helemaal geen weet van dat de OTP oorspronkelijk bedoeld was voor Charlie en accepteert de authenticatiepoging dus. Het komt er dus op neer dat ook al heb je als applicatie zijnde zelf de beveiliging op orde, dan is deze aanval in principe alsnog mogelijk als een andere applicatie dit niet heeft. Een oplossing tot dit specifieke voorbeeld en deze aanval in het algemeen is het gebruik van de challenge-response modus van de Yubikey, maar dat valt buiten dit onderzoek.

Lezen OTP's

Wanneer niet gebruik wordt gemaakt van een beveiligde verbinding is het lezen van OTP's erg gemakkelijk. In de eerste plaats kan je weinig met deze informatie (tenzij je het gebruikt op de hierboven beschreven manier), maar het kan wellicht wel helpen bij het uitvoeren van andere aanvallen. Denk bijvoorbeeld aan het kraken van de cryptografie. Wanneer een aanvaller alle inkomende communicatie op de server van een webapplicatie afluistert, dan is het mogelijk een zeer groot aantal OTP's te onderscheppen. Dit kan helpen de gebruikte cryptografie sneller te kraken. Bijvoorbeeld wanneer gebruik wordt gemaakt van de rainbow of codebook aanval (zie sectie 4.3.1).

Als er wel gebruik gemaakt wordt van een beveiligde verbinding, dan kan de aanvaller de OTP's niet lezen en zal deze aanval dus niet werken. Zodoende is ook bij deze aanval het aspect vertrouwelijkheid van grote invloed.

Hoofdstuk 6

Communicatie tussen (web)applicatie en validatieserver

De applicatie moet weten of de ontvangen OTP geldig is of niet. Over het algemeen kan de applicatie dit niet zelf controleren en moet dit gebeuren door een validatieserver. Dit onderdeel draait om de communicatie met de validatieserver. We gaan er in deze situatie vanuit dat de OTP gevalideerd wordt door de standaard Yubico validatieserver. Dit onderdeel is dus verantwoordelijk voor een (beveiligde) communicatie tussen applicatie en validatieserver. Deze communicatie gaat in twee richtingen (de applicatieserver stuurt de OTP naar de validatieserver en de validatieserver stuurt terug of deze geldig is of niet).

6.1 Werking

Het communiceren met de validatieserver gebeurt met het "Validation Protocol Version 2.0" ontwikkeld door Yubico. Dit protocol beschrijft hoe verzoeken naar de validatieserver gestuurd kunnen worden, hoe deze verzoeken eruit moeten zien en hoe het antwoord er uit ziet. Verder beschrijft het nog een aantal veiligheidsvoorschriften zoals het gebruik van handtekeningen en hoe deze te controleren.

Zoals in het protocol beschreven staat, gaat de communicatie door middel van een HTTP GET verzoek. De communicatie kan eventueel ook over HTTPS lopen (voor extra beveiliging). Ter beveiliging wordt over elk bericht (verzoek en antwoord) een soort digitale handtekening gegenereerd en meegestuurd. Voor de applicatie is dit optioneel (de handtekening wordt gecontroleerd door de validatieserver als deze wordt meegestuurd, maar anders niet). De validatieserver ondertekent de antwoorden altijd. De applicatie moet de handtekening controleren of het SSL-certificaat van de server verifiëren (of beiden). De keuze wordt in principe overgelaten aan de applicatie, maar minstens één van de twee moet gebeuren volgens het Yubico OTP-protocol. Het maken (en controleren) van de handtekeningen gebeurt met HMAC-SHA-1 op basis van een shared secret (gedeeld geheim). Dit geheim is van tevoren tussen applicatie en validatieserver overeengekomen. De handtekening dient voor zowel integriteit en authenticiteit van de berichten. Het idee is namelijk dat alleen met het geheim de juiste handtekening over het bericht gegenereerd kan worden. Als de handtekening afwijkt, dan staat of de authenticiteit of integriteit van het bericht niet vast (het is in principe niet te achterhalen welke van de twee) en wordt het bericht verworpen.

Uiteindelijk gaat een verzoek tot validatie van een OTP als volgt te werk. De applicatie doet een validatieverzoek en stuurt de OTP samen met een aantal (optionele) andere gegevens naar de validatieserver. De server valideert de OTP en stuurt terug of deze geldig is (of eventueel een foutcode) samen met nog eventuele andere informatie over de OTP. De applicatie gebruikt deze informatie vervolgens om een eindoordeel te vellen over de OTP (wat meestal inhoudt de OTP accepteren het antwoord de status 'OK' had of anders de OTP verwerpen).

6.1.1 Het verzoek

Het verzoek van de applicatie naar de validatieserver ziet er als volgt uit (bevat de volgende velden):

- 'id': Het ID van de applicatie (nodig voor het ophalen van het juiste gedeelde geheim om de handtekening te controleren).
- 'otp': De OTP om te laten valideren.
- 'h' (niet verplicht): De HMAC-SHA-1 handtekening over het verzoek met het gedeeld geheim.
- 'timestamp' (niet verplicht): Geef waarde 1 om timestamp en tellerwaarden uit de OTP op te halen en terug te geven in het antwoord.
- 'nonce': Een 16 tot 40 karakter grote willekeurige, unieke nonce (number used once).
- 'sl' (niet verplicht): Geeft aan hoeveel (percentage) van de sync-verzoeken beantwoord moeten worden (weiger het verzoek indien dit percentage niet behaald wordt). Hierover later meer in hoofdstuk 7.
- 'timeout' (niet verplicht): Geeft aan hoeveel seconden maximaal te wachten op antwoord bij het syncen.

6.1.2 Het antwoord

Het antwoord van de validatieserver naar de applicatie ziet er als volgt uit (bevat de volgende velden):

- 'h': De HMAC-SHA-1 handtekening over het antwoord met het gedeeld geheim.
- 't': De tijd van het versturen van het antwoord.
- 'status': Status indicator welke aangeeft of de OTP geldig is of niet (en waarom niet).
- 'timestamp' (alleen als timestamp=1 in het verzoek): De timestampwaarde uit de OTP.
- 'sessioncounter' (alleen als timestamp=1 in het verzoek): De sessietellerwaarde uit de OTP.
- 'sessionuse' (alleen als timestamp=1 in het verzoek): De sessiegebruiktellerwaarde uit de OTP.
- 'sl': Hoeveelheid servers (percentage) welke antwoord gaven op het sync-verzoek.
- 'otp': De OTP waarop het antwoord betrekking heeft.
- 'nonce': De nonce welke was meegestuurd met het verzoek.

De laatste twee velden (otp en nonce) worden overigens niet genoemd in de beschrijving van het "Validation Protocol Versie 2.0" [24]. Uit praktijktests is echter gebleken dat de validatieserver deze waarden wel meestuurt. Dit is belangrijk om bepaalde aanvallen tegen te gaan (zie sectie 6.3.3) en het is daarom vreemd dat deze velden niet genoemd worden in de beschrijving.

6.1.3 De handtekening

Om precies te zijn gaat het genereren van de handtekening (veld h) als volgt (zowel voor verzoek als antwoord):

1. Sorteer de verzameling van veldnaam en waarde alfabetisch (op veldnaam) met uitzondering van het veld h zelf.
2. Construeer één enkele regel door de gesorteerde veldnaam/waarde-paren achter elkaar te zetten, gescheiden met het karakter '&' en een '=' karakter tussen veldnaam en bijbehorende waarde. Voorbeeld: "a=1&b=2&c=3".
3. Pas het HMAC-SHA-1 algoritme toe op de zojuist geconstrueerde lijn, waarbij het gedeeld geheim als invoersleutel van het algoritme gebruikt wordt (zie [7] voor meer details over HMAC).
4. Pas base64 encoding toe op het resultaat volgens RFC 4648. Voorbeeld: "t2ZMtKeValdA+H0jVpj3LIichn4=".

5. Het resultaat is de sleutel en dient aan het bericht te worden toegevoegd bij veldnaam 'h'.

Zoals gemeld dient deze handtekening voor integriteit en authenticiteit van het bericht. Het valt op dat gekozen is voor het SHA-1 hash algoritme welke over het algemeen als gebroken wordt beschouwd. Er zijn in principe ook betere alternatieven voor het genereren van handtekeningen (denk aan asymmetrische encryptie met een public en private key). We zullen dit bespreken bij de beveiligingsanalyse.

6.2 Beveiligingsaspecten

De beveiligingseisen authenticiteit en robuustheid zijn erg belangrijk met betrekking tot dit onderdeel. De authenticiteiteis is van toepassing omdat de validatieserver via dit onderdeel (dit kanaal) antwoordt of een OTP geldig is of niet. Dit antwoord mag dus niet zomaar (ongedetecteerd) kunnen worden aangepast. Anders wordt het voor een eventuele aanvaller heel makkelijk om onrechtmatig te authenticeren (doe een authenticatiepoging met wat willekeurige data, onderschep het antwoord van de validatieserver, pas het aan en je bent binnen). Deze beveiligingseis speelt dus een belangrijke rol bij dit onderdeel en de beveiliging moet zorgen dat deze eis gewaarborgd blijft.

Hetzelfde geldt voor de robuustheideis. Als een kwaadwillende de antwoorden van de validatieserver kan onderscheppen en kan aanpassen, dan kan deze rechtmatige authenticatiepogingen blokkeren. Behalve het antwoord van de validatieserver is ook het verzoek naar de validatieserver van belang. Als iemand dit verzoek ongedetecteerd kan bewerken, dan kunnen rechtmatige authenticatiepogingen geweigerd worden.

Ook de beschikbaarheideis is zeker van belang (zij het in iets minder mate). Als communicatie tussen de applicatie en validatieserver niet mogelijk is, is controle ook niet mogelijk. Als gebruik wordt gemaakt van de standaard Yubico validatieserver, dan zal een internetverbinding dus vereist zijn. Bij problemen met de internetverbinding (om wat voor reden dan ook), zal de eis in het geding komen. De vraag is hoeveel hiertegen te doen valt binnen het Yubico OTP-protocol. Yubico heeft echter wel degelijk een aantal maatregelen genomen om beter te voldoen aan de eis, welke we dan ook zullen bespreken.

Wat betreft de beveiligingsaspecten zijn vertrouwelijkheid, integriteit, authenticiteit en beschikbaarheid allen belangrijk. Vertrouwelijkheid is om dezelfde reden belangrijk als bij het onderdeel communiceren OTP met applicatie, namelijk omdat het (op grote schaal) onderscheppen van OTP's op den duur zou kunnen leiden tot het kraken van de gebruikte cryptografie. Dit aspect speelt hier nog een grotere rol, omdat de validatieserver ook delen van de gebruikte plaintext van de OTP kan teruggeven (hierover meer in sectie 6.3.5). De integriteit is belangrijk om de authenticatie en robuustheideis te kunnen garanderen (zoals hierboven besproken is). Authenticiteit is met name belangrijk omdat de applicatie zeker moet weten met de validatieserver te communiceren en niet met een andere instantie (aanvaller). Dit ook om de authenticiteit en robuustheideis te kunnen garanderen. De beschikbaarheid is belangrijk om de beschikbaarheideis te kunnen waarborgen. Als het systeem niet beschikbaar is, dan is controle niet mogelijk en komt deze eis dus in het geding.

Het aspect onweerlegbaarheid doet (wederom) minder ter zake. Er zijn geen zaken waarvan onweerlegbaar vast dient te staan dat ze door een bepaalde partij zijn uitgevoerd. Het maakt in principe ook niet veel uit van wie de validatieverzoeken komen. Iedereen mag deze immers indienen. Vandaar dat het ondertekenen van validatieverzoeken (met de gedeelde sleutel) ook niet verplicht is.

6.3 Beveiligingsanalyse

In het antwoord van de validatieserver staat uiteindelijk of de OTP geldig is of niet. Een voor de hand liggende mogelijke aanval is dan ook om dit antwoord aan te passen en zodoende onrechtmatig te authenticeren. Dit is een vorm van een man-in-the-middle aanval, bedoeld om de authenticiteiteis te ondermijnen. We zullen deze aanval dus bespreken. Dit is niet de enige mogelijke aanval op de authenticiteiteis. Er zijn ook andere aanvallen, welke we ook zullen bespreken. Denk hierbij aan het kraken van de handtekening, voordoen als validatieserver, een replay aanval, het blokkeren van validatieverzoeken (zodat een ogenschijnlijk gebruikte OTP eigenlijk niet gebruikt is en dus nog geldig blijft) of een man-in-the-middle aanval met de focus op het verzoek (pas het verzoek aan, zodat minder strenge eisen gelden en dus wellicht een ongeldige OTP toch geaccepteerd wordt). We zullen al deze aanvallen behandelen.

Een andere belangrijke eis op dit onderdeel is de robuustheideis. Er is een aantal aanvallen denkbaar om deze eis te ondermijnen. Een mogelijkheid is bijvoorbeeld om het antwoord van de validatieserver aan te passen (immers als je het antwoord van niet geldig naar geldig kan wijzigen, ligt het voor de hand dat dit andersom ook kan). Een andere (waarschijnlijkere) mogelijkheid is het aanpassen van het verzoek. Dit kan door middel van een man-in-the-middle aanval en kan ertoe leiden dat rechtmatige authenticatiepogingen geblokkeerd worden (bijvoorbeeld als de OTP gewijzigd wordt in iets ongeldig, of als de timeout in een extreem lage waarde gewijzigd wordt).

Als laatste is er de beschikbaarheideis. Hier is één mogelijke (soort) aanval op, namelijk het blokkeren van de verbinding (dit kan op verschillende niveaus). Als de verbinding plat ligt en dus de verzoeken en/of de antwoorden niet aankomen, dan kan de OTP niet gecontroleerd worden en komt de beschikbaarheideis dus in het geding. We zullen deze aanval en de mogelijke gevolgen ervan bespreken.

Het Yubico OTP-protocol heeft binnen dit onderdeel een aantal keuzes en aanbevelingen qua beveiliging (net zoals in het vorige onderdeel de keuze was om van een HTTPS verbinding gebruik te maken of niet). Bij dit onderdeel gaat het om de volgende keuzes:

1. Wel of geen gebruik maken van een beveiligde verbinding (HTTPS).
2. Wel of niet ondertekenen van de verzoeken (het veld h, wat de handtekening over het verzoek bevat, is immers niet verplicht).
3. Wel of niet controleren van de handtekening op de antwoorden.
4. Wel of niet verifiëren van servercertificaat en dus authenticiteit van de server (indien gebruik wordt gemaakt van een HTTPS verbinding).

De eerste twee keuzen zijn volledig vrij en het Yubico OTP-protocol doet hier ook geen aanbevelingen voor, het biedt enkel de mogelijkheid het te doen. In beide gevallen geldt dat de beveiliging in principe beter is als het wel gebeurt. Bij de derde en vierde keus stelt het Yubico OTP-protocol wel een eis, namelijk dat of de handtekening over het antwoord van de validatieserver gecontroleerd moet worden of het servercertificaat moet geverifieërd worden (en dus gebruik worden gemaakt van een HTTPS verbinding). Echter controleert Yubico niet of applicaties dit ook daadwerkelijk doen en is het dus mogelijk dat er applicaties zijn die dit niet doen.

Merk overigens op dat de keuze tussen controleren handtekening of verifiëren servercertificaat (of beiden) verder niet uitmaakt voor de beveiliging. Beiden bieden immers zowel garantie van authenticiteit van de server en integriteit over het antwoord. Voor de handtekening geldt dit omdat er gebruik wordt gemaakt van een gedeeld geheim. Alleen met dit gedeelde geheim kan de juiste handtekening gegenereerd worden (en staat dus de authenticiteit vast) over het gehele bericht. Verder zal de handtekening niet meer overeenkomen als het antwoord gewijzigd wordt en staat dus ook de integriteit vast. Bij het verifiëren van de het servercertificaat geldt authenticiteit van de server door gebruik van het HTTPS/SSL protocol logischerwijs [6]. Verder garandeert het HTTPS protocol ook integriteit over het bericht [6]. Zodoende maakt het voor het bespreken van de beveiliging niet veel uit welke van de twee opties gekozen wordt.

De verschillende mogelijke keuzes leveren een aantal situaties op, welke van invloed zijn op de beveiliging en dus op hoe succesvol bepaalde aanvallen zijn. We zullen daarom voor elke aanval de verschillende situaties bespreken en hoe de kans op slagen beperkt wordt in een veiligere situatie. We beginnen met het bespreken van de basissituatie waarbij de beveiliging minimaal is en zullen eindigen bij de situatie waarin de beveiliging optimaal is. Om precies te zijn bespreken we de volgende situaties:

1. Geen HTTPS verbinding, niet ondertekenen van de verzoeken en niet controleren van de handtekeningen op de antwoorden. Deze situatie mag eigenlijk niet voorkomen volgens het Yubico OTP-protocol, maar dit is wel mogelijk en wordt dus toch besproken.
2. Geen HTTPS verbinding, wel ondertekenen van de verzoeken, maar niet controleren van de handtekeningen op de antwoorden. Ook deze situatie zou eigenlijk niet voor mogen komen.
3. Geen HTTPS verbinding, niet ondertekenen van de verzoeken, maar wel controleren van de handtekeningen. De minimale beveiliging welke wordt toegestaan door het protocol.

4. Geen HTTPS verbinding, wel ondertekenen van de verzoeken en controleren van de handtekeningen.

5. Wel HTTPS verbinding, niet ondertekenen van de verzoeken, niet controleren van de handtekeningen en niet verifiëren van servercertificaat. Deze situatie mag volgens het Yubico OTP eigenlijk ook niet voorkomen.

6. Wel HTTPS verbinding, niet ondertekenen van de verzoeken, maar wel controleren handtekening of verifiëren van servercertificaat.

7. Optimale beveiliging, dat wil zeggen: gebruik maken van HTTPS verbinding, ondertekenen van verzoeken. Verder controleren handtekening of verifiëren servercertificaat (of beiden).

Bij de verschillende situaties zijn verschillende beveiligingsaspecten wel of niet gewaarborgd. We zullen nu door middel van een tabel samenvatten welke aspecten wel en niet gewaarborgd zijn voor elke verschillende situatie.

	Vertrouwelijkheid		**Integriteit**		**Authenticiteit**	
	Verzoek	Antwoord	Verzoek	Antwoord	Applicatie	Server
Situatie 1	Nee	Nee	Nee	Nee	Nee	Nee
Situatie 2	Nee	Nee	Ja	Nee	Ja	Nee
Situatie 3	Nee	Nee	Nee	Ja	Nee	Ja
Situatie 4	Nee	Nee	Ja	Ja	Ja	Ja
Situatie 5	Ja	Ja	Ja	Ja	Nee	Nee
Situatie 6	Ja	Ja	Ja	Ja	Nee	Ja
Situatie 7	Ja	Ja	Ja	Ja	Ja	Ja

De verschillende keuzes en situaties zijn nu vastgesteld. We gaan nu over op het analyseren en bespreken van de beveiliging aan de hand van mogelijke aanvallen.

6.3.1 Kraken handtekening

In de situaties zonder HTTPS, wordt een handtekening gebruikt om authenticiteit en integriteit van op zijn minst het antwoord en eventueel het verzoek vast te stellen. Dit is van belang om diverse aanvallen tegen te gaan (zie secties 6.3.2 en 6.3.5). Het idee is dat alleen met het juiste geheim de juiste handtekening bij het bericht (verzoek of antwoord) gegenereerd kan worden. Dit aspect berust voor een groot deel op het gebruikte algoritme, in dit geval HMAC-SHA-1. De basis van dit algoritme is de hash-functie SHA-1. De sterkte van de handtekeningen hangt dus voor een groot deel af van de sterkte van SHA-1. Op het SHA-1 algoritme zijn diverse aanvallen mogelijk en het wordt dan ook al sinds enige tijd als gekraakt beschouwd [16]. De vraag is dus of de handtekeningen wel veilig genoeg zijn.

Er zijn diverse aanvallen op SHA-1 mogelijk. Het gaat in dit geval vaak wel om theoretische aanvallen met (nog steeds) een redelijke complexiteit (denk aan de orde van 2^{51} [11]). Het is dus niet mogelijk om SHA-1 zomaar even te kraken. Verder valt te betwijfelen hoe goed deze aanvallen toepasbaar zijn in de praktijk, bijvoorbeeld voor het vervalsen van een handtekening (op basis van HMAC-SHA-1) over een bericht. Daarbij komt nog eens dat HMAC-SHA-1 niet enkel bestaat uit het toepassen van SHA-1, maar twee keer SHA-1 en diverse XOR operaties worden toegepast. Wel moet gesteld worden dat HMAC-SHA-1 voor een groot deel qua beveiliging afhankelijk is van SHA-1, maar het betekent niet dat een theoretische aanval op SHA-1 direct toepasbaar is op HMAC-SHA-1. Hierdoor kunnen we in principe stellen dat het zomaar vervalsen van handtekeningen over de berichten op het moment nog niet praktisch haalbaar is. Het is wel iets om rekening mee te houden voor de toekomst, aangezien het niet ondenkbaar is dat er op kort termijn een meer praktische aanval op SHA-1 of HMAC-SHA-1 ontdekt wordt.

Een mogelijke oplossing voor deze beveiligingskwestie is het gebruik van HTTPS en verifiëren van het servercertificaat. Een andere mogelijke oplossing (waarmee ook de handtekeningen over het verzoek beter beveiligd zijn) is het gebruik van een ander protocol voor het maken van de handtekeningen. Denk bijvoorbeeld aan een systeem met asymmetrische cryptografie waarbij de privé sleutel gebruikt wordt voor het maken van de handtekening en de publieke sleutel voor het controleren van de handtekening. Waarom Yubico niet direct voor een dergelijke oplossing gekozen heeft is niet duidelijk.

In de rest van de beveiligingsanalyse op dit onderdeel zullen we ervan uitgaan dat de gebruikte methode voor het genereren van de handtekening sterk genoeg is en dus authenticiteit en integriteit van het bericht gegarandeerd is.

6.3.2 Voordoen als validatieserver

Een aanvaller zou zich tegenover de applicatie kunnen voordoen als validatieserver. Dit kan bijvoorbeeld door het verzoek van applicatie naar validatieserver te onderscheppen, niet door te sturen, maar hier zelf op te antwoorden. Zodoende kunnen de authenticiteit en robuustheideis geschonden worden. De aanvaller kan immers antwoorden dat een niet geldige OTP toch geldig is of juist andersom.

Deze aanval is alleen mogelijk als de authenticiteit van de validatieserver niet gewaarborgd is, oftewel in situaties waarbij geen controle van handtekening en geen verificatie van servercertificaat plaatsvindt (situaties 1, 2 en 5). In deze situaties is het vrij gemakkelijk de aanval uit te voeren en de applicatie kan in principe niet ontdekken dat deze helemaal niet met een (echte) validatieserver communiceert. Het is ook een zeer schadelijke aanval, want het wordt mogelijk twee eisen te schenden zonder veel moeite en ze zijn uit te voeren op elke gebruiker van die applicatie. Het lukt echter alleen maar in de situaties 1, 2 en 5. Al deze situaties zouden eigenlijk niet voor mogen komen. Als de applicatie zijn beveiliging dus op orde heeft en zich aan de eisen van het Yubico OTP-protocol houdt, dan is de aanval in principe niet uit te voeren en kan deze geen kwaad.

6.3.3 Replay aanval

De replay aanval kan worden uitgevoerd op het verzoek of op het antwoord. Het protocol is beschermd tegen een replay aanval uitvoeren op het verzoek (door het verzoek naar de validatieserver te herhalen). De validatieserver houdt namelijk bij welke OTP en nonce combinaties al voorbij zijn gekomen en neemt het verzoek niet in behandeling. Bovendien heeft de validatieserver de desbetreffende OTP al gezien en ook hierin (met behulp van de tellers) zitten maatregelen om replay-aanvallen te voorkomen.

Een replay aanval uitvoeren op het antwoord biedt echter meer mogelijkheden en de beveiliging hiertegen is minder goed. De aanval is in principe in de situaties (situaties 1, 2, 3 en 4) waarbij geen gebruik wordt gemaakt van een HTTPS verbinding makkelijk uit te voeren. In de overige situaties is de aanval niet uit te voeren. Het protocol maakt namelijk voor de encryptie gebruik van een tijdelijke sleutel, welke alleen binnen die sessie betekenis heeft. Als berichten later herhaald worden of in een andere sessie, dan is de oorspronkelijk sleutel niet meer beschikbaar en hebben de berichten dus geen betekenis.

We zullen nu de situatie bespreken waarbij de aanval (replay van het antwoord) wordt gebruikt om onrechtmatig te authenticeren. Het echter ook mogelijk om de aanvallen te gebruiken om rechtmatige authenticatie te blokkeren. Dit gaat vrijwel op dezelfde manier en zullen we dus niet apart bespreken.

De aanval is uit te voeren door eenmalig een antwoord van de validatieserver te onderscheppen waarin staat dat de OTP geldig is. Om dan vervolgens onrechtmatig te authenticeren dient de aanvaller een OTP in bij de applicatie zoals gebruikelijk. De applicatie zal dan een verzoek naar de validatieserver sturen. Dee aanvaller moet dit verzoek onderscheppen en als antwoord hierop het eerder onderschepte (geldige) antwoord van de validatieserver geven. De applicatie leest dit antwoord, controleert eventueel de handtekening (welke zal kloppen, want het antwoord is niet gewijzigd) en zal de OTP accepteren.

Deze aanval is een serieus probleem, aangezien door deze aanval de authenticiteiteis en robuustheideis in het geding komen. Dit terwijl het Yubico OTP-protocol er geen echte beveiliging tegen kent (of in ieder geval niet duidelijk specificeert in het protocol). Er is wel tegen te beveiligen door gebruik te maken van HTTPS, maar dit is in principe niet verplicht. Zodoende is deze aanval dus toepasbaar bij applicaties die gebruik maken van situaties 1, 2, 3 of 4 (waarvan situaties 3 en 4 gewoon zijn toegestaan binnen het Yubico OTP-protocol).

Er is echter wel een beveiliging mogelijk tegen deze aanval (in ieder geval sinds invoering van versie 2.0 van het Validation Protocol). De validatieserver stuurt namelijk ook de OTP en gebruikte nonce (de nonce gebruikt in het verzoek) mee terug bij het antwoord. Als de applicatie controleert of de ontvangen OTP en nonce in het verzoek overeenkomen met de verstuurde OTP en nonce (en het antwoord weigert als dit niet het geval is), dan is de aanval niet meer mogelijk. Voorwaarde is wel dat de handtekening wordt gecontroleerd, anders is de aanval alsnog mogelijk omdat de OTP en nonce dan kunnen worden aangepast (in situaties 1 en 2 is de aanval dan dus alsnog mogelijk, maar in situaties 3 en 4 niet meer).

We vermoeden dat de hierboven beschreven replay aanval in een redelijk aantal gevallen mogelijk is. Situatie 3 en 4 zijn namelijk gewoon toegestaan binnen het Yubico OTP-protocol en van de extra beveiliging (controleren van OTP en nonce) zullen niet alle applicaties gebruik maken. De extra beveiliging wordt immers niet genoemd in het Validation Protocol en is dus ook niet vereist of zelfs maar aangeraden. Bijvoorbeeld de PHP implementatie van de Web API (yubikeyPHPclass-0.96) doet deze extra controle dan ook niet (en deze implementatie wordt wel genoemd op de website van Yubico [26]). Als laatste is de invoering van het meesturen van de OTP en nonce in het antwoord vrij recent. De vorige versie van het Validation Protocol (versie 1.1) doet dit nog niet (deze versie heeft helemaal geen ondersteuning voor het meesturen van een nonce). In deze oudere versie is de hierboven beschreven replay aanval dus zeker mogelijk in de vier genoemde situaties. Om deze redenen verwachten wij dat een redelijk aantal applicaties gevoelig zal zijn voor deze (relatief simpel uit te voeren, maar toch krachtige) aanval. Wel moeten we vermelden dat beveiligingen hiertegen gemakkelijk is (maak bijvoorbeeld wel gebruik van een HTTPS verbinding of controleer wel op de OTP en nonce), maar dit wordt helaas niet in het protocol vermeld.

6.3.4 Blokkeren validatieverzoeken

Een mogelijke aanval is het blokkeren van validatieverzoeken. In dit geval niet met als doel de beschikbaarheideis te ondermijnen (over dit punt later meer), maar om de authenticiteiteis te ondermijnen. Als een gebruikte OTP immers niet bij de validatieserver aankomt, kan hij alsnog gebruikt worden (totdat er een nieuwere OTP van de desbetreffende Yubikey wel bij de validatieserver komt tenminste). Een mogelijke aanval is dus een validatieverzoek vanaf een applicatie te onderscheppen, de OTP uit te lezen en te gebruiken bij dezelfde of een andere applicatie om te authenticeren als de desbetreffende gebruiker. Deze aanval is mogelijk bij alle situaties waarbij geen HTTPS gebruikt wordt (anders kan de OTP namelijk niet worden uitgelezen). De aanval lijkt sterk op de man-in-the-middle aanval bij de communicatie tussen Yubikey en applicatie (zie sectie 5.3.3) en heeft dezelfde gevolgen. Ook deze aanval is in principe uit te voeren bij applicaties welke wel HTTPS gebruiken als de gebruikers ook een andere applicatie gebruiken zonder HTTPS (OTP's zijn namelijk niet specifiek voor één applicatie bedoeld en kunnen dus elders gebruikt worden dan waar ze onderschept waren).

Er is geen beveiliging tegen deze aanval mogelijk. Het Yubico OTP-protocol biedt immers geen garantie dat de validatieverzoeken ook daadwerkelijk bij de validatieserver aankomen. De aanval is dus in principe mogelijk, alhoewel het wellicht wel opvalt. Het lukt de rechtmatige eigenaar van de Yubikey immers niet om zich te authenticeren, omdat zijn pogingen geblokkeerd worden. Het ligt in de lijn der verwachtingen dat hij het net zolang blijft proberen tot het wel lukt (en dus alle eerdere OTP's die nog geldig waren in één klap ongeldig maakt) of dat hij contact opneemt met de helpdesk van de applicatie en/of Yubikey, waardoor zij maatregelen kunnen nemen (maar dit kan natuurlijk al te laat zijn).

6.3.5 Man-in-the-middle aanvallen

Er zijn diverse man-in-the-middle aanvallen mogelijk op dit onderdeel.Voor alle man-in-the-middle aanvallen geldt dat ze proberen het verzoek of het antwoord te lezen of te wijzigen. Hierdoor zijn ze enkel mogelijk in de situaties waarbij geen HTTPS gebruikt wordt. HTTPS zorgt immers voor zowel vertrouwelijkheid als integriteit [6]. Zodoende zijn de aanvallen hooguit in de situaties 1, 2, 3 en 4 mogelijk.

We zullen nu de verschillende aanvallen één voor één bespreken en kort toelichten. Ook zullen we behandelen in welke situaties een aanval juist wel of juist niet mogelijk is.

Lezen verzoek

Het lezen van het verzoek is mogelijk in alle situaties zonder HTTPS. In principe levert het lezen van het verzoek niet direct een gevaar op, aangezien het voor een tweede keer indienen van een OTP over het algemeen niet werkt. Het kan echter wel gebruikt worden bij het (later) uitvoeren van een aanval waarbij validatieverzoeken geblokkeerd worden (zie subsectie 6.3.4). Ook kan het gebruikt worden voor het onderscheppen van veel OTP's (ciphertexten) voor het uitvoeren van een aanval op de encryptie (zie sectie 4.3.1). Het Yubico OTP-protocol biedt geen vertrouwelijkheid. De enige bescherming hiertegen is dus gebruik maken van HTTPS (maar zelfs dan kan de applicatie nog kwetsbaar zijn doordat een andere applicatie, met dezelfde gebruiker, geen HTTPS gebruikt).

Wijzigen verzoek

Het verzoek kan (ongedetecteerd) gewijzigd worden in situaties waarbij er geen handtekening wordt meegestuurd over het verzoek en geen gebruik wordt gemaakt van HTTPS. Het gaat dus om de situaties 1 en 3. Belangrijk is daarbij om op te merken dat situatie 3 is toegestaan binnen het Yubico OTP-protocol en het dus in de lijn der verwachting ligt dat de aanvallen (in ieder geval) op enkele applicaties mogelijk zullen zijn.

We zullen beginnen met aanvallen met als doel de authenticiteiteis te schenden. Deze zijn vrij speculatief en zullen op zichzelf niet werken, maar kunnen gecombineerd worden met andere aanvallen om zodoende toch effect te hebben. Het gaat om het volgende:

- Timestampveld wijzigen. Wijzig het veld timestamp in de waarde 1 en de validatieserver geeft extra informatie terug over de OTP, namelijk wat de interne tellerwaarden en timestampwaarde was. Dit kan waardevolle informatie zijn (welke later uit het antwoord gelezen kan worden). De informatie zou bijvoorbeeld gebruikt kunnen worden bij het kraken van de cryptografie (zie sectie 4.3.1).
- Sl veld wijzigen. Wijzig het sl veld in een zeer kleine waarde en de validatieserver vereist minder sync-antwoorden en is dus minder zeker of de OTP echt geldig is of niet. Deze aanval kan gebruikt worden in combinatie met een aanval op de validatieserver (met name het sync-proces, zie sectie 7.3.2) om een ongeldige OTP toch goed te laten keuren.
- Timeoutveld wijzigen. Het timeoutveld zou gewijzigd kunnen worden in een zeer grote waarde. Dit betekent dat de validatieserver langer op sync-antwoorden wacht, wat een aanvaller mogelijk meer tijd geeft.

Er zijn ook aanvallen mogelijk om de robuustheideis te ondermijnen. Het gaat om de volgende aanvallen:

- OTP-veld wijzigen. Wanneer het OTP-veld gewijzigd wordt, dan zal de OTP hoogstwaarschijnlijk niet geaccepteerd worden door de validatieserver en dus de authenticatiepoging mislukken. Dit kan eventueel wel gedetecteerd worden door de applicatie, gezien de validatieserver (sinds validation protocol versie 2.0) de meegestuurde OTP ook weer terugstuurt. De kans bestaat dat applicaties hier niet op controleren (het is immers niet verplicht).
- Sl veld wijzigen. Wijzig het sl veld in een zeer grote (wellicht onrealistische waarde, zoals bijvoorbeeld meer dan 100%) en de validatieserver zal (wellicht) niet genoeg sync-antwoorden binnen krijgen en een OTP dus ten onrechte afkeuren.
- Timeoutveld wijzigen. Verlaag het timeoutveld in een zeer kleine waarde (misschien zelfs 0) en de validatieserver zal geen of te weinig sync-antwoorden binnen krijgen en dus de OTP ten onrechte afkeuren.
- Combinatie. Een combinatie (zeer grote sl en zeer kleine timeout) is natuurlijk bijzonder effectief en kan ertoe leiden dat elke OTP in de validatieverzoeken ten onrechte wordt afgekeurd.

Lezen antwoord

Voor deze aanval geldt ongeveer hetzelfde als voor de lezen verzoek aanval 6.3.5 en is ook in dezelfde situaties mogelijk. De aanval kan gebruikt worden om later een replay aanval uit te voeren (zie subsectie 6.3.3). Ook kan de aanval gebruikt worden voor het kraken van de encryptie. Dit geldt bij deze aanval zelfs nog sterker, aangezien ook delen van de plaintext (tellerwaarden en timestampwaarde) kunnen worden meegestuurd. Dit is het geval als het veld timestamp van het verzoek de waarde 1 heeft (of een aanvaller zorgt dat dit gebeurt, zoals beschreven in subsectie 6.3.5).

Wijzigen antwoord

Het wijzigen van het antwoord is enkel mogelijk in de situaties waarbij geen validatie van de handtekening en geen HTTPS verbinding gebruikt wordt (situaties 1, 2 en 5). Het gaat dus enkel om situaties die volgens het Yubico OTP-protocol niet voor mogen komen.

Als het antwoord op een validatieverzoek gewijzigd kan worden, dan is het erg gemakkelijk om de authenticiteiteis of robuustheideis te ondermijnen. Een aanvaller kan dan immers het statusveld aanpassen in elke waarde die hij wilt en dus onrechtmatige authenticatiepogingen laten accepteren of rechtmatige authenticatiepogingen blokkeren. Dit zou een zeer groot beveiligingsgebrek zijn en dit is dan ook hoogst waarschijnlijk de reden (samen met de " voordoen als validatieserver" aanval) dat het Yubico OTP-protocol verplicht stelt de handtekening of het certificaat te controleren. De aanval is dan niet mogelijk.

Het is natuurlijk de vraag of dit ook daadwerkelijk gebeurt. Bijvoorbeeld bij de PHP Web API (yubikeyPHPclass-0.96) [1] welke ook genoemd wordt op de Yubico website [26], wordt controle op de handtekening enkel gedaan als het gedeelde geheim wordt opgegeven (maar dit is niet verplicht). Een onoplettende programmeur, zou dit kunnen vergeten of met opzet niet doen (omdat hij denkt dat het toch niet nodig is) en desondanks geen gebruik maken van HTTPS. In dat geval zou de desbetreffende applicatie een zeer groot beveiligingslek kennen en zijn beide eisen ondermijnd. We gaan er echter over het algemeen wel vanuit dat programmeurs zich aan de eisen van het protocol zullen houden en hier wel opletten, maar het is zeker iets om in het achterhoofd te houden.

6.3.6 Verbinding blokkeren

Hierboven zijn veel aanvallen genoemd met betrekking tot de authenticiteit en robuustheideis. Echter speelt de beschikbaarheideis bij dit onderdeel ook zeker een rol en hier is ook een mogelijke aanval op. Als de applicatie namelijk niet in staat is verbinding te maken met (één van) de validatieserver(s), dan kan de OTP niet gecontroleerd worden en komt de beschikbaarheideis dus in het geding. Dit is op allerlei manieren mogelijk. Denk bijvoorbeeld aan het platleggen van het complete internet. De applicatie communiceert met de validatieserver over het internet, wat dan niet meer zal gaan. Dit is echter niet zo waarschijnlijk en ook erg risicovol. Er zijn ook minder rigoureuze aanvallen te bedenken om de beschikbaarheid te ondermijnen. Denk bijvoorbeeld aan het blokkeren van de verbinding tussen de applicatie en validatieserver, door alle communicatie te onderscheppen en te verwerpen of door een DDoS aanval op de validatieserver uit te voeren. Dit zorgt ervoor dat gebruikers niet meer bij de applicatie kunnen authenticeren en dit kan grote schadelijke gevolgen hebben.

Het Yubico OTP-protocol kent hier geen ultieme oplossing voor. Wel zijn er maatregelen genomen om het risico te verkleinen (deze maatregelen zijn tevens bedoeld om de gevolgen te verkleinen wanneer een validatieserver per ongeluk uitvalt, bijvoorbeeld door een stroomstoring). Het gaat om de maatregel door niet gebruik te maken van één validatieserver, maar meerdere servers (op dit moment vijf). Deze servers houden elkaar op de hoogte, zodat ze allen up-to-date zijn (hierover meer in hoofdstuk 7). Zodoende kan een applicatie eerst één server proberen. Als deze niet reageert (bijvoorbeeld doordat deze wordt aangevallen), dan kan de applicatie de volgende server proberen, net zolang tot het lukt.

Dit voorkomt dat authenticatie bij één of meerdere applicaties kan worden platgelegd door het aanvallen van één enkele validatieserver. Het is echter nog geen oplossing die deze aanval geheel voorkomt. Vijf servers is immers ook niet bijzonder veel en als één server kan worden platgelegd, dan lukt vijf servers misschien ook nog wel. We krijgen dan ook het idee dat deze oplossing meer bedoeld is om toevallige uitval te voorkomen, dan om echte aanvallen af te weren. Hoe dan ook zorgt het wel voor net wat extra moeite, wat aanvallen kan helpen voorkomen. Bovendien is het makkelijk om het aantal van vijf servers uit te breiden naar meer waardoor het nog lastiger wordt.

Verder helpt dit probleem ook niet wanneer het complete internet plat ligt of de communicatie dichtbij de bron wordt geblokkeerd (dichtbij de applicatie). Deze applicatie kan dan namelijk met geen enkele validatieserver communiceren. Overigens zullen andere applicaties in dit geval waarschijnlijk nog wel werken (en is het dus niet mogelijk heel veel applicaties tegelijk plat te leggen).

Een ander probleem is wanneer juist de validatieservers als bron worden gezien. Schakel de communicatiemogelijkheden van de vijf validatieservers uit en geen enkele applicatie kan meer OTP's laten valideren. Dit is een zeer groot probleem en we vinden daarom het aantal van slechts vijf validatieservers ook aan de lage kant. Het zal voldoende zijn voor toevallige uitval, maar niet bij geplande uitval door een aanval (en hier hebben dan wel zeer veel applicaties en gebruikers tegelijk last van). Hoe lang een dergelijke aanval standhoudt en niet getraceerd kan worden, is twijfelachtig. Het is echter wel iets om rekening mee te houden en we zouden dan ook aanraden het aantal validatieservers te vergroten (voorkomen is immers beter dan genezen).

Een andere zeer vervelende bijzaak is wanneer een applicatie lokaal gedraaid wordt. Er is dan toch

verbinding nodig met het internet om de OTP te laten valideren. Dit is eigenlijk onwenselijk, want als de verbinding met het internet niet beschikbaar is, kan je dus ook geen gebruik maken van lokale applicaties (terwijl dat anders wel zou kunnen). Een mogelijke oplossing hiervoor is om lokaal een validatieserver te draaien, maar dit levert weer het probleem op dat deze niet kan samenwerken met Yubico validatieservers (en dus compatibiliteit van de Yubikey met veel andere applicaties verloren gaat, wat juist één van de grote gemakken van de Yubikey is).

Hoofdstuk 7

Valideren OTP

De validatieserver is verantwoordelijk voor het bepalen of de OTP geldig is (en dus bij de meegestuurde publieke identiteit hoort) of niet. Het valideren kan gedaan worden door de standaard Yubico validatieserver of door een eigen validatieserver. We gaan er (net zoals in de vorige hoofdstukken) vanuit dat de OTP door de standaard Yubico validatieserver gevalideerd wordt (alhoewel de beschrijving voor het grootste deel ook geldt voor eigen validatieservers gebaseerd op het Yubico Validation Server Protocol).

7.1 Werking

Eigenlijk bestaat de Yubico validatieserver uit een netwerk van meerdere servers. Dit is om de availability (beschikbaarheid) te vergroten. De servers communiceren onderling en houden elkaar onderling up-to-date door middel van het Server Replication Protocol. We zullen dit protocol ook in deze sectie bespreken, aangezien het gebruikt wordt om de andere validatieservers up-to-date te houden, zodat allen in staat zijn een volgende OTP correct te kunnen valideren.

De validatieserver krijgt van een (web)applicatie het verzoek binnen om een OTP te controleren. Dit verzoek bestaat uit de OTP zelf samen met nog wat andere parameters (zie sectie 6.1.1). Dit verzoek is de complete invoer van de validatieserver. Het afhandelen van het verzoek gaat volgens het "Validation Server Algorithm", ontwikkeld door Yubico met als doel OTP's valideren, interne waarden up-to-date houden en eventueel andere validatieservers raadplegen en up-to-date houden. Dit algoritme valt onder te verdelen in een aantal delen. We zullen deze delen stuk voor stuk bespreken.

7.1.1 Controleren van het verzoek

De eerste stap is het parseren en controleren van het verzoek van de applicatie. Dit houdt in het controleren van de eventuele handtekening bij het verzoek en het controleren op replay-aanvallen van het verzoek (zoals besproken in sectie 6.3.3).

Het controleren van de handtekening gaat als volgt:

1. De server haalt het gedeelde geheim op bij de meegestuurde client ID van de applicatie
2. De server berekent de verwachte handtekening op basis van de parameters van het verzoek en het shared secret (zie sectie 6.1.3).
3. De server vergelijkt de verwachte handtekening met de handtekening in het verzoek (dit is de controle). Als er een afwijking is, wordt het verzoek niet uitgevoerd en een foutcode (zie sectie 7.1.5) geantwoord.

Het controleren op replay-aanvallen gebeurt door te kijken of de combinatie van OTP en nonce niet al voorkomt in de interne database. In een later stadium (zie subsectie 7.1.3) wordt de OTP en nonce combinatie opgeslagen zodat herhaling van het verzoek gedetecteerd kan worden. Als de OTP en nonce combinatie al voorkomt, dan is het verzoek herhaald en wordt het niet in behandeling genomen. De validatieserver zal een foutcode terugsturen.

7.1.2 Ontsleutelen van de OTP

Na het controleren van het verzoek, zal de validatieserver de OTP moeten ontsleutelen (decrypten) om deze verder te kunnen controleren. De validatieserver doet dit niet zelf, maar gebruikt hiervoor een YK-KSM (Yubikey Key Storage Module)[1]. Deze KSM kan op een andere server draaien dan de validatieserver (waardoor er weer onderlinge communicatie nodig is volgens een bepaald protocol), maar we gaan er in deze situatie vanuit dat de KSM op dezelfde server als de validatieserver draait. De KSM beschikt over de privé-identiteit en de AES-sleutel bij de publieke identiteit van elke Yubikey (ingevoerd in/ondersteund door deze KSM). Deze gegevens zijn nodig voor het ontsleutelen van de OTP. Om precies te zijn is de KSM verantwoordelijk voor de volgende zaken:

- Ontsleutelen van de OTP met behulp van de AES-sleutel behorende bij de publieke identiteit.
- Controleren van de opgeslagen privé-identiteit met de privé-identiteit uit de OTP (en melden als deze niet overeenkomen).
- Controleren van de CRC-checksum (en melden als iets niet klopt).
- Indien geen afwijkingen: teruggeven van de sessieteller, timestamp en sessiegebruikteller uit de OTP.

Om deze taken te kunnen uitvoeren beschikt de YK-KSM dus over een database met de publieke identiteit, de privé-identiteit en de AES-sleutel van elke ondersteunde Yubikey. De database wordt in principe alleen read-only gebruikt. De waarden worden alleen aangepast bij het produceren van een Yubikey (of het maken van een nieuwe Yubico OTP configuratie). Verder is het ook mogelijk om een Yubikey te revoceren (een Yubico OTP configuratie in te trekken/ongeldig te verklaren), dan wordt een waarde uit de database gehaald. De database wordt echter niet voor elk verzoek geüpdate (in tegenstelling tot de database van de validatieserver).

Als invoer neemt de YK-KSM de volledige OTP (dus publieke identiteit gevolgd door het dynamisch gegenereerde deel).

Als uitvoer kent de YK-KSM de volgende mogelijkheden:

- “ERR Syslog open error”: Het openen van het logbestand (intern) is mislukt.
- “ERR No OTP provided”: Er werd geen OTP meegegeven.
- “ERR Invalid OTP format”: Het formaat van de OTP is niet geldig.
- “ERR Database error”: Het opzetten van de database-connectie is mislukt.
- “ERR Database error”: Het uitvoeren van de SELECT query op de database is mislukt.
- “ERR Unknown yubikey”: De publieke identiteit van de OTP komt niet voor in de database.
- “ERR Corrupt OTP”: De CRC-checksum wijkt af of de privé-identiteit klopt niet.
- “OK counter=$counter low=$low high=$high use=$use” Het ontsleutelen van de OTP was succesvol. $counter is de sessieteller, $use is de sessiegebruikteller, $high en $low zijn beide delen van de timestampwaarde.

De uitvoer wordt teruggestuurd naar de validatieserver.

[1] We hebben er voor gekozen om de YK-KSM niet als apart onderdeel te noemen. Dit omdat de YK-KSM een subonderdeel van de validatieserver is, maar eigenlijk ook volledig binnen de validatieserver zelf geïntegreerd kan worden. We zien de taken die de YK-KSM uitvoert dus als taken van de validatieserver die toevallig worden uitbesteed (aan een aparte service, de YK-KSM). Yubico heeft gekozen de opsplitsing tussen validatieserver en YK-KSM te maken zodat de validatieserver en de KSM eventueel los van elkaar kunnen functioneren (en op aparte servers kunnen draaien), mocht dit gewenst zijn.

7.1.3 Controleren tellers

Het controleren van de OTP gebeurt deels door de KSM (klopt het formaat, komen de identiteiten overeen et cetera), maar de validatieserver doet zelf ook nog een controle. Deze berust op het controleren van de tellers (teruggekregen van de YK-KSM). De controle vindt plaats door de sessieteller en sessiegebruikteller waarden bij de publieke identiteit op te halen uit de lokale database. Vervolgens wordt gekeken of de lokale waarden hoger of gelijk zijn dan de waarden uit de OTP. Als dit het geval is, dan wordt de OTP niet geaccepteerd (het gaat om een herhaalde OTP en er is dus waarschijnlijk sprake van een replay aanval, zie sectie 5.3.2). Als dit niet het geval is (de waarden waren lager of er waren nog geen waarden bekend bij de publieke identiteit), dan wordt de OTP (op dit punt) goedgekeurd.

Na de lokale controles moet de interne database worden bijgewerkt. De combinatie van OTP en nonce uit het verzoek wordt toegevoegd aan de database van al voorgekomen OTP/nonce-combinaties om replay-aanvallen op het verzoek te voorkomen. Verder worden de sessieteller en sessiegebruikteller toegevoegd (als hier nog geen waarden voor waren bij de publieke identiteit) of opgehoogd om replay-aanvallen op de OTP te voorkomen. Tevens worden de velden high en low (uit het antwoord van de YK-KSM) opgeslagen in de database, maar hier gebeurt op het moment niets mee.

7.1.4 Syncen

De Yubico validatieserver bestaat feitelijk niet uit één enkele server, maar uit meerdere samenwerkende servers (zie ook het rechter deel van figuur 2.1). Hiervoor is gekozen zodat de werking van het systeem niet afhankelijk is van een enkele server. Als deze server uitvalt zou het hele systeem tijdelijk niet werken en kunnen OTP's niet gevalideerd worden (en dus gebruikers zich niet authenticeren). Het gebruik van meerdere servers verhoogt de beschikbaarheid van het systeem.

Om replay-aanvallen (beter) te kunnen voorkomen is het nodig dat de verschillende validatieservers elkaar op de hoogte houden. De servers updaten elkaar met diverse informatie als een nieuw validatieverzoek binnenkomt. Dit proces van elkaar updaten wordt syncen genoemd. Het syncen van een andere validatieserver gebeurt volgens het zogenaamde "Server Replication Protocol". We zullen dit protocol beschrijven en daarna uitleggen hoe het totale sync-proces in zijn werk gaat.

Server Replication Protocol

Dit protocol wordt door de Yubico validatieservers gebruikt om de andere servers te updaten als er een nieuwe OTP binnenkomt. De communicatie gaat via een HTTPS GET verzoek. Dit verzoek bevat de volgende velden:

- 'otp': De OTP afkomstig van de applicatie.
- 'modified': Een UNIX timestamp welke aangeeft wanneer de OTP is binnengekomen.
- 'nonce': De nonce afkomstig van de applicatie.
- 'yk_identity': De publieke identiteit in kwestie.
- 'yk_counter': De laatst geziene sessieteller door de zender (oftewel de sessieteller uit de OTP).
- 'yk_use': De laatst geziene sessiegebruikteller door de zender (ook uit de OTP).
- 'yk_high': Een deel van de laatst geziene timestamp (ook uit de OTP).
- 'yk_low': Een ander deel van de laatst geziene timestamp (ook uit de OTP).

De ontvangende server slaat hiervan de 'otp'/'nonce'-combinatie op. Verder worden de sessieteller en sessiegebruiktellerwaarden in de interne database bijgewerkt met de 'yk_counter' en 'yk_use' waarden uit het verzoek indien deze hoger zijn. Tevens vindt er logging plaats. Het verzoek wordt gelogd en er zijn een aantal controles welke vermeldingen of waarschuwingen kunnen opleveren. We zullen hier verder niet op ingaan.

De ontvangende server stuurt vervolgens een antwoord terug. Dit antwoord bevat bepaalde waarden uit de interne database (voordat deze waren bijgewerkt met informatie uit het verzoek). Het gaat om de volgende velden:

- 'modified': Een UNIX timestamp van wanneer de laatste OTP is binnengekomen.
- 'nonce': De laatst geziene nonce.
- 'yk_identitiy': De publieke identiteit in kwestie.
- 'yk_counter': De laatst geziene sessieteller.
- 'yk_use': De laatst geziene sessiegebruikteller.
- 'yk_high': Een deel van de laatst geziene timestamp.
- 'yk_low': Een ander deel van de laatst geziene timestamp.

Als het syncen altijd zou lukken, dan zouden alle servers dezelfde waarden in de interne database hebben (voordat er een nieuwe OTP binnenkomt tenminste) en is het dus niet nuttig om informatie terug te sturen (aangezien de versturende server deze informatie zelf ook in de database had staan). Yubico houdt er echter (terecht) rekening mee dat sync-verzoeken soms kunnen mislukken (bijvoorbeeld omdat ze niet aankomen of doordat de server tijdelijk is uitgevallen). Aangezien het onwenselijk is dat replay-aanvallen soms zouden kunnen lukken (omdat het verzoek van de applicatie naar een niet volledig bijgewerkte validatieserver wordt verstuurd), sturen de validatieservers bij een sync-verzoek ook altijd hun eigen informatie terug. Zodoende kunnen hier controles op plaatsvinden om replay-aanvallen te helpen voorkomen.

Het sync-proces

Het totale sync-proces gaat nu als volgt. De validatieserver heeft op dit punt een ontsleutelde OTP en daar al een aantal controles op gedaan. De laatste stap bij het verwerken van een validatieverzoek is het syncen van alle servers en een laatste controle op replay-aanvallen (voor het geval de validatieserver zelf niet up-to-date is). De validatieserver genereert een sync-verzoek en verstuurt deze. Vervolgens wacht de server op antwoord van de andere servers. Dit wachten gebeurt net zolang tot de OTP wordt afgekeurd (aan de hand van onderstaande criteria), de time-out waarde overschreden wordt (het veld 'timeout' in het validatieverzoek) of tot een groot genoeg aantal (percentage) van de andere servers antwoord heeft gegeven en de OTP nog niet afgekeurd is (afhankelijk van de het veld 'sl' in het validatieverzoek).

Als een antwoord op een sync-verzoek binnenkomt, dan vinden de volgende controles plaats:

- Als de tellerwaarden uit het antwoord hoger zijn dan van de validatieserver (van de OTP die gecontroleerd dient te worden), dan worden de waarden van de validatieserver bijgewerkt en de OTP afgekeurd (OTP replay aanval).
- Als de tellerwaarden gelijk zijn, maar de noncewaarde uit het antwoord verschilt met die van de validatieserver, dan wordt de OTP afgekeurd (OTP replay aanval).
- Als de tellerwaarden en de nonce gelijk zijn, dan wordt de OTP goedgekeurd.

Als een OTP al is goedgekeurd (genoeg servers hebben geantwoord) en er komt alsnog een antwoord binnen wat de OTP eigenlijk zou afkeuren, dan wordt dit gelogd als een fout (het alsnog afkeuren van de OTP gaat immers niet, aangezien al naar de applicatie verstuurd is dat de OTP goedgekeurd is en de applicatie dus de gebruiker al toegang heeft verleend). Verder vinden er ook nog andere controles plaats op timestamp-, teller- en noncewaarden. Dit kan vermeldingen of waarschuwingen opleveren (bijvoorbeeld omdat een server niet meer gesynchroniseerd was) en deze worden ook gelogd. We zullen hier verder niet op ingaan.

7.1.5 Resultaat

Het "Validation Server Algorithm" levert uiteindelijk een resultaat op. In dit resultaat staat of een OTP is goedgekeurd of afgekeurd (en waarom). De volgende waarden zijn mogelijk:

- "OK": De OTP is geldig.

- "BAD_OTP": Het formaat van de OTP is ongeldig (wordt bepaald door de YK-KSM).
- "REPLAYED_OTP": De OTP is herhaald, er is sprake van een replay aanval (wordt bepaald door het controleren van de tellers en de sync-antwoorden).
- "BAD_SIGNATURE": De handtekening van het verzoek klopt niet (wordt bepaald door het controleren van het verzoek).
- "MISSING_PARAMETER": Het verzoek mist een verplicht argument (wordt bepaald door het controleren van het verzoek).
- "NO_SUCH_CLIENT": ID van de applicatie is niet bekend (wordt bepaald door het controleren van het verzoek).
- "OPERATION_NOT_ALLOWED": De applicatie mag geen OTP's laten controleren (wordt bepaald door het controleren van het verzoek).
- "BACKEND_ERROR": Er ging intern iets mis.
- "NOT_ENOUGH_ANSWERS": Er waren niet genoeg sync-antwoorden (afhankelijk van het veld 'sl') binnen de opgegeven timeoutwaarde (wordt bepaald door het syncen).
- "REPLAYED_REQUEST": Het verzoek is herhaald, er is sprake van een replay aanval (wordt bepaald door het controleren van het verzoek).

Dit resultaat wordt teruggestuurd naar de applicatie als het veld 'status' in het validatie-antwoord (zie sectie 6.1.2).

7.2 Beveiligingsaspecten

Dit onderdeel is van toepassing op alle drie de beveiligingseisen. We zullen per eis kort toelichten waarom precies en in welke mate.

Het onderdeel heeft een grote rol bij het garanderen van de authenticiteiteis. Bij het valideren van de OTP wordt immers bepaald of de OTP geldig is of niet. Dit moet correct gebeuren, anders komt de eis in gevaar (en wordt onrechtmatige authenticatie mogelijk). Ook moet een aantal subtaken correct worden uitgevoerd (zoals het syncen) zodat de eis ook in de toekomst gegarandeerd kan worden.

Ook de beschikbaarheideis speelt bij dit onderdeel een grote rol. Zonder validatieserver is controle simpelweg niet mogelijk. Vandaar dat binnen het onderdeel maatregelen genomen zouden moeten zijn om ervoor te zorgen dat controle altijd mogelijk is, ook al is een validatieserver (door een kwaadwillende) niet beschikbaar.

De robuustheideis is tevens belangrijk bij dit onderdeel. De validatieserver bepaalt immers of een OTP geldig is of niet en kan dus ook bepalen dat een rechtmatige authenticatiepoging geweigerd wordt. In een normale situatie zal dit niet gebeuren, maar de beveiliging van dit onderdeel moet er in principe voor zorgen dat dit ook niet met een aanval kan gebeuren. Dit is onder andere van toepassing op het sync-proces en dat moet dus goed beveiligd zijn.

Op het gebied van beveiligingsaspecten zijn alle aspecten met uitzondering van onweerlegbaarheid erg belangrijk. Vertrouwelijkheid is zeer belangrijk, want de validatieserver kent in principe alle geheimen (gebruikte AES-sleutel en privé-identiteit) over alle Yubikeys. Dit mag dus niet zomaar uitlekken. Ook bij het sync-proces is vertrouwelijkheid belangrijk, want als alle OTP's die verstuurd worden vanaf een validatieserver gelezen kunnen worden (en eventueel ook de antwoorden welke informatie over de plaintext bevatten), dan levert dit nuttige informatie op.

De integriteit is belangrijk vanwege de authenticiteit en robuustheideis. De data op de validatieserver mag niet kunnen worden aangepast, aangezien anders rechtmatige authenticatie mogelijk wordt (verlaag bijvoorbeeld de tellerwaarden) of juist rechtmatige authenticatiepogingen mislukken (verander bijvoorbeeld een privé-identiteit). Dit geldt ook voor de sync-berichten. Als de inhoud hiervan kan worden aangepast, dan is het ook mogelijk om het te laten lijken dat de andere validatieserver een bepaalde OTP juist wel of niet geldig vindt (terwijl dit niet het geval is).

De authenticiteit is met name van belang vanwege de authenticiteiteis en de robuustheideis. Dit heeft opnieuw voornamelijk te maken met het sync-proces. De validatieserver moet namelijk zeker weten dat sync-verzoeken van een andere (geldige) validatieserver komen en de antwoorden hierop ook. Als dit niet het geval is, dan kan onrechtmatige authenticatie wellicht mogelijk worden of rechtmatige authenticatie geblokkeerd worden (om ongeveer dezelfde redenen genoemd bij integriteit).

De beschikbaarheid van dit onderdeel is ook zeer belangrijk. Immers als het systeem niet beschikbaar is, kunnen OTP's niet gevalideerd worden en komt de beschikbaarheideis dus in het gedrang. De beveiliging van dit onderdeel moet er dus in principe (redelijkerwijs) voor zorgen dat dit niet kan gebeuren. Een enkele validatieserver mag best onbeschikbaar zijn, zolang het systeem als geheel maar wel beschikbaar blijft.

Onweerlegbaarheid is vanwege dezelfde redenen eerder genoemd bij andere onderdelen opnieuw niet echt van belang.

7.3 Beveiligingsanalyse

Veel van de controles bij dit onderdeel (zoals het controleren van tellerwaarden) dienen om te beschermen tegen aanvallen bij andere onderdelen (zoals een replay aanval op het onderdeel communicatie tussen Yubikey en applicatie). We zullen deze aanvallen hier niet opnieuw bespreken, aangezien ze al behandeld zijn. Wel zullen we een aantal nieuwe aanvallen specifiek op dit onderdeel behandelen.

Dit onderdeel bevat veel belangrijke waarden welke geheim moeten blijven en/of niet zomaar mogen veranderen. Dit is belangrijk om de authenticiteiteis te kunnen garanderen. Mogelijke aanvallen op dit onderdeel zijn dan ook het aanvallen van (één van) de validatieserver(s) om deze waarden te verkrijgen of aan te passen. Denk bijvoorbeeld aan proberen onrechtmatig toegang te verschaffen tot de YK-KSM om zo de privé-identiteit en AES-sleutel te achterhalen. Als dit gebeurt kan een aanvaller zich daarna heel makkelijk als iemand anders authenticeren. Ook het wijzigen van informatie kan gevolgen hebben voor de authenticiteiteis. Denk bijvoorbeeld aan het verlagen van de tellerwaarden zodat gebruikte OTP's weer geldig worden (en dus een replay aanval mogelijk is). Veel van deze aanvallen kunnen tevens gebruikt worden om de robuustheideis te ondermijnen.

Een andere mogelijkheid om de authenticiteiteis in het geding te brengen is door het sync-proces aan te vallen. Dit kan op diverse manieren, bijvoorbeeld door middel van een man-in-the-middle aanval of juist door het blokkeren van sync-verzoeken. Deze aanvallen kunnen ook (op een aangepast manier) gebruikt worden tegen de robuustheideis.

Als laatste is er nog een aanval op de beschikbaarheideis. Deze aanval lijkt sterk op degene genoemd bij het onderdeel communicatie tussen applicatie en validatieserver. Het idee daar was echter om de communicatie te blokkeren, terwijl het hier juist gaat om het uitschakelen van de server (en zo de beschikbaarheid te ondermijnen).

7.3.1 server-aanval

De meest effectieve aanval op dit onderdeel is waarschijnlijk de server-aanval. De validatieserver/YK-KSM kennen immers alle geheimen van de (alle) Yubikeys. Ze hebben de AES-sleutel, de privé-identiteit, de tellerwaarden et cetera. Al deze gegevens zijn in principe onversleuteld opgeslagen op de server(s). Zelfs als de data versleuteld zou zijn opgeslagen, zou deze waarschijnlijk alsnog te achterhalen zijn, aangezien de decryptiesleutel ook op de server moet zijn opgeslagen (de validatieserver en YK-KSM hebben deze data immers continu nodig). Als een aanvaller dus onrechtmatige toegang kan krijgen tot de server, kan hij deze informatie bemachtigen en/of wijzigen (en dus gebruiken om de authenticiteiteis of robuustheideis te doorbreken). Het spreekt voor zich dat als een aanvaller deze gegevens kan inzien en/of kan wijzigen het mogelijk is de eisen op veel verschillende manieren te ondermijnen. We zullen de tientallen manieren waarop dit kan niet allemaal bespreken en toelichten (dit zijn er simpelweg te veel en we hebben al enkele voorbeelden genoemd).

Voor dit onderdeel is het dus essentieel te zorgen dat een aanvaller geen onrechtmatige toegang tot een server kan verkrijgen. De mogelijkheden van een aanvaller om dit te doen zijn te verdelen in twee categorieën: een fysieke server-aanval (waarbij de aanvaller fysiek bij de server kan en dus bijvoorbeeld de harde schijf verwijdert om de informatie te lezen) en een externe aanval (waarbij de aanvaller op afstand toegang verkrijgt). Binnen deze categorieën zijn diverse mogelijkheden en gradaties. Een aanvaller kan

bijvoorbeeld toegang verkrijgen door een hardware keylogger te plaatsen tussen toetsenbord en server en zo het (root) wachtwoord onderscheppen (dit is duidelijk een fysieke aanval). Een aanvaller kan ook proberen fouten/kwetsbaarheden te vinden in software die op de server draait, bijvoorbeeld de database software (dit kan wellicht op afstand plaatsvinden). Er zijn tal van mogelijkheden te bedenken waarop dit zou kunnen, maar tegen de meeste zal Yubico hoogstwaarschijnlijk maatregelen genomen hebben. Of een aanvaller zich onrechtmatig toegang kan verschaffen tot een server is afhankelijk van veel specifieke factoren (zoals welke software erop draait). Deze factoren zijn in het geval van de Yubico validatieservers niet bekend en kunnen bovendien snel verouderen. Aangezien deze aanvallen eigenlijk ook niet zozeer betrekking hebben op het Yubico OTP-protocol (de aanvallen vinden immers buiten het protocol om plaats) zullen we deze mogelijkheden en de risico's hiervan verder niet bespreken.

Kort samengevat is het in ieder geval zaak voor Yubico (of iemand die een eigen validatieserver of YK-KSM draait) zorg te dragen voor de beveiliging van de server. Als de server namelijk kwetsbaar is, kan het hele Yubico OTP-protocol kwetsbaar zijn.

7.3.2 Sync-aanval

Met een sync-aanval wordt een aanval bedoelt op het sync-proces. Dit kan bijvoorbeeld een replay aanval zijn of een man-in-the-middle aanval. Zoals misschien wel opgevallen is, wordt bij het sync-proces eigenlijk geen beveiliging in het bericht zelf gestopt (er wordt bijvoorbeeld geen handtekening over het bericht gedaan ten behoeve van integriteit en authenticiteit). Dit komt doordat voor de communicatie gebruik gemaakt wordt van een beveiligde HTTPS verbinding waarbij zowel het servercertificaat (ontvanger van sync-verzoek) als cliëntcertificaat (verzender van sync-verzoek) gecontroleerd worden [23]. Zodoende biedt de verbinding dus vertrouwelijkheid, integriteit en authenticiteit en is dus vergelijkbaar met situatie 7 (optimale beveiliging) besproken bij de beveiligingsanalyse van het vorige onderdeel (zie sectie 6.3).

Door de gebruikte communicatievorm worden aanvallen als replay aanval, man-in-the-middle aanval of voordoen als validatieserver in één klap onbruikbaar. Er vallen dus veel aanvallen op dit gebied gelijk af. De enige aanval die overblijft, is het blokkeren van sync-verzoeken.

Bij het blokkeren van sync-verzoeken wordt communicatie tussen validatieserver A (die het verzoek wilt doen) en validatieserver B (die in sync gebracht dient te worden) geblokkeerd. Het verzoek wordt dan onderschept en weggegooid en komt zodoende nooit aan bij validatieserver B. Dit lijkt weinig nut te hebben, maar het kan helpen bij een aanval. Stel bijvoorbeeld dat er twee applicaties zijn, applicatie 1 en applicatie 2, welke respectievelijk gebruik maken van validatieserver A en validatieserver B. Een aanvaller kan OTP's onderscheppen bij applicatie 1 en de sync-verzoeken van validatieserver A naar B tegenhouden. Validatieserver B is dan niet in sync en dus kunnen de OTP's hergebruikt worden bij applicatie 2. De applicatie gaat immers bij validatieserver B navragen of de OTP's al gebruikt zijn. Server B kent de OTP's nog niet en zal dus (onterecht) antwoorden dat ze geldig zijn.

In principe is deze aanval uit te voeren. Het vergt zeker wat werk, expertise en moeite, maar het is niet onmogelijk. Een factor die het uitvoeren van de aanval lastiger maakt is dat over het algemeen een bepaald percentage van de validatieservers moet antwoorden op de sync-verzoeken (en anders de OTP wordt afgekeurd). Dit percentage wordt bepaald door het veld sl in het verzoek (en als dit niet gevuld is, wordt de standaardwaarde gebruikt). Aangezien altijd minimaal één server sowieso in up-to-date is (waar het validatieverzoek op binnenkomt), zal de aanval mislukken bij applicaties die van 100% van de servers antwoord vereisen. Standaard wordt echter maar een sl van 25% vereist. Met vijf servers, betekent dit dat slechts (5 - 1) * 25% = 1 andere validatieserver op het sync-verzoek moet antwoorden. In dat geval is de aanval dus op zich wel uit te voeren. De aanvaller zorgt ervoor dat twee validatieservers niet up-to-date zijn (met de hiervoor beschreven methode) namelijk de validatieserver B waar de applicatie het verzoek standaard heen stuurt en een willekeurige andere server X. Als de applicatie het validatieverzoek naar server B verstuurt en server B vervolgens de sync-controle wil doen, dan moeten alle sync-verzoeken (of de antwoorden hierop) geblokkeerd worden, behalve degene die naar validatieserver X gaat. Zowel B als X zijn niet up-to-date en zullen dus een herhaalde OTP accepteren en teruggeven dat ze die nog niet eerder hebben gezien. Zodoende kan een aanvaller zich onrechtmatig authenticeren en dus de authenticiteiteis schenden.

De vraag is hoe reëel deze aanval is. Een aanvaller moet namelijk wel erg veel zaken tegelijk doen (hij moet eerst sync-antwoorden blokkeren zodat genoeg servers niet meer up-to-date zijn, hij moet een pas gebruikte OTP onderscheppen en hij moet daarna bij het gebruiken van de OTP zorgen dat alleen

de niet meer up-to-date servers antwoorden op de sync-antwoorden). Theoretisch gezien is het mogelijk, maar in de praktijk zal de aanval zeer lastig uit te voeren zijn. Bij een sl-waarde van 0% is het nog wel haalbaar, maar bij 25% wordt het al zeer lastig. Hoe hoger de sl-waarde is, hoe lastiger het wordt (en bij 100% is het in principe onmogelijk deze aanval uit te voeren). De reden dat applicaties niet standaard voor 100% kiezen is simpel, het komt de beschikbaarheid niet ten goede. Immers hoeft er maar één server niet te antwoorden op een sync-verzoek (bijvoorbeeld door een aanval) en authenticeren bij die applicatie mislukt. Er bestaat dus een duidelijke trade-off tussen de authenticiteiteis en de beschikbaarheideis bij het kiezen van de waarde voor sl. We denken dat voor de meeste applicaties de standaardwaarde van 25% voor de authenticiteiteis eigenlijk al voldoende is. Bij zeer belangrijke applicaties kan wellicht voor een iets hogere waarde gekozen worden. Applicatie-ontwikkelaars zouden voor zichzelf moeten nagaan wat zij belangrijker vinden voor hun applicatie en daarop hun keuze baseren.

Wel zou het voor Yubico aan te raden zijn de hoeveelheid validatieservers te vergroten. Dit komt namelijk zowel de authenticiteiteis (aanval is lastiger uit te voeren) en beschikbaarheideis (minder gevoelig voor uitvallen enkele servers) ten goede. Eventueel kan dan ook de standaardwaarde voor sl verhoogd worden, zonder te veel in te leveren qua beschikbaarheid.

7.3.3 Beschikbaarheid-aanval

De laatste mogelijke soort aanval op dit onderdeel is de Beschikbaarheid-aanval. Deze aanval is, zoals de naam al doet vermoeden, gericht op de beschikbaarheideis. Het idee is het onmogelijk maken van het valideren van OTP's en dus het onmogelijk maken van authenticeren bij een bepaalde applicatie. Dit kan onder andere door de verbinding tussen applicatie en validatieserver te blokkeren, zoals besproken in subsectie 7.3.2. De aanval op dit onderdeel richt zich echter op andere punten, namelijk: het aanvallen de validatieserver(s)/YK-KSM's of het blokkeren van sync-berichten.

Aanvallen van server(s)

Server(s) kunnen op allerlei manieren worden aangevallen en 'uit de lucht' gehaald worden. Het gaat hier om het aanvallen van de validatieserver(s) en eventueel de YK-KSM server(s) Dit heeft voor een deel overlap met de server-aanvallen besproken in subsectie 7.3.1. Echter richt de aanval zich in dit geval niet op het extraheren of aanpassen van gegevens, maar op het onbeschikbaar maken van de server (wat in sommige gevallen makkelijker is). Ook dit kan op vele manieren, welke we niet allemaal in detail zullen bespreken, maar enkele voorbeelden zijn: stroomaansluiting van de server verwijderen (fysieke aanval) of server extreem overbelasten (externe aanval).

In principe is het Yubico OTP-protocol hier wel tegen beschermd door het gebruik van meerdere servers, maar zoals al eerder aangegeven zijn vijf servers nog redelijk kwetsbaar. Als iemand in staat is één server aan te vallen, is het niet onwaarschijnlijk dat dit bij meerdere servers lukt (afhankelijk van de soort aanval). Bovendien hoeven niet eens alle servers worden aangevallen om validatie onmogelijk te maken. Het Yubico OTP-protocol vereist immers een bepaald niveau van syncen (standaard 25%) waardoor valideren al mislukt als vier van de vijf servers niet beschikbaar zijn. Bij sommige applicaties, met een hoge sl-waarde, hoeven er zelfs nog minder servers te worden aangevallen.

Aanvallen van sync-berichten

Een andere manier, waarbij geen complete servers onbeschikbaar gemaakt hoeven te worden, is het blokkeren van de sync-berichten. Deze manier valt waarschijnlijk minder op dan wanneer gehele servers onbeschikbaar gemaakt worden en de aanval is wellicht ook makkelijker uit te voeren. Het idee is om bij een validatiepoging de sync-verzoeken of antwoorden te blokkeren, zodat het minimum aan aantal succesvolle sync-pogingen niet behaald wordt.

Het Yubico OTP-protocol kent hier geen beveiliging tegen. De HTTPS verbinding biedt vertrouwelijkheid, integriteit en authenticiteit, maar helaas geen beschikbaarheid. In theorie is de aanval dus uit te voeren (alhoewel dit in de praktijk waarschijnlijk nog steeds niet gemakkelijk zou gaan) en dus iets om rekening mee te houden. Een mogelijke oplossing hiervoor is door simpelweg het aantal validatieservers te vergroten en eventueel applicaties in te stellen om een andere validatieserver te proberen mocht de eerste mislukken (het kan namelijk voorkomen dat alle communicatie direct bij de bron onderschept wordt, waardoor één validatieserver gevoelig is voor deze aanval, maar anderen niet).

Hoofdstuk 8

Overzicht (mogelijke) aanvallen

Gedurende het onderzoek is er voor elk onderdeel een aantal mogelijk aanvallen genoemd, wat het risico op de aanval is en wat de gevolgen ervan zijn. We zullen nu een overzicht geven van al deze mogelijke aanvallen en deze op een aantal punten beoordelen. Hierbij zullen we meenemen:

- Eisen (welke eisen de aanval mogelijk schendt): authenticiteiteis, beschikbaarheideis en/of robuustheideis.
- Bescherming (hoe goed is het protocol beschermd tegen de aanval): Goed, gemiddeld, slecht of situatie-afhankelijk.
- Waarschijnlijkheid (hoe gemakkelijk is de aanval uit te voeren, hoe waarschijnlijk is de aanval): Waarschijnlijk, gemiddeld, onwaarschijnlijk, niet mogelijk of situatie-onafhankelijk.
- Schade (hoeveel schade kan ermee worden aangericht): Groot, middelmatig, klein, niet tot nauwelijks.
- Verbeteringen (zijn er verbeteringen mogelijk om de bescherming tegen de aanval te vergroten): Gemakkelijk, mogelijk, moeilijk, niet nodig.

Bij schade houden we rekening met welke eisen eventueel geschonden worden. Hierbij beschouwen we de authenticiteiteis als het belangrijkst (zoals in hoofdstuk 3 vermeld).

Aan de hand van dit overzicht kan snel worden ingeschat waar (bij welk onderdeel) de grootste beveiligingsgebreken zitten, waar de meeste ruimte voor verbetering is en kan een eindconclusie gedaan worden over de (beveiliging van de) Yubikey.

8.1 Genereren OTP

1. Kraken cryptografie
   - Eisen: authenticiteiteis (met name) en robuustheideis (in iets mindere mate).
   - Bescherming: Goed, dankzij het gebruik van AES.
   - Waarschijnlijkheid: Onwaarschijnlijk, geen aanval binnen afzienbare tijd mogelijk.
   - Schade: Groot, de authenticiteiteis en robuustheideis zouden volledig doorbroken zijn.
   - Verbeteringen: Niet nodig (op het moment).
2. Geheimen achterhalen (zoals AES-sleutel)
   - Eisen: authenticiteiteis (met name) en robuustheideis (in kleine mate).
   - Bescherming: Goed, de geheimen zijn niet zomaar via de interface uit te lezen.
   - Waarschijnlijkheid: Onwaarschijnlijk, de bescherming tegen uitlezen via de interface is prima. Het is wellicht wel mogelijk de Yubikey fysiek uit te lezen, maar dit is niet gemakkelijk.

- Schade: Groot, de authenticiteiteis en robuustheideis zouden volledig doorbroken zijn.
- Verbeteringen: Mogelijk, de Yubikey zou fysiek beschermd kunnen worden tegen deze aanval.

3. Yubikey fysiek aanvallen (stelen, kapot maken, et cetera)
   - Eisen: robuustheideis.
   - Bescherming: Slecht, de Yubikey kent hier geen bescherming tegen binnen het protocol zelf.
   - Waarschijnlijkheid: Gemiddeld, er is namelijk wel fysieke toegang tot de Yubikey nodig.
   - Schade: Middelmatig, enkel robuustheideis doorbroken en meestal is er een een applicatiespecifieke oplossing.
   - Verbeteringen: Moeilijk, bijvoorbeeld mogelijkheid om te authenticeren zonder Yubikey maken.
4. Data wijzigen
   - Eisen: robuustheideis.
   - Bescherming: Goed, door middel van 48 bits sleutel.
   - Waarschijnlijkheid: Waarschijnlijk, dit omdat de bescherming niet standaard gebruikt wordt.
   - Schade: Middelmatig, zelfde redenen als bij vorige aanval.
   - Verbeteringen: Gemakkelijk, namelijk de configuraties op de Yubikey standaard beschermen.

8.2 Communiceren OTP met (web)applicatie

1. Reflection aanval
   - Eisen: authenticiteiteis.
   - Bescherming: Goed, dankzij startende partij zich eerst (en als enige) te laten authenticeren.
   - Waarschijnlijkheid: Niet mogelijk.
   - Schade: Middelmatig, een aanvaller kan zich met wat moeite voordoen als een ander persoon.
   - Verbeteringen: Niet nodig.
2. Replay aanval
   - Eisen: authenticiteiteis.
   - Bescherming: Goed, doordat een OTP maar eenmalig werkt.
   - Waarschijnlijkheid: Onwaarschijnlijk, door gebruikte bescherming.
   - Schade: Middelmatig, een aanvaller kan zich met wat moeite voordoen als een ander persoon.
   - Verbeteringen: Niet nodig.
3. Man-in-the-middle aanval: Wijzigen OTP
   - Eisen: robuustheideis.
   - Bescherming: Slecht, geen standaard bescherming. Wel mogelijkheid om HTTPS te gebruiken.
   - Waarschijnlijkheid: Gemiddeld als HTTPS gebruikt wordt, anders onwaarschijnlijk.
   - Schade: Klein, de robuustheideis is eenmalig voor één gebruiker doorbroken. Aanval valt op.
   - Verbeteringen: Moeilijk, toevoegen goede klokwaarde in OTP en gebruiken bij valideren.
4. Man-in-the-middle aanval: Onrechtmatig authenticeren
   - Eisen: authenticiteiteis.
   - Bescherming: Slecht, geen standaard bescherming. Wel mogelijkheid om HTTPS te gebruiken.
   - Waarschijnlijkheid: Gemiddeld, afhankelijk van het specifieke geval (applicatie en verbinding).

- Schade: Middelmatig, de authenticiteiteis is eenmalig voor een gebruiker doorbroken.
- Verbeteringen: Mogelijk, verplicht gebruik laten maken van HTTPS.

5. Man-in-the-middle aanval: Lezen OTP's

   - Eisen: authenticiteiteis en robuustheideis.
   - Bescherming: Slecht, geen bescherming tegen. Wel mogelijkheid om HTTPS te gebruiken.
   - Waarschijnlijkheid: Waarschijnlijk, indien niet gebruik wordt gemaakt van HTTPS.
   - Schade: Klein tot geen, niet zomaar te gebruiken dankzij cryptografie.
   - Verbeteringen: Niet nodig.

8.3 Communiceren tussen (web)applicatie en validatieserver

Zoals besproken zijn er bij de communicatie tussen applicatie en validatieserver veel situaties mogelijk qua beveiliging (waarbij de keuze in principe bij de applicatie ligt). Deze situaties hebben invloed op de waarschijnlijkheid van en de bescherming tegen bepaalde aanvallen. Aangezien het weinig zinvol zou zijn iedere keer alleen "situatie-afhankelijk" te antwoorden, zullen we bij het bespreken de situaties noemen waarvoor de beoordeling verschilt en wat de beoordeling in dat geval is. We zullen hierbij de situaties die niet mogen voorkomen volgens het protocol (situatie 1, 2 en 5) buiten beschouwing laten. Wel is het belangrijk in het achterhoofd te houden dat bij deze situaties zeer veel aanvallen mogelijk zijn en het zeker niet ondenkbaar is dat een (klein) aantal applicaties toch van deze situaties gebruik maakt.

1. Kraken handtekening

   - Eisen: authenticiteiteis en robuustheideis.
   - Bescherming: Gemiddeld (voor het moment prima, maar het kan beter).
   - Waarschijnlijkheid: Onwaarschijnlijk (op het moment).
   - Schade: Groot, maakt diverse andere aanvallen gemakkelijk.
   - Verbeteringen: Mogelijk, gebruik HTTPS of asymmetrische cryptografie.

2. Voordoen als validatieserver

   - Eisen: authenticiteiteis en robuustheideis.
   - Bescherming: Goed (tenminste in de toegestane situaties).
   - Waarschijnlijkheid: Niet mogelijk.
   - Schade: Groot, authenticiteiteis en robuustheideis volledig doorbroken.
   - Verbeteringen: Niet nodig, eventueel zorgen dat applicaties toegestane situatie gebruiken.

3. Replay aanval (op het antwoord)

   - Eisen: authenticiteiteis.
   - Bescherming: Bij gebruik HTTPS goed, anders mogelijk (maar niet standaard).
   - Waarschijnlijkheid: Indien geen gebruik van HTTPS of controle OTP/nonce waarschijnlijk.
   - Schade: Middelmatig, aanvaller kan met wat moeite authenticiteiteis of robuustheideis ondermijnen voor één gebruiker.
   - Verbeteringen: Mogelijk, verplicht stellen gebruik HTTPS of controleren OTP/nonce.

4. Blokkeren validatieverzoeken

   - Eisen: authenticiteiteis.
   - Bescherming: situatie-afhankelijk, deels bij gebruik HTTPS en in andere situaties geen.
   - Waarschijnlijkheid: Gemiddeld, aanval is in principe uit te voeren, maar valt op.

- Schade: Middelmatig, een aanvaller kan zich met wat moeite voordoen als een ander persoon.
- Verbeteringen: Moeilijk, toevoegen goede klokwaarde in OTP en gebruiken bij valideren.

5. Man-in-the-middle aanval: Lezen verzoek

    - Eisen: authenticiteiteis.
    - Bescherming: Bij gebruik HTTPS goed, in andere situaties geen.
    - Waarschijnlijkheid: Niet mogelijk bij gebruik HTTPS, in andere situaties waarschijnlijk.
    - Schade: Klein, niet direct schadelijk gevolgen, maar kan gebruikt worden bij andere aanval.
    - Verbeteringen: Mogelijk, verplicht gebruik maken van HTTPS.

6. Man-in-the-middle aanval: Wijzigen verzoek

    - Eisen: authenticiteiteis en robuustheideis.
    - Bescherming: In situatie 3 geen, in andere situaties goed.
    - Waarschijnlijkheid: in situatie 3 waarschijnlijk, in andere situaties niet mogelijk.
    - Schade: Middelmatig, robuustheideis kan doorbroken worden. authenticiteiteis niet direct doorbroken (maar het kan helpen bij andere aanvallen).
    - Verbeteringen: Mogelijk, situatie 3 ook verbieden.

7. Man-in-the-middle aanval: Lezen antwoord

    - Eisen: authenticiteiteis en robuustheideis.
    - Bescherming: Bij gebruik HTTPS goed, in andere situaties geen.
    - Waarschijnlijkheid: Niet mogelijk bij gebruik HTTPS, in andere situaties waarschijnlijk.
    - Schade: Klein, kan niet op zich zelf gebruikt worden, maar kan helpen bij andere aanvallen.
    - Verbeteringen: Mogelijk, zoals verplicht gebruik HTTPS (maar eigenlijk niet nodig).

8. Man-in-the-middle aanval: Wijzigen antwoord

    - Eisen: authenticiteiteis en robuustheideis.
    - Bescherming: Goed (tenminste in de toegestane situaties).
    - Waarschijnlijkheid: Niet mogelijk.
    - Schade: Groot, authenticiteiteis en robuustheideis volledig doorbroken.
    - Verbeteringen: Niet nodig, eventueel zorgen dat applicaties toegestane situatie gebruiken.

9. Verbinding blokkeren

    - Eisen: beschikbaarheideis
    - Bescherming: Gemiddeld.
    - Waarschijnlijkheid: Gemiddeld (aanvaller moet meerdere verbindingen tegelijk blokkeren).
    - Schade: Klein, het doorbreekt enkel de beschikbaarheideis en valt bovendien erg op.
    - Verbeteringen: Mogelijk, meer Yubico validatieservers of gebruik eigen, lokale validatieserver.

8.4 Valideren OTP

1. server-aanval
    - Eisen: authenticiteiteis, beschikbaarheideis en robuustheideis.
    - Bescherming: Onbekend (niet besproken binnen het Yubico OTP-protocol).
    - Waarschijnlijkheid: Niet waarschijnlijk, maar wellicht wel mogelijk
    - Schade: Groot, meerdere eisen volledig doorbroken.
    - Verbeteringen: Onbekend, eventueel lokale validatieserver gebruiken (en zelf beschermen).
2. sync-aanval: replay aanval, man-in-the-middle aanval, e.d.
    - Eisen: authenticiteiteis en robuustheideis.
    - Bescherming: Goed (door gebruik van HTTPS met server en verificatie cliëntcertificaat).
    - Waarschijnlijkheid: Niet mogelijk.
    - Schade: Groot, meerdere eisen doorbroken.
    - Verbeteringen: Niet nodig.
3. sync-aanval: blokkeren sync-verzoeken
    - Eisen: authenticiteiteis.
    - Bescherming: Gemiddeld (door gebruik meerdere validatieservers), maar niet optimaal.
    - Waarschijnlijkheid: Niet waarschijnlijk (er moet veel tegelijk gebeuren), maar wel mogelijk.
    - Schade: Middelmatig, een aanvaller kan zich met (veel) moeite voordoen als een ander persoon.
    - Verbeteringen: Moeilijk, toevoegen goede klokwaarde in OTP en gebruiken bij valideren.
4. Beschikbaarheid-aanval: server-aanval
    - Eisen: beschikbaarheideis.
    - Bescherming: Gemiddeld (door gebruik meerdere validatieservers), maar niet optimaal.
    - Waarschijnlijkheid: Gemiddeld.
    - Schade: Afhankelijk van situatie en duur. Kan groot zijn (bijvoorbeeld alle validatieservers onbeschikbaar voor lange tijd), maar het kan ook erg meevallen. Aanval valt snel op.
    - Verbeteringen: Mogelijk, meer Yubico validatieservers.
5. Beschikbaarheid-aanval: Sync-berichten aanval
    - Eisen: beschikbaarheideis.
    - Bescherming: Gemiddeld, door genoegen nemen met beperkt aantal sync-antwoorden.
    - Waarschijnlijkheid: Gemiddeld.
    - Schade: Afhankelijk van situatie en duur.
    - Verbeteringen: Mogelijk, meer Yubico validatieservers of gebruik lokale validatieserver. Kies goede sl-waarde (afweging tussen authenticiteiteis en beschikbaarheideis).

Hoofdstuk 9

Discussie en conclusie

Dit onderzoek heeft twee hoofddoelen, namelijk het beschrijven van het Yubico OTP-protocol (hoe werkt het, welk onderdeel doet wat, waarom gebeurt dit zo en hoe zit het met de beveiliging) en het doen van een beveiligingsanalyse waarbij gezocht wordt naar mogelijke aanvallen, gebreken en zwakheden. Om met het eerste deel te beginnen. We hebben een overzicht gegeven van de exacte werking van het protocol. We begonnen met het algemeen beschrijven van wat er gebeurt en hebben dit daarna voor elk onderdeel specifiek en uitgebreid toegelicht. Deze informatie is grotendeels afkomstig uit een literatuuronderzoek, waarbij meerdere bronnen gebruikt werden om de werking van het gehele Yubico OTP-protocol in kaart te brengen. Behalve een literatuuronderzoek hebben we ook praktijktests gedaan en specifieke implementaties van onderdelen bestudeerd, aangezien niet alles volledig of duidelijk beschreven stond. De beschrijving is zo goed als compleet, elk belangrijk onderdeel en proces wordt genoemd en uitgelegd. Wel hebben we enkel het (standaard) Yubico OTP-protocol besproken, maar dit was een bewuste en duidelijk aangegeven keuze.

Wat betreft het tweede doel. We hebben een beveiligingsanalyse van het Yubico OTP-protocol gegeven. Dit hebben we gedaan door eerst drie beveiligingseisen op te stellen en een aantal (standaard) beveiligingsaspecten te noemen welke we gebruikten om de beveiliging te bespreken. We hebben kort besproken welke beveiligingsaspecten er komen kijken bij de Yubikey in een groter geheel en daarna is op elk onderdeel een specifieke beveiligingsanalyse gedaan. Dit door eerst te bepalen welke beveiligingseisen en aspecten van toepassing zijn op het onderdeel en daarna op basis van deze eisen en aspecten de analyse te doen. De analyse is gedaan door mogelijke aanvallen te bedenken en te behandelen (uitleggen, risico inschatten, schade bespreken et cetera). Als laatste hebben we een samenvattend overzicht van alle mogelijke aanvallen gegeven, samen met enkele belangrijke eigenschappen. Dit overzicht geeft een goed beeld van de punten waar wel goed tegen beveiligd is en waar juist niet goed beveiligd is. Voor de beveiligingsanalyse hebben we gebruik gemaakt van literatuurstudies, praktijktests, specifieke (open-source) implementaties en eigen inzicht.

Een belangrijk discussiepunt is of de beveiligingsanalyse volledig is geweest. Dit is niet met zekerheid te zeggen, aangezien het altijd mogelijk is dat we iets over het hoofd gezien hebben of ergens niet aan gedacht hebben. Bovendien hebben we bepaalde zaken geabstraheerd. Zo hebben we geen uitgebreide analyse gedaan van AES, maar gaan we er vanuit dat de beveiliging hiervan goed (genoeg) is (gebaseerd op andere onderzoeken). Het is dus lastig om zeker te kunnen zijn dat er niet nog ergens een verborgen gebrek zit. Wel kunnen we stellen dat we de beveiliging van elk onderdeel op zich zo volledig mogelijk geanalyseerd hebben en dit op een systematische manier. Hierbij hebben we gekeken naar drie eisen en vijf aspecten en hieruit mogelijke aanvallen afgeleid en deze besproken. Door de systematische aanpak is de kans dat we iets over het hoofd zien klein. Soms hebben we zaken met opzet niet besproken (omdat het buiten de scope van het onderzoek valt), maar dit is dan duidelijk bij de analyse vermeld en toegelicht. Daarom gaan we ervan uit dat de beveiligingsanalyse (in ieder geval voor het grootste deel) volledig is.

De hoofdvraag bij dit onderzoek was of de Yubikey beveiligingsgebreken kent en zo ja, welke. In principe kent de Yubikey (in ieder geval in bepaalde situaties) inderdaad beveiligingsgebreken. Welke precies kan gevonden worden in het overzicht in hoofdstuk 8 en bij de beveiligingsanalyse van de specifieke onderdelen. Over het algemeen is de beveiliging van de Yubikey echter prima in orde. In ieder geval voor de meeste applicaties en al helemaal als de Yubikey als toevoeging op het standaard wachtwoord

gebruikt wordt (in plaats van vervanging). We kunnen ook stellen dat de Yubikey een betere beveiliging is dan het standaard wachtwoord (of er een goede toevoeging op vormt), aangezien de Yubikey niet of minder gevoelig is voor enkele aanvallen waar het standaard wachtwoord zeer gevoelig voor is (denk aan replay-aanvallen). Echt volledig veilig is de Yubikey niet, aangezien deze ondanks dat de beveiliging over het algemeen prima in orde is, toch een aantal gebreken kent (zie het overzicht) welke mogelijk uit te buiten zijn. Ze zijn niet allemaal even spannend, waarschijnlijk of gevaarlijk. De belangrijkste gebreken (waar Yubico een oplossing voor zou moeten bedenken of applicaties rekening mee dienen te houden) zijn volgens ons:

- Diverse onderschep- en blokkeeraanvallen (zie sectie 5.3.3, 6.3.4 en 7.3.2). Een aanvaller blokkeert de OTP voordat deze bij de (alle) validatieserver(s) is en gebruikt hem daarna zelf. Dit is een vrij ernstig probleem, waar geen goede oplossing voor is binnen het Yubico OTP-protocol. Bovendien is de aanval op meerdere plekken (bij meerdere onderdelen) toepasbaar. Een mogelijke toekomstige oplossing is het gebruik van een klok met eigen stroomvoorziening (welke dus continu doortelt en redelijk synchroon blijft met de validatieservers) en vervolgens OTP's afkeuren die te oud zijn (huidige tijd min OTP tijd overschrijdt een bepaalde waarde).

- Data wijzigen op Yubikey (zie sectie 4.3.4). Er is wel bescherming tegen deze aanval, maar die wordt standaard niet gebruikt en we denken dan ook dat weinig mensen deze bescherming gebruiken. Dit is zonde, want de aanval vormt wel een beveiligingsrisico welke eventueel zelfs op afstand kan worden toegepast.

- Replay aanval op het antwoord van de validatieserver (zie sectie 6.3.3). Er is wel bescherming mogelijk tegen deze aanval (tenminste in nieuwere versies van het validation server protocol), maar is niet verplicht en wordt ook niet standaard gebruikt. We denken daarom dat deze aanval bij veel applicaties wel zou werken.

- Diverse aanvallen op beschikbaarheid (zie sectie 6.3.6 en 7.3.3). Er zijn diverse aanvallen mogelijk met betrekking tot de beschikbaarheid door het platleggen van validatieservers of de communicatie hiervan. Door het gebruik van meerdere servers, wordt dit wel beperkt, maar het is desondanks nog steeds mogelijk (zeker met maar vijf validatieservers).

De bovengenoemde aanvallen zijn vaak situatie-afhankelijk, voornamelijk met betrekking tot of HTTPS gebruikt wordt of niet. Het gebruik van HTTPS kan veel van de bovengenoemde aanvallen voorkomen of moeilijker maken en we raden daarom dan ook alle applicaties aan dit te doen (zowel bij communicatie met gebruiker als met applicatie). Ook zou het voor Yubico aan te raden zijn om meer validatieservers te gebruiken. Dit maakt diverse aanvallen een stuk lastiger.

Wat ons betreft is de Yubikey (ondanks een aantal gebreken) een goed authenticatiemiddel ter vervanging of als toevoeging op het standaard wachtwoord. Het is in ieder geval veiliger en voor een groot deel is de beveiliging prima in orde. Er zijn helaas wel een klein aantal gebreken, maar deze zijn vaak makkelijk op te lossen of te beperken door geen genoegen te nemen met de minimale beveiliging, maar iets verder gaan (bijvoorbeeld HTTPS gebruiken). Ook doen applicaties er goed aan om de implementatie goed te controleren en zoveel mogelijk van de beveiligingen die het protocol biedt gebruik te maken (denk aan controleren van OTP en nonce bij het antwoord van de validatieserver, zie sectie 6.3.3).

De Yubikey is met name geschikt voor (online) applicaties waarvoor een gemiddelde beveiliging vereist is. Qua beveiliging is de Yubikey te plaatsen tussen bijvoorbeeld het standaard wachtwoord en Public Key Infrastructure (PKI) met PIN authenticatie. De Yubikey zou bijvoorbeeld prima gebruikt kunnen worden voor het inloggen op een PayPal account. Bij PayPal is zeker (minimaal) een gemiddelde beveiliging gewenst, aangezien er geldzaken bij betrokken zijn. Normaal gesproken wordt alleen een wachtwoord gevraagd, wat niet erg veilig is. Met de Yubikey erbij wordt de beveiliging een stuk sterker zonder het erg omslachtig te maken (althans volgens diverse reviews met betrekking tot het gebruiksgemak).

Een lokaal/offline voorbeeld waarvoor de Yubikey goed gebruikt zou kunnen worden is in de zorg voor de toegang tot (lokale) dossiers. Nu gebeurt het vaak dat mensen, als ze even weggaan, het systeem ingelogd laten (om tijd en moeite te besparen). De Yubikey kan in dit geval een gemakkelijke, maar toch veilige oplossing zijn om de beveiliging (van toegang tot dossiers) te vergroten. Wel zou het gebruik van een lokale validatieserver dan zeker aan te raden zijn (om niet al te veel te hoeven vertrouwen op Yubico en tevens een aantal mogelijke aanvallen onbruikbaar te maken).

Bibliografie

[1] Yubikey php webservice class. http://code.google.com/p/yubikey-php-webservice-class/, 2011.

[2] Federal Information Processing Standards Publication 197. Announcing the advanced encryption standard (aes). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001.

[3] Stuart Allman. Encryption and security: the advanced encryption standard. EDN (2002) October, p26-30, 2002.

[4] Kemal Bicakci and Nazife Baykal. One-time passwords: Security analysis using ban logic and integrating with smartcard authentication. ISCIS (2003), LNCS 2869, p794-801, 2003.

[5] Sang-Soo Yeo Binod Vaidya, Jong Hyuk Park and Joel J.P.C. Rodrigues. Robust one-time password authentication scheme using smart card for home network environment. Computer Communications (2011) 34, p326-336, 2011.

[6] David Wagner David Wagner. Analysis of the ssl 3.0 protocol. http://www.schneier.com/paper-ssl.pdf, 1996.

[7] R. Canetti H. Krawczyk, M. Bellare. Hmac: Keyed-hashing for message authentication. http://www.ietf.org/rfc/rfc2104.txt, 1997.

[8] Vincent Rijmen Hans Dobbertin and Aleksandra Sowa. Advanced encryption standard - aes. 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, 2004.

[9] Paul Brusil John Hale. Secur(e/ity) management: A continuing uphill climb. Netw Syst Manage (2007) 15, p525-553, 2007.

[10] Simon Josefsson. Yubikey security evaluation. http://static.yubico.com/var/uploads/pdfs/Security_Evaluation_2009-09-09.pdf, 2009.

[11] Stéphane Manuel. Classification and generation of disturbance vectors for collision attacks against sha-1. Journal Designs, Codes and Cryptography 59, p247-263, 2011.

[12] Dirk Merkel. Yubikey one-time password authentication. Linux journal (2009) 177, p46-55, 2009.

[13] Ing-Yi Chen Neng-WenWang, Han-Chieh Chao and Yueh-Min Huang. A novel user's authentication scheme for pervasive on-line mediaservices. Telecommun Syst (2010) 44, p181-190, 2010.

[14] Leo Laporte Steve Gibson. Security now 143: Yubikey (netcast). http://www.twit.tv/sn143, 2008.

[15] Marries van de Hoef. Inleiding tot de advanced encryption standard. http://www.cs.uu.nl/docs/vakken/b3crp/Symp1001/AES.pdf, 2011.

[16] Elisabeth Oswald Vincent Rijmen. Update on sha-1. A.J. Menezes (Ed.) CT-RSA, LNCS 3376, p58-71, 2005.

[17] Ilsun You. A one-time password authentication scheme for secure remote access in intelligent home networks. KES (2006), Part II, LNAI 4252, p785-792, 2006.

[18] Yubico. Key lifecycle management. http://static.yubico.com/var/uploads/pdfs/Key_Lifecycle_Management_2009-09-09.pdf, 2009.

[19] Yubico. Validation service reliability. http://static.yubico.com/var/uploads/pdfs/Validation_Service_Reliability_2009-09-09.pdf, 2009.

[20] Yubico. Yubikey configuration manual. http://static.yubico.com/var/uploads/pdfs/YubiKey_Configuration_Manual_2009-09-09.pdf, 2009.

[21] Yubico. The yubikey manual. http://static.yubico.com/var/uploads/pdfs/YubiKey_Manual_2010-09-16.pdf, 2010.

[22] Yubico. Low-level library. http://www.yubico.com/low-level-library, 2011.

[23] Yubico. Server replication protocol. http://code.google.com/p/yubikey-val-server-php/wiki/ServerReplicationProtocol, 2011.

[24] Yubico. Validation protocol v2.0. http://code.google.com/p/yubikey-val-server-php/wiki/ValidationProtocolV20, 2011.

[25] Yubico. Validation server. http://www.yubico.com/validation-server, 2011.

[26] Yubico. Web api & clients. http://www.yubico.com/web-api-clients, 2011.