



# Abgabedokument Lab1

## Introduction to Security

183.594 – WS 2020

07.12.2020

Team 15

Name	MatrNr.
Maximilian Hagn	11808237
Pascal Poremba	11809911
Hannes Rokitte	11806914
Paul Schmitt	11714338

# Inhaltsverzeichnis

<b>1</b>	<b>Auf die Plätze, fertig, skripten!</b>	<b>3</b>
1.1	Zip Maze . . . . .	3
1.2	Scrambled Image! . . . . .	4
<b>2</b>	<b>Crack It</b>	<b>6</b>
2.1	Call John* . . . . .	6
2.2	/etc . . . . .	7
2.3	Passwort? . . . . .	8
2.4	Imagemagick . . . . .	10
<b>3</b>	<b>Die Rechnung, bitte.</b>	<b>11</b>
3.1	Block 1-5 . . . . .	11
<b>4</b>	<b>Kommando Marsch!</b>	<b>12</b>
4.1	Stage 1* . . . . .	12
4.2	Stage 2* . . . . .	12
4.3	Stage 3 . . . . .	12
4.4	Stage 4 . . . . .	12
4.5	Stage 5 . . . . .	13
4.6	Stage 6 . . . . .	13
4.7	Stage 7 . . . . .	13
4.8	Stage 8 . . . . .	13
4.9	Stage 9 . . . . .	14
4.10	Stage 10 . . . . .	14
4.11	Stage 11 . . . . .	14
4.12	Stage 12 . . . . .	14
<b>5</b>	<b>Web-Challenges</b>	<b>15</b>
5.1	Next Door* . . . . .	15
5.2	You Got Selected! . . . . .	16
5.3	All Inclusive . . . . .	16

# 1 Auf die Plätze, fertig, skripten!

## 1.1 Zip Maze

In einem ersten Schritt wurden alle Zip Dateien im Angabeordner mit dem Befehl in **Listing 1** rekursiv entpackt. **Listing 1** zeigt, dass alle gefundenen Dateien mit der Endung .zip gesucht werden. Solange noch gefundenen Dateien im Speicher sind werden diese nacheinander, in das ursprüngliche Dateiverzeichnis entpackt. Da durch das entpacken neue .zip Dateien gefunden werden können, wird dieses Prozedere wiederholt, bis alle Dateien entpackt sind.

```
1 while true; do find . -name "*.zip" | while read filename; do
    unzip -o -d "$(dirname "$filename")" "$filename"; done;
```

Listing 1: Alle .zip Dateien rekursiv entpacken

Anschließend konnte das EsseFlag bereits vom Mac Os Finder gefunden werden. Das File könnte jedoch auch mit dem in **Listing 2** beschriebenen Kommando über die Befehlszeile ausgelesen werden. **Listing 2** zeigt einen grep ( <https://wiki.ubuntuusers.de/grep/> ) Befehl der rekursiv alle Dateien nach dem regulären Ausdruck "^flag"durchsucht.

```
1 grep -r "^flag" /
```

Listing 2: In allen Dokumenten nach 'flag' suchen

**Lösung: EsseFlag{jzbdkNYtMpsu9XJ4kT6x}**

## 1.2 Scrambled Image!

Um das durcheinander gebrachte Bild wiederherzustellen wurde die Programmiersprache Python mit dem Random Module ( [https://www.w3schools.com/python/module\\_random.asp](https://www.w3schools.com/python/module_random.asp) ) und dem Image Module ( <https://pillow.readthedocs.io/en/stable/reference/Image.html> ) verwendet. Der Großteil des Codes wurde in einem Stackoverflow Beitrag ( <https://stackoverflow.com/questions/17777760/mixing-pixels-of-an-image-manually-using-python> ) gefunden.

In **Abbildung 1** ist das Ausgangs Bild zu erkennen. Dieses wird von dem, auf **Seite 5** beschriebenen Python Programm, eingelesen und bearbeitet. **Abbildung 2** zeigt das Endresultat, dass von dem Programm ausgegeben wird. Es enthält das Flag „EsseFlag-cWap0osv56aOhGJksqtE“.

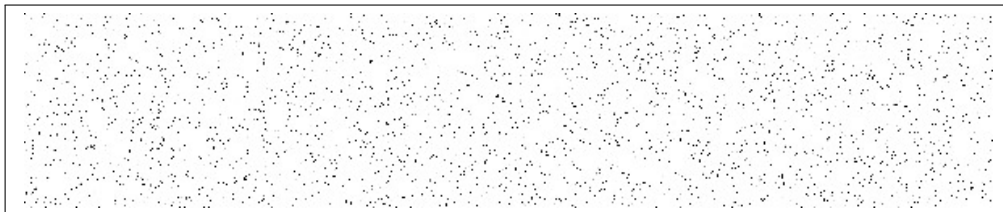


Abbildung 1: Scrambled Input Image



Abbildung 2: Scrambled Output Image

Der verwendete Code wird in Listing 3 dargestellt. Zuerst werden die oben erwähnten Module importiert. Anschließend wird Ausgangsbild geöffnet und der Random Seed auf das meist verwendete Passwort ("123456", Quelle: <https://t3n.de/news/beliebteste-passwort-deutschen-1235494/>) gesetzt. Dann wird die Höhe und Breite des Bildes in den Variablen `w`, `h` festgehalten. Des Weiteren wird ein Array `pxs` erstellt, dass später die einzelnen Pixel des Bildes enthält. Nun wird über alle Pixel des Bildes iteriert und diese werden nach und nach an das Array `pxs` angehängt. Die Pixel werden nun in einem Temporären Array `idx` gespeichert, da das Kommando `random.shuffle()` das Array selbst verändert. Nun wird ein neues Array `out`, mit der Größe von `pxs` erstellt. In einer `for`-Schleife, werden nun alle ursprünglichen Pixel an die Position nach dem `random.shuffle()` Befehl getauscht. Die `out` Liste enthält nun alle zurück getauschten Pixel. Mit Hilfe von `Image.new()` wird ein neues Bild mit der gleicher Größe des Ausgangsbildes erstellt. Durch einen Iterator `pxIter` der alle Pixel wird nun ein Image `outImg` erstellt. Dieses wird schlussendlich abgespeichert.

```
from PIL import Image
2 import random

4 img = Image.open("img.png")
  random.seed("123456")

6
  w, h = img.size
8 pxs = []
  for x in range(w):
10     for y in range(h):
        pxs.append(img.getpixel((x, y)))
12
  idx = list(range(len(pxs)))
14 random.shuffle(idx)

16 out = list(range(len(pxs)))
  cur = 0
18 for i in idx:
        out[i] = pxs[cur]
20     cur += 1
  pxs = out
22
  outImg = Image.new("RGB", img.size)
24 pxIter = iter(pxs)
  for x in range(w):
26     for y in range(h):
        outImg.putpixel((x, y), pxIter.__next__())
28     outImg.save("unscrambled.png")
```

Listing 3: Pixel werden mittels Random Seed zurückgetauscht

**Lösung: EsseFlagcWap0osv56aOhGJksqtE**

## 2 Crack It

### 2.1 Call John\*

Beim ersten Beispiel der Crack It Reihe war ein mit Passwort geschütztes ZIP-File gegeben. Es wurde sich dafür entschieden das Beispiel mit dem Commandlinetool `fcrackzip` zu versuchen. Da ein Bruteforceangriff mit verschiedenen Einstellungen nicht funktionierte, griffen wir auf ein Dictionary zurück. Das dictionary heißt 'rockyou' und wurde von <https://github.com/mishrasunny174/WordLists> heruntergeladen. Mit dem Befehl `fcrackzip --dictionary crackit_team15.ba86c799edbf35f4280685729814ce5.zip -up rockyou.txt` konnte das Passwort dann letztendlich gecrackt werden.

**Lösung: fireman**

## 2.2 /etc

Nach der Call John Aufgabe konnte das ZIP-File entpackt werden. In dem File war ein Ordner 'data' und ein weiteres ZIP-Archiv mit dem Namen etc. In dem 'data'-Ordner wiederum war ein passwd und ein shadow File. Der Benennung und dem Inhalt nach war es ein Shadow File, welches Benutzernamen und deren verschlüsselte Passwörter enthält. Zuerst wird die Information der beiden Files mithilfe von `unshadow` kombiniert. Das Ergebnis haben wir in ein neues Textfile gespeichert. Danach wurde mithilfe von 'John the Ripper' (Commandlinebefehl `john`) und einer wordlist (`rockyou.txt` aus der vorherigen Aufgabe) das Passwort geknackt: `john --wordlist=rockyou.txt passwords.txt`.

**Lösung: trustno1**

## 2.3 Passwort?

Um diese Aufgabe abzuschließen wurden die Tools John the Ripper ( <https://www.openwall.com/john/> ) und Hashcat ( <https://hashcat.net/hashcat/> ) verwendet. Des Weiteren wurde die Wordlist rockyou.txt ( <https://github.com/mishrasunny174/WordLists> ) verwendet. Weitere Informationen zur Durchführung wurden unter <https://madcityhacker.com/2018/11/04/cracking-keepass-databases-with-hashcat/> gefunden.

Im ersten Schritt wurde das Master Passwort der .kdbx Datenbank mit dem in Listing 4 aufgezeigten Befehl extrahiert. Der extrahierte Hash ist in Listing 5 zu sehen. Der Name "mySecretDB:" muss für folgende Schritte entfernt werden und gehört nicht zum eigentlichen Hash, sondern ist nur ein Identifikator im Terminal.

```
python keepass2john.py mySecretDB.kdbx
```

Listing 4: Befehl um den Hash des Master Passworts zu extrahieren

```
mySecretDB:$keepass$*2*60000*222*e8b0e1859210ccf8e9a3be8e0df7
d57756d349fec0408eaabe11d15bcfc603f*1be536b2a87911ac6c91de59
5818330184d20f538764e4f71b9d0e884bbe57e4*83e440abbab38ecb3f0a
c4832b1fde05*e09d87e96842e3ce2944fe92a71369bd8b908818531df87f
7f93e3cbc30b6ad4*24956253f60ab0a2bf8c66333ba040b9c43216efb2c2
7f0930b8b4f1040ad4b3
```

Listing 5: Master Password Hash

Anschließend wurde eine neue Datei keepass.txt erstellt, in dem sich der Hash des Master Passworts befindet. Die Datei wurde zusammen mit der rockyou.txt im Hashcat Ordner abgespeichert. Mit dem in Listing 6 angeführten Befehl konnte schlussendlich das Passwort "milkshake" herausgefunden werden. -a 0 bezieht sich auf den "Attack Mode", in diesem Fall wird eine Attacke basierend auf einer Wortliste ausgeführt. -m 13400 definiert den verwendeten "Hash Mode". Für .kdbx Datenbanken wird KeePass 1 (AES/Two-fish) und Keepass 2 (AES) verwendet (Quelle: <https://madcityhacker.com/2018/11/04/cracking-keepass-databases-with-hashcat/> ). In keepass.txt befindet sich der zuvor extrahierte Passwort Hash und in rockyou.txt eine bekannte Wortliste.

```
hashcat -a 0 -m 13400 keepass.txt rockyou.txt
```

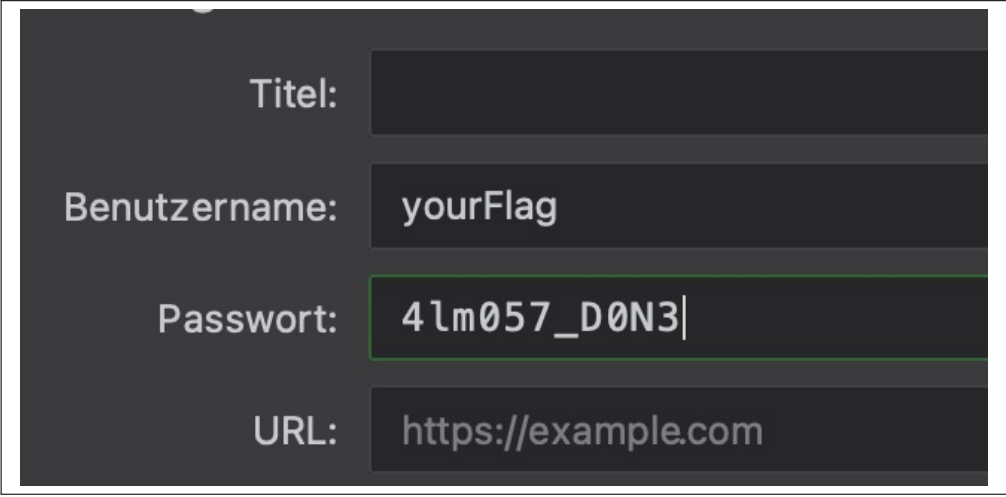
Listing 6: Befehl um Passwort zu cracken



Die Ausführung des Befehls in Listing 6 führt zu der Ausgabe in Listing 7. Am Schluss des Hashes wird nun das gefundene Passwort "milkshake" ausgegeben. Dieses konnte anschließend verwendet werden um die .kdbx Datenbank zu öffnen. Diese wurde über das Programm <https://keepassxc.org> geöffnet. So konnte das Passwort für die Passwort?.zip Datei herausgefunden werden. Das gefundene Passwort / das gefundene Flag ist in Abbildung 3 ersichtlich.

```
$keepass$*2*60000*222*e8b0e1859210ccf8e9a3be8e0df7d57756d349f  
ec0408eaabe11d15bcfcb603f*1be536b2a87911ac6c91de595818330184d  
20f538764e4f71b9d0e884bbe57e4*83e440abbab38ecb3f0ac4832b1fde0  
5*e09d87e96842e3ce2944fe92a71369bd8b908818531df87f7f93e3cbc30  
b6ad4*24956253f60ab0a2bf8c66333ba040b9c43216efb2c27f0930b8b4f  
1040ad4b3:milkshake
```

Listing 7: Master Password Hash



The image shows a screenshot of a KeePassXC entry form. The form has a dark background with light-colored text. The fields are labeled 'Titel:', 'Benutzername:', 'Passwort:', and 'URL:'. The 'Benutzername:' field contains the text 'yourFlag'. The 'Passwort:' field contains the text '4lm057\_D0N3|' and is highlighted with a green border. The 'URL:' field contains the text 'https://example.com'.

Abbildung 3: Gefundenes Passwort / Flag in der .kdbx Datenbank

**Lösung:** 4lm057\_D0N3

## 2.4 Imagemagick

Um diese Aufgabe zu lösen wurde die Software ImageMagick ( <https://legacy.imagemagick.org/> ) verwendet. Es sollen 225 verschiedene .png Bilder, die nach Reihen und Spalten beschriftet sind, zu einem Bild zusammengefügt werden. Die Software ImageMagick bietet das Kommando `montage` bei dem mehrere Bilder zu einem beliebigen Raster hinzugefügt werden können. Der verwendete Befehl ist in [Abbildung 8](#) angeführt. `ls *.png` listet alle Dateien mit der Endung .png und `sort -V` sortiert diese nach Versionsnummer, sprich row-1-col-1, row-1-col-2, usw. ( Quelle: <https://wiki.ubuntuusers.de/sort/> ). Dieser Befehl muss verwendet werden, da die Bilder sonst falsch aneinander gereiht werden. `-tile 15x15` erstellt ein Grid mit 15 Spalten und 15 Reihen, damit alle 225 Bilder reinpassen. `-geometry 15x15` legt fest, dass jedes Bild 15 Pixel hoch und 15 breit ist. `../out.png` bestimmt, wo das zusammengefügte Bild gespeichert werden soll. Die Lösung ist in [Abbildung 4](#) zu sehen.

```
montage $(ls *.png | sort -V) -tile 15x15 -geometry 15x15 ../out.png
```

Listing 8: Bilder mit ImageMagick zusammenfügen



Abbildung 4: Zusammengefügtes ImageMagick Bild

Dieser QR-Code ergibt gescannt das Flag `SSm9vX0NSNENrM0RfMTc=`. Dieses Flag ist Base64 verschlüsselt und wurde deswegen mit einem Base64 Decoder ( [base64decode.org](#) ) entschlüsselt. Entschlüsselt ergibt sich daraus die Lösung: `"Joo_CR4Ck3D_17"`.

**Lösung: Joo\_CR4Ck3D\_17**

## 3 Die Rechnung, bitte.

### 3.1 Block 1-5

Bei diesen Aufgaben wurde ein PDF mit geschwärzter IBAN gegeben. Es war also klar, dass das PDF zu Text konvertiert werden muss. Dafür wurde `pdftotext` in der commandline verwendet. Mit dem Befehl wurde ein `.txt` erzeugt, in welchen die ungeschwärzte IBAN zu sehen ist. Aus dieser wurden dann die jeweilig geforderten Blöcke herausgelesen. Für Block 2 war nach der Erzeugung des Fließtextes nur 0— zu erkennen.

**Lösung: AT20 0— 0501 7669 7980**

## 4 Kommando Marsch!

Bei dieser Aufgabengruppe geht es darum für den jeweils nächsten Benutzer das Passwort herauszubekommen. Es beginnt wenn man sich mit dem Benutzer 'is\_team15' und dem gegebenen Passwort auf den 10.10.10.100 Server verbindet. Hier bekommt man direkt nach dem Verbinden in einer Ausgabe die Anmeldedaten für den 'stage00' Benutzer.

### 4.1 Stage 1\*

Nach dem anmelden mit Benutzer 'stage00' findet man in dessen Home-Verzeichnis eine Datei 'Cookie', in der das Passwort für 'stage01' in Plaintext enthalten ist.

**Lösung:** uGauth3Shaip

### 4.2 Stage 2\*

Im Verzeichnis von 'stage01' ist mit dem normalen `ls`-Befehl keine Datei aufzufinden. Listet man jedoch auch versteckte Dateien auf, findet man schnell die '.hidden'-Datei in der das Passwort in Plaintext enthalten ist.

**Lösung:** HaTh5Quahvoh

### 4.3 Stage 3

Meldet man sich nun mit 'stage02' an, findet man wieder eine versteckte Datei, diesmal aber eine '.gz'. Man kann sie einfach mit `gunzip .compressed.gz --stdout` auslesen.

**Lösung:** iehet7ieW7ye

### 4.4 Stage 4

Der Benutzer 'stage03' findet ähnlich wie im vorherigen Beispiel eine versteckte Datei: '.compressed.unknown.rar'. Um hier Einsicht zu erlangen wird `uncompress --stdout .compressed.unknown.rar` verwendet.

**Lösung:** Dae7queph2ga

## 4.5 Stage 5

'stage04' findet ein .encrypted File, dessen inhalt auf eine base64-Verschlüsselung schließen lässt. Da wir auf dem Server keine rechte haben neue files zu erzeugen wurde der folgende Befehl lokal mit einer kopie des 'encrypted'-Files ausgeführt: `base64 --decode .encrypted > decrypted.zip`. In der entschlüsselten ZIP-Datei ist ein File welches das Passwort für 'stage05' enthält.

**Lösung:** laoloh7iePei

## 4.6 Stage 6

In Stage 6 ist ein .boxed file gegeben. Dieses haben wir mittels base64 codiert. Danach wurde lokal mit `base64 --decode passwd > file.zip` ein ZIP-File erzeugt, welches ein base64-Codiertes File enthält. Decodiert man dieses erhält man die Anmeldedaten für 'stage06'.

**Lösung:** eaNgooJ3ieGh

## 4.7 Stage 7

Meldet man sich mit Benutzer 'stage06' an ist ein Labyrinth aus Ordnern in dem Folder 'hidden' vorzufinden. Dieses haben wir uns mit dem Befehl `ls -Ral` ausgegeben. Das Argument '-R' ist dabei für die Rekursive Ausgabe zuständig. Mit etwas Zeit würde dann das File in einem Ordner gefunden, welches das Passwort für die nächste Stage beinhaltet.

**Lösung:** Kahp7io5Zie5

## 4.8 Stage 8

Hier war nur ein 'shadow.bak'-File gegeben. Der Name und Inhalt ließen auf ein Shadow File schließen. Um das zu entschlüsseln wurde ein passendes passwd-File benötigt. Das fanden wir in '/etc/passwd'. Lokal wurden die beiden Files zuerst mittels `unshadow` kombiniert und dann mit `john --wordlist=rockyou.txt passwords.txt` geknackt. Die Wordlist ist die gleiche wie in dem Crack It Beispiel '/etc'.

**Lösung:** 4478

## 4.9 Stage 9

In diesem Beispiel war wieder eine Ordnerstruktur gegeben die wir uns mit `ls -Ra` ausgegeben haben. Es wurden schnell zwei Dateien gefunden, mit deren Hilfe das gesuchte Passwort zusammengesetzt werden konnte.

**Lösung:** vahn5AeYooHa

## 4.10 Stage 10

Hier konnte der zweite Teil des Passwortes direkt der Datei im Homeverzeichnis entnommen werden. Der erste Teil des Passwortes wurde in einer versteckten Datei in dem Verzeichnis `'stage09b/.bash'` gefunden.

**Lösung:** WooB3shaec4w

## 4.11 Stage 11

Der Benutzer `'stage10'` findet in seinem Home-Verzeichnis ein kleines Programm, dass den Inhalt einer gewünschten Datei (als Parameter der Name der Datei) im `'secrets'`-Ordner ausgibt. Nach Analyse des Algorithmus bzw. der Folder wurde mit `./sdv .p1` der erste Teil des Passwortes gefunden. Nach langen herumprobieren konnte der zweite Teil mithilfe des Programmes und unter verwendung von Wildcats gefunden werden: `./sdv ../.other_secrets/*.*`

**Lösung:** phaith6Ieyoh

## 4.12 Stage 12

Im letzten Beispiel dieser Reihe war wieder ein kleines Programm der Ausgangspunkt. Auch hier wurde, ähnlich dem letzten Beispiel, nach einer Analyse der Ordnerstruktur (soweit die Berechtigungen vorhanden waren) das Passwort wie folgt gefunden: `./sdv2 ../.secrets.bak/p`.

**Lösung:** Kahphee4Ahka

## 5 Web-Challenges

### 5.1 Next Door\*

Bei der ersten Webchallenge betrachtet man zunächst einmal eine schlichte Login-Seite. Mit dem Hinweis darauf, dass eventuell Zwischenschritte zur Wiederherstellung gesichert sind, die man ungeschützt aufrufen kann, macht man sich auf die Suche nach ungesicherten Back-Up Dateien. Bei Aufruf der Seite `localhost:12345/index.php~` wird die `index.php` Datei nicht ausgeführt sondern heruntergeladen. Innerhalb der Datei befindet sich ein Kommentar, welches beschreibt unter welcher URL sich eine Log-File eines Logins befindet. Der relative Datenpfad lautet `/5fca6ab17d1dadf9b45b616fa306fce3.log`.

Dort finden sich folgende Zugangsdaten:

username: johnny\\_atkins, pw: 405b96886b9115ae1dca2c4d7f847288

Decodiert lautet das Passwort: cashmoney

Nach dem Einloggen stellt man in der URL den Query-Parameter auf `id=4` um und gelangt somit auf der Seite, welche die Flag enthält.

**Lösung: FLAG\_f399bac294b2b8db4lW4Y5\_CHECK\_4UTH0R1z4t10n**

## 5.2 You Got Selected!

Wie zuvor bei der ersten Web-Challenge startet auch "You Got Selected!" mit einer normalen Login-Seite. Auf Grund des Namens der Challenge kann davon ausgegangen werden, dass die Lösung mit sql-injections zutun hat. Um einen erfolgreichen Login zu erreichen wurde zuerst das im Hintergrund ausgeführte sql-Statement angenommen. Basierend auf dieser Annahme wurde eine sql-injection entwickelt, welche das Zusammenspiel der beiden Felder *Username* und *Password* ausnutzt. Mit dem Username "\ " und dem Passwort "or 1;#" kann man einen erfolgreichen Login ohne validen Zugangsdaten erreichen. Nach dem Login wird man zu einer einfachen Seite, auf der man nach Produkten suchen kann, geleitet. Hier war naheliegend, das gleiche Verfahren wie beim Login zu verwenden. Nach wenigen Eingaben in der Suchleiste war klar, dass sql-injections möglich sind, aber in Anfragen Keywords von sql gelöscht werden. Um diesen Mechanismus zu umgehen, wurden jegliche Keywords in der sql-injection doppelt verfasst (z.B: *Seleselectect*). Mit ein wenig Erfahrung kann man schnell feststellen in welchem sql-Dialekt die Datenbank verfasst ist. Da es sich wahrscheinlich um eine MySql Datenbank handelt, konnte durch die sql-injection eine Liste aller Tables ausgegeben werden. Nach wenigen Versuchen wurde im Table secrets eine Flag gefunden.

```
Username: \  
Password: or 1;#
```

Listing 9: Befehle zum Einloggen

```
Weight' UNUNIONION SESELECTLECT TABLE_NAME, TABLE_SCHEMA, 3,  
4, 5 FRFROMOM information_schema.tables;#  
  
Weight' UNUNIONION SESELECTLECT flag, 2, 3, 4, 5 FRFROMOM  
secrets;#
```

Listing 10: Ausgabe aller Tabellen und Abfrage der Flag

**Lösung: FLAG\_c71f52b9d79836210HH\_N0\_3V3RytH1N95\_53L3Ct48l3**

## 5.3 All Inclusive

Diese Aufgabe wurde nicht bearbeitet.