Überschrift

Veronika	Oliver	Lucasz
Tyshchenko	Oberdick	Kotula
Matrikel	Matrikel	Matrikel

11. Oktober 2025

Zusammenfassung

Beschreibung zur APL Programmierung 2 der Gruppe HH05

Inhaltsverzeichnis

1 Story

1.1 Vorgeschichte:

Dein Rechner wurde von irgendjemandem kompromittiert.
Freundlicherweise wurden dir einige hinweise hinterlassen.
Folge ihnen um die Kontrolle zurück zu erhalten

Du schaltest deinen Computer ein und siehst nur eine eigenartige Eingabemaske vor dir. Irgendwer Irgendetwas scheint ihn verändert zu haben. Jetzt mußt du die Kontrolle zurückerlangen. 'Freundlicherweise' wurden dir einige Hinweise hinterlassen. Finde und nutze sie!

1.2 Levelvorbereitung

1: erstellen eines zufälligen Verschlüsselungs Key in einer Variable oder speichern in einer Datei 'Dateiname Sec_Key_HH05.key' zur weiteren nutzung.

2: evtl könnte auf die Speicherung des Key in einer Datei verzichtet werden, wenn die Eingaben für die Level 1-?? vor dem Level Start generiert werden, dann könnte der Key in einer Variable verbleiben

3: Verstecken des Key in einer Zufällig ausgewählten Bild-Datei (Eingabedaten für Level 2) Bild-Datei immer wieder neu erstellen und die alte überschreiben.

4: generieren einer Log-Datei Bsp. wie die ausgabe von
netstat -l — grep LISTEN
mit zufällig ergänzten Portnummern, welche nicht offen sein sollten.

5: Nutzen des Key für die Verschlüsselung der Log-Datei auf Bitebene (Eingabedaten für Level 4)

6: ?

2 Levelbeschreibung

2.1 Level 1

Hey Buddy, ich habe jetzt die Kontrolle.
Deine Dateien sind verschluesselt.
Wenn du dein Passwort wiederhaben willst, folge den Anweisungen.
Hier ist mein Wallet: Diese Cockies sind nicht lecker!

- Schaue die Webseite an und danach stelle fest, sind die Cockies lecker und was die Gangster mit ASCII zu tun haben.

2.2 Level 2

Your encrypted file wird hier benannt, finde das unten:
Nachrichten ansehen

- Schreibe deine Lösung so, dass du die Endausgabe Datei liest und die UTC-Zahlen ersetzt.

2.3 Level 3

Hi,
das ist zwar kein CTF, aber ein flag ist trotzdem zu suchen

- schau mal im Bild!
- suche nach dem flag=
- Eingabedaten sind der Dateiname des Bildes
- mit jedem Bild oder neuanfang bekommst du auch eine andere flag
- speichern kann nicht schaden, Bsp. game.key
- als encoding wurde 'ISO-8859-1' verwendet
- in einem Linux Terminal funktioniert auch der Befehl 'strings [Dateiname]' v

2.4 Level 4

Du hast jetzt einen Dateinamen static/text.crypt, schon mar reingeschaut?
zur kontrolle, zeig mir die Zeichen 20 - 70

- kannst du den Inhalt lesen?
- Hattest du die flag gespeichert? Bsp. game.key?
- Bitweises XOR schon mal gesehen?
- als Rückgabewert die Zeichen 20 - 70 als String zum alsolvieren dieses Level sollten erstmal reichen
- Denke drann den Inhalt des Key.File zu nutzen, nicht den Dateinamen
- den Key kannst du auch mehrfach hintereinander schreiben, falls er nicht lang genug ist
- trotzdem solltest du die komplette Datei bearbeiten und auch wieder speichern. Bsp. ausgabe_encrypt.txt

2.5 Level 5

Level 4: Logfile-Analyse
Du hast ein Logfile erhalten, das verdächtige Netzwerkaktivitäten enthält.
Deine Aufgabe: Extrahiere alle Ports aus dem Logfile und bestimme ihren Status.
Achte auf Schlüsselwörter wie secure, attempt, filtered.
Lernziele: Textanalyse, Reguläre Ausdrücke, Listen und Dictionaries

- Nutze re.findall(rport (+)", line), um Portnummern zu extrahieren.
- Verwende line.lower().strip(), um die Zeile zu normalisieren.
- Prüfe mit if, ob bestimmte Schlüsselwörter enthalten sind.

2.6 Level 6

Erstmal nur ein Platzhalter!

- ??

3 Programmcode

3.1 Excape Room

Der Raum:

```

import random
import string
from EscapeRoom import EscapeRoom

import time
import re
import lib.stego as STEGO # Funktionssammlung Oliver Level 3
import lib.crypt as CRYPT # Funktionssammlung Oliver Level 4

class Gruppenarbeit_kombiniert(EscapeRoom):

    def __init__(self, response=None):
        super().__init__(response)

        self.set_metadata("Veronika, Lucasz & Oliver", __name__)

        ## Fuer Level 3-4
        self.key = CRYPT.schluessel_erstellen(30) #schluessel erstellen
        self.bild = "static/KEY.jpg"
        STEGO.random_bild(self.bild) # zufaelliges Bild ermitteln und umkopieren
        STEGO.im_bild_verstecken(self.bild , self.key)
        self.verschluesselt = "static/text.crypt"
        CRYPT.schluesselanwendung_datei("static/originale/test.log" ,self.verschluesselt , self.key )

        ## Fuer Level 5-6

        self.add_level(self.create_level1()) # Veronika
        self.add_level(self.create_level2()) # Veronika
        self.add_level(self.create_level3()) # Oliver
        self.add_level(self.create_level4()) # Oliver
        self.add_level(self.create_level5()) # Lucasz
        self.add_level(self.create_level6()) # Lucasz

    ### LEVELS ###
    # Level 1
    def create_level1(self):
        cookie = self.ascii_cookie()
        task_messages = [
            "Hey Buddy, ich habe jetzt die Kontrolle. ",
            "Deine Dateien sind verschluesselt. ",
            "Wenn du dein Passwort wiederhaben willst, folge den Anweisungen.",
            "Hier ist mein Wallet: Diese Cockies sind nicht lecker!"
        ]

        hints = [
            "Schaue die Webseite an und danach stelle fest, sind die Cockies lecker und was die Gangster mit ASCII zu tun haben."
        ]

        self.response.set_cookie("hint", cookie)

        return {
            "task_messages": task_messages,
            "hints": hints,
            "solution_function": self.solution_level1,
            "data": cockie
```

```
        }

    # Level 2
    def create_level2(self):
         # Define file paths
        path = "static/template.txt"
        output_path = "static/output.txt"

        # Define placeholders
        self.placeholders = ["{key1}", "{key2}", "{key3}"]

        # Generate decrypted file
        decrypted_path = self.generate_decrypted_file(path, output_path)

        # Count occurrences for internal testing
        solution = self.count_decrypted_words(output_path)
        print("Level 2 solution:", solution)  # z.B."343"

        # Messages for the user
        task_messages = [
            "Your encrypted file wird hier benannt, finde das unten:",
            f"<a href='{decrypted_path}' target='_blank'>Nachrichten ansehen</a>"
        ]

        hints = [
            "Schreibe deine Loesung so, dass du die Endausgabe Datei liest und die UTC-
    Zahlen ersetzt."
        ]

        return {
            "task_messages": task_messages,
            "hints": hints,
            "solution_function": self.count_decrypted_words,  # This should be your checker
            "data": decrypted_path
        }

    # Level 3
    def create_level3(self):
        task_messages = [
            "  <img src=" + self.bild + " alt='The Key you looking for' height='150'/> ",
            "Hi,",
      "das ist zwar kein CTF, aber ein flag ist trotzdem zu suchen",
        ]
        hints = [
            "schau mal im Bild!",
            "suche nach dem flag= ",
            "Eingabedaten sind der Dateiname des Bildes",
            "mit jedem Bild oder neuanfang bekommst du auch eine andere flag",
            "speichern kann nicht schaden, Bsp. game.key",
            "als encoding wurde 'ISO-8859-1' verwendet",
            "in einem Linux Terminal funktioniert auch der Befehl 'strings [Dateiname]' "
        ]
        return {"task_messages": task_messages, "hints": hints, "solution_function": STEGO.
    im_bild_finden, "data": self.bild}

    # Level 4
    def create_level4(self):
        task_messages = [
            "Du hast jetzt einen Dateinamen " + self.verschluesselt + ", schon mar
    reingeschaut?",
            "zur kontrolle, zeig mir die Zeichen 20 - 70"
        ]
        hints = [
            "kannst du den Inhalt lesen?",
            "Hattest du die flag gespeichert? Bsp. game.key?",
```

```
            "Bitweises XOR schon mal gesehen?",
            "als Rueckgabewert die Zeichen 20 - 70 als String zum alsolvieren dieses Level sollten erstmal reichen",
            "Denke drann den Inhalt des Key.File zu nutzen, nicht den Dateinamen",
            "den Key kannst du auch mehrfach hintereinander schreiben, falls er nicht lang genug ist",
            "trotzdem solltest du die komplette Datei bearbeiten und auch wieder speichern. Bsp. ausgabe_encrypt.txt"
        ]
        return {"task_messages": task_messages, "hints": hints, "solution_function": CRYPT.entschluesseln, "data": self.verschluesselt}

    # Level 5
    def create_level5(self):
        log_data = """
        Secure connection established on port 443
        Unauthorized access attempt on port 8080
        Port 22 is filtered
        Connection accepted on port 8443
        Unknown activity on port 9999
        """

        parsed_ports = self.parse_logfile(log_data)
#        self.set_solution("malware_ports", parsed_ports)

        task_messages = [
            "<b> Level 4: Logfile-Analyse</b>",
            "Du hast ein Logfile erhalten, das verdaechtige Netzwerkaktivitaeten enthaelt.",
            "Deine Aufgabe: Extrahiere alle Ports aus dem Logfile und bestimme ihren Status.",
            " Achte auf Schluesselwoerter wie <i>secure</i>, <i>attempt</i>, <i>filtered</i>.",
            " Lernziele: Textanalyse, Regulaere Ausdruecke, Listen und Dictionaries"
        ]

        hints = [
            " Nutze <code>re.findall(r\"port (\\d+)\", line)</code>, um Portnummern zu extrahieren.",
            " Verwende <code>line.lower().strip()</code>, um die Zeile zu normalisieren.",
            " Pruefe mit <code>if</code>, ob bestimmte Schluesselwoerter enthalten sind."
        ]

        return {
            "task_messages": task_messages,
            "hints": hints,
            "solution_function": self.check_ports_level4,
            "data": log_data
        }

    # Level 6
    def create_level6(self):
        task_messages = [
            "  <img src=" + self.bild + " alt='The Key you looking for' height='200'/> ",
            "Hi,",
      "das ist zwar kein CTF, aber ein flag ist trotzdem zu suchen"
        ]
        hints = [
            "schau mal im Bild!",
            "suche nach dem flag= ",
            "Eingabedaten sind der Dateiname des Bildes",
            "mit jedem Bild oder neuanfang bekommst du auch eine andere flag",
            "speichern kann nicht schaden, Vorschlag game.key",
            "als encoding wurde 'ISO-8859-1' verwendet",
            "in einem Linux Terminal funktioniert auch der Befehl 'strings [Dateiname]' "
        ]
```

```
        return {"task_messages": task_messages, "hints": hints, "solution_function": STEGO.
    im_bild_finden, "data": self.bild}

    ### Hilfsfunktionen ###

        # Level 1. Aufgabe
    def ascii_cockie(self):
        return "67 111 111 107 105 101 109 111 110 115 116 101 114"

        # Level 2. Aufgabe
    def generate_decrypted_file(self, template_path, output_path):
        # Generate UTCs and save as instance variable
        self.utc_list = [
            f"-{self.random_utc_timestamp()}" for _ in self.placeholders]

        with open(template_path, "r", encoding="utf-8") as f:
            text = f.read()

        for ph, utc in zip(self.placeholders, self.utc_list):
            text = text.replace(ph, utc)

        with open(output_path, "w", encoding="utf-8") as f:
            f.write(text)

        return output_path

    @staticmethod
    def random_utc_timestamp(start_year=2000, end_year=2025):
        start = int(time.mktime(time.strptime(
            f"{start_year}-04-12", "%Y-%m-%d")))
        end = int(time.mktime(time.strptime(f"{end_year}-10-31", "%Y-%m-%d")))
        return random.randint(start, end)

    # Level 5. Aufgabe
    def parse_logfile(self, log_text):
        results = []
        lines = log_text.strip().split("\n")

        for line in lines:
            line = line.lower().strip()
            matches = re.findall(r"port (\d+)", line)
            for match in matches:
                port = int(match)
                if "secure" in line or "accepted" in line:
                    status = "open"
                    reason = "secure/accepted"
                elif "attempt" in line or "exposed" in line or "unauthorized" in line:
                    status = "open"
                    reason = "attempt/exposed/unauthorized"
                elif "filtered" in line:
                    status = "closed"
                    reason = "filtered"
                else:
                    status = "closed"
                    reason = "default"

                results.append({
                    "port": port,
                    "status": status,
                    "reason": reason,
                    "raw_line": line
                })

        return results

```

```
241
242     ### SOLUTIONS ###
243
244         # Level 1. Loesung
245     def solution_level1(self, cockie):
246         return "".join(chr(int(n)) for n in cockie.split())
247
248         # Level 2. Loesung
249     def count_decrypted_words(self, output_path):
250         # Datei lesen
251         with open(output_path, "r", encoding="utf-8") as f:
252             text = f.read()
253
254         # Alle UTCs im Text finden
255         utc_list = re.findall(r"-\d+", text)
256
257         # Vorkommen zaehlen
258         counts = {utc: text.count(utc) for utc in utc_list}
259
260         for utc, count in counts.items():
261             text = text.replace(utc, f"{count}")
262
263         # Concatenate counts into string like "433"
264         name_exe = "".join(str(counts[utc]) for utc in utc_list)
265         return name_exe
266
267     # Level 5. Loesung
268     def check_ports_level4(self, log_data):
269         return self.parse_logfile(log_data)
```

Listing 1: der Raum

Level 1

```
1 #!/usr/bin/python3
2
3 print(" ist nur ein Platzhalter")
```

Listing 2: Ein Beispiel

Level 2

```
1 #!/usr/bin/python3
2
3 print(" ist nur ein Platzhalter")
```

Listing 3: Ein Beispiel

Level 3

```
1 #!/usr/bin/python3
2
3 import random
4 import os
5
6 # """ Steganographie
7 # verstecken und auslesen von Nachrichten in einem Bild.
8 #
```

```
# Oliver Oberdick Matrikel:548933
# """

# Hilfsfunktion nur fuer den EscapeRoom, damit unterschiedliche Bilder genutzt werden
def random_bild(ziel_bild):
  nummer = random.randint(1, 9)
  dst = ziel_bild
  src = "static/originale/Bild_Schluessel_" + str(nummer) + ".jpg"
  if os.name == 'nt':  # pruefen ob Windows
    kopierbefehl = f'copy "{src}" "{dst}"'
  else:
    kopierbefehl = f'cp "{src}" "{dst}"'
  os.system(kopierbefehl)

# Funktion zum Vorbereiten der Level
def im_bild_verstecken(bild_datei, schluessel):
    bild = open(bild_datei, encoding="ISO-8859-1", mode="a+")
    bild.write("flag=" + schluessel)
    bild.close()

# Kontrollfunktion
def im_bild_finden(bild_datei, was="flag="):
    bild = open(bild_datei, encoding="ISO-8859-1", mode="r")
    search = was
    try:
        txt = ""
        byte = bild.read(1)
        while byte != "":
            txt = txt + byte
            byte = bild.read(1)
        pos = txt.find(search) # position des suchstring finden
        pos = pos + len(search) # laenge des suchstrings ueberspringen
        with open("tmp/game.key", 'w') as tmp: # gefundenen Schluessl zwischenspeichern
            tmp.writelines(txt[pos:])
        bild.close()
        return txt[pos:]
    except:
        bild.close()

##
if __name__ == "__main__":
    pass
```

Listing 4: Funktionen für Level 3 (Steganographie)

.

Level 4

```
#!/usr/bin/python3

import random

# """ Verschluesselung
# Symetrische Verschluesselung mittels XOR auf Bit-Basis
#
# Oliver Oberdick Matrikel:548933
# """

# Hilfsfunktion zum erstellen eines Verschluesselungs key in beliebiger laenge
def schluessel_erstellen(laenge):
        ergebnis = ""
        while len(ergebnis) < laenge:
```

```
            zahl = random.randint(48, 122)
            if ((zahl >= 48 and zahl <= 57) or (zahl >= 65 and zahl <= 90) or (zahl >= 97
    and zahl <= 122)):
        #  Damit der Schluessel nur aus Zahlen, Grossbuchstaben und Kleinbuchstaben besteht
                ergebnis += chr(zahl)
        return ergebnis

def string_to_binaer(nachricht):
  ergebnis = ""
  for c in nachricht:
    ergebnis += ''.join(format(ord(c), '08b'))
  return ergebnis

def binaer_to_string(nachricht):
  ergebnis = ""
  for i in range(0, len(nachricht), 8):
    ergebnis += chr(int(nachricht[i: i+8], 2))
  return ergebnis

# Ver oder Entschluesseln eines String mittels XOR (Symetrisch)
def schluesselanwendung(was, womit):
  ergebnis = ""
  schluessel = ""
  while len(schluessel) < len(was):
    schluessel += womit
  binaer_schluessel = string_to_binaer(schluessel)
  binaer_nachricht = string_to_binaer(was)
  for i in range(len(binaer_nachricht)):
    ergebnis += str(int(binaer_nachricht[i]) ^ int(binaer_schluessel[i]))
  return binaer_to_string(ergebnis)

# erweiterung, damit auch Dateien ver und entschluesselt werden koennen
def schluesselanwendung_datei(eingabe_datei, ausgabe_datei, schluessel):
  counter = 0  # nur zur kontrolle
  ergebnis = ""  # nur zur kontrolle
  with open(eingabe_datei, 'r') as in_file:
    with open(ausgabe_datei, 'w') as out_file:
      for line in in_file.read():
        counter += 1  # nur zur kontrolle
        tmp = schluesselanwendung(line, schluessel)
        out_file.write(tmp)
        if (counter >= 20 and counter >= 70): # nur zur kontrolle
          ergebnis += tmp # nur zur kontrolle
  return ergebnis # nur zur kontrolle im EscapeRoom Spiel

# Angepasste Funktion, damit zum entschluesseln der Schluessel aus einer Daten genutzt
    werden kann
def entschluesseln(eingabe, ausgabe="tmp/ausgabe_encrypt.txt", schluessel="tmp/game.key"):
  key = ""
  with open(schluessel, "r") as f:
    key = f.readline()
  return schluesselanwendung_datei(eingabe, ausgabe, key)

##
if __name__ == "__main__":

  key1 = schluessel_erstellen(20)

  print(key1)

  text = "Hallo du da im Radio!"

  print("Original Text")
  print(text)
  print("Verschluesselt mit Key1")
```

```
  text_verschluesselt = schluesselanwendung(text, key1)
  print(text_verschluesselt)
  print("Entschluesselt mit Key1")
  text_entschluesselt = schluesselanwendung(text_verschluesselt, key1)
  print(text_entschluesselt)
```

Listing 5: Funktionen für Level 4 (Symetrische Verschluesselung)

.

Level 5

```
#!/usr/bin/python3

print(" ist nur ein Platzhalter")
```

Listing 6: Ein Beispiel

Level 6

```
#!/usr/bin/python3

print(" ist nur ein Platzhalter")
```

Listing 7: Ein Beispiel

3.2 Beispiellösungen

Level 1

```
def run(eingabe):
  result = ""
  tmp = eingabe.split()
  for i in tmp:
    result += "".join(chr(int(i)))
  return result # tmp


## Loesung ist: 67 111 111 107 105 101 109 111 110 115 116 101 114.
```

Listing 8: Beispiellösung Level 1

.

Level 2

```
import re


def run(eingabe):
  with open(eingabe, "r", encoding="utf-8") as f:
    text = f.read()

  # Alle UTCs im Text finden
  utc_list = re.findall(r"-\d+", text)

```

```
  # Vorkommen zaehlen
  counts = {utc: text.count(utc) for utc in utc_list}

  for utc, count in counts.items():
    text = text.replace(utc, f"{count}")

  # Concatenate counts into string like "433"
  name_exe = "".join(str(counts[utc]) for utc in utc_list)
  return name_exe
```

Listing 9: Beispiellösung Level 2

.

Level 3

```
# Beispielloesung Level 2

def run(wo, was="flag="):
    bild = open(wo, encoding="ISO-8859-1", mode="r")
    search = was
    try:
        txt = ""
        byte = bild.read(1)
        while byte != "":
            txt = txt + byte
            byte = bild.read(1)
        pos = txt.find(search)
        pos = pos + len(search)
        with open("tmp.txt", 'w') as tmp: # gefundenen Schluessel zwischenspeichern
            tmp.writelines(txt[pos:])
        return txt[pos:]
    except:
        bild.close()
```

Listing 10: Beispiellösung Level 3

.

Level 4

```
# Beispielloesung Level 4

def run(eingabe):
  return schluesselanwendung_datei(eingabe, "ausgabe_encrypt.txt", "tmp.txt")

def string_to_binaer(nachricht):
  ergebnis = ""
  for c in nachricht:
    ergebnis += ''.join(format(ord(c), '08b'))
  return ergebnis

def binaer_to_string(nachricht):
  ergebnis = ""
  for i in range(0, len(nachricht), 8):
    ergebnis += chr(int(nachricht[i: i+8], 2))
  return ergebnis

def schluesselanwendung(was, womit):
  ergebnis = ""
  schluessel = ""
```

```
  while len(schluessel) < len(was):
    schluessel += womit
  binaer_schluessel = string_to_binaer(schluessel)
  binaer_nachricht = string_to_binaer(was)
  for i in range(len(binaer_nachricht)):
    ergebnis += str(int(binaer_nachricht[i]) ^ int(binaer_schluessel[i]))
  return binaer_to_string(ergebnis)

def schluesselanwendung_datei(eingabe_datei, ausgabe_datei, schluessel):
  key = ""
  with open(schluessel, "r") as f:
    key = f.readline()
  counter = 0
  ergebnis = ""
  with open(eingabe_datei, 'r') as in_file:
    with open(ausgabe_datei, 'w') as out_file:
      for line in in_file.read():
        counter += 1
        tmp = schluesselanwendung(line, key)
        out_file.write(tmp)
        if (counter >= 20 and counter >= 70):
          ergebnis += tmp
  return ergebnis
```

Listing 11: Beispiellösung Level 4

.

Level 5

```
import re

def run(input_data):
    results = []
    lines = input_data.strip().split("\n")

    for line in lines:
        line = line.lower().strip()
        matches = re.findall(r"port (\d+)", line)
        for match in matches:
            port = int(match)
            if "secure" in line or "accepted" in line:
                status = "open"
                reason = "secure/accepted"
            elif "attempt" in line or "exposed" in line or "unauthorized" in line:
                status = "open"
                reason = "attempt/exposed/unauthorized"
            elif "filtered" in line:
                status = "closed"
                reason = "filtered"
            else:
                status = "closed"
                reason = "default"

            results.append({
                "port": port,
                "status": status,
                "reason": reason,
                "raw_line": line
            })

    return results
```

Listing 12: Beispiellösung Level 5

.

Level 6

```
# Beispielloesung Level 2

def run(wo, was="flag="):
    bild = open(wo, encoding="ISO-8859-1", mode="r")
    search = was
    try:
        txt = ""
        byte = bild.read(1)
        while byte != "":
            txt = txt + byte
            byte = bild.read(1)
        pos = txt.find(search)
        pos = pos + len(search)
        with open("tmp.txt", 'w') as tmp: # gefundenen Schluessel zwischenspeichern
            tmp.writelines(txt[pos:])
        return txt[pos:]
    except:
        bild.close()
```

Listing 13: Beispiellösung Level 6

.

A Eigenständigkeitserklärungen