Überschrift

Oliver Oberdick

5. Oktober 2025

Zusammenfassung

Beschreibung zur APL Programmierung 2 der Gruppe HH05

Inhaltsverzeichnis

1 Story

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!

1.1 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

Bsp.: Du schaltest dein Rechner ein und bekommst nur einen eigenartigen Eingabeprompt

- ??

2.2 Level 2

‘Finde das flag / den Key in dem Bild

- auf einer normalen Kommandozeile könntest du ihn mit dem Kommando Strings auslesen.
- merke dir den Key gut.
- ?
- nur in dieser Runde lautet der Key ???

2.3 Level 3

Bsp.: Deine Dateien sind nicht da wo sie sein sollten, suche sie!

- ??

2.4 Level 4

die gefundene Datei muß entschlüsselt werden.

- Vielleicht könnte dir der “gemerkte Key” dir behilflich sein.
- Verschlüsselung mit XOR
- die Verschlüsselung erfolgte auf Bitebene.
- wenn der Key nicht lang genug ist, Hilft vielleicht eine aneinanderreihung

2.5 Level 5

Bsp: Mal schauen was in der Datei zu finden ist

- ??

2.6 Level 6

Bsp: Was sollte jetzt mit den gefundenen Informationen angestellt werden.

- ??

A Programmcode

A.1 Vorbereitungen

Generell:

```
import random
import string
from EscapeRoom import EscapeRoom

import lib.stego as STEGO # Funktionssammlung Oliver Level 2
import lib.crypt as CRYPT # Funktionssammlung Oliver Level 4

class Gruppe_HH_05(EscapeRoom):

    def __init__(self):
        super().__init__()
        self.set_metadata("Veronika, Lucasz & Oliver", __name__)
        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 )

        self.add_level(self.create_level1()) # Veronika
        self.add_level(self.create_level2()) # Oliver
        self.add_level(self.create_level3()) # Veronika
        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):
        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, 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}

    # Level 2
    def create_level2(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: 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 3
    def create_level3(self):
        task_messages = [
            "  <img src=" + self.bild + " alt='The Key you looking for' height='400'/> ",
            "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):
        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}

    # 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 ###



    ### SOLUTIONS ###
```

Listing 1: Genereller Escape-Room

Level 1

Level 2

```
#!/usr/bin/python3

import random
import os

# """ Steganographie
# verstecken und auslesen von Nachrichten in einem Bild.
#
# 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 2: Funktionalitäten für Level 2

Level 3

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)
  key2 = schluessel_erstellen(20)

  print(key1)
  print(key2)

  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)
  print("entschluesselt mit Key2")
  text_entschluesselt = schluesselanwendung(text_verschluesselt, key2)
  print(text_entschluesselt + "  - mit falschem Key")

  print("-------------------------------------")

  schluesselanwendung_datei("test.txt", "test.crypt", key1)

  schluesselanwendung_datei("test.crypt", "test_entcript.txt", key1)

  print("Dateien fertig")
```

Listing 3: Funktionalitäten für Level 4

Level 5

Level 6

A.2 Beispiellösungen

Level 1

Level 2

```python
# 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 4: Beispiellösung Aufgabe 2

Level 3

Level 4

```python
# 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]))
```

```
27   return binaer_to_string(ergebnis)
28
29 def schluesselanwendung_datei(eingabe_datei, ausgabe_datei, schluessel):
30   key = ""
31   with open(schluessel, "r") as f:
32     key = f.readline()
33   counter = 0
34   ergebnis = ""
35   with open(eingabe_datei, 'r') as in_file:
36     with open(ausgabe_datei, 'w') as out_file:
37       for line in in_file.read():
38         counter += 1
39         tmp = schluesselanwendung(line, key)
40         out_file.write(tmp)
41         if (counter >= 20 and counter >= 70):
42           ergebnis += tmp
43   return ergebnis
```

Listing 5: Beispiellösung Aufgabe 4

Level 5

Level 6