Prüfung: Fake News, Phishing, Authentifizierung

Erwartete Note: ____ **Punkte:** ____ **/ 24** **Note:** ____

Aufgabe 1: Phishing ..... **/ 6 P.**

Definiere die folgenden drei Begriffe in je einem Satz und mache je ein Beispiel.

- *Spear Fishing*
- *Whaling*
- *Social Engineering*

Lösung:

- *Whaling* nennt man Phishing mit dem Ziel, eine besonders wertvolles Opfer zu attackieren, meist durch *Spear Phishing*. *Beispiel*: Gezieltes Email-Phishing auf einen CEO.
- *Spear Fishing* nennt man Phishing mit dem Ziel, nur eine bestimmte Person zu attackieren, im Gegensatz zu einer breit gestreuten Attacke via Massen-Emails. Dafür wird die Attacke individuell auf die Zielperson angepasst. Wird oft für Whaling verwendet. *Beispiel*: Gezieltes Email-Phishing auf einen Lehrer.
- *Vishing* nennt man Attacken per Telefon (*Voice Phishing*), bei denen mehr psychologischer Druck aufgebaut werden kann als per Email. *Beispiel*: Schockanruf.
- *Baiting* nennt man Attacken, bei denen Köder so platziert werden, dass potentielle Opfer diese finden und anwenden. *Beispiel*: USB-Sticks, QR-Codes.
- *Social Engineering* ist ein Vorgehen, bei dem nicht eine technische Attacke im Vordergrund steht, sondern psychologische Tricks angewandt werden, um eine Zielperson dazu zu bringen, Informationen preiszugeben oder Handlungen ausführen. *Beispiel*: Eine Person wird durch einen Schockanruf dazu gebracht, einer fremden Person Geld zu überweisen.

Bewertung: je 1P für richtige Definition und pro richtigem Beispiel.

Aufgabe 2: Authentifizierung ..... **/ 11 P.**

Du baust eine Webseite zur Rettung des Eiffelturms, bei der sich die Spender:innen mit Benutzername und Passwort einloggen (authentifizieren) müssen.

(a) (2 P.) Weshalb bietet dieses Vorgehen nur beschränkte Sicherheit? Was ist das Hauptproblem von Passwörtern? Beschreibe in maximal 3 Sätzen.

Lösung:

- **Passwörter** 2P. Benutzer verwenden meist auf vielen Webseiten die gleichen Benutzernamen und Passwörter. Wird auch nur eine einzige Webseite gehackt, so können die gleichen Angaben auf anderen Webseiten probiert werden.
  [1P. für andere Nennungen: Brute-Force, Dictionary-Attack...]

Für mehr Sicherheit baust du zusätzlich zum Passwort einen zweiten Sicherheitsfaktor ein: die Kunden müssen ausser ihrem Passwort noch ihr Geburtsdatum nennen.

(b) (2 P.) Nenne zwei Probleme dieses Vorgehens!

(c) (1 P.) Was wäre eine bessere Wahl für den zweiten Faktor?

Lösung:

- **Problem** 1P. Passwort und Lieblingsfarbe sind beides *Wissensfaktoren*. Beide haben die gleiche Anfälligkeit: ist das Geheimnis anderenorts abgespeichert und wird gehackt, können dieselben Angaben auf deiner Webseite verwendet werden.
  1P. Die Lieblingsfarbe ist zudem nicht besonders geheim - ein Angreifer könnte sie beispielsweise mit Social Engineering herausfinden, indem er Bekannte des Kunden kontaktiert.
- **Besserer 2. Faktor** 1P. Der zweite Faktor sollte aus einer anderen Kategorie mit einem anderen Angriffsprofil stammen. Zum Beispiel der Besitz eines Smartphones (über eine 2FA-App) oder eines Hardware-Tokens (*Besitzfaktoren*), oder ein Fingerabdruck (*Inhärenzfaktor*).

Du hast Zugriff auf eine geheime Datenbank, die für jede:n Schüler:in den Namen des passenden Totem-Tiers speichert. Allerdings ist der Zugang mit einem Passwort gesichert. Der Benutzername ist dein KSR-Benutzername (ohne `@ksr.ch`, also beispielsweise `mafathal`). Das Passwort besteht aus drei Zeichen, wobei die Kleinbuchstaben `abcdef` sowie die Ziffern von 0-9 vorkommen können.

(a) (1 P.) Wieviele Passwort-Kombinationen sind möglich?

Lösung: 1P - Es gibt $16^3 = 4096$ Möglichkeiten. $\frac{1}{2}$P fürs Erkennen des Alphabets und richtige Formel.

(b) (4 P.) Am Computer: Schreibe eine Brute-Force-Attacke auf die Datenbank und finde dein geheimes Totem-Tier heraus (Abgabe des Codes via Teams).

Anleitung:

- Lade die Dateien `animals.py` und `hacking.py` von der Teams-Aufgabe herunter und speichere sie im gleichen Ordner ab.
- Führe den Code `hacking.py` aus - du solltest `Forbidden` zurück erhalten.
- Verändere den Code in `hacking.py` so, dass du das richtige Passwort herausfindest.
- Gib nur `hacking.py` per Teams ab!

```python
import animals  # Animals-Datenbank importieren

username = "pascherr"  # Hier soll DEIN Benutzername stehen!
password = "ab23"       # Nur ein Beispiel!
response = animals.login3(username, password)  # Zugriff auf Datenbank
print(response)

# Response ist "Forbidden", wenn das Passwort falsch ist
# Andernfalls ist response das geheime Totem-Tier.
```

Lösung:

```python
import animals
import itertools

alphabet = "abcdef0123456789"

def attack(username):
    for guess in itertools.product(alphabet, repeat=4):
        pw = ''.join(guess)
        response = animals.login3(username, pw)
        if response != "Forbidden":
            print("Passwort gefunden: " + pw)
```

```
12              return response
13
14      print("Nicht gefunden")
15
16  print(attack('pascherr'))
```

Bewertung:

- 1P für die korrekte Antwort (Tier)
- 4P für das funktionierende Programm, davon
  - 1P für eine sinnvolle Codierung des Alphabets.
  - 1P für eine Schleife mit login3-Aufruf
  - 1P für den richtigen Test mit "Forbidden"
  - 1P für die Kombination von drei Buchstaben

(c) (1 P.) Schreibe dein Totem-Tier hier in die Papier-Prüfung!

Lösung:

Aufgabe 3: Python String-Operationen ..... / 7 P.

Alle Lösungen sind als Dateien in der Teams-Aufgabe abzugeben.

(a) (3 P.) Schreibe eine Python-Funktion `count_letters(text, letter)`, die als Input einen Text (einen String) und einen Buchstaben erhält und zurückgibt, wie oft der Buchstabe `letter` im String `text` vorkommt.

Der folgende Beispiel-Code soll 3 ausgeben.

```
print(count_letters("Ein heller Morgen ohne Sorgen", "o"))
```

Lösung:

```
def count_letters(text, letter):
    count = 0
    for l in text:
        if l == letter:
            count += 1
    return count
```

Bewertung: Volle Punktzahl, wenn der Code das gewünschte Resultat liefert. Pro Fehler ½P. Abzug.

(b) (4 P.) Schreibe eine Python-Funktion `domain_part(email_address)`, die die *Domain* einer Email-Adresse zurückgibt, also den Teil *nach* dem `@`.

Der folgende Beispiel-Code soll `ksr.ch` auf der Konsole ausgeben:

```
print(domain_part('hof@ksr.ch'))
```

Lösung:

```
def local_part(email_address):
    result = ''
    for letter in email_address:
        if letter == '@':
            return result
        result = result + letter
    return result
```

Bewertung: Volle Punktzahl, wenn der Code das gewünschte Resultat liefert, egal wie die Unterscheidung vor / nach @ erreicht wird. Keine Punkte, wenn die Grundkonstruktion fehlt (`for l in address` o.ä.) Pro Fehler ½P. Abzug.