Sigurnost Tjedni Izvještaji

Prvi tjedan:

Cilj-Ubacit evil-station u komunikacijski kanal između station-1 i station-2

WSL (za linux)

mkdir bdimic - Napravio direktorij bdimic

cd bdimic - uški u direktorij

git clone "https://github.com/mcagalj/SRP-2021-22"

cd SRP-2021-22/arp-spoofing/ - ušli u ovaj direktorij (sadrži start.sh i stop.sh)

cd arp-spoofing/ - pristupili tom direktoriju

Otvaramo containere:

docker exec -it station-1 bash

docker exec -it station-2 bash

docker exec -it evil-station bash

Zapravo što radimo jest da naš evil-station imitira station-2 te tako informacije prolaze od station-1 preko evil-stationa do station-2 i tako prisluškujemo razmjenu informacija dvaju stationa

ping- simple utility used to check whether a network is available and if a host is reachable. With this command, you can test if a server is up and running.

tcpdump- dump traffic

arpspoof- intercept packets on a switched LAN

netcat- command-line utility that reads and writes data across network connections, using the TCP or UDP protocols.

Drugi tjedan:

Za enkripciju smo koristili ključeve 22 bita entropije

Koristili smo fernet sustavTrebali smo brute forsati ključ pomoću kojeg smo dobili poruku

Koristili smo fernet sustav:

Fernet koristi sljedeće *low-level* kriptografske mehanizme:

- AES šifru sa 128 bitnim ključem
- CBC enkripcijski način rada
- HMAC sa 256 bitnim ključem za zaštitu integriteta poruka
- Timestamp za osiguravanje svježine (*freshness*) poruka

Ključevi su generirani na sljedeći način:

```
 # Encryption keys are 256 bits long and have the following format:
#
```

```
#             0...000b[1]b[2]...b[22]
#
# where b[i] is a randomly generated bit.
key = int.from_bytes(os.urandom(32), "big") & int('1'*KEY_ENTROPY, 2)

# Initialize Fernet with the given encryption key;
# Fernet expects base64 urlsafe encoded key.
key_base64 = base64.urlsafe_b64encode(key.to_bytes(32, "big"))
fernet = Fernet(key_base64)
```

Učitavanje i spremanje datoteka u Pythonu:

```
# Reading from a file
with open(filename, "rb") as file:
    ciphertext = file.read()
    # Now do something with the ciphertext
```

```
# Writing to a file
with open(filename, "wb") as file:
    file.write("Hello world!")
```

Treći tjedan:

Cilj vježbe je provjeriti integritet poruka

-Pri tome ćemo koristiti simetrične i asimetrične krito mehanizme: *message authentication code (MAC)* i *digitalne potpise* zasnovane na javnim ključevima

Koristimo HMAC za zaštitu integriteta

1. U lokalnom direktoriju kreirajte tekstualnu datoteku odgovarajućeg sadržaja čiji integritet želite zaštititi.
2. Učitavanje sadržaja datoteke u memoriju.

   ```
   # Reading from a file
   with open(filename, "rb") as file:
       content = file.read()
   ```

3. Funkcija za izračun MAC vrijednosti za danu poruku.

   ```
   from cryptography.hazmat.primitives import hashes, hmac
   ```

```
def generate_MAC(key, message):
    if not isinstance(message, bytes):
        message = message.encode()

    h = hmac.HMAC(key, hashes.SHA256())
    h.update(message)
    signature = h.finalize()
    return signature
```

4. Funkcija za provjeru validnosti MAC-a za danu poruku.

```
from cryptography.hazmat.primitives import hashes, hmac
from cryptography.exceptions import InvalidSignature


def verify_MAC(key, signature, message):
    if not isinstance(message, bytes):
        message = message.encode()

    h = hmac.HMAC(key, hashes.SHA256())
    h.update(message)
    try:
        h.verify(signature)
    except InvalidSignature:
        return False
    else:
        return True
```

-Za preuzeti sve izazove sa servera:

- `wget.exe -r -nH -np --reject "index.html*" http://a507-server.local/challenges/<prezime_ime>/`

Otvaranje datoteka:

```
or ctr in range(1, 11):
    msg_filename = f"order_{ctr}.txt"
    sig_filename = f"order_{ctr}.sig"
    print(msg_filename)
    print(sig_filename)

    is_authentic = ...

    print(f'Message {message.decode():>45} {"OK" if is_authentic else "NOK":<6}')
```

Četvrti tjedan:

U ovom tjednu ćemo proći hash kriptografskim funkcijama za sigurnu pohranu zaporki i izvođenje enkripcijskih ključeva

Lozinke su najzastupljeniji način autentikacije korisnika pa je bitno da razumijemo osnovne koncepte za sigurnu pohranu lozinki

Okviran popis aktivnosti; detaljne upute ćemo dati u realnom vremenu:

- Usporedba *brzih* i *sporih* kriptografskih *hash* funkcija.
- Razumijevanje suštine pojmova *spore/brze* funkcije.
- Demonstracija *memory-hard* funkcija.

Peti tjedan:

Za računati moguće sifre od 4-6 znamenki, triba izracunat od 26 mogućih znamekni(Suma od i=4 do 6) = 26na6+26na5+26na4(mozemo zanemarit 26na5 i 26na4 zato sto je 26na6 ogroman broj

-primjer online brute forcea-koristimo hidru

U ovoj vježbi pokusavamo pronaći lozinku na offline i online način:

Online Password Guessing

1. Open bash shell in WSL on your local Windows machine.
2. Check that you can reach the lab server by pinging it.

   ```
   ping a507-server.local
   ```

3. Install `nmap` application. In the bash shell execute the following commands.

   ```
   sudo apt-get update
   sudo apt-get install nmap

   # Test it
   nmap
   ```

Try to understand what is `nmap` used for (Google it).

4. Next, run the following command.

   ```
   nmap -v 10.0.15.0/28
   ```

   Comment the results.

5. Open the following web page http://a507-server.local/ and note down the **IP address** of a Docker container and **username** dedicated to you.

6. Try to open a remote shell on the dedicated machine using previous information. Use `ssh` client from your local shell.

   ```
   # ssh <username>@<your IP address>
   ssh cagalj_mario@10.0.15.1
   ```

7. Install `hydra` application. Get to know it. Now, try to perform an **online password guessing attack** against your account. You know the following information about the used password:

   - it is comprised of lowercase letters
   - its length between 4 and 6 characters

   Q1: Estimate the password space.

   ```
   # hydra -l <username> -x 4:6:a <your IP address> -V -t 1 ssh
   hydra -l cagalj_mario -x 4:6:a 10.0.15.1 -V -t 1 ssh
   ```

   Q2: Using the output produced by `hydra` try to estimate your effort, that is how long it would take, on average, before you succeed. You can also try to play with parameter `-t` (**IMPORTANT:** Test values 2, 3, 4 but please do not exaggerate to avoid crushing the server).

   Q3: What do your do if the estimated time from the previous question is prohibitively large?

8. Get the dictionary from http://a507-server.local:8080/ as follows (please mind the **group ID**).

   ```
   # For GROUP 1 (g1)
   wget -r -nH -np --reject "index.html*" http://a507-server.local:8080/dictionary/g1/
   ```

9. Finally, use `hydra` with the dictionary as shown below (IMPORTANT: use `dictionary_online.txt`).

   ```
   # hydra -l <username> -P dictionary/<group ID>/dictionary_online.txt 10.0.15.1 -V -t 4
   ssh
   hydra -l cagalj_mario -P dictionary/g1/dictionary_online.txt 10.0.15.1 -V -t 4 ssh
   ```

10. Try to login to your machine using the discovered password. Locate password hashes, select one account (different from your own) and try to learn the corresponding password using **offline password guessing** attack as outlined in the sequel.

Offline Password Guessing

1. For this task, use `hashcat` tool. Install it on your local machine as follows.

   ```
   sudo apt-get install hashcat

   # Test it
   hashcat
   ```

2. Save the password hash obtained in the previous task into a file. To make this step somewhat easier, open the present folder in Visual Studio Code by running the following command.

   ```
   code .
   ```

3. Start offline guessing attack by executing the following command. As in the previous task you know the following about the password:

   - it is comprised of lowercase letters
   - its length is exactly 6 characters

   ```
   # hashcat --force -m 1800 -a 3 <password_hash_file> ?l?l?l?l?l?l --status --status-timer 10
   hashcat --force -m 1800 -a 3 hash.txt ?l?l?l?l?l?l --status --status-timer 10
   ```

   Q1: Estimate the password space.

   Q2: Using the output produced by `hashcat` try to estimate your effort, that is how long it would take, on average, before you succeed.

   Q3: What do your do if the estimated time from the previous question is prohibitively large?

4. If the attack from the previous step is not feasible approach, try a dictionary-based guessing attack (**IMPORTANT:** use `dictionary_offline.txt`).

   As before you can get the dictionary by executing the following in the local (WSL) bash shell (in the same directory where you stored the password hash file).

```
# For GROUP 1 (g1)
wget -r -nH -np --reject "index.html*" http://a507-server.local:8080/dictionary/g1/
```

Now start `hashcat` using the following command.

```
# hashcat --force -m 1800 -a 0 <password_hash_file> <dictionary_file> --status --status-timer 10
hashcat --force -m 1800 -a 0 hash.txt dictionary/g1/dictionary_offline.txt --status --status-timer 10
```

5. Test validity of the cracked password by logging into the remote machine as follows.

```
# ssh <username>@<your IP address>
ssh jean_doe@10.0.15.1
```

Šesti tjedan:

U ovoj vježbi smo se upoznali sa ACL-om i postupkom upravljanja korisničkim računima na Linux OS-u.

U Linux-u svaka datoteka ili program (*binary executable file*) ima vlasnika (*user or owner*). Svakom korisniku pridjeljen je jedinstveni identifikator *User ID (UID)*. Svaki korisnik mora pripadati barem jednoj grupi (*group*), pri čemu više korisnika može dijeliti istu grupu. Linux grupe također imaju jedinstvene identifikatore *Group ID (GID)*.

1. Identifikatore `uid`, `gid` i pripadnost grupama možete provjeriti kako je prikazano u nastavku (ovu naredbu izvršite u *shell-u* u WSL na vašem lokalnom računalu):

   ```
   id
   ```

   Uvjerite se da pripadate administratorskoj grupi `sudo`; za ispis samo grupa kojima pripadate možete koristiti naredbu `groups`.

2. Kreirajte novi korisnički račun (npr. `alice`). Pri tome možete koristiti naredbu `adduser` (odnosno `deluser` za brisanje postojećeg korisničkog računa). Logirajte se kao novi korisnik i saznajte vaš `uid`, `guid` i sve dodatne grupe kojim pripadate.

   NAPOMENA: Ovo možete izvršiti samo ako imate administratorske ovlasti (odnosno pripadate grupi `sudo`).

   ```
   sudo adduser alice
   ```

3. Logirajte se kao novi korisnik i saznajte odgovarajuće identifikatore korisnika i grupa kojima korisnik pripada.

   ```
   su - alice
   ```

4. Kreirajte još jedan korisnički račun (npr. `bob`).

   NAPOMENA: Ne zaboravite da za ovo trebate administratorske ovlasti. Izvršavanjem naredbe `exit` u *shell*-u prethodnog korisnika `alice` vraćate se u `shell` korisnika koji ima administratorske ovlasti (odnosno član je grupe `sudo`).

Standardna prava pristupa datotekama

1. Logirajte se u sustav kao novi korisnik (npr. `alice`). U korisnikovom *home* direktoriju (`/home/alice`) kreirajte novi direktorij `srp` te u njemu datoteku `security.txt` (upišite proizvoljan tekst).

   ```
   # navigate to home directory
   cd

   # create a new directory
   mkdir

   # create a file with text
   echo "Hello world" > security.txt

   # print file content
   cat security.txt
   ```

2. Izlistajte informacije o novom direktoriju i datoteci. Odredite vlasnike ovih resursa (korisnike i grupe) kao i dopuštenja (*access permissions*) definirana na njima. Pri tome možete koristiti neku od sljedećih naredbi: `ls -l` ili `getfacl`.

   ```
   ls -l .
   ls -l srp
   ls -l srp/security.txt
   ```

```
getfacl srp
getfacl srp/security.txt
getfacl -t srp/security.txt
```

3. Oduzmite pravo pristupa datoteci `security.txt` vlasniku datoteke modifikacijom dopuštenja (*access permissions*). Za promjenu dopuštenja koristite naredbu `chmod`. Testirajte ispravnost vaših rješenja.

   NAPOMENA: U dokumentu *Linux Security* možete naći primjere primjene chmod naredbe.

   U nastavku su dani neki primjeri primjene `chmod` naredbe:

   ```
   # Remove (u)ser (r)ead permission
   chmod u-r security.txt

   # Add (u)ser (r)ead permission
   chmod u+r security.txt

   # Remove both (u)ser and (g)roup (w)rite permission
   chmod ug-w security.txt

   # Add (u)ser (w)rite and remove (g)roup (r)ead permission
   chmod u+w,g-r security.txt

   # Add (u)ser (r)read, (w)rite permissions and remove e(x)ecute permpission
   chmod u=rw security.txt
   ```

4. Oduzmite pravo pristupa datoteci `security.txt` vlasniku datoteke na način da mu u tom postupku **ne odzimate `(r)ead` dopuštenje nad datotekom**.

   NAPOMENA: Ovo možete realizirati modifikacijom dopuštenja na *parent* direktoriju (`srp`). Oduzimanjem `(r)ead` prava na direktoriju oduzimate pravo ispisivanja/listanja (`ls`) njegovog sadržaja, dok oduzimanjem `e(x)ecute` prava oduzimate pravo ulaska (`cd`) u direktorij.

5. U dopunskom terminalu logirajte se kao drugi korisnik (npr. `bob`) i pokušajte pročitati sadržaj datoteke `security.txt` (koja pripada drugom korisnku, npr. `alice`). Oduzmite prava pristupa novom korisniku (`bob`) sadržaju ove datoteke.

6. Sada ponovo omogućite novom korisniku (`bob`) pristup sadržaju datoteke `security.txt` ali na način da taj korisnik ima pristup datoteci isključivo ako je član grupe koja je vlasnik predmetne datoteke `security.txt`.

   ```
   # 1. Learn the group that owns the filw using getfacl command
   # 2. Add new user (bob) to that group (requires administrator privileges)
   usermod -aG <owner_group> bob
   ```

```
# 3. Logout and login for this change to take effect
# 4. Verify the group membership of bob
id
# 5. Finally, try to read the file content
```

7. Logirajte se kao jedan dodanih korisnika i pokušajte pročitati sadržaj datoteke `/etc/shadow` u koju Linux pohranjuje *hash* vrijednosti korisničkih zaporki. Što ste uočili prilikom pokušaja pristupa navedenim datotekama? Objasnite razlog tog ishoda; odredite vlasnike navedenih datoteka (korisnike i grupe) kao i dopuštenja definirana na njima.

   Koristeći saznanja iz prethodnih koraka omogućite korisniku pristup sadržaju datoteke `/etc/shadow`.

8. Uklonite novog korisnika iz grupa `alice` i `shadow` (potrebne su administratorske ovlasti).

   ```
   # gpasswd -d <user> <group>
   gpasswd -d bob alice
   gpasswd -d bob shadow
   ```

   Ovim postupkom smo naučili kako dodavati i oduzimati prava useru, grupi i ostalima te kako funkcionira cijeli sistem.

C. Kontrola pristupa korištenjem *Access Control Lists (ACL)*

Za inspekciju i modifikaciju ACL-ova resursa (datoteka, direktorija) koristimo programe `getfacl` i `setfacl`.

1. Uvjerite se da novi korisnik `bob` nema pristup sadržaju datoteke `security.txt` korisnika `alice`.

2. U prethodnom zadatku pristup sadržaju smo omogućili dodavanjem novog korisnika u grupu koja je vlasnik predmetne datoteke. Korištenjem ACL, ovo možemo jednostavnije rješiti tako da u ACL datoteke `security.txt` dodamo novog korisnika sa `(r)ead` ovlastima (potrebne su administratorske ovlasti).

   ```
   # 1. Read/record current permissions defined on the file
   getfacl security.txt
   ```

```
# 2. Add (u)ser bob to the ACL list of the file with (r)ead premission
setfacl -m u:bob:r security.txt

# 3. Check the updated permissions defined on the file
getfacl security.txt

# 4. Login as bob, navigate to the file and try to read its content
cat security.txt
```

3. Uklanjanje zapisa iz ACL-a.

```
# Removing one entry from ACL
setfacl -x u:bob security.txt

# Removing the complete ACL
setfacl -b security.txt
```

4. Pokušajte omogućiti novom korisniku pristup sadržaju datoteke `security.txt` ali kroz članstvo u grupi (novu grupu možete prigodno nazvati `alice_reading_group`).

   Novu grupu možete kreirati kako slijedi (zahtjeva administratorske ovlasti):

   ```
   groupadd alice_reading_group
   ```

   Također smo naučili i da osin dodavanja u grupe, možemo dodati korisnika direkt u ACL gdje on može imati svoja posebna prava neovisno o grupi/ostalima.