ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK	Hallgató neve:	**Török Zoltán**
2025	Hallgató törzskönyvi száma:	**T009496/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai rendszerek Intézet

SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Török Zoltán**
Törzskönyvi száma: T009496/FI12904/N
Neptun kódja: H1H8WT

A dolgozat címe:

Behatolásdetektáló és csali rendszerek összehasonlító elemzése
Comparative analysis of intrusion detection and honeypot systems

Intézményi konzulens: Balázsné Dr. Kail Eszter
Külső konzulens:

Beadási határidő: 2023. december 15.

A záróvizsga tárgyai: Informatikai rendszerek üzemeltetésének biztonsága
IT hálózatbiztonság

A feladat

Az utóbbi években megszaporodtak a komplex és kifinomult kibertámadások. A védekezés egyik fontos eleme a támadások detektálása, tanulmányozása, következtetések levonása. A támadások detektálásának népszerű eszközei az IDS, valamint a csali (honeypot) rendszerek, melyek segíthetnek azonosítani akár egy addig még nem ismert támadást is.
A szakdolgozat célja a Honeypotok és az IDS rendszerek összehasonlító vizsgálata.
A hallgató feladata az IDS és Honeypot technológiák működésének, és a szolgáltatott adatok használhatóságának elméleti, valamint virtuális környezetben implementált megvalósításaiknak gyakorlati összehasonlítása.

A dolgozatnak tartalmaznia kell:

- napjainkban használt Honeypot és IDS rendszerek bemutatását,
- már meglévő rendszerek összehasonlító elemzését,
- az elkészítendő tesztkörnyezet rendszertervét,
- a rendszer implementációját virtuális környezetben,
- tapasztalatok ismertetését és kiértékelését,
- jövőbeli fejlesztési irányok megfogalmazását.

ÓBUDAI EGYETEM Neumann János Informatikai Kar 1. Budapest

Ph.

.....
Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2025. december 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

....
külső konzulens intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat / diplomamunka saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban / diplomamunkámban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2024. december 14.

Tóth Zoltán

.....

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve: Neptun Kód: Tagozat:

Török Zoltán ..... H1H8WT ..... Levelező

Telefon: Levelezési cím (pl.: lakcím):

06 70 5122372 ..... 2045 Törökbálint Szív u. 10.....

Szakdolgozat / Diplomamunka[1] címe magyarul:

Behatolásdetektáló és csali rendszerek összehasonlító elemzése

Szakdolgozat / Diplomamunka[2] címe angolul:

Comparative analysis of intrusion detection and honeypot systems

Intézményi konzulens: Külső konzulens:

Balázsné Dr. Kail Eszter ....

Kérjük, hogy az adatokat nyomtatott nagy betűkkel írja!

Alk.	Dátum	Tartalom	Aláírás
1.	2024.09.25	Honeypot - ok kiválasztása, megvitatása	Kail E
2.	2024.10.11	IDS – ek kiválasztása	Kail E
3.	2024.11.21	Gyakorlati szempontok átbeszélése	Kail E
4.	2024.12.04	Összehasonlítás szempontjainak megbeszélése	Kail E

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat / Szakdolgozat I. / Szakdolgozat II. / Projektlabor 1 / Projektlabor 2 / Projektlabor 3 / Záródolgozati projekt / Diplomamunka I / Diplomamunka II / Diplomamunka III / Diplomamunka IV [3] tantárgy követelményét teljesítette, beszámolóra / védésre [4]bocsátható.

A konzulens által javasolt érdemjegy: jeles (5)

Budapest, 2024.12.12

Kail E

Intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

ABSZTRAKT

Az utóbbi években a kibertámadások jelentős mértékben megnőttek. Mind az állami mind a magánszemélyek szintjén is. Nincs ország és ember, aki biztonságban lenne ezekkel a veszélyekkel szemben. Ezért a védelem kifejezetten fontos. De hogyan védekezzünk olyasmi ellen, amit nem ismerünk, ill. nem elég mélyen? Ezért a támadásokat nem csak elhárítani kell, hanem mindent megtanulni ezekről a tevékenységekről, hogy ebből tanulva még hatékonyabbá tegyük a védelmünket.

A szakdolgozatom célja, bemutatni olyan eszközöket, amelyekkel nem csak a védekezés valósítható meg, hanem támadások technikája és módja is kifürkészhető. Az általam végzett összehasonlításban igyekeztem érzékeltetni a „csak” védekezést megvalósító eszközök és a támadás technikáját kémlelő szoftverek lehetőségeit és azok közötti különbségeket.

ABSTRACT

In recent years, cyberattacks have increased significantly, both at the state level and for private individuals. No country or person is safe from these threats. Therefore, protection is especially important. But how can we defend ourselves against something we don't know or don't understand deeply enough? For this reason, it is not only necessary to prevent attacks but also to learn everything possible about these activities so that we can make our defenses even more effective.

The goal of my thesis is to present tools that not only enable defense but also allow us to uncover the techniques and methods of attacks. In my comparison, I have tried to highlight the capabilities and differences between tools that focus "only" on defense and software designed to analyze and explore attack techniques.

TARTALOM

1 BEVEZETÉS 3
2 IDS RENDSZEREK BEMUTATÁSA 4
2.1 Alkalmazásuk előnyei/hátrányai 4
2.2 IDS – ek csoportosítása 4
2.3 Alkalmazásuk előnyei/hátrányai 5
2.4 Néhány IDS rövid ismertetése 5
3 HONEYPOT RENDSZEREK BEMUTATÁSA 8
3.1 Alkalmazásuk előnyei/hátrányai 8
3.2 Honeypot – ok csoportosítása 9
3.3 Egy speciális csoport, a Honeynet 11
3.4 Néhány honeypot rövid ismertetése 11
4 4. IDS VS. HONEYPOT 23
5 5. RENDSZERTERV 25
5.1 IDS 25
5.2 Honeypot 25
6 6. MEGVALÓSULT IMPLEMENTÁCIÓK 29
6.1 Cowrie honeypot – SSH 29
6.2 Cowrie honeypot – Telnet 32
6.3 Valhala honeypot – Telnet 34
6.1.1 6.4 Valhala honeypot – FTP 37
6.5 Snort – FTP 40
6.1.2 6.6 Suricata – SSH 42
7 ÖSSZEHASONLÍTÁS 46
7.1 Honeypot: Cowrie, IDS: Suricata, Protokoll: SSH 46
7.2 Honeypot: Valhala, IDS: Snort, Protokoll: FTP 46
8 JÖVŐBELI FEJLESZTÉSEK IRÁNYA 48

9 ÖSSZEFOGLALÁS 49
10 SUMMARY 50
11 KÖSZÖNETNYÍLVÁNÍTÁS 51
12 ÁBRAJEGYZÉK 52
13 FELHASZNÁLT SZAKIRODALOM 54

1 BEVEZETÉS

Az elmúlt években szinte már megszokhattuk, hogy a médiából hetente jönnek a különböző IT biztonsággal kapcsolatos hírek a világ minden tájáról. Az olyan szavak és kifejezések, mint az adatlopás, feltörték, túlterheléses támadás stb. mindennapos, agyoncsépelt szófordulatokká váltak. Mindenki tudomásul vette: ez van. Azonban a 2022 évben rá kellett ébredni, hogy ezek a hírek és események nagyon közel kerülhetnek hozzánk. Természetesen itt most a nemzetközi színtéren történő eseményekre is gondolok. Mindannyian tudjuk, nem csak az analóg világban, de a digitális világban is komoly csaták zajlanak. Ebből kiindulva még hangsúlyosabbá váltak a felderítő és védekező mechanizmusok fejlesztése és üzemeltetése az kibertérben. Az időbeni felderítés és elhárítás, majd az abból való tovább építkezés a hatékonyabb védelem céljából elengedhetetlen. A témafelvetésem, mely az IDS és a honeypot - ok összehasonlító elemzése, ezért is aktuális és – szerintem megkerülhetetlen – minden informatikával foglalkozó szakember, szervezet és ország számára.

2 IDS RENDSZEREK BEMUTATÁSA

Az IDS (Intrusion Detection System) egy behatolást és riasztást küldő védelmi rendszer az informatikai hálózatokban. Figyeli és elemzi a hálózati folyamatokat valós időben. Képes ellenőrizni a csomagok fejléceit rosszindulatú támadások után kutatva, mint pl. TCP SYN támadás, töredezett csomag támadás [1]. Vannak típusok, melyek legfőképpen anomáliákat keres, ha talál, akkor riasztást küld az előre beállított csatornákon. Közben naplózást és kategorizálást is végez az általa érzékelt tevékenységekről.

2.1 Alkalmazásuk előnyei/hátrányai

Előnyök:

Számos támadási formát felismernek, ill. ezen eszközök alkalmazása alapvetően nem érinti a már meglévő infrastruktúrát. Kivételt képez ha IPS-t (Intrusion Prevention) is telepítünk, akkor az lassíthatja a válaszidő átfutását. Az IPS abban különbözik az IDS -től, hogy incidens esetén nem csak riaszt, hanem védekező mechanizmusok is beindulnak az előre beprogramozott szabályok alapján. Az IPS eszközök valós idejű monitorozásra képesek, így azonnal jeleznek, ha incidenst érzékel.

Hátrányok:

Gyors emberi beavatkozást igényel incidens esetén. A felhasználói azonosítást és hitelesítést sem váltja ki. Nem jelent megoldást minden biztonsági kihívásra. Ezen kívül ott van még a „fals pozitív" riasztások problémaköre, ez azt jelenti, ha túlságosan „érzékeny" a rendszerben lévő szabály, akkor a normál tevékenységet is rosszindulatúnak minősít /anomália alapúak/. De előfordulhat az ún. „fals negatív" jelenség is, amikor a rosszindulatú tevékenységet nem veszi észre /anomália alapúak/.

2.2 IDS – ek csoportosítása

- localizáció alapján:

Host jellegű IDS (HIDS): ennél az elrendezésnél egy konkrét eszközt, kiszolgálót figyel, mely lehet egy szerver vagy végponti eszköz. Nincs kapcsolata más komponensekkel és a hálózattal.

Network jellegű IDS (NIDS): hálózati forgalmat figyeli. De nem az aktív hálózati eszközök belső tevékenységét, hanem a köztük lévő forgalmat követi nyomon.

- működésük alapján:

Szabály alapú IDS: egy konkrét szabályrendszerrel feltöltött adatbázissal rendelkezik. A szabályok alapján próbálja kiszűrni a támadásra utaló jeleket. A felhasználónak van lehetősége saját szabályokat létrehozni, így tudja hangolni a saját igényei szerint.

Anomália alapú IDS: ezek az eszközök azzal kezdik a működésüket, hogy figyelik és felmérik a hálózat működését. Ez lesz a „baseline", a kiindulási pont. Minden eseményt, eszközhöz kapcsolódó forgalmat és hálózati jellemzőt elment, ebből kialakít egy standard hálózati karakterisztikát. A működési elve azon alapszik, hogy amikor hálózati incidens történik, az nyilvánvalóan eltérő forgalmat és aktivitást fog eredményezni a hálózatban, ahhoz képest, amit eddig az IDS tapasztalt. Érdemes megjegyezni, hogy ennél a típusnál kezd megjelenni a Mesterséges Intelligencia alkalmazása.[2]

2.3 Alkalmazásuk előnyei/hátrányai

Szabály alapú IDS előnye, viszonylag gyors és egyszerű telepítése. Nem igényel mélyebb szaktudást, kivéve, ha saját szabályokat akarunk bevinni a rendszerbe.

Hátrányuk, amenyiben az adatbázis nincs frissítve, pl.: gyártói támogatás által, akkor az újabb támadási technikákat nem fogja detektálni.

Anomália alapú előnye, az hogy maga az IDS az „átlagos" hálózati forgalomból tanulja meg, hogy mi a standard. Minden, ami ettől eltérő, azt incidensnek detektálja, így jóval nehezebb kijátszani.

Ám ebből fakad a hátránya is, hiszen hálózati zaj terhelheti az IDS működését. Ráadásul a „tanítás" alatt nem lehetünk biztosak abban, hogy éppen nem zajlik egy támadás, mert ha igen, akkor ezt normális működésnek fogja venni. A nagy zajban és forgalomban könnyen elvegyülhetnek a támadások. Ha pedig a rendszer a kelleténél érzékenyebb, akkor jönnek a fals pozitív riasztások, melyek hosszú távon a riportok átláthatatlanságához vezethet.

2.4 Néhány IDS rövid ismertetése

Samhain, [3] egy nyílt forráskódú, host jellegű HIDS rendszer, amelyet a Samhain Labs fejlesztett ki. A Samhain - t lehet centralizáltan vagy az egyes számítási csomópontokon egyedi megvalósításként is telepíthető. A Samhain HIDS a következő funkciókat biztosítja. Fájlintegritás-ellenőrzés, a fájlfigyelés és elemzés, Root - kit észlelés, Rogue SUID észlelés és rejtett folyamat figyelése és elemzése.

Jellemzők: A Samhain több platformot támogat, például Linux, Mac OS, Windows POSIX emulátorral. A Samhain architektúrája lehetővé teszi az egyetlen gazdagép megvalósítását, valamint a központosított kliens/szerver megvalósítást. Az ilyen típusú telepí-

téssel a menedzsment központosítása valósul meg. A Samhain kliens/szerver megvalósítása olyan összetevőkből áll, mint az Integritás-ellenőrző, a server-komponens, az kliens - összetevő.

A Samhain rendelkezik egy adatbázissal a naplók és jelentések karbantartására. A relációs adatbázisok, például az Oracle, a MySQL és a PostgreSQL szintén támogatottak. A Beltane a Samhain web alapú felhasználói felülete, ez egy a jelentések megtekintésére, kliens adatbázisok frissítésére szolgál a szerveren.

A Samhain gazdagép integritását különféle modulok segítségével figyelik. Minden modul bővíthető. A Samhain Windows alapú rendszerleíró adatbázis-ellenőrzést, Linux alapú rendszerek esetén pedig Kernel Integrity Check szolgáltatást nyújt. Továbbá, a Samhain külön modulokkal rendelkezik a nyitott port figyeléshez, a naplófájlok figyeléséhez és elemzéséhez, a bejelentkezési/kijelentkezési események figyeléséhez és a csatlakoztatott eszközök ellenőrzéséhez. A Samhain beállítható úgy, hogy elrejtse az identitását. Ezzel gyakorlatilag láthatatlan a behatolók számára.

Suricata, nyílt forráskódú hálózat alapú behatolásérzékelő rendszer (NIDS). A legnépszerűbb NIDS rendszerek a Suricata, a Snort és a Bro. A Bro egy Unix/Linux alapú, és nem támogatja a Windows szervereket. A Suricata - t az Open Information Security Foundation (OISF) fejlesztette ki. Suricata néhány funkciója.

Skálázhatóság: sokszor a tűzfalak és az IDS-k/IPS-ek szűk keresztmetszetet jelentenek a hálózati információs rendszerek teljesítményében. A Suricata azonban tudja azt, hogy az IDS-folyamat minden egyes példányát több szálon, különböző processzorokon futtassa keresztül, ügyelve a terheléselosztásra és a teljesítményre. Ezért skálázható a Suricata.

Protokoll azonosítás és támogatás: bevett gyakorlat a forgalom port szintű szűrése. A támadók azonban könnyen megkerülik az ilyen szűrést. A hálózaton futó gyakori protokollok felismerése után a szabályok a protokollra íródnak, nem pedig a várt portra. Ez a képesség biztosít a Suricatának lehetőséget, hogy a rosszindulatú programokat elemezni és ellenőrizni tudja. Továbbá a kulcsszavak a protokoll mezőkön belül is egyeztethetők, az egyszerű http fejlécektől az SSL tanúsítvány azonosítókig terjed. Támogatja a 2. rétegű, 3. és 4. rétegű protokollok csomag dekódolását, mint például a PPP, PPPoE, IPv4, IPv6, TCP, UDP, SCTP, ICMPv4, ICMPv6, GRE, SLL, VLAN, QINQ. A HTTP, SSL, TLS, SMB, SMB2, DCERPC, SMTP, FTP, SSH és DNS alkalmazási réteg dekódolása szintén támogatott.

Fájl azonosítás: MD5 hash – t alkalmazz, a rendszer menet közben számítja ki a fájlok kicsomagolása és lemezre írása során. Ezután az IDS dönti el, hogy ezeket a fájlokat megtartja, vagy kihagyja a rendszerből.

Teljesítmény: mivel ez az IDS támogatja a már említett teljes többszálú feldolgozást, amely felgyorsítja a hálózati forgalom elemzését így sokkal nagyobb teljesítményt biztosít, mint más bevett IDS, például a Snort rendszer.

Korábbi tanulmányok szerint a Suricata produkálta a legkevesebb csomagveszteséget. Nagyobb teljesítményt is megerősítettek, ha nagyobb sebességű hálózatokat használtak, mint pl.: a BSD free-n [4].

Snort, [5] szintén egy nyílt forráskódú IDS, képes valós idejű forgalomelemzésre és csomagnaplózásra az IP hálózatokon. Tud végezni protokollelemzést, tartalomkeresést/egyeztetést és számos támadás és vizsgálat észlelésére használható, mint például puffertúlcsordulás, lopakodó port ellenőrzés, CGI és SMB támadás, operációs rendszer fingerprinting kísérlet és még sok mást. Az internetről letölthető, és szinte minden hardver platformon és operációs rendszeren fut. A Snort folyamatának munkája hat részből áll: adatcsomag elfogása, adatkód elemzése, csomag elő - feldolgozása, szabály elemzés, motor észlelése és naplózása. Emellett a Snort egyszerű, rövid és jó programozhatósággal rendelkezik. Strukturált és könnyen olvasható.

3 HONEYPOT RENDSZEREK BEMUTATÁSA

Miközben az IDS technológia folyamatosan fejlődik, az IDS-ek megkerülésére szolgáló módszerek is egyre elterjedtebbek. Idővel megjelentek az olyan módszerek, mint pl.: a mimicry [6], melyek egy legális alkalmazást szimulálva próbál behatolni – az IDS számára észrevétlenül - a hálózatba.

A honeypot egy olyan csapda, amely az illetéktelen behatolót észleli, eltérítheti, megfigyeli, miközben minden információt begyűjt róla. A honeypot úgy viselkedik, mint egy szerver vagy akár – több honeypot esetén – mint egy hálózat. De nincsenek valós információk és szolgáltatások sem. Minden, amely egy támadónak érdekes lehet általában szimulálva van. Ez a rendszer úgy van kialakítva, hogy egy átlagos felhasználó nem fog ott megjelenni, mivel semmi dolga nincs ott. Nem is tud ezekről a szerverekről, hálózatokról. Ha egy honeypot „környékén" bárki megjelenik, az támadó.

Sokszor ezek a csalik egy valódi hálózat részének tűnnek, de valójában elszigetelt és megfigyelt, amely úgy tűnik, információkat vagy értékes forrást tartalmaz a támadók számára. Egy honeypot úgy működik, hogy a támadókkal elhiteti, hogy ez egy legitim rendszer. A támadók anélkül támadják meg a rendszert, hogy tudnának arról, hogy megfigyelés alatt állnak. Amikor egy támadó megpróbál feltörni egy honeypot - ot, a támadással kapcsolatos információk, mint pl.: a támadó IP-címét, begyűjtik. A támadó által végzett tevékenység értékes információkat nyújt a támadási technikák elemzésére.

3.1 Alkalmazásuk előnyei/hátrányai

Előnyök közt kell megemlíteni, hogy egy honeypot csak akkor gyűjt adatokat, ha valaki vagy valami interakcióba lép velük. Cégek vagy szervezetek naponta több ezer riasztást is naplózhatnak a hagyományos technológiákkal. Ez a szám honeypot - ok esetében nagyságrendileg is kevesebb lehet. Ez azért fontos, mert egyszerűbbé teszi a riportotok készítését és elemzését és kevesebb erőforrást igényel.
A már korábban említett fals pozitív problémakör is lényeges, amikor a hagyományos rendszer olyan aktivitást érzékel incidensnek, mely pl.: ritka, de normális hálózati forgalom. A legtöbb detektálási technológia egyik legnagyobb kihívása, az ilyen jelzések korlátozása. A honeypot - ok nem produkálnak ilyen téves jelzéseket.
A hagyományos technológiák másik kihívása az ismeretlen támadások észlelésének fel nem ismerése. Ez egy kritikus különbség a honeypot - ok és a hagyományos számítógépes biztonsági technológiák között, A honeypot könnyen azonosíthatja és elfoghatja az új jellegű támadásokat. Bármilyen tevékenység a honeypot - tal egy anomália, ami miatt az új vagy a még nem ismert támadások könnyen kibukhatnak.
Előnyök között említhetjük még a titkosított adatforgalommal végrehajtott támadásokat, ugyanis ebben az esetben nincs jelentősége annak, hogy egy támadás vagy rosszindulatú tevékenység titkosítva van, a honeypot rögzíti tevékenységet ebben az esetben is.

A honeypot - ok rendkívül alkalmazkodóak, rugalmasak és sokféle környezetben használhatók. Épp úgy mint ahogy minimális erőforrást igényelnek, még a nagyobb hálózatokon is. Meg kell jegyeznem, ez azért nem minden esetben igaz, amint azt a gyakorlati tesztelés során látni fogjuk.

Hátrányok között kell megemlíteni a honeypot- ok alkalmazásának a kockázatát, pl.: ha nincs kellő körültekintéssel elszigetelve az olyan hálózattól/szervertől, amely már nem része a honeypot rendszernek, akkor ebben az esetben fennáll annak a veszélye, hogy a támadók bejutnak az éles rendszerbe. Ez a kockázat a használt honeypot típusától függően változik. Például, az olyan egyszerű honeypot - nak, mint a KFSensor, nagyon kicsi a kockázata. A kockázati szintek változnak a különböző típusú honeypot telepítésnél. Itt is működik az a szabály, hogy minél bonyolultabb egy rendszer, annál nagyobb a kockázat a hibalehetőségekre.
Meg kell említeni a korlátozott látómező problémáját. A honeypot csak azt látja vagy rögzíti, ami interakcióval lép vele. Ezek nem egy passzív hálózati eszköz, amely az összes többi rendszer tevékenységét rögzíti. Ehelyett csak akkor naplóz, ha közvetlenül érintkeznek velük. Hasonlattal élve, a honeypot - ok olyanok, mint a mikroszkópok. Korlátozott látómezővel rendelkeznek, de amit lát, arról nagyon sok részletet mutat meg nekünk. Végül meg kell említeni egy másik lehetséges hátrányos tulajdonságot ezen eszközök alkalmazásakor. Ez pedig nem más, mint a lelepleződés esélye. Bár a honeypot felfedezésének kockázata kicsi, de azért mindig van esély arra, hogy egy gyakorlott, komoly tudással rendelkező hacker kiszúrja, hogy egy szimulált rendszerrel van dolga. Elég számára egy hibásan írt szó a szolgáltatásban, gyanúsnak tűnő tartalom és máris felkeltheti a gyanúját. Ezzel a tudással már akár ki is kerülheti a honeypot - ot és támadhatja az éles rendszert [7].

3.2 Honeypot – ok csoportosítása

Telepítés szerint:

A produktív /éles/ környezetbe telepített honeypot - ok viszonylag könnyen használhatók, de csak korlátozottan képesek az információgyűjtésre. Elsősorban a vállalatok használják. A cég vagy szervezet más, valódi kiszolgálók mellé helyezi el az éles hálózatban, ezzel is, mint egy növelve a cég informatikai biztonságát. Általánosan megállapítható, hogy az éles környezetben használt honeypot - okat könnyebb telepíteni, viszont kevesebb információt ad a támadásokról, mint egy kutatói környezet esetében.

Kutatói környezetben telepített honeypot - ok az elsődleges célja, hogy a lehető legtöbb információt gyűjtsenek a hackerek támadási technikájukról. Ezekkel a honeypot - okkal lehetséges új, eddig nem ismert támadási módszereket felismerni, tanulmányozni. Ezen új információk birtokában lehetséges egy új védekező „scenariot” kialakítani. Továbbá, olyan fontos és értékes információkat is kinyerhetünk egy jól beállított honeypot - ból,

hogy mik az új „trendi” célpontok a hackerek számára. Gondoljunk például az utóbbi években megsokasodó kriptovalutákhoz köthető támadásokra, mint a zsarolóvírusok (ransomware), melyek bejutva egy cég hálózatbába, igen komoly váltságdíjak kifizetését tudták a elérni. Aztán ott vannak az ezekkel szoros kapcsolatban lévő kripto tőzsdék ellen végrehajtott támadások is. A kutatói honeypot - okat bonyolultabb telepíteni és fenntartani, mint a produktív hálózatban lévőket, ezért ezeket átfogó információk gyűjtéséhez, elsősorban kutatási, katonai vagy kormányzati szervezetek használják.

Interaktivitás szerint:

Alacsony interaktivitású honepot – ok (LIH - Low Interaction Honeypots) a támadók által leggyakrabban felkeresett és támadott szolgáltatásokat szimulálják. Kevés az erőforrás igényük, ezért akár több virtuális gép is könnyen elhelyezhető egy fizikai szerveren, a virtuális rendszerek rövid válaszidővel rendelkeznek, és kevesebb kódra van szükség, ami nem jár a biztonsági kockázat növekedésével. Ezért kevésbé kockázatosak és könnyebben karbantarthatók. Nem irányítják a támadókat a gyökérrendszerre. Az ilyen típusú honeypot - ok hátránya, hogy a támadó számára nagyobb valószínűséggel tűnnek hamisnak, mert jelentősen korlátozott lehet az elérhető interakció [8]. Az alacsony interakciójú honeypot - ok alkalmasak a botok és rosszindulatú szoftverek támadásainak felderítésére.

Közepes interaktivitású honeypot – okkal (MIH - Medium Interaction Honeypots) a szakirodalom kutatása közben, egy – két helyen találkoztam ezzel a kategóriával [2]. Nem igazán elterjedt, azért fontosnak éreztem ennek a megemlítését is. Ebben az olvasatban a honeypot - okat az alacsony interaktivitású honeypot - ok kiterjesztett, de a magas interaktivitású honeypot - oknál szegényesebb interaktivitást biztosít a támadó számára. Egy közepes interakciójú honeypot teljes mértékben megvalósíthatja pl.: a HTTP protokoll egy jól ismert gyártói implementációját, például egy Apache szervert. Az interakcióknak azonban nincsenek teljes értékű megvalósításai, a funkciók és szolgáltatások csak részlegesen vannak megvalósítva. Érdemes megemlíteni, sok kutató és szakember ezeket az eszközöket inkább az alacsony interaktivítású honeypot- ok közé sorolja.

Magas interaktivitású honeypot – ok (HIH – High Interaction Honeypots) [2], különböző, valós rendszerek tevékenységét szimulálják, azokat elérhetővé teszik az interneten sok más szolgáltatással együtt. A cél az, hogy a potenciális támadó abban a hitben legyen, hogy valódi éles rendszerrel van dolga. Bármilyen parancs vagy alkalmazás, amelyet a végfelhasználó elvárna, elérhető, és általában kevés vagy nincs is korlátozás arra vonatkozóan, hogy a hacker mit tehet, miután bejutott a rendszerbe. Amíg a honeypot feltörésére pazarolja az idejét, számos információt gyűjt be róla a rendszer. A magas interakciós honeypot célja az is, hogy a támadót ösztönözze a root - vagy rendszergazdai szintű – hozzáférés megszerzésére (Escalation of privilege), mindeközben figyelemmel kísérje a támadó tevékenységét.

3.3 Egy speciális csoport, a Honeynet

A honeynet [2], [14] többféle honeypot - ok kombinációja, amelyek egy közös hálózatban vannak telepítve. A honeynet - ben lévő honeypot - nak megvan a maga szerepe és saját funkcionalitása, melyet megvalósít. Fontos tényt, honeynet kivitelezése homogén honeypot - okkal egyszerű, de nem olyan hatékony, mint ha a honeynet kivitelezése heterogén honeypot - okkal rakjuk össze. Ez viszont nem egyszerű feladat, ebből született igény egy közös, mindenki által használható honeynet keretrendszert /framemework/ leíró nyelv megvalóítása. Ezzel lehetséges integrálni az összese honeypot - ot a honeynet infrastruktúrájába.

A TIHDL (Technology Impartial Honeynet Description Language) egy szabványos nyelv, amelyet arra terveztek, hogy interpretálja honeynet - et mindezt az XML szintaxis használatával [9].

3.4 Néhány honeypot rövid ismertetése

Conpot [10] egy alacsony interakciójú nyílt forráskódú, szerveroldali honeypot. Könnyű telepíteni, módosítani és bővíteni. Egy sor általános ipari vezérlőprotokoll beépítésével létrehozhatjuk a saját rendszerünket. Amely képes összetett infrastruktúrák emulálására, hogy meggyőzze a támadót arról, hogy egy hatalmas ipari komplexumot talált. A megtévesztő képességek javítása érdekében lehetőség van arra, hogy egy egyedi ember - gép interfész - HMI - érintőképernyő - (Human Machine Interface) is beépíthető legyen, hogy növeljük a honeypot - tok támadási felületét. A szolgáltatások válaszideje mesterségesen késleltethető, hogy utánozza a rendszer állandó terhelés alatti viselkedését. Mivel a protokollok teljes halmazát implementálták, ezzel biztosítva van az, hogy a Conpot elérhető a produktív HMI-ken keresztül is, vagy valódi hardverekkel bővíthető. A Conpot a Honeynet Project égisze alatt fejlesztették.

A Cowrie [11] egy közepes és magas interakciós SSH és Telnet honeypot, amelyet a brute force támadások és a támadó által végrehajtott shell interakciók naplózására terveztek. Közepes interakciós módban (shell) egy UNIX rendszert emulál Pythonban. Magas interakciós módban (proxy) SSH és telnet proxyként működik, hogy megfigyelje a támadók viselkedését. Szintén nyílt forráskódú.

Heralding, [12] néha csak egy egyszerű honeypot - ra van szükségünk, amely csupán a hitelesítő adatokat gyűjti, semmi mást. Ez a honeypot ennek tesz eleget. Jelenleg a következő protokollok támogatottak: ftp, telnet, ssh, http, https, pop3, pop3s, imap, imaps, smtp, vnc, postgresql stb. Nyílt forráskódú [10].

SNARE/TANNER, [13] egy webes honeypot, A SNARE lehetővé teszi a meglévő weboldalak támadási felületekké alakítását azzal, hogy a weboldalunkat klónozza és azt teszi ki a netre. A SNARES egy Nginx típusú webes alkalmazás kiszolgálót szimulál a külvilág

felé. TANNER a SNARES "agya". A SNARE minden eseményt, amely az oldalon történik, elküld a TANNER-nek, kiértékelésre, és a TANNER eldönti, hogyan reagáljon a SNARE a „látogatónak”. Ez lehetővé teszi számos érzékelő viselkedésének megváltoztatását valós időben. Szintén nyílt forráskódú.

KFSensor, [14] ez egy nem nyílt forráskódú honeypot. Fizetős, dobozos termék. Free trial verzió letölthető és 30 napig használható. Windows alapú és teljes GUI – val rendelkezik. Minden portot figyel, az összes TCP- és UDP-portot figyeli, valamint észleli az ICMP - üzeneteket. A KFSensor Enterprise kiadás több honeypot telepítésének kezelését és felügyeletét is tartalmazza. A hálózat különböző érzékelőitől származó események valós időben összefűződnek, lehetővé téve a támadások azonnali megfigyelését. Könnyű frissíteni a szabálybázist és új szabályokat létrehozni közvetlenül egy eseményből.

T-Pot, az „All In One Honeypot” platform [15].
Az elmúlt néhány évben a Deutsche Telekom honeypot projektjében számos honeypot - ot állított fel a saját hálózataiban, a Telekom hálózataiban és a partnereknél világszerte. A honeypot - ok által gyűjtött adatok a korai előrejelző rendszerük alapvető elemei, és a Sicherheitstacho / "Securitydashboard" adatait táplálják.
A számos helyszínen, a honeypot rendszerek felállítása során szerzett tapasztalataik azt mutatták, hogy sokan érdeklődtek valamilyen honeypot érzékelő működtetése iránt, de a bevezetési eljárás és a karbantartás kissé túlterhelő volt számukra. A múltban a Deutsche Telekom szakemberei sok tapasztalatot szereztek a konfigurációk kezelésével kapcsolatban, és végül úgy döntöttek, hogy létrehoznak egy olyan honeypot rendszert, amely könnyen telepíthető, alacsony karbantartási igényű, és a legjobb honeypot technológiák közül néhányat egyesít egy rendszerben. Mindezt egy új technológia, a docker segítségével.
A T-Pot telepítésénél létrehoz egy olyan rendszert, amelynek teljes TCP hálózati tartománya, valamint néhány fontos UDP szolgáltatás honeypot - ként működik, és minden bejövő támadási forgalmat továbbít a legmegfelelőbb honeypot démonokhoz, hogy azok válaszoljanak és dolgozzák fel az aktuális támadást.
A technikai koncepció:
A T-Pot egy Ubuntu ISO képfájlra épült. A honeypot démonokat és a többi támogató komponenst docker segítségével paravirtualizálták. Ez lehetővé tette, hogy több honeypot démont fussunk ugyanazon hálózati interfészen anélkül, hogy problémák merülnének fel, és ezért az egész rendszer kevés karbantartást igényel.
A honeypot démonok beágyazása a dockerbe jó izolációt biztosít a futási környezeteknek, így egyszerűek a frissítési mechanizmusok.

A T-Potban kombinálják a meglévő honeypot - okat, (kezdetben a glastopf, kippo, honeytrap és dionaea) a hálózati IDS/IPS suricata-val, az adatmonitorozás és vizualizáció tekintetében az elasticsearch-logstash-kibana „hármast”, valamint a saját adatbeküldőt, az ewsposter –t, amely már támogatja a hpfeeds honeypot adatmegosztását.

A dockerben tárolt adatok mindig időlegesek. Amikor egy docker konténer összeomlik, benne tárolt adatok elvesznek, és újraindításra kerül egy friss példány. Ezért néhány olyan adat, amelynek meg kell maradnia, például konfigurációs fájlok stb., egy tartós adattárolóban van csatlakoztatva a gép /data/ könyvtárához, hogy elérhető legyen a konténerek vagy a rendszer újraindításakor.
Az ewsposter rendszeresen ellenőrzi a honeypot konténereket az új támadások szempontjából, és adatokat küld a közösségi háttér rendszerbe. A T-Pot projekt keretében minden eszközt és dokumentációt biztosítanak annak érdekében, hogy saját honepot rendszert hozhasson létre bárki.
A T-Pot jelenleg 16db. Honeypot - hoz kínál docker image-eket.

Támogatott eszközök száma, szintén docker tekintetében, jelenleg 9 db.

Megvalósult honeypot projektek/tanulmányok:

A bemutatott tanulmányokkal szeretném érzékeltetni, hogy a honepot technológia mennyire sokoldalú és alkalmazható a legkülönbözőbb környezetekben.

Rosszindulatú felhasználók profilozása az Ethereum hálózaton.[16]
Az Ethereum hálózaton, amely egy blokklánc, interfészként a JSON-RPC protokollt használják. Ennek a protokollnak a lényege: egy webes alkalmazás interakcióba léphessen az Ethereum blokklánccal (azaz blokklánc adatokat olvasson és/vagy tranzakciókat küldjön a hálózatba), melynek kapcsolódnia kell egy Ethereum csomóponthoz.
Erre a célra minden Ethereum kliens megvalósítja a JSON-RPC specifikációt, így létezik egy egységes metódus készlet, amelyre az alkalmazások támaszkodhatnak.
A JSON-RPC egy állapotmentes, távoli eljáráshívási (RPC) protokoll. Számos adatszerkezetet és azok feldolgozásával kapcsolatos szabályokat definiál.
Napvilágot látott olyan támadás az Ethereum kliensek /pl.: Go-Ethereum, Parity/ ellen, ahol a fent említett JSON-RPC API végpontot kívülről elérhető állapotba „helyezték". Ennek eredménye, a támadó a klienst, mint tulajdonos, tudta kezelni, vagyis ellopta a kripto valutákat a számlatulajdonostól. A 2018 – as NICTER jelentése szerint a darkneten a 8545/TCP porthoz kapcsolódó csomagok az összes megfigyelt csomagok 1,7% - át tették ki. Ez alapján a JSON-RPC a nyolcadik legtámadottabb célport. Igaz, ez 2018 – évi adat. Tudni kell, hogy néhány évre rá beköszöntött az ú.n. kripto tél, mely 2023 elején ért véget. Így az ilyen jellegű támadások valószínűleg újult erővel fognak visszatérni.
Ebben a kutatásban, abból a célból, hogy az ilyen eseteket kivizsgálják, kilenc országban (Ausztrália, Kanada, Németország, Franciaország, Japán, Hollandia, Szingapúr, Egyesült Királyság, USA.) helyeztek el honeypot - okat, az Ethereum hálózatra fókuszálva. A begyűjtött adatok alapján lehet kialakítani valamilyen profilt, a támadok viselkedési mintái és indítékaik alapján.

A vizsgálatban egy konkrét Ethereum alkalmazást – a már említett Go Ethereum – csomópontot, hivatalos nevén Geth. Egy ilyen alkalmazást szimuláló honeypot - ot telepítettek, melyeken megnyitották a sérülékeny 8545/TCP portot.
A Geth szó a „Go Ethereum" szóból származik. A „Go" és az Ethereum programozási nyelv keveréke. Ez teszi lehetővé a fejlesztők számára parancsok futtatását az Ethereum csomópontokon. Ezenkívül lehetővé teszi kriptovaluták bányászását és intelligens szerződések végrehajtását.
A csomópont a Rinkeby nevezetű Ethereum teszthálozatán lett telepítve, mivel a teszthálózaton ugyanazt a számú portot használják, mint az éles hálózaton, így a támadók nem tudták, hogy melyik hálózathoz tartozik a csomópont, amíg el nem végzi a vizsgálatokat. Így ugyan az a támadás figyelhető meg, mintha az éles rendszert támadnák. A honeypot - ok telepítése a már említett országokban történt, és igyekeztek országokra lebontva adatokat gyűjteni. Az adatokat a kutatók nyolc napon keresztül gyűjtötték 2019. április 30 és május 7 közötti időszakban. A begyűjtött kérések száma 956,151 db. volt.
A támadások és támadók részletes elemzése a következő három adattípus kombinálásával készült.

a. rosszindulatú kommunikációs előzmények, melyeket a honepot - okra küldtek,
b. Ethereum hálózati információk,
c. Darknet - ből érkező csomagok

Az adatokból kiderül, hogy a támadók nem az országokat támadják, hanem a sérülékenységeket keresik, földrajzi fekvéstől függetlenül. Ezt támasztja alá az is, hogy az összes országban a kéréseket küldő első öt IP cím megegyezett. Továbbá, az egyes országokra küldött kérések számának arányában nem volt szignifikáns eltérés.
Az egyedi IP címek tekintetében azt lehet megállapítani, hogy azok nem nagyon változtak a vizsgált időszak alatt. Ám a kérések száma gyorsan emelkedett. A rosszindulatú IP címek elemzése megmutatta, hogy az összes kérés 98,4% - át az első tíz IP címről érkezett, az első öt IP címről a kérések 91,6% - át tette ki. Az első három IP cím pedig a kérések 76.6% - át adta. Így kiderült, hogy egy bizonyos IP címekről érkező kérések tették ki a legnagyobb forgalmat.
A kérések tartalmát tekintve a fő cél az EAO – account hozzáférésének a megszerzése. Az EOA /externally-owned account/ fiókot az kontrolálja, aki birtokában van a privát kulcsnak. A támadók első körben mindig a csomóponti információkat próbálták felderíteni. Számlaegyenleg, elérhető műveletek és jogosultságok, beleértve az rendszergazdai jogosultságot is. Ezen információk birtokában igyekeznek megszerezni az utaláshoz szükséges jogosultsági szintet. Mivel a jelszó hiányában ez nem lehetséges, így bruteforce vagy szótár alapú jelszó töréssel próbálkoztak. Ez azért is lehetséges, mert a Geth a specifikációjában nem korlátozza a jelszó kísérletek számát egy bizonyos időn belül.
Az elemzésből kiderült, hogy érkeztek csomagok a Darknet hálózatból is. Lehetett látni, hogy az Ethereum szolgáltatási port - tal folytatott kommunikáció nem csak az olyan „kutatási" célzatú kommunikáció volt, mint pl.: a Shodan, hanem támadási kommunikáció is volt benne, amely az Ethereum - ot célozta.

Konklúzió: számos támadási kísérlet volt megfigyelhető az olyan sebezhető csomópontok ellen, melyek a JSON-RPC API – t használták a 8545/TCP - vel. Kimutatható volt, ha bekövetkezett a legrosszabb forgatókönyv - sikeres támadás – akkor ott jogosulatlan kriptovaluta utalás kockázata állt fenn.

Honeypot - tal és gépi tanuláson alapuló detektálási keretrendszerrel való védekezés az IoT eszközökkel végrehajtott botnetes DDos támadások ellen. [17]
A dolgok internete, amely az összekapcsolt dolgok hálózata, anélkül, hogy emberi beavatkozásra lenne szükség, mostanra sok DDoS támadás forrásává vált. Az IoT eszközök könnyebben sebezhetőek, mint az asztali számítógépek. Ezért van egy jelentősen növekedés az IoT alapú botnet támadások számában. A botnet a botok hálózataként összekapcsolt botok hálózata (kompromittált IoT eszközök).
A Nemzeti Kibervédelmi Intézet jelentése [18] szerint, 2021 – ben körülbelül 25 milliárd ilyen eszköz volt csatlakoztatva az internetre. A növekedés megállíthatatlannak tűnik. Vannak becslések melyek szerint 2030 – ra ez a szám elérheti a 75 – 100 milliárdot is. Nyilván ezek a potenciálisan sebezhető eszköz nem maradnak észrevétlenek a kiberbűnözők számára sem. Néhány ilyen támadás csak az iparból:
2003, USA, néhány államban a Blaster worm nevű vírus mely egy Windowsos sérülékenységet használt ki. Néhány órától több hétig tartó áramkimaradást okozott. [18]
2010, Irán, Stuxnet malware. Windows sérülékenységet kihasználva a támadók az urándúsító centrifugákat vezérlő PLC-ket célozták meg és tették is használhatatlanná. [18]
2015, Ukrajna. Áramszolgáltatók elleni támadás egy komplex művelet volt, támadták a SCADA rendszereket és ezzel egy időben DDoS támadással az áramszolgáltató webszervereit és ügyfélszolgálatait is elérhetetlenné tették. SCADA: olyan rendszerek, melyek lehetővé teszik az ipari folyamatok vizualizációját, a vizuális felület általi beavatkozást, valamint az adatgyűjtést. A rövidítés az angol “Supervisory Control And Data Acquisition” kifejezésből ered. [18]
2019, Dél Afrikai Köztársaság. Zsarolóvírus miatt több ezer ember maradt áramszolgáltatás nélkül. [18]
2020, Portugália. Közműszolgáltatót ért zsarolóvírus támadás és adatlopás, amelynek következtében több terabájt adatot loptak el és titkosítottak. [18]
Persze a sort lehetne folytatni más területekről is.
Az említett támadások azért voltak említésre méltóak, mert nagyon sok embert érintettek. Az anyagi károkról nem is beszélve.
Ebben a projektben a javasolt megoldás egy honeypot keretrendszer alkalmazása volt, abból a célból, hogy elkapják a kísérletet, amikor a malware próbál települni egy IoT eszközre. A napló fájlokból begyűjtött információk lesznek az inputok, a gépi tanulási modell gyakorlásához. A honeypot használatának előnye, szemben az adathalmazokkal és a modell képzéséhez az, hogy képesek lesznek a kutatók a modellt a rosszindulatú programok ismeretlen variánsaira is megtanítani, ahelyett, hogy csak korlátozottan, az ismert adatokat használnánk.

A gépi tanulás koncepcióját használták a megoldásukban, hogy automatizálják az IoT-eszközökre érkező biztonsági fenyegetések észlelésének és előrejelzésének folyamatát a megfelelő tanulási algoritmusok és technikák segítségével.
A tanulási algoritmusok általában felügyelt és felügyelet nélküli tanulási módszerekre oszthatók. A felügyelt tanulás esetén a tanítási folyamat során címkéket kell hozzárendelni az osztálykategóriákhoz, amelyek alapján a címkéket előrejelzésként lehet használni, ha a megfelelő jellemzők hasonlósága közelítenek egymáshoz. A felügyelet nélküli tanulás során nem szükséges az ilyen címke – osztály összerendelése. Mivel az algoritmus a tanítási adathalmaz különböző jellemzői közötti hasonlóságok alapján osztályoz. A megoldásunkban a felügyelet nélküli tanulási algoritmust részesítjük előnyben, mivel nem szeretnének emberi beavatkozást a folyamatba, mert egy szakértőre lenne szükség a szabályok kialakításához és a címkék hozzárendeléséhez. A leggyakoribb felügyelet nélküli tanulási módszerek közé tartoznak a klaszterezés, az anomália felismerés és a neurális hálózatok. A kártevők detektálása leginkább klaszterezési vagy osztályozási feladat. Az osztályozási probléma esetén ismert adatpéldák állnak rendelkezésre, ezért a felügyelt tanulási módszer mellett döntöttek, hogy osztályokra sorolják a probléma jellegét. A klaszterezési problémában ismeretlen káros programtípusokat, több klaszterbe rendeznek, azonosított, bizonyos tulajdonságok alapján, amit egy felügyelet nélküli tanulási algoritmus végez.
Ezen túlmenően a gépi tanulás alkalmazásának előnye, hogy képes kisebb mennyiségű a hamis pozitív és hamis negatív eredményeket produkálni, mint a más anomália-felderítési módszer.
A dolgozatom korábbi részében már említésre került az anomália alapú detektálás, amely nem használ szabályokat, hanem egy küszöbértéket állít be a normál felhasználói viselkedéshez, és bármilyen eltérés azonnal jelzi a lehetséges rosszindulatú viselkedést. Az ilyen rendszerek magas hamis pozitív arányokkal küzdenek, mert a támadók most már utánozhatják a normál viselkedést is. A gépi tanuláson alapuló megoldás képes kezelni ezt a problémát, mivel tud tanulni és idővel tanítani. Így hatékony és frissített adatokkal való modellképzéssel egy pontosabb osztályozás érhető el kevesebb hamis pozitív eredménnyel. A gépi tanulás koncepciója a honeypot által előállított dinamikus adatok hatékonyabb felhasználására szolgál, és növeli a jövőbeli támadások előrejelzésének lehetőségét.
Számos gépi tanulás módszer is javasolt a DDoS támadások azonosítására, amelyek statisztikai jellemzők kiválasztásán alapulnak, többek között a felügyelt tanulási algoritmusok. Azonban ezek a módszerek nagy hálózati szaktudást igényelnek a megfelelő jellemzők kiválasztásához az adathalmazból, és általában csak egy vagy néhány DDoS támadási vektorra korlátozódnak. Emellett rendszeres frissítéseket igényelnek annak érdekében, hogy különböző helyzetekben is megfelelően működjenek.
Egy másik gépi tanuláson alapuló megoldás a DDoS támadások észlelésére az ún. mély tanulási modellek, mint a Konvolúciós Neurális Hálózat (CNN) [19], a Rekurzív Neurális Hálózat (RNN) [20]. Egy hálózati alapú anomáliás észlelési módszer is felmerült, amely

kivonatolja a hálózat viselkedési pillanatképeit és mély autoenkóderek (ez egy felügyelet nélküli tanulást megvalósító neurális hálózat) segítségével észleli a kompromittált IoT eszközökből származó anomális hálózati forgalmat [21]. Azonban a mély tanulási modelleknek nagy mennyiségű adatra van szükségük a pontos eredmények előállításához. Ennek eredményeképpen rendkívül számításigényesek és bonyolultak a tanítási folyamatuk, és gyakran jelentős időt igényelnek a tanuláshoz. Az IoT eszközök nem engedhetik meg maguknak az ilyen kiterjedt eljárásokat, mivel erőforrások tekintetében korlátozottak és valós idejű szolgáltatásokat kell nyújtaniuk a felhasználónak. Emellett szükség van új módszerek kidolgozására az észlelésre, amelyek megkülönböztetik az IoT eszköz alapú támadásokat órás és milliomod másodperces támadások között.
A javasolt megoldásban nem csak a malware detektálására koncentráltak, hanem az is érdekelte a kutatókat, hogy az ismeretlen malware családokat is azonosítsák, amelyek, pl. felelősek a Zero-Day DDoS támadásokért. A Zero-Day támadásokat a még nem azonosított malware okozták, amelyek ellen nincs védelem. Ezt a problémát honeypot - tal oldják meg, egy gépi tanuláson alapuló detektálási keretrendszerrel. A honeypot itt is arra szolgál, hogy szándékosan becsalogatjuk a támadókat azzal a céllal, hogy a malware - ek tulajdonságait és a biztonsági problémákra gyakorolt hatását rögzítsék az IoT eszközökön. Emellett egy gépi tanuláson alapuló detektálási keretrendszert is alkalmaznak, amely a honeypot által generált naplófájlok alapján előre jelzi az abnormális aktivitás lehetőségét, egy osztályozó algoritmussal, mely egy felügyelet nélküli algoritmust. Az architektúra: A folyamat akkor kezdődik, amikor egy támadó megpróbálja bejuttatni a malware - t egy nyitott porton (23 vagy 2323), többféle username/password kombináció használatával bejelentkezve egy IoT eszközbe. A honeypot ebben a helyzetben szándékosan lehetővé teszi a támadó számára, hogy hozzáférjen az eszközhöz, megsértve a védelmi vonalat. A fő szándék az információ megszerzése a malware - ről, a támadóról, továbbá az eszköz és a behatoló közötti minden tevékenységről, melyeket bekerülnek a naplóállományokba. Ezek a fájlok olyan információkat rögzítenek, amelyek lehetővé teszik a kutatók számára, hogy megtudjuk az új malware - ek jellegét, változatait, a célozott eszközök típusát, valamint a C&C szerver IP-címét, portszámát stb. A gépi tanulási modellünk tanításához át kell alakítanunk a log fájl adatait a megfelelő táblázatos formátumba. Az osztályozáshoz előnyös egy memória hatékony megoldás, amely minimális edzési adatokat használ fel a hasznos információk előrejelzéséhez, hogy elkerülje az IoT eszközök terhelését. Végül, az osztályozás eredménye alapján megfelelő intézkedést hozunk. Az edzés folyamata ismétlődik minden alkalommal, amikor meghaladja az engedélyezett edzési adatok méretét, hogy a folyamat dinamikus legyen, és könnyen futtatható legyen az erőforrásokban korlátozott IoT eszközökön.

Megvalósítási szempontok:

A megvalósítás elengedhetetlen része bármilyen új megközelítésnek vagy ötletnek annak ellenőrzéséhez, hogy megvalósítható-e és értékelhető-e a hatékonysága a jelenleg

elérhető hasonló megoldásokhoz képest. Ahogy a fentebb tárgyalt részben ismertetésre került, javasolt megközelítés különböző egymást követő lépésekből áll. Minden lépésnél alkalmazhatóak a legújabb módszerek, hogy megfelelően, frissített megoldás álljon a rendelkezésre a jelenlegi IoT kihívások kezelésére. A következőkben felsorolásra kerülnek azok a fejlesztések, amelyek az elmúlt években történtek az IoT csalihálózatok és a valós idejű gépi tanulás alapú detektálás területén, és ezek a két legfontosabb lépést jelentik megközelítésben a kívánt megvalósítás elvégzéséhez:

IoT Virtuális honeypot:

Javasolt megközelítés első lépése, hogy csaljuk be a támadókat, akik szándékosan kihasználják az IoT eszközökben található sebezhetőségeket. Ennek a viselkedésnek a szimulálásához olyan rendszerre vagy eszközre van szükségünk, amely pontosan úgy működik, mint egy kihasználható IoT eszköz, és arra ösztönzi a támadót, hogy rosszindulatú lépéseket tegyen anélkül, hogy megkérdőjelezné az exploitok valódiságát. Az ilyen rendszereket általában IoT honeypot - oknak nevezik. Ezen honeypot - ok interakciós szintje a közepes kategóriába sorolható (MIH), mivel erőforrások korlátozottsága miatt nem lehetséges magas interakciós honeypot - ot (HIH) beállítani. Ezért hívjuk ezeket IoT "virtuális" honeypot - oknak, mivel ebben az esetben, az így szimulált IoT platformon keresztül valósítjuk meg a honeypot - ot az IoT kommunikációs protokollok használatával. A támadási stratégiák, mint például a hálózati forgalom, a payload, a kártevő minták, a támadó által használt eszközkészlet stb. rögzíthetők a honeypot által.

Az alábbiakban ismertetek néhány, a közelmúltban kifejlesztett IoT honeypot - ot a DDoS detektáláshoz:

IoTPOT [22]: ez a honeypot különböző IoT eszközök Telnet szolgáltatásainak emulálására szolgál, áll egy frontend, alacsony interakciós felülettel és egy backend, magas interakciós virtuális környezettel, ez az IoTBOX. A két komponens együttműködve képes különböző CPU-architektúrákon működni.

Telnet IoT honeypot [23]: ebben a honeypot - ban a Telnet szervert használják az IoT csapda megvalósításához.

HoneyThing [24]: ez a honeypot egy sebezhető modem/router emulátora (RomPager beágyazott webszerverrel), és TR-069 (CPE WAN kezelési protokoll) specifikus.

Dionaea [25]: ez a honeypot MQTT protokollt használ az IoT viselkedésének szimulálására. Az MQTT protokoll (Message Queue Telemetry Transport) egy ISO szabvány, publish/subscribe (one-to-many) alapú, pehelysúlyú üzenetküldő protokoll. A protokollnak alacsony a sávszélesség igénye. Szükséges hozzá egy szerver ún. broker. A broker feladata a beérkező üzenetek továbbküldése a klienseknek, téma alapján. Azaz a kliensek

„feliratkoznak” különböző témákra, majd a broker a témákba érkezett üzeneteket továbbítja a feliratkozott klienseknek. Nyílt, ingyenes, a küldő félnek semmit sem kell tudnia a vevő(k)ről. Ideális, zajokkal terhelt hálózatoknál, pl. kis sávszélesség, nagy késleltetés, csomagvesztés, instabil kapcsolatok. Illetve szűkös erőforrásokkal rendelkező eszközök számára pl.: kevés memória, kis teljesítményű processzor. A célja, megbízható üzenetküldés a lehető legegyszerűbb üzenetformátumban.

ZigBee Honeypot [26]: ez a honeypot egy ZigBee átjárót szimulál. A Zigbee egy IEEE 802.15.4 hálózati szabványon alapuló, vezeték nélküli kommunikációs nyelv, bizonyos értelemben a Wi-Fi és a Bluetooth alternatívájára az olyan az alacsony fogyasztású eszközöknél, amik nem igényelnek nagy sávszélességet (IoT), A Zigbee nem a ponttól-pontig való kommunikációra készült, mint a Bluetooth, hanem mindenképpen szükséged van egy helyileg telepített központi egységre, egy hub-ra vagy gateway-re, mellyel minden eszköz kommunikál. A Zigbee előnye, hogy egy tisztán Zigbee-alapú rendszernél pl.: smart home, csak a hub/gateway rendelkezik WiFi-s vagy vezetékes internetkapcsolattal, azaz a sok okoseszköz nem terheli túl a WiFi hálózatot, hiszen az egyes eszközök egy kifejezetten okoseszközöknek fenntartott, a WiFi-től eltérő frekvenciasávokat használnak a rendszerben, melyen keresztül kommunikálnak egymással.

Többcélú IoT honeypot [27]: IoT honeypot a Telnet, SSH, HTTP és CWMP-re összpontosít. A CWMP - CPE WAN Management Protocol - egy alkalmazási rétegű protokoll. Célja, hogy távelérréssel lehessen kezelni az ügyfél helyszínén lévő eszközöket (CPE - customer premise equipment), amelyek az Internet Protocol (IP) hálózatra vannak csatlakoztatva. Ez a kétirányú menedzsment protokoll a CPE és az automatikus konfigurációs szerver (ACS) közötti kommunikációra szolgál. A protokoll támogatja az automatikus konfigurációt, a szoftver- vagy firmware képfájlkezelést, a szoftvermodul kezelést, az állapot- és teljesítménykezelést, valamint a diagnosztikát. Tartalmazza a biztonságos automatikus konfigurációt és más CPE kezelési funkciók vezérlését egy integrált keretrendszeren belül. A protokoll telekommunikációs eszközökre, mint például modemek, routerek, átjárók, valamint olyan végfelhasználói eszközökre, amelyek az Internetre csatlakoznak, például set-top boxok és VoIP-telefonok, vonatkozik.

ThingPot [28]: ez az IoT honeypot képes egy teljes IoT platformot szimulálni, nem csak egyetlen alkalmazási réteg kommunikációs protokollját (pl. Telnet, HTTP, stb.).

A legmegfelelőbb IoT honeypot - oknak nem csak néhány kiválasztott IoT kommunikációs protokollt kell szimulálnia, hanem képesnek kell lennie a teljes IoT platform és az összes támogatott protokoll szimulálására is. Az IoT kommunikációhoz használt néhány népszerű alkalmazási protokoll: mint az előbb röviden ismertetett az MQTT (Message Queue Telemetry Transport), vagy az IBM által fejlesztett XMPP(Extensible Messaging and Presence Protocol), amely alapvető azonnali üzenetküldést(IM) és

jelenlétet biztosít. Az AMQP(Advanced Message Queuing Protocol), amely a pénzügyi iparból származik, a CoAP (Constrained Application Protocol), a korlátozott erőforrásokkal rendelkező eszközökre lett tervezve, az UPnP (Universal Plug and Play), a hálózati eszközök felfedezésére szolgáló hálózati protokollhalmaz, valamint az HTTP REST. A REST egy olyan architekturális vonal, amelyet széles körben használnak gép-gép közötti (M2M) kommunikációban és IoT platformokon. A felsorolt honeypot - ok közül a Thing-Potot használhatjuk a kártékony támadásunk céljára.

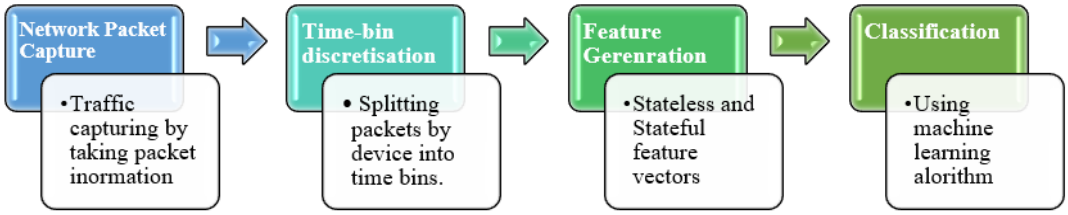
Valós idejű gépi tanuláson alapuló detektálási keretrendszer:
A gépi tanuláson alapuló detektálási keretrendszer egy másik, fontos lépése, osztályozás a DDoS detektálásában. Számos gépi tanulási algoritmus áll rendelkezésre ennek elvégzéséhez. Azonban nem csak a kártevők osztályozása érdekel minket, hanem olyan gépi tanulási megoldást szeretnénk, amely valós időben alkalmazható és pontosan képes azonosítani a kártevő jellemzőit, anélkül hogy hamis pozitív eredményeket eredményezne. Az IoT eszközökben valós idejű gépi tanuláson alapuló detektálás területén végzett kutatások közé tartozik a R. Doshi és munkatársai [29] által javasolt megoldás, amely bizonyította, hogy 0,99 pontossággal képes osztályozni a kártevőket. A megoldás kifejezetten az IoT botnet támadásokra összpontosít, amelyek az utóbbi években drasztikusan megnőttek.

Az IoT forgalma másként viselkedik, mint a hagyományos laptopok és okostelefonok forgalma, mivel az eszközök kis hatótávolságon belül kommunikálnak a végpontokkal, nem pedig nagy web szerverekkel. Az IoT forgalom ilyen jellegzetes viselkedése egy gépi tanulási folyamatban megfigyelhető. A folyamat több lépésből áll, kezdve az adatgyűjtéstől, a feature extraction - ból, amely a gépi tanulási modell által feldolgozott bemeneti adatok átalakítását jelenti, könnyen feldolgozható formába és ezzel javítva a teljesítményt. A befejező lépés a bináris osztályozás. Így kinyert adatok, főleg az IoT-specifikus hálózati viselkedésekre és hálózati forgalom jellemzőire összpontosítanak, mint például a csomaghossz, az időközök a csomagok között és az érintett protokoll. Többféle osztályozó alkalmazható támadások észlelésére, beleértve a Random Forest, a K-nearest neighbors – t, a Support Vector Machine - t (SVM) és a neuronhálózat osztályozók különösen hatékonyaknak bizonyultak [29]. Az IoT-specifikus hálózati viselkedés, például az endpointok számának korlátozása, a csomagok közötti rendszeres időközök stb. használhatóak az adatokból kinyert tulajdonságok kiválasztására, hogy magasabb pontosságot érjünk el az IoT-hálózati forgalomban bekövetkező DDoS támadások észlelésében a különböző gépi tanulási algoritmusok, például a neurális hálózatok segítségével.

Az anomália észlelés egy folyamat, amely különböző fázisokon halad keresztül, kezdve az adatforgalom rögzítésével, majd az adatcsomagok csoportosításával, eszköz és idő szerint, majd a jellemzők kinyerésével, és végül a bináris osztályozási fázisban végződik. Az adatforgalom rögzítése az IoT eszköz által küldött összes IP csomag, forrás IP-címének, forrás portjának, célcímének, célportjának, csomagméretének és

időbélyegzésének rögzítéséről szól, ami egy okos otthon alkalmazás része. Ez a DDoS forgalom gyűjtése meglehetősen nagy kihívást jelent a biztonsági kockázatok és a bonyolultság miatt. A leggyakoribb DDoS támadás három változatát, azaz a TCP SYN forgalmat, UDP forgalmat és az HTTP GET forgalmat szimulálják, hogy rögzítsék az újonnan érkező variánsokat a kártevők tulajdonságaiban. Az IoT eszközök csomagjait csoportosítják a forrás IP-cím alapján, a korábban rögzített időbélyegek alapján.

A jellemzők kinyerése, az a folyamat, amely felelős az állapotfüggő és állapotmentes jellemzők előállításáért minden csomag esetében, az IoT eszköz viselkedésétől függően. Az állapotmentes jellemzők azok a jellemzők, amelyek minden küldött csomag független jellemzőiből származnak, azaz anélkül generálódnak, hogy az IP-forrástól érkező forgalmat szétvágnánk. Másrészről, az állapotfüggő jellemzők arról szólnak, hogy az összesített áramlási információt rögzítik a hálózati forgalomban az adott időtartam alatt. A csomagméretet és a csomagok közötti idő állapotmentes jellemzőnek tekintik, míg a sávszélességet, az IP-cím gyakoriságát állapotfüggő jellemzőknek nevezik. Végül a bináris osztályozás folyamata következik, amely a különböző osztályozási algoritmusok (például Random Forest, a K-nearest neighbors, a Support Vector Machines (SVM) és a neuronhálózat) felhasználásával történik, hogy megkülönböztessék a normál forgalmat a DDoS forgalmától. A 30. forrású cikk, teljes folyamatábrája látható a 2. ábrán.



1. ábra, valós idejű gépi tanulás teljes folyamatábrája

A javaslat összefoglalása, az IoT-eszközök különböző fázisain keresztül haladó anomáliakeresési folyamat megvalósítható egy olyan IoT honeypot használatával, amelyet a ThingPot inspirált, és amely képes különböző IoT kommunikációs protokollok utánzásával és az egész IoT platform viselkedésének emulálásával különböző botnet binárisokat elkapni. Az IoT korlátai miatt nem lehetséges minden eszközön osztályozót implementálni, ezért a router szintjén kell megvalósítani. Emellett, az adott IoT eszközre érkező forgalom mennyisége nem add elegendő információt a szükséges gépi tanuló modellek képzéséhez, ezért bármely IoT hálózat szimulátor használható az IoT forgalom megfelelő mennyiségének generálására. A log fájlok átalakítása a szükséges formátumá alakítása a bemeneti adatokhoz való felhasználás céljából Linux bash szkriptek használatával végezhető el. Az IoT szkenneléshez használható eszközök a Microsoft Azure, MATLAB stb. gépi tanulási eszközök virtuális környezetben történő használatával megvalósíthatók. A

valós idejű gépi tanulásos észlelő keretrendszer megvalósításához az előbb említett módszert használhatjuk.

A dolgok internete (IoT) a világ egyik legmeghatározóbb technológiája, emiatt növekszik az ilyen jellegű kibertámadások száma. Számos védelmi mechanizmust javasoltak a területen, de képtelenek kezelni az új IoT botnet támadások változatait. A cikk egy honeypot alapú megoldást javasol a DDoS észlelésére, amely valós idejű gépi tanulású észlelő keretrendszerre épül. A honeypot - ok hálózatának használata biztosítja a malware tulajdonságainak naplózását, amelyek hatékonyan képzik a gépi tanulási algoritmusokat.

4 4. IDS VS. HONEYPOT

Az IDS-ek magas arányban adhatnak fals pozitív riasztást. Minden honeypot értékét az általa begyűjtött információk határozzák meg. A honeypot - ba be - és kilépő adatok figyelése lehetővé teszi számunkra, hogy minden apró mozzanatot gyűjtsünk, loggoljunk. A rosszindulatú viselkedés észleléséhez az IDS - nek szüksége van az ismert támadások karakterisztikájára. Ellenben honeypot olyan támadási technikát és sebezhetőségeket is észlelhet, amelyek még nem publikusak vagy nem ismertek. Ilyenek pl.: a zero day sebezhetőség, bár ez akár anomália alapú IDS – re is igaz. Továbbá, pl.: egy honeypot - ból gyűjtött adatok törvényszéki elemzése kisebb valószínűséggel vezet fals pozitív eredményhez, mint az IDS által gyűjtött adatok. Az "elméleti" összehasonlítás az 1. számú táblázatban látható.

	IDS	**Honeypot**
Cél	Detektál és riaszt	Csali rendszer/riaszt
Működés	Felügyeli a hálózati forgalmat és elemzi a hálózati csomagokat.	Passzív figyelés a hálózati forgalom irányában, támadók tevékenységét rögzíti.
Fő feladata	Azonosítja a kártékony tevékenységeket és riasztást ad az incidensekről. IPS esetén beavatkozik.	Csaló rendszerként működik, hogy elcsábítsa a támadókat, és információkat gyűjtsön a támadási módszerekről és célpontokról.
Védelem	Passzív védelem, nem csatlakozik aktívan a támadás elleni küzdelemhez.	Nincs aktív védelem, szándékosan sebezhetőnek tűnik.
Előnyök	Gyorsan észleli a hálózati támadásokat és incidenseket.	Kiváló adatforrás a támadási módszerek, célpontok és sebezhetőségek elemzéséhez. Segíti a védelmi stratégiák fejlesztését.
Korlátaik	Nem képes megállítani a támadásokat vagy a kártékony tevékenységeket, csak észleli azokat és riaszt. Sok hamis pozitív riasztást okozhat.	Nem képes figyelni a teljes hálózatot. A támadók képesek felismerni és kikerülni.
Felhasználás	Vállalatok, szervezetek és szolgáltatók használják a hálózati biztonság növelésére.	Biztonsági szakemberek és kutatók használják leginkább, a támadási módszerek és fenyegetések elemzésére. Cégeknél is megjelenik.

Átláthatóság	Az IDS rendszerek általában jól láthatók a hálózaton.	A honeypot- ok is láthatóak, csak a mibenlétük nem.
Jogszabályok	Az IDS rendszerek alkal-mazásához nincs különleges jogi szabályozás.	Adatvédelmi kérdések.

1. táblázat, IDS és Honeypot elméleti, táblázatos összehasonlítása

5 5. RENDSZERTERV

Kiválasztottam két IDS és két honeypot rendszert. A kiválasztást három kritérium alapján hoztam meg. Az első, hogy álljon rendelkezésre megfelelő mennyiségű dokumentáció. A második, a költségmentes vagy próbaverzió elérhetősége és nem utolsó sorban, a rendekezésemre álló gépi erőforrás megfelelő legyen.

	Dokumentáció	Elérhető verzió	erőforrás	egyéb
Snort	Megfelelő	Igen	Megfelelő	-
Suricata	Megfelelő	Igen	Megfelelő	Tanultam
Cowrie	Megfelelő	Igen	Megfelelő	-
Conpot	Megfelelő	Igen	Nem áll rend.	Ipari környezet
Heralding	Kevés	Igen	Megfelelő	-
Snare/Tanner	Megfelelő	Igen	Megfelelő	web
KFSensor	Megfelelő	Igen (30 nap)	Megfelelő	30 nap

2. *táblázat, számomra "használható" rendszerek összefoglalása*

5.1 IDS

Suricata, elegendő dokumentáció, nyílt forráskódú és a rendelkezésemre álló virtuális környezetbe telepíthető, számomra alkalmas a tesztelésre. Suricata szabályokat már korábbi tanulmányaim alatt tanultam. Ezért erre esett a választásom. Egy másik IDS rendszert is teszteltem, ez pedig a Snort. Szintén bőséges a rendelkezésre álló dokumentáció és elegendő a rendelkezésemre álló erőforrás.

5.2 Honeypot

Corwie honeypot rendszert telepítettem a saját virtuális környezetembe, igyekeztem kiismerni és megtanulni a használatukat. Majd kiválasztottam egy támadási formát. Első lépés mindenképpen a felderítés lesz, hiszen a valóságban is ezzel indul minden támadás. A támadó gép Kali linux operációs rendszerrel telepített gép, a felderítést az nmap programmal végeztem, így láttam majd, hogy hogyan reagálnak. Mivel jó lenne látni egy windows alapú honeypot – ot is működés közben, sikerült találnom egy műödőképes szoftvert, ez pedig a Valhala nevezetű „csali" szoftver.

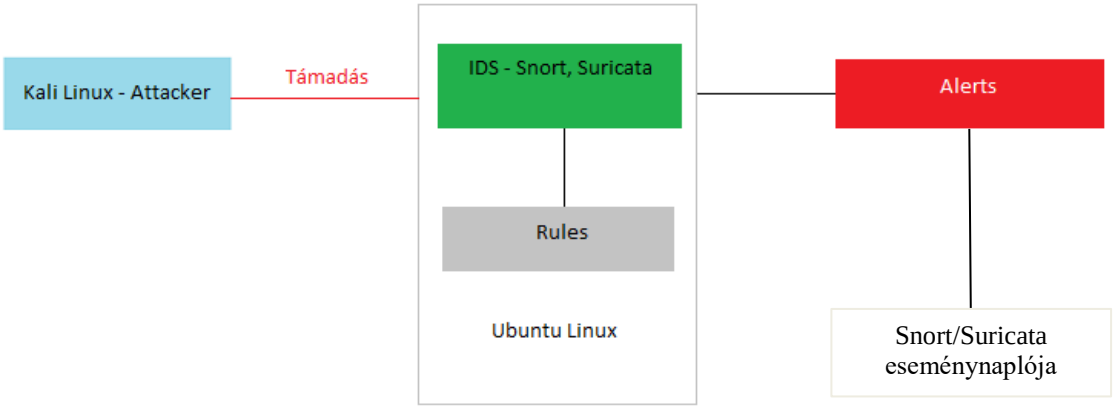
A telepítési környezet a fizikai gépemen történt. Melynek paraméterei a következők:

CPU - Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz 2.81 GHz,

RAM – 16 GB,

OP. rendszer Windows 10 pro, ver. 22H2.

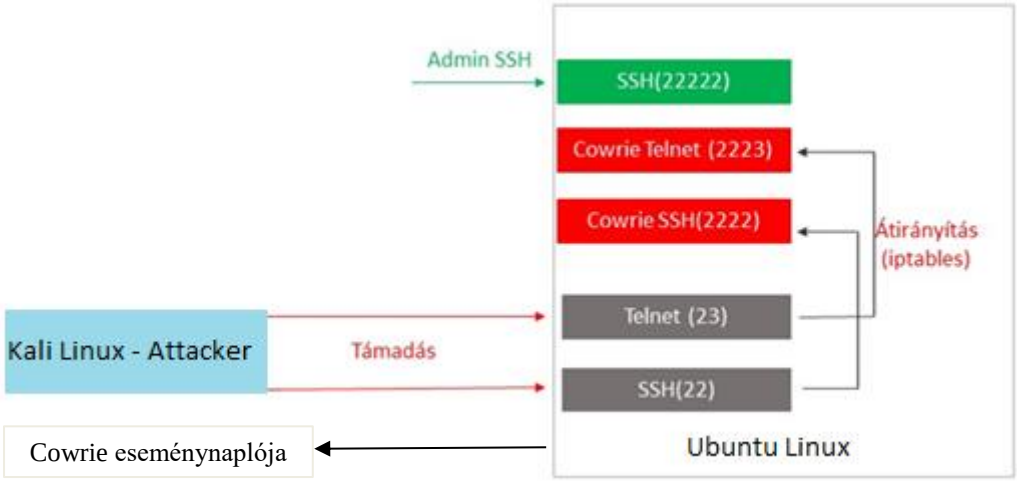
IDS: a Suricata és a Snort szoftvert telepítettem VMware Workstation virtuális környezetben. Az ott használt operációs rendszer az Ubuntu Linux. A logikai topológia a 2. ábrán látható. Mind a Suricata, minda Snort esetén ezt a topológiát valosítottam meg.



2. ábra, Sort/Suricata - val megvalósított topológia

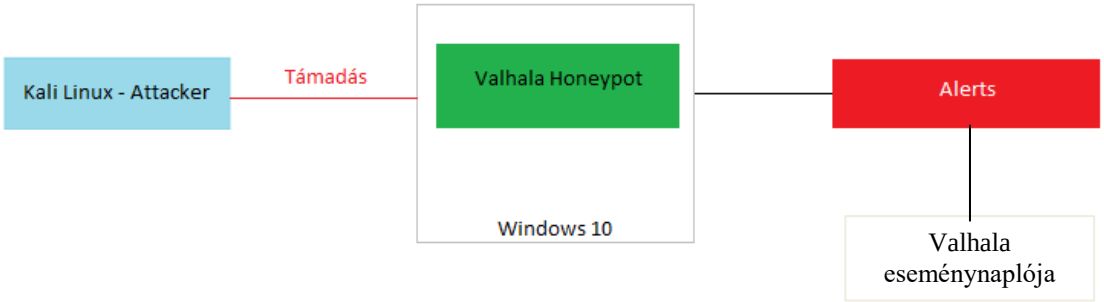
Cowrie Honeypot: a Cowrie szoftvert telepítettem virtuális környezetbe. Az ott használt operációs rendszer Ubuntu Linux. A beállításom hasonló a Cowrie alapértelmezett telepítéséhez. A hoszt SSH daemonja a 22222 porton futott, a Cowrie pedig a 2222-es porton, és a 22-es port (az alapértelmezett SSH) átirányításra került a 2222-es portra az iptables segítségével.

Tehát az SSH bot, vagy támadó, a 22-es porthoz csatlakozott, majd átirányításra került a honeypotomra a 2222-es porton. A logikai topológia a 3. ábrán látható.



3. ábra, Cowrie honeypot - tal megvalósított topológia

Valhala Honeypot: A Valhala nevezetű honeypot - ot egy windows 10 operációs rendszerre telepítettem virtuális gépen. A támadó gép szintén egy Kali Linux és a virtuális környezeten belül a két gép „látja” egymást. A topológiát a 4. ábra mutatja.



4. ábra, Valhala honeypot - tal megvalósított topológia

A számomra szóba jöhető támadási formák, az SSH és a Telnet kapcsolatokat támadó technikák összehasonlítása, a 3 számú táblázatban látható.

Támadási technikák összehasonlítása		
Támadás technika	**Telnet**	**SSH**
Biztonság	Titkosítatlan adatátvitel	Titkosított adatátvitel
Brute force	Felhasználónevek/jelszavak korlátlan próbája. Nincs beépített védelem a túlzott bejelentkezési kísérletek ellen	Kriptográfiai védelem és próbálkozások száma korlátozott
Dictionary támadás	Előre elkészített szótárat (dictionary) használ a hitelesítő adatok kitalálására. A szótár tartalmazhat gyakori jelszavakat, elterjedt felhasználóneveket és egyéb kombinációkat.	Előre elkészített szótárat (dictionary) használ a hitelesítő adatok kitalálására. A szótár tartalmazhat gyakori jelszavakat, elterjedt felhasználóneveket és egyéb kombinációkat.

Man-in-the-middle támadás	A támadó közbeiktatja magát, hogy megfigyelhesse vagy módosíthassa az adatfolyamot.	Titkosítva van. Csak privát kulccsal.
Kulcslopás	nincs kulcs	privát kulcs megszerzése.
Exploit kihasználás	Telnet szerver sebezhetőség kihasználása	SSH szerver sebezhetőség kihasználása

3. táblázat, Támadási technikák összehasonlítása

6 6. MEGVALÓSULT IMPLEMENTÁCIÓK

6.1 Cowrie honeypot – SSH

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# nmap -sC -sV 192.168.88.132
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-28 02:12 EDT
Nmap scan report for 192.168.88.132
Host is up (0.013s latency).
Not shown: 999 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
2222/tcp open  ssh     OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
```

5. ábra. Cowrie honeypot SSH kapcsolat felderítése

Telepítése egy Ubuntu linux virtualis (VMware) gépre történt. A tesztelést egy támadással végeztem. Felderítés, mint első lépés látható az 5. ábrán. Ebben a szakaszban, mint a továbbiakban is, az Nmap szoftvert használtam. Az nmap kapcsolói közül az –sC és az –sV opciókat vettem ígénybe. Az –sC kapcsolóval engedélyezem az NSE scriptek futtatását. Az -sV kapcsoló a fingerprinting technológiával probálja meghatározni a futó szolgáltatások verzióját.

Kali linux rendszer alatt:

Eredményből látható az SSH nyitott port. Ezen a ponton már megvalósul a „csali” szerep, mert a támadó talált egy nyitott portot! A rendszertervben megfogalmazott felépítés tükröződik a felderítés eredményéből. Alapértelmezetten az ssh port a 22 – es port. Az átirányítás miatt viszont a 2222 –es port látható a nyitott ssh portnak.

Mindeközben a honeypot érzékeli a letapogatást, 6. ábra:

```
root@cowrie-ubuntu:/home/xy/Desktop/cowrie/var/log/cowrie# tail cowrie.json
["eventid":"cowrie.session.connect","src_ip":"192.168.88.129","src_port":41872,"dst_ip":"192.168.88.132","dst_po
:41872 (192.168.88.132:2222) [session: 1b68391cd266]","sensor":"Cowrie-Ubuntu","timestamp":"2024-07-20T08:55:46.
["eventid":"cowrie.client.version","version":"SSH-2.0-Nmap-SSH2-Hostkey","message":"Remote SSH version: SSH-2.0-
```

6. ábra. Cowri honeypot érzékeli a portscann -t

A fenti, 6. ábra, mely egy errősen megvágott képernyőkép, helyhiány miatt nem minden látsztik. Amit a honeypot lát egy portscann alkalmával: a forrás ip címe és a port száma, a cél ip címe és port száma. Láthatóak továbbá az időbélyegek, minden egyes eseményhez. Az eseményről egy szöveges üzenet. Az érintett protokoll neve és verziószáma. A sensor – jelen esetben Cowrie – neve és annak operációsrendszer pltaformja. Ez talán egy kis magyarázatra szorul, hogy miért is kell a honeypotnak azonosítani magát. Erre azért

van szükség, ha több honeypotunk van becsatornázva valamilyen SIEM felületre vagy bármilyen dashboard – ra, akkor tudnunk kell egyértelműen azonosítani adat forrását.

A felderítés után megkiséreltem belépni (7. ábra), kiváncsi voltam a honeypot reagálására:

```
login as: tesztelek
tesztelek@192.168.23.130's password:
Access denied
tesztelek@192.168.23.130's password:
Access denied
```

7. ábra. belépési kísérlet a Cowrie honeypot-ba a támadó gépről

Világosan látszik, a honeypot teszi a dolgát, minden bejelentkezési kísérlet megjelenik a naplóállományaiban, minden beírt user/password probálkozás látható, továbbá a támadó ip címe és hogy sikeres vagy sikertelen volt a belépés, 8. ábra:

```
cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'tesztelek' trying auth b
HoneyPotSSHTransport,1,192.168.23.1] Could not read etc/userdb.txt, default data
HoneyPotSSHTransport,1,192.168.23.1] login attempt [b'tesztelek'/b'rosszjelszo']
cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'tesztelek' failed auth b'
cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] unauthorized login: ()
cowrie.ssh.userauth.HoneyPotSSHUserAuthServer#debug] b'tesztelek' trying auth b'
HoneyPotSSHTransport,1,192.168.23.1] Could not read etc/userdb.txt, default data
```

8. ábra. Cowrie honeypot reagálása a belépési kísérletre

Kipróbáltam egy windowsos gépről is a bejelentkezést, miután sikeresen bejelentkeztem, a kiadott parancsokat is lehetett látni, 9. ábra:

```
root@svr04:/# ls
bin        boot       dev        etc        home       initrd.img lib
lost+found media      mnt        opt        proc       root       run
sbin       selinux    srv        sys        test2      tmp        usr
var        vmlinuz
root@svr04:/# cd boot
root@svr04:/boot# ls
System.map-3.2.0-4-686-pae config-3.2.0-4-686-pae
grub                       initrd.img-3.2.0-4-686-pae
vmlinuz-3.2.0-4-686-pae
root@svr04:/boot#
```

9. ábra. Navigálás a Cowrie honeypot fájlrendszerében, windows alól bejelentkezve

Windows alól bejelentkezve a kiadott parancsok a Cowrie eseménynaplóban, 10. ábra:

```
2024-07-31T14:27:42.795354Z [HoneyPotSSHTransport,0,192.168.88.1] CMD: ls
2024-07-31T14:27:42.799012Z [HoneyPotSSHTransport,0,192.168.88.1] Command found: ls
2024-07-31T14:27:48.951148Z [HoneyPotSSHTransport,0,192.168.88.1] CMD: cd ..
2024-07-31T14:27:48.954909Z [HoneyPotSSHTransport,0,192.168.88.1] Command found: cd
2024-07-31T14:27:51.719300Z [HoneyPotSSHTransport,0,192.168.88.1] CMD: ls
2024-07-31T14:27:51.723320Z [HoneyPotSSHTransport,0,192.168.88.1] Command found: ls
```

10. ábra. A kiadott parancsok

```
root@svr04:/# ls
bin        boot       dev        etc        home       initrd.img lib        lost+found
media      mnt        opt        proc       root       run        sbin       selinux
srv        sys        test2      tmp        usr        var        vmlinuz
root@svr04:/# cd media
root@svr04:/media# ls
cdrom  cdrom0
root@svr04:/media# cd ..
root@svr04:/# cd usr
root@svr04:/usr# ls
bin     games   include lib     local   sbin    share   src
root@svr04:/usr# cd ..
root@svr04:/# cd root
```

11. ábra. Playlog.py "működés" közben, a kedvenc eszközöm

Cowrie honeypot – ban van egy fantasztikus eszköz, a playlog.py nevű segédprogram. Ezzel az eszközzel a bináris naplóállományokat tudjuk megtekinteni, amely valójában visszajátsza a támadó monitoráról készült „felvételt". Onnantól kezdve, hogy a támadó bejutott a rendszerünkbe, minden kiadott parancsot visszajátszik nekünk. Láthatóak az elgépelt utasítások, visszatörlések, fájlmozgatások stb (11. ábra). Ez az állomány nem az eddig ismertetett naplóállományok. Az eddigieket a /cowrie/var/log/ útvonalon találhatjuk .log kiterjesztéssel, aktuális dátummal generált fájlnévvel. A playlog programmal visszajátszható állományok a cowrie/var/lib/cowrie/tty/ útvonalon találhatók. Itt is egy generált fájlnevet kapun, mely 64 karakteres. Egy konkrét példa: ffa5b2bd73783f644f213695b41f6cef4ee60a5c057f16ad109901189509f05.

6.2 Cowrie honeypot – Telnet

Az előbb ismertetett honeypotról van szó. A tesztelést brute force támadással végeztem, a Kali linuxban található metasploit framework segítségével (8.kép). Előtte létrehoztam egy password szótárt és egy user szótárt. Majd ezeket meghívva futtatam a msf – t, 12. ábra:

```
msf6 auxiliary(scanner/telnet/telnet_login) > set PASS_FILE /home/kali/Desktop/pwd.txt
PASS_FILE => /home/kali/Desktop/pwd.txt
msf6 auxiliary(scanner/telnet/telnet_login) > set RHOSTS 192.168.88.132
RHOSTS => 192.168.88.132
msf6 auxiliary(scanner/telnet/telnet_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS => true
msf6 auxiliary(scanner/telnet/telnet_login) > set USER_FILE /home/kali/Desktop/user.txt
USER_FILE => /home/kali/Desktop/user.txt
msf6 auxiliary(scanner/telnet/telnet_login) > run

[!] 192.168.88.132:23      - No active DB -- Credential data will not be saved!
[-] 192.168.88.132:23      - 192.168.88.132:23 - LOGIN FAILED: user:12345678 (Incorrect: )
[-] 192.168.88.132:23      - 192.168.88.132:23 - LOGIN FAILED: user:87654321 (Incorrect: )
[-] 192.168.88.132:23      - 192.168.88.132:23 - LOGIN FAILED: user:Password (Incorrect: )
[-] 192.168.88.132:23      - 192.168.88.132:23 - LOGIN FAILED: user:password01 (Incorrect: )
[-] 192.168.88.132:23      - 192.168.88.132:23 - LOGIN FAILED: user:11111111 (Incorrect: )
[+] 192.168.88.132:23      - 192.168.88.132:23 - Login Successful: root:12345678
[*] 192.168.88.132:23      - Attempting to start session 192.168.88.132:23 with root:1234567
[*] Command shell session 1 opened (192.168.88.129:34369 -> 192.168.88.132:23) at 2024-04-2
[*] 192.168.88.132:23      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/telnet/telnet_login) > run
```

12. ábra. Brute force támadás metasploit framework programmal

Miután a törés sikeres volt – meg is nyitott egy sessiont, melyet a honeypot is érzékelt, 13. ábra:

```
[-] Cowrie Version 2.5.0
[-] Loaded output engine: jsonlog
[twisted.scripts._twistd_unix.UnixAppLogger#info] twistd 24.3.0 (/home/xy/Desktop/cowrie/cow
[twisted.scripts._twistd_unix.UnixAppLogger#info] reactor class: twisted.internet.epollreact
[-] CowrieSSHFactory starting on 2222
[cowrie.ssh.factory.CowrieSSHFactory#info] Starting factory <cowrie.ssh.factory.CowrieSSHFac
[-] Ready to accept SSH connections
[-] HoneyPotTelnetFactory starting on 2223
[cowrie.telnet.factory.HoneyPotTelnetFactory#info] Starting factory <cowrie.telnet.factory.H
[-] Ready to accept Telnet connections
/Desktop/cowrie/var/log/cowrie# tail cowrie.log
[CowrieTelnetTransport,4,192.168.88.129] login attempt [b'user'/b'11111111'] failed
[cowrie.telnet.userauth.HoneyPotTelnetAuthProtocol#warn] state changed and new state returne
[CowrieTelnetTransport,4,192.168.88.129] Connection lost after 0 seconds
[cowrie.telnet.factory.HoneyPotTelnetFactory] New connection: 192.168.88.129:34369 (192.168.
[CowrieTelnetTransport,5,192.168.88.129] Could not read etc/userdb.txt, default database act
[CowrieTelnetTransport,5,192.168.88.129] login attempt [b'root'/b'12345678'] succeeded
[CowrieTelnetTransport,5,192.168.88.129] Initialized emulated server as architecture: linux-
[cowrie.telnet.userauth.HoneyPotTelnetAuthProtocol#warn] state changed and new state returne
[CowrieTelnetTransport,5,192.168.88.129] CMD: echo veCz8WYWKsB6doL1GetVnOm
[CowrieTelnetTransport,5,192.168.88.129] Command found: echo veCz8WYWKsB6doL1GetVnOm
/Desktop/cowrie/var/log/cowrie#
```

13. ábra. a feltört "telnet"

A Cowrie naplózta a sikertelen és a sikeres belépési kisérleteket, így a lehető legtöbb információt szolgáltatva pl. egy Blue Team csapatnak.

Tapasztalataim: a „hagyományos” naplóállományok, melyek már a felderítés első pillanatától kezdve naplóznak minden eseményt, és a bináris állományok, melyek a bejutás pillanatától kezdve elkezdik „felvenni” a támadó minden billentyűzet leütését, ezen két forrás feldolgozásával kialakítható egy teljes támadási profil. Vegyük az én támadásomat. A naplóállománybó kiolvasható, hogy milyen felhasználó nevek és jelszavak kombinációival próbálkoztam. Az is látható, az időbélyegeknek köszönhetően, időben milyen gyorsasággal történik. Mikor csak manuálisan próbálkoztam, akár másodpercek is eltelhet a két próbálkozás között. Amikor a brute force technikát alkalmaztam, akkor jóval gyorsabb az időbélyegek keletkezése, hiszen minden belépési kísérlet egy egy új esemény és ezért új időbélyeg is generálódik hozzá. Sikeres bejelentkezés után látható az összes kiadott parancs, ha lefutott, ha nem. Láthaó, hogy a támadó a fájlrendszerben miket néz meg, mit keres, mit tölt le és fel. Ezek nagyon értékes információk a támadó szándékáról, például a fájlműveletekből következtetni lehet, hogy jogosultsági szintet /privilege escalation/ szeretne első körben, esetleg továbbmegy egy fájlfeltöltéssel egy backdoor nyitást valósít meg.

Cowrie honeypot telepítésekor kapunk egy teljes fáljstruktúrát. A támadó ezzel a rendszerrel találja szembe magát, megerősítve, hogy egy produktív környezetbe sikerült bejutnia. A fájlrendszert ki tudjuk bővíteni, átalakítani, ahogy éppen jónak látjuk. Abban az esetben, ha a támadó valamit töröl, feltölt, átnevez vagy valamilyen paranccsal bármit módosít, azt könnyen vissza lehet állítani.

Cowrie honeypot telepítése viszonylag egyszerű volt, a rengeteg dokumentációnak köszönhetően gyorsan sikerült beüzemelni és működésre bírni. A bejövő interfészt természetesen a konfigurációs fájlban be kellett állítani. Az SSH protokoll „beindítása” és az iptables konfigurálása is elsőre sikerült. Hasonlóan, a Telnet protokollt is könnyedén konfiguráltam. Összességében talán egy tucatnyi linux parancsot kellett kiadni az egész honeypot telepítéséhez. Zárojeles megjegyzés: gyakran lefagyott a virtuélis gépem, melyen a Cowrie – t futattam, ennek valószínűleg kapacitáshiány állhatott a háttérben. Próbálkoztam többféle memória kiosztással 4, 8 és 16 Gb RAM – al, sajnos egyik sem segített. Lefagyás nélkül kb. 10 – 15 percet futott átlagban, utána újra kellett indítani az egész gépet. Egy élesen működő rendszerben erre fokozottan figyelni kell.

6.3 Valhala honeypot – Telnet

Eredetileg a KFSensor nevezetű Windows platformon futó honeypotot szerettem volna tesztelni. Sajnos időközben a gyártó honlapján csak lejárt licencű szoftvert lehetett letölteni. Eredetileg 30 napos a használata, de hiába telepítettem fel egy teljesen újonnan installált Windowsra, akkor is a „lejárt a 30 napos próbaidőszak” figyelmeztetést kaptam. Jeleztem a gyártó felé a problémát, sajnos választ nem kaptam. Így egy másik Windows alatt futó honeypot után kezdtem kutatni. Így találtam rá a Valhala nevű programra [33]. Ez egy alacsony interaktivitású Windows alatt futó honeypot, a következő szolgáltatásokat tudja szimulálni: Ftp, Telnet, SMTP, IIS, POP3. Elsőnek a Telnet kapcsolatot vizsgáltam, kezdve a felderítéssel. Az egyszerűség kedvéjért csak az első 100 portot vizsgáltam, ezt a –p kapcsolóval értem el. Az -sV kapcsoló a fingerprinting technológiával probálja meghatározni a futó szolgáltatások verzióját. (14. ábra):

```
┌──(root㉿kali)-[/home/kali]
└─# nmap 192.168.88.131 -p 1-100 -sV
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-26 05:26 EDT
Nmap scan report for 192.168.88.131
Host is up (0.0024s latency).
Not shown: 89 closed tcp ports (reset)
PORT   STATE SERVICE     VERSION
7/tcp  open  echo
9/tcp  open  discard?
13/tcp open  daytime?
17/tcp open  qotd        Windows qotd (English)
19/tcp open  chargen
21/tcp open  ftp
22/tcp open  tcpwrapped
23/tcp open  backdoor    OptixPro backdoor (**BACKDOOR**)
25/tcp open  smtp?
79/tcp open  reverse-ssl SSL/TLS ClientHello
80/tcp open  http        Microsoft IIS httpd 10.0
```

14. ábra. Valhala honeypot felderítése

Vizsgálat előtt a honeypot összes „szolgáltatását” bekapcsoltam, hogy kiderüljön, hogy a felderítés során valóban működik – e a csali szerep. Az eredményből látható a nyitott portok listája, működik a csali!

Természetesen a honeypot is azonnal érzékelte a felderítést. Ez a képernyő kép csak egy részlet, helyhiány miatt csak egy kis szeletét raktam be ide, de az összes aktív protokollra megjelent a riszatás. (15. ábra):

```
IP 192.168.88.129 tried to invade by ftp (connect)
IP 192.168.88.129 tried to invade by smtp (connection)
IP 192.168.88.129 tried to invade by extra port (22)
IP 192.168.88.129 tried to invade by telnet (connection)
IP 192.168.88.129 tried to invade by ftp ( )
IP 192.168.88.129 tried to invade by ftp ( )
IP 192.168.88.129 tried to invade by finger ( @192.168.88.131)
IP 192.168.88.129 tried to invade by telnet ( )
IP 192.168.88.129 tried to invade by finger ( using the request: GET / HTTP/1.0@192.168.88.131)
IP 192.168.88.129 tried to invade by finger ( using the request: HELP@192.168.88.131)
IP 192.168.88.129 tried to invade by finger ( using the request: OPTIONS / HTTP/1.0@192.168.88.131)
IP 192.168.88.129 tried to invade by finger ( using the request: OPTIONS / RTSP/1.0@192.168.88.131)
IP 192.168.88.129 tried to invade by telnet ( )
IP 192.168.88.129 tried to invade by ftp (connect)
IP 192.168.88.129 tried to invade by ftp (disconnect)
```

15. ábra. Valhala honeypot reagálása

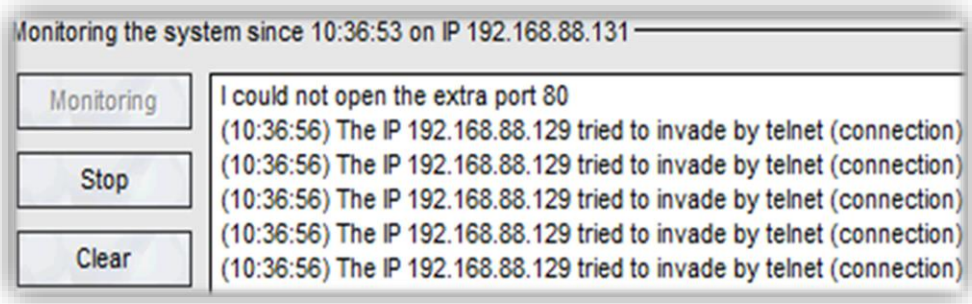
Felderítés után jöhetett a támadás. Itt a Kaliban található Hydra nevezetű szoftvert használtam. Elkészítettem a password szótár fájlt majd elkezdtem a bruteforce támadást. A 16. ábrán látható a sikeres törés, melyet a Hydra visszaadott eredményül:

```
┌──(root㉿kali)-[/home/kali]
└─# hydra -l user -P pwd.txt telnet://192.168.88.131
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in mili
l purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-04-26 04:23:48
[WARNING] telnet is by its nature unreliable to analyze, if possible better choose
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per
[DATA] attacking telnet://192.168.88.131:23/
[23][telnet] host: 192.168.88.131   login: user   password: 12345678
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-26 04:24:22
```

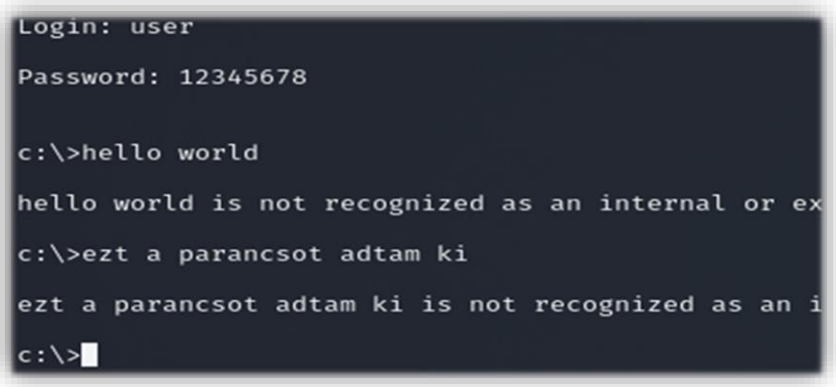
16. ábra. Hydra sikeres törése

A következő képen (17. ábra) a honeypot oldalon megjelenő üzenetek láthatóak. Azaz a támadó IP címe, időpont, a támadott protokoll és a kísérletek száma:



17. ábra. Valhala honeypot real - time naplója

A következő lépés (18. ábra), a megtalált user/password információval belépek Telnettel a „gépre”. Itt szándékosan egy windowsos gépet használtam, hogy lát – e külömbséget a honeypot a Windows és Linux között. Ilyet nem tapasztaltam:



18. ábra. Belépés telnet kapcsolattal

Ellenben, minden parancs, amit kiadtam, azzonal megjelent a honeypot naplójában, beleértve a felhasználónevet és a jelszót is (19. ábra):

19. ábra. A kiadott parancsokat mutatja a Valhala honeypot

A Valhala honeypot is tartalmaz egy fájlrendszert (20. ábra), igaz ez sokkal szegényesebb és szerényebb funkcionalításokkal bír, mint a Cowrie fájlrendszere. Ezzel sok mindent nem lehet kezdeni, nem véletlen, hogy alacsony interaktivitású eszköznek van minősítve. Ezen a ponton, egy gyakorlott támadó jó eséllyel felismeri, hogy egy csali rendszerbe futott bele. Arról a hibáról nem is beszélve, melyett én is tapasztaltam, nevezetesen a parancsokat duplán kell kiadni, mert csak akkor hajtja végre az utasítást. Ez már a bejelentkezésnél is és később pl.: a dir parancs kiadásakkor is feltűnt. Egy rutinosabb hacker erre az anomáliára könnyen felfigyelhet. Nyilvánvalóan egy „bug" – ról van szó, mely le tudja buktatni a honeypot - ot, de erre már utaltam a szakdolgozatom 3.1 – es fejezetébe a „Felderítés" részben.

```
Login: user

Password: 11111111


F:\>dir

Volume in drive F is SISTEMA
Volume Serial Number is F078-2A14

 Directory of F:\

22/03/2001  14:53    <DIR>          .
22/03/2001  14:53    <DIR>          ..
22/03/2001  14:53    <DIR>          arquivos
22/03/2001  14:53    <DIR>          backup
22/03/2001  14:53    <DIR>          clientes
22/03/2001  14:53    <DIR>          documentos
22/03/2001  14:53    <DIR>          funcionarios
22/03/2001  14:53    <DIR>          windows
               0 File(s)           0 Bytes.
               6 Dir(s)   49.962.553.344 Bytes free.
```

20. ábra. Valhala honeypot szimulált fájlrendszere

6.1.1 6.4 Valhala honeypot – FTP

Az előző pontban végzett felderítés eredményét felhasználva kezdtem az FTP kapcsolat támadását. Első lépésben itt is létrehoztam egy password szótárt. Szándékosan beletéve a helyes jelszót. Ismételten a Hydra programot használtam, a 21 - es ábrán látható a szótár fájl létrehozása és a Hydra használata. A hacker program megtalálta a helyes felhasználónév/jelszó párost, mint ahogy látható is az alábbi képen:

```
┌──(root㉿kali)-[/home/kali]
└─# cat >pwd.txt
password
jelszo
pass123
passw0rd123
12345678
11111111
87654321

┌──(root㉿kali)-[/home/kali]
└─# hydra -l user -P pwd.txt ftp://192.168.88.131
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do
l purposes (this is non-binding, these *** ignore laws and ethics

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 202
[DATA] max 7 tasks per 1 server, overall 7 tasks, 7 login tries (l
[DATA] attacking ftp://192.168.88.131:21/
[21][ftp] host: 192.168.88.131   login: user   password: 12345678
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 202
```

21. ábra. Hydra megtalálja a helyes jelszót

Természetesen a támadott oldal is azonnal reagált a scannelésre, és pontosan nyomon lehetett követni a behatolási szándék tényét (22. ábra). Pontosan látszik minden kísérlet. Szépen látható a Hydra program által kipróbált jelszavak, időbélyeg és az támadó ip cime. Ez a honeypot ennél több információt nem tud szolgálni. Sajnos itt is csak egy részét tudtam beilleszteni a képernyő képből, de ez egy hosszabb és részletesebb lista volt:

```
(10:07:33) The IP 192.168.88.129 tried to invade by ftp (PASSWORD pass123)
Authentication failure on FTP
(10:07:33) The IP 192.168.88.129 tried to invade by ftp (PASSWORD 11111111)
Authentication failure on FTP
(10:07:33) The IP 192.168.88.129 tried to invade by ftp (PASSWORD password)
Authentication failure on FTP
(10:07:33) The IP 192.168.88.129 tried to invade by ftp (PASSWORD jelszo)
Authentication failure on FTP
(10:07:33) The IP 192.168.88.129 tried to invade by ftp (PASSWORD passw0rd123)
Authentication failure on FTP
```

22. ábra. Hydra próbálkozásai

Miután „megszereztem” a jelszót, megkíséreltem belépni annak használatával, (23. ábra):

```
┌──(root㉿kali)-[/home/kali]
└─# ftp 192.168.88.131
Connected to 192.168.88.131.
220 Wu-ftpd 1.7.0
Name (192.168.88.131:kali): user
331 Password required for user.
Password:
230 User user logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
227 Entering Passive Mode (192,168,88,131,194,10).
150 Opening data connection for directory list.
drw-rw-rw-   1 ftp      ftp            0 Oct 05  2023 .
drw-rw-rw-   1 ftp      ftp            0 Oct 05  2023 ..
drw-rw-rw-   1 ftp      ftp            0 Apr 26 09:36 log
226 File sent ok
ftp> ls
227 Entering Passive Mode (192,168,88,131,194,11).
150 Opening data connection for directory list.
drw-rw-rw-   1 ftp      ftp            0 Oct 05  2023 .
drw-rw-rw-   1 ftp      ftp            0 Oct 05  2023 ..
drw-rw-rw-   1 ftp      ftp            0 Apr 26 09:36 log
226 File sent ok
ftp> cd ..
```

23. ábra. Beléptem a Valhala honeypotba a Hydra által kiderített jelszóval

A sikeres belépést követően a honeypot tette a dolgát és részletesen, az általam kiadott összes paroncsot megjelenítette az IP cimmel és időponttal együtt (24. ábra):

```
(11:55:32) The IP 192.168.88.129 tried to invade by ftp (connect)
(11:55:45) The IP 192.168.88.129 tried to invade by ftp (USER user)
(11:55:49) The IP 192.168.88.129 tried to invade by ftp (PASSWORD 12345678)
User authenticated on FTP
(11:55:49) The IP 192.168.88.129 tried to invade by ftp (SYST )
(11:55:49) The IP 192.168.88.129 tried to invade by ftp (FEAT )
(11:55:53) The IP 192.168.88.129 tried to invade by ftp (EPSV )
(11:55:53) The IP 192.168.88.129 tried to invade by ftp (PASV )
(11:55:53) The IP 192.168.88.129 tried to invade by ftp (LIST )
(11:56:14) The IP 192.168.88.129 tried to invade by ftp (PASV )
(11:56:14) The IP 192.168.88.129 tried to invade by ftp (LIST )
(11:56:43) The IP 192.168.88.129 tried to invade by ftp (CWD ..)
(11:57:21) The IP 192.168.88.129 tried to invade by ftp (CWD syst)
```

24. ábra. Valhala honeypot munka közben

Tapasztalataim: a Cowrie honeypot – hoz hasonlóan, itt is van lehetősége a támadónak „valódi” fájlrendszertben navigálni, de ez, a Cowrie honeypot - hoz képest korlátozott. Míg a Cowrie – ban lehet mappa és fájl műveleteket végezni, erre itt nincs lehetőség. A Cowrie honeypotban van mód a fájlrendszer személyre szabására, itt erre sincs lehetőség. Ezen a ponton lehet érzékelni, hogy a Valhala honeypot egy alacsony interaktivitású eszköz, míg a Cowrie honeypot közepes/magas interaktivitású szoftvernek van titulálva. Ez valóban egy jól megfogható külömbség az interaktivitások szintjei között.
Egy másik hátránya a Valhala honepot – nak, nincs paylog „szolgáltatása”. Azaz, nem lehet visszajátszani a támadó monitoráról készült „felvételt”, amely nagyon mgkönnyíti a támadó tevékenységének elemzését.

A Valhala honeypotot az interneten megtalálni nehezebb volt, mint beüzemelni. Miuttán rátaláltam, már dokumentációt is sikerült fellelni hozzá. Futtatásához semmiféle extra követelmény nem volt, kellett egy windowsos gép (10, VMWare), 2 Gb RAM - al, majd azon futtatni. Stabilan tette a dolgát, lefagyást, lassulást nem tapasztaltam. Az aktív interfészt megtalálta és azt el is kezdte figyelni, nem kellett konfiguráni sem. Mind a Telnet és az FTP protokollt csak be kellett kattintani a configurációs grafikus felületen, és már működött is a monitoring rendszere. Ez egy igazi „dobozos” termék, ráadásul még korlátozás sincs rajta. Aki kezdő szinten szeretne ismerkedi egy kicsit a honeypotok - kal és a Windowsos felület szimpatikusab számára mint a linuxos környezet, annak tökéletes válsztás lehet.

6.5 Snort – FTP

A Snort teszteléséhez egy windowsos gépet használtam, melyen aktiváltam az

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# nmap -sC -sV 192.168.88.131
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-28 05:45 EDT
Nmap scan report for 192.168.88.131
Host is up (0.0011s latency).
Not shown: 990 closed tcp ports (reset)
PORT     STATE SERVICE       VERSION
7/tcp    open  echo
9/tcp    open  discard?
13/tcp   open  daytime?
| fingerprint-strings:
|   GenericLines, NULL:
|_     11:45:31 2024. 04. 28.
17/tcp   open  qotd          Windows qotd (English)
19/tcp   open  chargen
21/tcp   open  ftp           Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_09-17-23  10:13AM                    0 index.html
| ftp-syst:
|_  SYST: Windows_NT
```

25. ábra. FTP szerver felderítése

FTP szolgáltatást, 25. ábra. A támadást az FTP ellen végzem úgy, hogy a Snort közbeékelődve tudjon reagálni. A felderítés eredményéből látható a nyitott ftp port és a windows környezet. A Snort IDS üzemmódban van beállítva.
A felderítést a Suricata is érzékelte. Látható a forgalom “osztályozása”, amely jelenleg potenciális veszélyforrásnak van minősítve.

Kiolvasható a kommunikációban szereplő forrás és a cél IP cím, valamint az érintett protokollok. Ez egy erőssen megvágot képernyőkép (26. ábra):

```
Unknown Traffic] [Priority: 3] {TCP} 192.168.88.129:57236 -> 192.168.88.131:8
[**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.88.

[**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.88.

[**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.88.

: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.88.129:48744 -> 192.168
lly Bad Traffic] [Priority: 2] {TCP} 192.168.88.129:48786 -> 192.168.88.131:21
lly Bad Traffic] [Priority: 2] {TCP} 192.168.88.129:48786 -> 192.168.88.131:21
[Classification: Generic Protocol Command Decode] [Priority: 3] {TCP} 192.168.
```

26. ábra. Suricata reagálása a felderítésre

Egy portscann-t még jó eséllyel riasztja a Snort - ot, mivel egy alap rules állomány be van töltve, de mindenképpen finomhangolni kell a rendszert a saját igényeinknek megfelelően.
Pl. egy konkrét FTP bejelentkezési kísérletre csak egy, már betöltött, erre a célra írt szabály esetében risztott. Ebben az esetben a megfelelő rule - t betöltve a local.rules állományban. Így az általam írt egyszerű Snort szabály volt aktiválva:
alert tcp any any -> any 21 (msg:"FTP USER command detected"; sid:100001;)
A szabály figyelmeztetést generál minden olyan TCP forgalomra, ami a 21-es porton érkezik. Ha a csomag tartalmazza a "USER" szót, akkor kiad egy üzenetet, hogy érzékelt egy FTP USER parancsot, (26. ábra). Az any any azt jelenti, hogy a forrás és a célcímek bármi lehetnek. Az alert az, hogy figyelmeztetést generálunk, és a sid (Signature ID) egy azonosító a szabálynak, ami egyedi azonosítóként szolgál a naplózásban és az eseménykezelésben.

Miután a szabály aktiválása megtörtént, kipróbáltam egy brute foce támadást a már jól ismert Hydra – val, 27. ábra:

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# hydra -l xy -P pwd.txt ftp://192.168.88.131
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or s
oses (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-04-28 06:22:28
[DATA] max 5 tasks per 1 server, overall 5 tasks, 5 login tries (l:1/p:5), ~1 try per task
[DATA] attacking ftp://192.168.88.131:21/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-28 06:22:29
```

27. ábra. Hydra támadása az FTP szerver ellen

Ez most egy sikertelen támadás volt, de a Snort a szabálynak megfelelően viselkedet. Minden, az FTP irányában történő kapcsolat felvételi kísérletre reagált. Itt látható (28. ábra) az FTP kapcsolódási kísérlet, ip címekkel és időbélyeggel együtt. A Snort riasztásai:

```
FTP USER command detected [**] [Priority: 0] {TCP} 192.168.88.129:42776 -> 192.168.88.131:21
FTP USER command detected [**] [Priority: 0] {TCP} 192.168.88.129:42750 -> 192.168.88.131:21
FTP USER command detected [**] [Priority: 0] {TCP} 192.168.88.129:42746 -> 192.168.88.131:21
FTP USER command detected [**] [Priority: 0] {TCP} 192.168.88.129:42748 -> 192.168.88.131:21
FTP USER command detected [**] [Priority: 0] {TCP} 192.168.88.129:42774 -> 192.168.88.131:21
FTP USER command detected [**] [Priority: 0] {TCP} 192.168.88.129:42776 -> 192.168.88.131:21
FTP USER command detected [**] [Priority: 0] {TCP} 192.168.88.129:42746 -> 192.168.88.131:21
```

28. ábra. Snort risztása az FTP csatlakozási kísérletre

Majd megpróbáltam manuálisan csatlakozni az FTP kiszolgálóhoz (29. ábra):

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# ftp 192.168.88.131
Connected to 192.168.88.131.
220 Microsoft FTP Service
Name (192.168.88.131:kali): xy
331 Password required
Password:
530 User cannot log in.
ftp: Login failed
ftp> 12345678
?Invalid command.
ftp> dir
530 Please login with USER and PASS.
530 Please login with USER and PASS.
```

29. ábra. FTP belépési kísérlet Kali linux alól

A Snort risztása a „túloldalon”, Szintén látható az időbélyeg, IP címek és az incidensben érintett protokoll, és maga a tevékenység, jelen esetben egy „USER command” detektálása, melynek a besorolás szerinti meghatározása szintén egy potenciális veszély lehet, szintén egy erőssen megvágott kép, (30. ábra):

```
USER command detected [**] [Priority: 0] {TCP} 192.168.88.129:56314 -> 192.168.88.131:21
utive TCP small segments exceeding threshold [**] [Classification: Potentially Bad Traff
USER command detected [**] [Priority: 0] {TCP} 192.168.88.129:50444 -> 192.168.88.131:21
utive TCP small segments exceeding threshold [**] [Classification: Potentially Bad Traff
USER command detected [**] [Priority: 0] {TCP} 192.168.88.129:54320 -> 192.168.88.131:21
utive TCP small segments exceeding threshold [**] [Classification: Potentially Bad Traff
```

30. ábra. Jól látható a szabálynak megfelelő risztások sorozata

6.1.2 6.6 Suricata – SSH

A Suricata behatolásfigyelő rendszert egy Ubuntu VMware környezetben telepítettem és teszteltem. A képen (31. ábra) a beüzemelt Suricata látható:

```
suricata.service - LSB: Next Generation IDS/IPS
     Loaded: loaded (/etc/init.d/suricata; generated)
     Active: active (running) since Fri 2024-04-26 15:08:00 CEST; 4min 57s
       Docs: man:systemd-sysv-generator(8)
      Tasks: 8 (limit: 4554)
     Memory: 428.1M
        CPU: 38.194s
     CGroup: /system.slice/suricata.service
             └─1033 /usr/bin/suricata -c /etc/suricata/suricata.yaml --pidf

ápr 26 15:08:00 SSH-Ubuntu systemd[1]: Starting LSB: Next Generation IDS/IP
ápr 26 15:08:00 SSH-Ubuntu suricata[899]: Starting suricata in IDS (af-pack
ápr 26 15:08:00 SSH-Ubuntu systemd[1]: Started LSB: Next Generation IDS/IPS.
```

31. ábra. az aktív Suricata

Itt az SSH – t teszteltem, ezért egy SSH servert telepítettem az Ubuntu operációs rendszerre. A Suricata – t ráállítottam a megfelelő interfészre, majd betöltöttem egy ssh – ra vonatkozó szabályt a suricata.yaml fájlba.
Az általam írt és használt rule: alert tcp any any -> any 22 (msg:"SSH Connection Established"; sid:1000001;)
Ez a szabály azt jelzi, ha bármely IP-címről bármely portról SSH kapcsolat létesül valamely másik IP-címre a 22-es porton. A szabály meghatározza a figyelmeztetés üzenetét valamint az egyedi azonosítót (sid).

A felderítés is a szokásos Nmap programmal történt, az eredményből szépen látszik a nyitott SSH port, (32. ábra):

```
┌──(kali㉿kali)-[~]
└─$ nmap -sC -sV 192.168.88.133
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-26 10:24 EDT
Nmap scan report for 192.168.88.133
Host is up (0.0010s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.7 (Ubuntu Linux
| ssh-hostkey:
|   256 fa:a9:7d:3a:2e:54:b7:4e:73:a4:77:40:4b:0a:95:28 (ECDSA)
|_  256 07:6d:55:30:47:0f:4d:df:2b:ad:ae:9f:42:ca:b5:09 (ED25519)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results a
Nmap done: 1 IP address (1 host up) scanned in 13.69 seconds
```

32. ábra. az SSH szerver felderítése

A portscann is megjelent a Suricata rendszerében, (33. kép):

```
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP}
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP}
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP}
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP}
```

33. ábra. Suricata jelzése a portscann megjelenésére

Itt persze még nics kapcsolat felépülve, ez csak a szabályban lévő szöveg miatt jelent így meg. Látható, hogy a kapcsolódási kíséerletnek látszó letapogatás prioritása 3 – as értéket kapot. A prioritás kulcsszóhoz tartozik egy kötelező numerikus érték, amely 1 és 255 között lehet. Az 1 és 4 közötti értékeket használják általában. A legmagasabb prioritás az 1. A magasabb prioritású aláírások kerülnek először vizsgálatra. Általában az aláírások prioritása osztálytípus definíción keresztül van meghatározva. Az osztálytípus definíció felülírható a prioritás kulcsszó megadásával az aláírásban.

Megpróbáltam egy windowsos gépről az ssh bejelentkezést (34. ábra):

```
icrosoft Windows [Version 10.0.19045.4291]
(c) Microsoft Corporation. Minden jog fenntartva.

:\Users\mrrob>ssh user@192.168.88.133
he authenticity of host '192.168.88.133 (192.168.88.133)' can't be established.
CDSA key fingerprint is SHA256:zx+pnGSMW0iheKoxihbJe4LBSdZtTM2OpeYzlSAzaTA.
re you sure you want to continue connecting (yes/no/[fingerprint])? y
lease type 'yes', 'no' or the fingerprint: y
```

34. ábra. SSH bejelentkezés windows alól

És egyértelműen riasztott, ip címek, protokollok és időbélyeggel (35. ábra):

```
04/26/2024-15:46:47.529472  [**] [1:2022973:1] ET POLICY Possible Kali Lin
Classification: Potential Corporate Privacy Violation] [Priority: 1] {UDP}
04/26/2024-16:07:13.966699  [**] [1:1000001:0] SSH Connection Established
3] {TCP} 192.168.88.1:59331 -> 192.168.88.133:22
```

35. ábra. Windows operációs rendszerből indított belépés risztása

És ha már itt „jártam”, akkor a Kali alól is, (36. ábra):

```
┌──(kali㉿kali)-[~]
└─$ ssh 192.168.88.133
kali@192.168.88.133's password:
Permission denied, please try again.
kali@192.168.88.133's password:
Permission denied, please try again.
kali@192.168.88.133's password:
```

36. ábra. SSH belépési kísérlet Kali alól

Persze így is riasztott (37. ábra), szintén ip címek, időbélyeg és az érintett szabály által megadott szöveg megjelenése a naplóállományban:

```
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP} 19
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP} 19
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP} 19
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP} 19
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP} 19
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP} 19
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP} 19
SSH Connection Established [**] [Classification: (null)] [Priority: 3] {TCP} 19
```

37. ábra. Suricata riasztása Kali alól történő belépési kísérletre

Tapasztalataim: a behatolás érzékelő rendszerek főleg a hálózati forgalom ellenörzésére van kihegyezve. Milyen csomagok, kérések és ezek iránya adja vizsgálat tartományát. Itt a fő cél a behatolás érzékelése, megelőzése és annak megállítása.
Az IDS/IPS rendszer beüzemelése és finomhangolása a saját ígényeinkre szabva jóval több szakértelmet és odafigyelét követelt meg részemről, mint a honeypot – ok esetében. Kezdve az interfészek konfigurálásától a naplóállományok beállításán keresztül a szabályok kiválasztás/írás/aktíválásáig. Sokkal több mindenre kellett figyelnem, ahhoz hogy egyáltalán elinduljon és érzékeljen valamilyen forgalmat. Még a .log könyvtárakhoz is külön jogosultságokat kellett kiosztanom a linuxon és a kofigurációs állományokkal is több munka volt, mint a honeypotok esetében.

7 ÖSSZEHASONLÍTÁS

Az előző fejezetben ismertettem az általam elkészített gyakorlati teszteket. Most ezeket fogom – vizsgált protokollok alapján, IDS és Honeypotok szembeállításával – összehasonlítani a megfigyeléseim alapján.

7.1 Honeypot: Cowrie, IDS: Suricata, Protokoll: SSH

Kezdjük a hasonlóságokkal. Amit tapasztaltam, hogy a portscann – t mindegyik rendszer érzékelte. Mind a két rendszer naplózta a támadó IP címét, a protokollt és az időbélyeget. Itt persze máris van egy különbség, mégpedig, az hogy az IDS esetén milyen szabályok futnak. Míg a honeypot esetén nem kellett külön létrehoznom semmilyen szabályt – lévén a Cowrie egy SSH és FTP célszoftver – az IDS esetén létre kellett hoznom egy szabályt, amely figyeli az SSH kapcsolatfelvételi kísérleteket. Persze az más kérdés, hogy a Suricata – ban van egy kiinduló szabálykészlet a szabályokból és azt engedélyezve máris érzékeli az ilyen jellegű kapcsolatokat. De alapvetően nekünk kell finomhangolni az igényeink és a hálózati adottságoknak megfelelően. Igazából a hasonlóságok sora itt véget is ér. Mert onnantól kezdve, hogy megtörtént a kapcsolatfelvétel, jól kirajzolódott a két rendszer különbsége. Méghozzá az, hogy melyik eszköz mit is látott. Míg a Suricata a támadó által kiadott parancsok hatását/eredményét/következményét tudta csak detektálni, addig a honeypot a kiadott parancsokat is. Konkrétan azt is naplózza, hogy milyen parancsok lettek kiadva, függetlenül attól, hogy azoknak aztán milyen hatással volt a rendszerre, egyáltalán le futott vagy megakadt valahol. És ez komoly külömbség. Gondoljunk csak bele, a Honeypot – oknak aza lényegük, hogy kiismerjük a támadó technikáját, eszköztárát. A kiadott parancsokból sok minden leszürhető és sok következtetés vonható le. Pl.: milyen parancsokat adott ki és azok közül melyek vezettek eredményre ill. segítették a támadót közeleb jutni a céljához.

Most pedig nézzünk meg egy alacsony interaktivitású honeypot – ot és egy másik IDS rendszert. Ebben az esetben az FTP protokollt használtam.

7.2 Honeypot: Valhala, IDS: Snort, Protokoll: FTP

A hasonlóságok itt is alapvetően kimerültek a portscann művelet befejezéséig. Mind a honeypot, mind az IDS rendszer érzékelt és riasztott ebben a fázisban. Lehetett látni az IP címeket, az időbélyeget és az érintett protoklot. Természetesen itt is igaz volt a szabályok elkészítésére vonatkozó tennivalók és azok „bekapcsolása” az IDS esetén. A honeypot – nál ezzel szintén nem kellett foglakozni, autómatikusan beindult. Ebben az esetben is, mint az előzőben, a honeypot megjelenítette a kiadott parancsokat is, szemben az IDS - el, ahol szintén csak a kiadott parancsok hatását tudta detektálni és jelezni. Az előző és a mostni tesztben is megmutatkozott a két honeypot közötti különbség

(Cowrie, Valhala). Itt az interaktivítási szintre kell gondolni. Cowrie honeypot – hoz hasonlóan, itt is van lehetősége a támadónak „valódi” fájlrendszerben navigálni, de ez, a Cowrie honeypot - hoz képest korlátozott. Míg a Cowrie – ban lehet mappa és fájl műveleteket végezni, erre itt nincs lehetőség. A Cowrie honeypot - ban van mód a fájlrendszer személyre szabására, itt ez nem lehetséges.

8 JÖVŐBELI FEJLESZTÉSEK IRÁNYA

Honeypot esetében, úgy látom, hatalmas potenciál van a lehetőségek tekintetében. Ahogy terjednek az új technológiák, azzal együtt jelennek meg az újabb és egyre kreatívabb csalási/támadási formák is. Az új technológia alatt leginkább az mesterséges intelligenciát és a blokkláncot értem. Az AI berobbanása nagyon látványos volt, nem érdemtelenül, ám közben a blockchain technológia is szép csendben egyre több helyen megjelenik. Én, mint köztiztviselő, nekem is volt alkalmam részt venni olyan tanfolyamon, mely a blokklánc technológia felhasználási lehetőségeit ismertette a közszférában. Sok olyan projektet hallottam, mint pl. a svédországi, ahol a földhivatali nyilvántartást ültették át erre, az új technikára. A Magyar Nemzeti Bank is komoly munkát fektett a blokklánc tanulmányozásába és hasznosításába. Megjelent az NFT (nem helyettesíthető tokenek) eszköz, melyek szintén a blokkláncra épűlnek. Sok más felhasználási területük is van, ilyen pl. okosszerződések, egy választás lebonyolítása, ellátási lánc felügyelete. Sok más bank is felismerte, hogy a struccpolitika, ezen a téren nem kifizetődő. Gyakorlatilag már minden ország foglalkozik ezzel a területtel. Jó pár egyetemen továbbképző kurzusok is indulnak ebben a témakörben. Ínnentől kezdve, a kíberbiztonsági szakembereknek fel kell készülni ezen infrastrukúra elleni támadásrokra. Meglátásom szerin, érdemes lenne, blokláncra kihegyezett honeypot - okat létrehozni, kombinálva IDS eszközökkel. Ezeket megtámogatni, már csak a nagy mennyiségű adatt miatt is, a mesterséges intelligencia valamely formájával. Bizonyára már lehetnek ilyen eszközök, de egyre fog nőni az igény, az ilyen jellegű csali rendszerekre és az ilyen irányú tudással rendelkező IT biztonsági mérnökökre.

9 ÖSSZEFOGLALÁS

Az IDS/IPS rendszerek és a honeypot – ok közötti legnagyobb különbség: míg az előbbi csoport tagjai igyekeznek távol tartani a rosszindulatú tevékenységet a hálózattól/végponttól, addig a honeypotok pont az ellenkezőjére törekszenek, „behívják”, be engedik a támadót. Míg a behatolás érzékelő rendszerek, detektálva az anomáliát, jelzik és – ha IPS üzemmód aktív – ellenlépéseket tesznek ennek elhárítására, addig a csali rendszerek „bíztatják” a támadót, hogy kibontakoztassa képességeit és megmutassa tudását. Ebből látható, a két rendszer közvetlen céljai mások. Az egyik, gátolja, akadályozza a támadást, a másik engedi, élősegíti a behatolást. A közvetett célok viszont ugyanazok, a hálózat/végpont védelem.

Az általam tesztelt honeypot - ok erőssége abban állt, hogy könnyen és gyorsan be lehetett üzemelni. Probléma nélkül elindultak és azonnal tették a dolgukat. Ezekre az eszközökre, mint célszerszámok érdemes tekinteni. Amely feladatra fejlesztették őket, arra tökéletesen megfelelnek. Nyilván, a teljesítmény függvényében kell hozzájuk megfelelő erőforrást allokálni – mint ahogy már jeleztem is a Cowrie esetében – de ha ezt megkapják, akkor – az én szerény házilaborom környezetében is – egy nagyon megbízható biztonsági eszközt tudhatunk a hálózatunkban.

A teszteléseim során használt IDS/IPS eszközök beüzemelése koránt sem volt olyan egyszerű, mint a honeypot - ok esetében. Itt mélyebre kellett ásnom a dokumentációkban. A különböző verziószámok és frissítésekre is figyelni kellett. A napló állományok sem voltak egyértelműek, a configurációs állományokban több helyen is módosítani kellett. Azán ki kellett ismernem magam a szabályok kezelésében és a saját szabályok írását/felhasználhatóságát át kellett látnom. Szóval egy sor olyan dologgal kellett foglalkozni, amelyek nem jöttek elő a honeypot - oknál. Cserében viszont egy nagyon rugalmas és sokoldalú, mondhatnám univerzális eszközt kapunk.

Egy komlex hálózatvédelemben erre a két eszközre nem egymás kiváltására vagy helyettesítésére, egymást kiegészítésére érdemes gondolni.

10 SUMMARY

The main difference between IDS/IPS systems and honeypots lies in their purpose: while the former aim to keep malicious activities away from the network or endpoints, honeypots do the exact opposite by "inviting" and allowing attackers in. Intrusion detection systems (IDS), when they detect anomalies, alert the administrator and if operating in IPS mode take countermeasures to block the threat. In contrast, honeypots encourage attackers to exploit their skills and reveal their methods. This highlights that the direct objectives of the two systems differ: one aims to block and hinder attacks, while the other facilitates and attracts intrusions. However, their indirect goal is the same: to protect the network or endpoint.

The honeypots I tested proved to be strong tools due to their ease and speed of deployment. They started without issues and immediately fulfilled their function. These tools should be viewed as specialized instruments, perfectly suited for the specific tasks they were designed for. Of course, sufficient resources must be allocated to them depending on performance requirements, as I noted in the case of Cowrie, but if properly equipped, even in my modest home lab environment, they proved to be highly reliable security tools for the network.

In contrast, deploying IDS/IPS tools during my testing was far less straightforward than working with honeypots. I had to dig deeper into the documentation, keep track of various versions and updates, and carefully analyze log files, which were often unclear. Multiple configuration files required adjustments, and I also had to learn how to manage rules effectively and understand the creation and application of custom rules. These tasks were unnecessary for the honeypots. However, in return, IDS/IPS systems offer a highly flexible, versatile, and even universal tool.

In a complex network defense strategy, these two tools should not be considered replacements or substitutes for one another but rather as complementary components.

11 KÖSZÖNETNYÍLVÁNÍTÁS

Köszönetet mondok konzulensemnek, Balázsné Dr. Kail Eszternek, a dolgozattal kapcsolatos szakmai iránymutatásainak, rengeted segítsége és végtelen türelme nélkül nem sikerült volna eljutnom idáig.

Továbbá szeretnék köszönetet mondani Vörösné Dr. Bánáti-Baumann Annának és Rigó Ernőnek. Sokat segítettek szakmai észrevételeikkel és segítő kérdéseikkel, mellyel előre lendítették a téma kibontását.

12 ÁBRAJEGYZÉK

1. ábra, valós idejű gépi tanulás teljes folyamatábrája 21

2. ábra, Sort/Suricata - val megvalósított topológia..... 26

3. ábra, Cowrie honeypot - tal megvalósított topológia..... 26

4. ábra, Valhala honeypot - tal megvalósított topológia 27

5. ábra. Cowrie honeypot SSH kapcsolat felderítése..... 29

6. ábra. Cowri honeypot érzékeli a portscann -t 29

7. ábra. belépési kísérlet a Cowrie honeypot-ba a támadó gépről 30

8. ábra. Cowrie honeypot reagálása a belépési kísérletre 30

9. ábra. Navigálás a Cowrie honeypot fájlrendszerében, windows alól bejelentkezve .. 30

10. ábra. A kiadott parancsok 31

11. ábra. Playlog.py “működés” közben, a kedvenc eszközöm..... 31

12. ábra. Brute force támadás metasploit framework programmal..... 32

13. ábra. a feltört “telnet”..... 32

14. ábra. Valhala honeypot felderítése..... 34

15. ábra. Valhala honeypot reagálása 35

16. ábra. Hydra sikeres törése..... 35

17. ábra. Valhala honeypot real - time naplója 36

18. ábra. Belépés telnet kapcsolattal..... 36

19. ábra. A kiadott parancsokat mutatja a Valhala honeypot 36

20. ábra. Valhala honeypot szimulált fájlrendszere..... 37

21. ábra. Hydra megtalálja a helyes jelszót 38

22. ábra. Hydra próbálkozásai 38

23. ábra. Beléptem a Valhala honeypotba a Hydra által kiderített jelszóval 39

24. ábra. Valhala honeypot munka közben..... 39

25. ábra. FTP szerver felderítése 40

26. ábra. Suricata reagálása a felderítésre..... 41

27. ábra. Hydra támadása az FTP szerver ellen 41
28. ábra. Snort risztása az FTP csatlakozási kísérletre 42
29. ábra. FTP belépési kísérlet Kali linux alól 42
30. ábra. Jól látható a szabálynak megfelelő risztások sorozata 42
31. ábra. az aktív Suricata 43
32. ábra. az SSH szerver felderítése 43
33. ábra. Suricata jelzése a portscann megjelenésére 44
34. ábra. SSH bejelentkezés windows alól 44
35. ábra. Windows operációs rendszerből indított belépés risztása 44
36. ábra. SSH belépési kísérlet Kali alól 45
37. ábra. Suricata riasztása Kali alól történő belépési kísérletre 45

13 FELHASZNÁLT SZAKIRODALOM

[1] Tejvir Kaur, Vimmi Malhotra, Dr. Dheerendra Singh: Comparison of network security tools-Firewall, Intrusion Detection System and Honeypot, International Journal of Enhanced Research in Science Technology & Engineering, Vol. 3 Issue 2, February-2014, pp: (200-204)

[2] Prof. Raj Jain, Eric Peter, Todd Schiller: A Practical Guide to Honeypots, www.cse.wustl.edu/~jain/cse571-09/ftp/honey/index.html

[3] Sreenivas SremathTirumala, Hira Sathu, Abdolhossein Sarrafzadeh, Free and open source intrusion detection systems: A study. (IEEE) 2015 International Conference on Machine Learning and Cybernetics (ICMLC)

[4] K. Thongkanchorn, S. Ngamsuriyaroj and V. Visoottiviseth: "Evaluation studies of three intrusion detection systems under various attacks and rule sets", 2013 IEEE International Conference of IEEE Region 10 (TENCON 2013)

[5] Zhimin Zhou, Zhongwen Chen, Tiecheng Zhou, Xiaohui Guan: The study on network intrusion detection system of Snort, 2010 International Conference on Networking and Digital Society

[6] H. Güneş Kayacık, A. Nur Zincir-Heywood: Mimicry Attacks Demystified: What Can Attackers Do To Evade Detection? , Sixth Annual Conference on Privacy, Security and Trust, 2008

[7] kaspersky.com/resource-center/threats/what-is-a-honeypot
(utolsó megtekintés: 2023.05.22)

[8] Mahmoud T. Qassrawi, Hongli Zhang: Client Honeypots: Approaches and Challenges, 4th International Conference on New Trends in Information Science and Service Science, IEEE 2010

[9] Akshat Divya, Anchit Bhushan, Nihal Anand, Rishabh Khemka, Sumithra Devi K.: HONEYPOT: Intrusion Detection System, International Journal of Education, Science, Technology and Engineering, vol. 3, no. 1, pp. 13-18, June 2020

[10] Lukas Rist: Conpot Honeypot,

github.com/mushorg/conpot, utolsó megtekintés: 2023.05.22.

[11] Michel Oosterhof: Cowrie SSH/Telnet Honeypot,

github.com/cowrie/cowrie, utolsó megtekintés: 2023.05.22.

[12] Johnny Vestergaard: Heralding Honeypot,

github.com/johnnykv/heralding, utolsó megtekintés: 2022.05.22.

[13] Lukas Rist, Johnny Vestergaard, Daniel Haslinger, Andrea Pasquale: Snare Honeypot,

github.com/mushorg/snare, utolsó megtekintés: 2022.05.22.

[14] Göcs László, Johanyák Zsolt Csaba, Kovács Szilveszter: Trap on the network, Gradus Vol 3, No 2 (2016) 55-60

[15] Deutsche Telekom Security GmbH,

https://github.com/telekom-security, utolsó megtekintés: 2022.05.22.

[16] Kazuki Hara, Teppei Sato, Mitsuyoshi Imamura, Kazumasa Omote: Profiling of Malicious Users Using Simple Honeypots on the Ethereum Blockchain Network, 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)

[17] A Honeypot with Machine Learning based Detection Framework for defending IoT based Botnet DDoS Attacks

[18] Nemzeti Kibervédelmi Intézet, CTI Jelentés IoT eszközök biztonsági kérdései - Az ipar, https://nki.gov.hu/it-biztonsag/elemzesek/iot-eszkozok-biztonsagi-kerdesei-az-ipar/

[19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, (1997).

[20] Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., & Elovici, Y. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. IEEE Pervasive Computing, 17(3), 12-22. (2018).

[21] Nawrocki, M., Wählisch, M., Schmidt, T. C., Keil, C., & Schönfelder, J., A survey on honeypot software and data analysis. arXiv preprint arXiv:1608.06249. (2016).

[22] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. IoTpot: A novel honeypot for revealing current iot threats. Journal of Information Processing, 24(3):522–533, 2016.

[23] Phype. Telnet IoT honeypot. https://github.com/Phype/telnet-iot-honeypot, utolsó megtekintés: 2023.05.22

[24] Omererdem. Honeything. https://github.com/omererdem/honeything, utolsó megtekintés: 2023.05.22

[25] DinoTools. dionaea - catches bugs, github.com/DinoTools/dionaea/blob/master/README.md.

[26] S. Dowling, M. Schukat, and H. Melvin. A zigbee honeypot to assess iot cyberattack behaviour. In 2017 28th Irish Signals and Systems Conference (ISSC), pages 1–6, June 2017.

[27] Roy T. Fielding and Richard N. Taylor. Architectural styles and the design of network-based software architectures. University of California, Irvine Doctoral dissertation, 2000.

[28] Wang, Meng, Javier Santillan, and Fernando Kuipers. "ThingPot: an interactive Internet-of-Things honeypot." arXiv preprint arXiv:1807.04114 (2018).

[29] Doshi, R., Apthorpe, N., & Feamster, N. Machine Learning DDoS Detection for Consumer Internet of Things Devices, arXiv preprint arXiv:1804.04159 (2018).

[30] Singh, K., Guntuku, S. C., Thakur, A., & Hota, C. (2014). Big data analytics framework for peer-to-peer botnet detection using random forests. Information Sciences, 278, 488-497.

[31] Gor Saribekyan: Brute Force – Metasploit,

https://www.youtube.com/watch?v=ihQ6Ih-HOSo, utolsó megtekintés: 2024.04.28.

[32] Dr. Zag: Snort Windows 10 Run alerts,

https://www.youtube.com/watch?v=ZEdfnXt-ptM, utolsó megtekintés: 2024.04.21.

[33] Marcos Flavio: Valhala Honeypot,

https://sourceforge.net/projects/valhalahoneypot/ utolsó megtekintés: 2024.05.01