Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Dynamická analýza malware s cílem získávání indikátorů kompromitace a jejich následném využití

Diplomová práce

Bc. Martin Kunc

Školitel: Ing. Petr Břehovský

České Budějovice 2019

Kunc, M., 2019: Dynamická analýza malware s cílem získávání indikátorů kompromitace a jejich následném využití.

[Dynamic malware analysis with aim for gathering indicators of compromise and their utilization. Mgr. Thesis, in Czech.] – 45 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic

Anotace

Diplomová práce se zabývá převážně sběrem síťových indikátorů kompromitace získaných dynamickou analýzou malware v reálném prostředí. Především rozebírá možnosti, jak takový sběr provádět. A následně z nich vybere nejvhodnější řešení. Získané indikátory kompromitace jsou poté analyzovány a využity pro zlepšení kybernetické bezpečnosti v prostředí České republiky.

Abstract

This master thesis focuses on collecting network indicators of compromise gathered by using dynamic malware analysis in real environment. It speculates on possibilities on how to approach such collection and the most suitable solution is selected. Gathered indicators of compromise are thoroughly analyzed and utilized for improving cyber-security of Czech Republic.

Poděkování

Předem bych chtěl poděkovat Petru Břehovskému, mé rodině a mé přítelkyni za trpělivost. Pavlu Baštovi za cenné rady a na závěr bych rád poděkoval kakaovému nápoji Granko.

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

České Budějovice, 17. 04. 2019

Martin Kunc

Obsah

1 Úvod

Jednou ze stěžejních disciplín kybernetické ochrany je odhalování napadených strojů. Mezi rozšířené standardy patří řešení v podobě různých antivirových produktů. S rostoucí rozmanitostí zařízení, připojených do internetu ovšem klesá podpora antivirových řešení. Představa, že by chytrá lednice nebo webkamera měla svůj vlastní antivirus je v dnešní době sice poněkud úsměvná, nicméně nedá se říci, že by to bylo úplně zbytečné. IoT zařízení jsou v roce 2018 běžným cílem kybernetických útoků.

Antivirové společnosti rozhodně neztrácejí krok s dobou a již rozpoznávají i malware pro jiné zařízení než s OS Windows a bývají dostupné i pro mobilní telefony. Nicméně pokrýt všechna zařízení je nemožné, především z důvodu nedostatečného výkonu a kapacity některých zařízení.

Jedním z řešení nastalé situace je hlídání sítě na jejím perimetru. Samo o sobě se ještě nejedná o inovaci. Firewall na vstupu sítě se počítá mezi nejzákladnější dobré praktiky, nicméně inovativní je přístup sledování odchozích spojení a jejich porovnání s nežádoucími IP adresami.

Získat seznam nežádoucích IP adres lze několika způsoby. Prvním a nejsnazším způsobem je najít otevřené zdroje na internetu. Avšak pro takové zdroje je zapotřebí nejprve otestovat jejich kvalitu, aby nedošlo k zablokování legitimních služeb. Tyto zdroje obsahují velké množství IP adres, ale často obsahují také velké množství false-positive. Jejich další podstatná nevýhoda spočívá v tom, že sdružují IP adresy, které byly zaznamenány při útoku či scanu. Zpravidla tedy neobsahují IP adresy řídících serverů botnetu, kam by se nakažený stroj připojoval.

Druhým způsobem je statická analýza, která je ale časově náročná a nelze ji tedy aplikovat na širší množství vzorků. Jinými slovy, tento způsob je příliš finančně náročný a jeho návratnost mizivá.

Třetím způsobem je spuštění malware ve virtualizovaném prostředí pouze emulujícím internet. Na tento způsob už si ale autoři malware začali dávat pozor a snaží se proti němu bránit, a to jak detekováním běhu v emulovaném prostředí, tak detekováním emulace internetu. To se projevilo například u ransomware *WannaCry* (rok 2017, který při vlastním šíření testoval,

zda se mu podaří přeložit doménu „iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com“. Zatímco nástroje emulující internet odpovídají na všechny DNS dotazy kladně a s výsledkem, stáva se, že ve skutečnosti nikdo nemá doménu zaregistrovanou. Malware tedy fungoval pouze pokud byl připojen k reálnému internetu. Tedy do doby, než si toho někdo všiml a doménu si zaregistroval.

Čtvrtým způsobem, který se nabízí, je spouštět malware na fyzickém stroji připojeném do internetu a po každém běhu použití ho uvést do původního stavu. Navrácení do původního stavu ovšem v případě klasických PC může trvat velmi dlouho a nelze vyloučit infikování jiných součástí jako je například BIOS.

1.1 Motivace

Díky dostatečně vysokému výkonu dnešního hardwaru je možné detekovat infikované počítače i bez přítomnosti antiviru. Příkladem může být velké množství odchozích TCP spojení na různé IP adresy zaznamenané síťovou bránou na portu 25 od tiskárny, aniž by tiskárnu někdo využíval k tisku. V takovém případě je více než pravděpodobné, že je dané síťové zařízení infikované a zneužívané k rozesílání SPAMu. Před pár lety by mohlo být podezřelé už jen to, že se tiskárna vůbec k internetu připojuje, což dnes může být důsledek automatických aktualizací. Rozlišit oprávněné a neoprávněné příchozí či odchozí spojení se tak stává kritickou součástí náplní práce síťových administrátorů.

Pracovníci, pověření kybernetickou obranou svěřených počítačů, mezi sebou sdílí takzvané indikátory kompromitace (viz. 2.5 Indikátory kompromitace) za účelem usnadnění detekce a následného odstranění nákazy. Síťové indikátory kompromitace pak tvoří podkategorie, které popisují IP adresy a porty, kam se malware připojuje, či specifický způsob komunikace. Dynamická analýza malware pak umožňuje přesně a relativně snadno získat seznam síťových indikátorů kompromitace. Například porovnáním seznamu IP adres, na které se malware připojoval, a logů, obsahujících IP adresy (na které se připojovaly stroje ve svěřené síti), dokážeme získat seznam nakažených strojů. V případě zavedení blokovacích pravidel na firewallu tak můžeme dokonce zamezit jakékoliv komunikaci malware s řídícím serverem.

Získání síťových indikátorů kompromitace tak hraje zásadní roli v celém popsaném procesu.

Kvalitní a ověřené indikátory kompromitace, se těžko shání, případně jsou zpoplatněné. Proto vznikl nápad připravit takové prostředí, které by s minimálním rizikem umožňovalo vytvářet vlastní seznamy IP adres řídících serverů botnetu. Díky spouštění aktuálních vzorků malware by pak mohl vznikat důvěryhodný a aktuální seznam takových IP adres. Uplatnění takového seznamu by se pak našlo například v projektu Turris (viz. Definice pojmů a zkratek) či při tvorbě statistik pro oddělení honeypotů sdružení CZ.NIC. V případě nalezení IP adres patřících strojům na území České republiky by z těchto informací mohl profitovat i projekt PROKI (viz. Definice pojmů a zkratek).

1.2 Cíle práce

Cílem této práce je získání indikátorů kompromitace ze síťové komunikace dynamickou analýzou malware. Získané indikátory kompromitace se následně porovnají s dostupnými databázemi jako například VirusTotal. O nabyté informace se následně obohatí výstupy z PassiveDNS (viz. Definice pojmů a zkratek). Takto získané indikátory kompromitace se následně poskytnou pro použití v projektech Turris a PROKI (viz. Definice pojmů a zkratek).

Ke splnění hlavních cílů bude zapotřebí realizovat několik dílčích úkolů:

- získat alespoň 100 aktuálních vzorků malware
- vytvořit nevirtualizovaného run-time prostředí s možností opakovaného spouštění malware s omezenou rychlostí síťového připojení
    - prozkoumat možnosti emulace jiných procesorových architektur
- získat indikátorů kompromitace ze síťové komunikace
    - porovnat získané indikátory kompromitace s dostupnými zdroji (např. VirusTotal)
    - obohatit získané IP adresy o informace z dostupných zdrojů (např. Passive DNS)
- prozkoumat možnosti automatizace procesu

Jako vedlejší leč nezanedbatelný produkt vznikne srovnání zastoupení jednotlivých procesorových architektur v získané sadě malware, což může sloužit jako ukazatel toho, jaká architektura bývá oblíbeným cílem útočníků. Dále vznikne návrh prostředí pro opakované spouštění malware na fyzickém hardwaru (bez virtualizace). Tím se umožní sledování malware i v případě, že má implementované ochrany proti běhu ve virtualizovaném prostředí,

čímž se snaží zabránit tomu, aby byl analyzován. Vzhledem k očekávanému využití honeypotů k získání potřebných vzorků k analýze dojde rovněž k analýze souborů nahraných útočníky na honeypoty.

2 Použité technologie

Pro účely této práce bylo využito několik technologií, které jsou uvedeny v následujících podkapitolách.

2.1 Honeypot

Jako honeypot se označuje počítač, přesněji však služba, která na něm běží. Úkolem této služby je emulovat jednu nebo více známých zranitelností systému. Díky honeypotu je tak možné zachytit a zaznamenat případné útočníky nebo jejich automatizované nástroje, které mají za úkol známou zranitelnost zneužít a získat tak kontrolu na cílovým systémem. Typicky pak dochází k uložení škodlivého kódu, který chtěl útočník na napadeném stroji spustit. Analýzou zachyceného škodlivého kódu a útoku lze následně zjistit podstatné informace, které obvykle zahrnují IP adresu útočníka, velikost a kryptografický otisk škodlivého kódu (např. SHA256), ale i IP adresu tzv. Command and Control serveru (viz. Definice pojmů a zkratek).

Na základě dlouhodobého pozorování podobných malware, užitých nástrojů a způsobů komunikace je následně možné přibližně určovat rodiny malware a jeho vývoj. Stejně tak využitím shodné infrastruktury pro řízení botnetu umožňuje odhadnout shodný zdroj útoku.

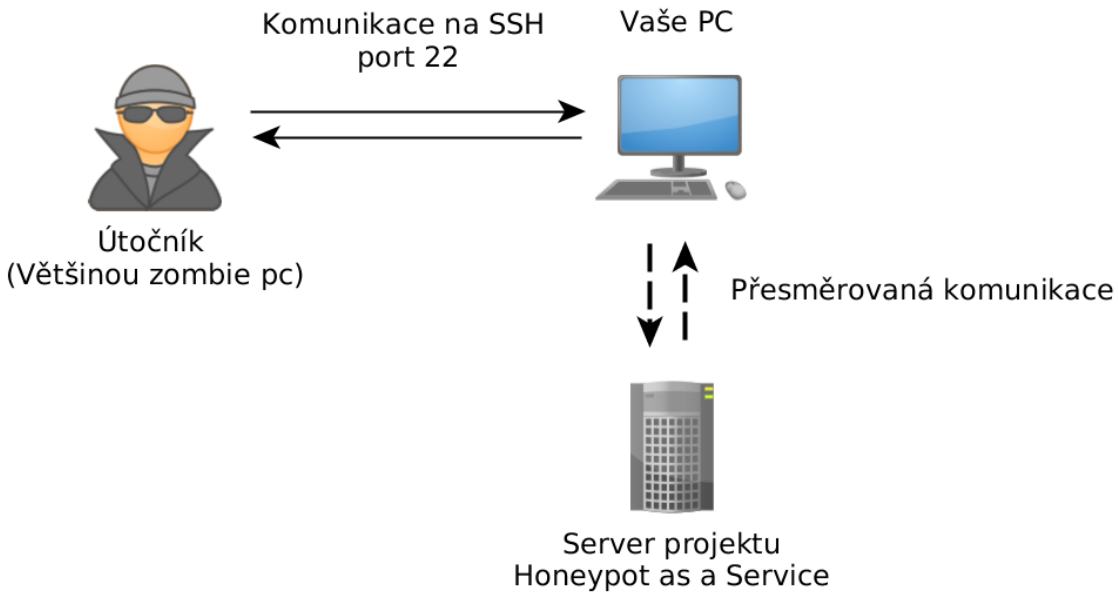
2.2 Kryptografický otisk

Kryptografický otisk škodlivého kódu, který je typicky distribuován ve formě skriptu nebo spustitelného souboru, se pak dá využít jak pro antivirová pravidla, tak pro komunikaci s ostatními CSIRT týmy či online službami pro jednoznačnou identifikaci konkrétního zachyceného vzorku. Výhoda kryptografického otisku spočívá ve velmi malé pravděpodobnosti kolize – při vhodné volbě algoritmu pro kryptografický otisk (tedy například sha256) je téměř vyloučeno, že by dva různé vzorky měly stejný otisk (hash). Nevýhoda využití této metody spočívá v nemožnosti najít příbuzné vzorky, protože z principu fungování kryptografických otisku vyplývá, že dva téměř stejné vzorky budou mít velmi odlišný otisk. To je jedna z vlastností, které se snaží přiblížit v práci použitý projekt SSDeep [1], jehož výstupem je řetězec podobný kryptografickému otisku. Avšak na rozdíl od kryptografických otisků, lze řetězce generované nástrojem SSDeep porovnávat a na základě počtu shodných znaků u různých vstupních souborů se dá odvodit míra jejich shodnosti.

SSDeep je de facto považován za standard a používá se například i pro základní identifikaci malware (například v projektu VirusTotal).

2.3 Honepot as a Service

Honeypot as a Service (HaaS) je projekt [2] provozovaný sdružením CZ.NIC. HaaS volně přeloženo znamená „Honeypot jako služba“. Tento projekt má za cíl realizovat aplikovaný výzkum v oblasti kybernetické bezpečnosti. Pro tyto účely využívá dobrovolníků pro přesměrování útoků jdoucích na zařízení v různých sítích poskytovatelů internetu (ISP), čímž podstatně zvyšuje šance na zachycení činnosti šířících se botnetů. K tomu využívá dvou komponent. V prvé řadě se jedná o proxy, která je na straně uživatele a jejímž úkolem je útok přesměrovat na honeypot. Proxy je v tomto případě jednoduchý program naslouchající na TCP portu 22, případně 2222. V případě, že na tento port přijde příchozí spojení, veškerá přijatá data – příkazy, jsou přeposlána na honeypoty Cowrie[1] služby HaaS. Tam dojde k jejich zpracování a odpovědi se vrací zpět přes proxy k útočníkovi. Takto způsobem jsou zachyceny pokusy ovládnout počítač a je zachycen malware, který měl být na cílovém počítači spuštěn. Využití proxy se v tomto případě nelze vyhnout, protože jiná řešení, jako například přesměrování v iptables, neumožňují zaznamenání IP adresy útočníka.



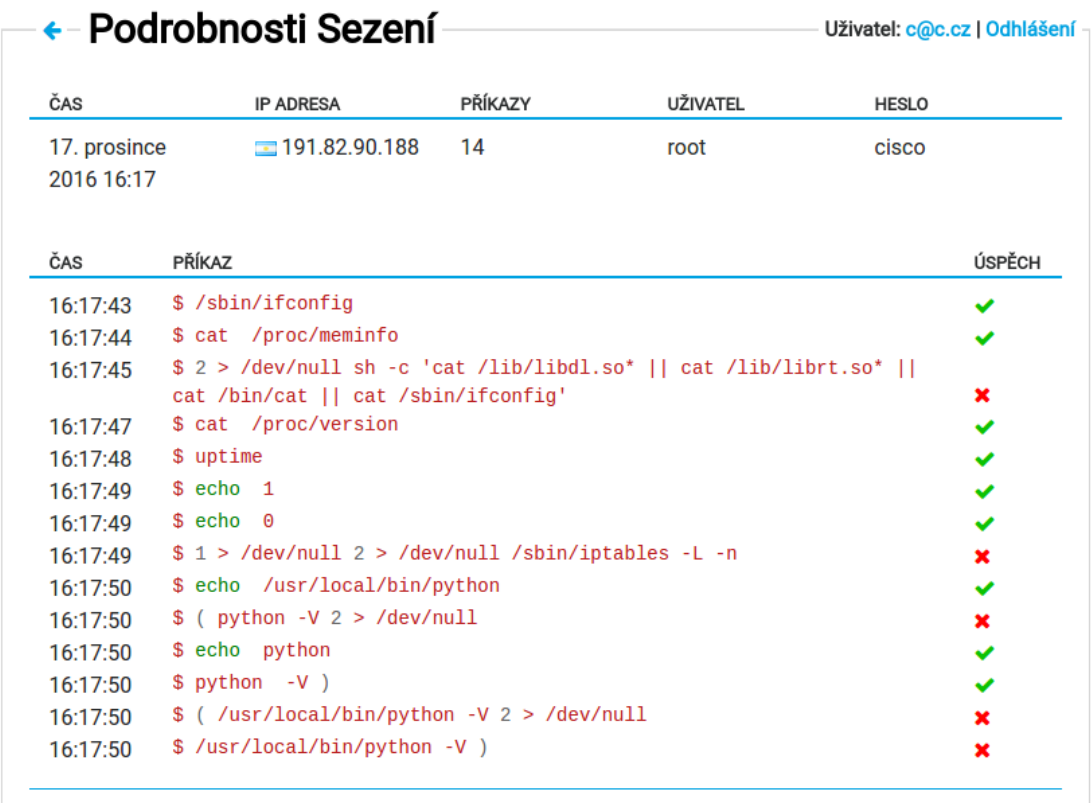
Obrázek 1: Schéma komunikace při zachycení přeposílání sezení (Zdroj: projekt HaaS)

[1]https://github.com/cowrie/cowrie

Výhoda pro uživatele pak spočívá v tom, že nemusí provozovat svůj vlastní honeypot, a riskovat tak jeho případně zneužití. Využitím proxy se veškerá potenciálně škodlivá činnost provádí mimo infrastrukturu uživatele. Jako další výhodu lze uvést přehledné grafické prostředí, ve kterém si uživatel může prohlédnout průběh útoku.

Všechny získané a analyzované vzorky pocházejí právě z projektu HaaS. Hlavní předností pro využití v této práci je, že poskytuje reálná syrová data tak, jak byla nahrána během kybernetického útoku a nejedná se tedy pouze o „sadu“ zajímavých vzorků.

Obrázek 2: Grafické rozhraní projektu Haas zobrazující zaznamenané útoky



Obrázek 3: Grafické rozhraní projektu Haas zobrazující průběh útoku

2.4 C&C server

Command and Control neboli C&C server je typicky počítač nebo server, který řídí botnet útočníka. Útočník se snaží napadnout a následně ovládnout co možná největší počet počítačů, které pak následně ovládá právě z řídícího serveru. Komunikace mezi ovládnutým počítačem (tzv. zombie) a Command and Control serverem pak probíhá například využitím IRC protokolu. Občas se využívá vlastních TCP spojení, která mohou občas být v dnešní době navíc i šifrovaná, a dokonce se objevuje i využití distribuovaných protokolů založených na principu decentralizovaných P2P sítí.

2.5 Indikátory kompromitace

Indikátory kompromitace (angl. Indicators of Compromise, IOC), jsou indikátory, pomocí kterých lze rozpoznat napadení konkrétním typem malware. Každý škodlivý kód má tedy vlastní indikátory kompromitace. Jedná se v zásadě o soubor stop, jejichž přítomnost na systému prokazuje přítomnost škodlivého programu, přestože tam již v danou chvíli ani nemusí být. Mezi takovéto indikátory patří například konkrétní záznamy v logu počítače, výskyt určitých souborů, záznamů v registru (v případě OS Windows), či běžící procesy

a vztahy mezi nimi. Indikátory kompromitace se ale soustředí i na síťovou komunikaci, ať už se jedná o DNS dotaz na konkrétní doménu, či pokus o spojení na konkrétní IP adresu. [3]

3 Postup

Jako první je zapotřebí nalézt vhodné prostředí pro spouštění malware. Dále je nutné vybrané prostředí připravit na základě předem daných parametrů, a co možná nejvhodněji uzpůsobit pro účely této práce.

3.1 Volba prostředí

Pro opakované spouštění malware se nabízí následující možnosti realizace prostředí: virtuální počítač, dedikovaný počítač nebo vlastní řešení.

3.1.1 Virtuální počítač

Tato možnost má obrovskou výhodu díky své jednoduchosti a nejmenší prodlevě mezi spouštěním různých vzorků malware. Zároveň dokáže v některých variantách emulovati různé procesorové platformy, což umožňuje snadno spouštět malware, který je kompilován pro jinou platformu. Na druhou stranu má tato varianta i několik nevýhod. Hlavní z nich je možnost úniku malware z virtualizovaného prostředí. Únikem malware se rozumí neoprávněné ovlivnění nadřazeného systému – typicky fyzického hardwaru nebo operačního systému. Mezi tyto jevy se řadí třeba neoprávněné čtení či zápis do paměti, která není vyhrazená virtualizovanému počítači, nebo útok typu *denial of service*. V nejhorším případě pak může dojít ke spuštění libovolného kódu. [4] Jako další nevýhodu lze uvést snahu malware o detekci prostředí (zdali neběží právě v tom virtualizovaném) a v případě, že ano, upraví své chování tak, aby se navenek jevil jako neškodný proces. [5] Po uvážení těchto nevýhod nebyla nakonec tato možnost vybrána.

3.1.2 Dedikovaný počítač

Dedikovaný počítač má oproti předchozímu řešení tu výhodu, že jej malware nebude detekovat jako virtualizované prostředí. Bohužel i u tohoto řešení nalezneme celou řadu nevýhod, mezi které patří například delší prodleva mezi novým čistým spuštěním nebo vyšší náklady. Asi největší nevýhodou je hrozba persistence malware, ať už v BIOSu/UEFI, či u jiných součástí počítače. [6] Existuje sice možnost tuto hrozbu částečně eliminovat extenzivními testy infikovaného počítače, avšak nikdy ji nelze stoprocentně vyloučit.

3.1.3 Vlastní řešení

Cílem vlastního řešení je vyhnout se nedostatkům uvedených prostředí a maximalizovat výhody obou. Jako fyzická platforma bylo vybráno Raspberry Pi 2 B+. Mezi výhody patří nízké pořizovací i provozní náklady, nízký výkon, který omezí možnosti malware, možnost vyloučit persistenci malware jinde než na vloženém paměťovém mediu (Raspberry nemá BIOS, apod.) a nulová šance na detekci emulace, protože malware bude spuštěn přímo na fyzickém procesoru. Mezi nevýhody pak patří delší čas při spuštění. Podpora dalších architektur je řešitelná (viz kapitola 3.2.2 Možnosti emulace jiných procesorových architektur). Po zvážení všech výhod a nevýhod bylo nakonec zvoleno vlastní řešení. Porovnání dostupných řešení lze vidět vedle sebe v Tabulce 1:

	Podpora více architektur	Hrozba persistence	Detekce Emulace	Cena	Průměrné pořadí
Fyzický počítač	3	3	1	3	2,5
Virtuální počítač	1	2	3	2	2
Vlastní řešení	2	1	2	1	1,5

Tabulka 1: Srovnání dostupných řešení – worst-case pro zvolené řešení

Vlastní řešení vychází jednoznačně nejlépe, ale je třeba osvětlit několik detailů. V kategorii detekce emulace se vlastní řešení snadno dostane na úroveň fyzického počítače, pokud budeme počítat pouze nativní vzorky. Stejně tak co se podpory více architektur týče, vlastní řešení je srovnatelné s virtuálním počítačem, protože dokáže emulovat nenativní vzorky. Ovšem v případě jejich emulace, vlastní řešení něco málo ztratí potenciální detekcí této emulace. Taktéž je možné namítnout, že virtuální počítač má menší náklady než vlastní řešení, nicméně v takovém případě by došlo k zhoršení hodnocení v kategorii „Hrozba persistence“ vzhledem k útokům, které umožňují spustit kód na fyzickém hardware. Právě vzhledem k tomu má virtuální počítač v kategorii „Cena“ horší hodnocení, protože je nutné toto řešení provozovat na samostatném počítači. Nicméně v případě best-case scénáře by hodnocení pro zvolené řešení vycházelo ještě o něco výhodněji, jak ukazuje tabulka 2:

	Podpora více architektur	Hrozba persistence	Detekce Emulace	Cena	Průměrné pořadí
Fyzický počítač	3	3	1	3	2,5
Virtuální počítač	1	2	3	2	2
Zvolené řešení	1	1	1	1	1

Tabulka 2: Srovnání dostupných řešení – best-case pro zvolené řešení

3.2 Příprava vlastního řešení

Pokud jde o operační systém, byl vybrán Raspbian, především kvůli podpoře vybraného hardwaru, a také proto, že je založen na osvědčené linuxové distribuci Debian.

Pro vyloučení persistence malware na zařízení bylo potřeba zařídit, aby veškerá dostupná non-volatilní paměťová média (tj. taková, která udrží obsah i po výpadku napájení), byla při startu v původním stavu. Takové médium existuje pro Raspberry Pi 2 B+ pouze jedno, a to paměťová microSD karta. Existují dvě možnosti, jak toho dosáhnout. Buď lze kartu microSD po každém spuštění malware přepisovat, nebo zařídit, aby malware nemohl na kartu nic zapsat. Vzhledem k tomu, že první možnost by byla časově náročná a zvyšovala by opotřebení karty, byla vybrána možnost druhá.

K zabránění zápisu na microSD kartu se opět nabízí několik možností. První je microSD kartu vůbec nepoužívat, což by ale způsobilo, že není z čeho systém zavést. Druhou možností je při startu připojit souborový systém pouze pro čtení (read-only). Tato možnost se dá ale snadno obejít příkazem:

```
mount -o remount rw /
```

Protože tato varianta tedy neposkytuje dostatečné záruky, jevilo se využití takzvané „write lock“ funkce microSD karty jako lepší možnost. Ani tato funkce sice proti obecnému přesvědčení netvoří hardwarovou bariéru proti zápisu na microSD kartu, ale její obejití by vyžadovalo změnu modulu jádra a jeho kompilaci pro konkrétní architekturu a verzi linuxového jádra, což lze považovat za dostatečně velkou překážku s únosnou mírou rizika. Jako protiopatření byla zavedena kontrola souborového systému jednou denně kryptografickým otiskem (sha256) generovaným v jiném zařízení. Tato kontrola zabrala přibližně 20 minut.

Empirickým pozorováním bylo zjištěno, že write-block pin není u Raspberry Pi zapojen (tzn. Write-block pin od microSD slotu není vyveden do procesoru či samostatného řadiče). To se projevilo tak, že při zamčení microSD karty nedošlo k znemožnění zápisu. Tento fakt bohužel nebylo možné ověřit z dokumentace, protože tato část zapojení ve veřejně dostupné dokumentaci chybí.

Obrázek 4: SD se zámkem proti zápisu

Obrázek 5: MicroSD se zvýrazněnou mezerou jejímž zaslepením se karta zamyká proti zápisu

Nakonec bylo přistoupeno k záložnímu řešení, u kterého se bootloader uložil na zapisovatelnou microSD kartu, která byla vložena do Raspberry a kořenový systém souborů se uložil na druhou microSD kartu, která byla vložena do čtečky paměťových karet a následně připojena přes USB 2.0 sběrnici k Raspberry Pi. Tímto způsobem stačilo kontrolovat kořenový systém souborů podle plánu jednou denně – to pro případ, že by některý ze vzorků úspěšně obešel nastavenou ochranu a zapsal na disk. Oddíl s bootloaderem, který byl zapisovatelný, se kontroloval stejným principem při každém spuštění (po každém restartu), což ale zabralo pouze několik vteřin.

3.2.1 Síťová konektivita

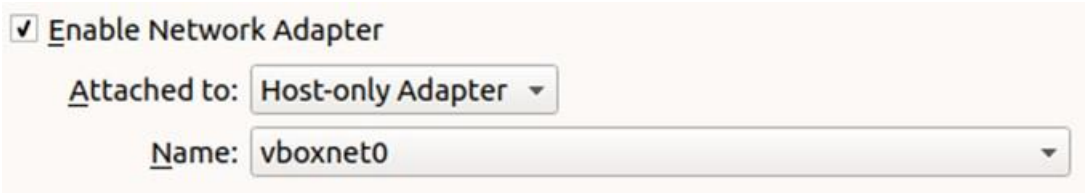
Za účelem získání indikátorů kompromitace ze síťové komunikace bylo potřeba zajistit síťovou konektivitu pro sledované zařízení takovým způsobem, aby bylo možné provoz nejenom sledovat, ale také zpomalit, a v případě potřeby i zastavit.

Z výše uvedených důvodů bylo zvoleno následující řešení: pracovní počítač s připojením k internetu, virtuální počítač spojen dvěma linkami k pracovnímu počítači a Raspberry Pi připojené k virtuálnímu počítači přes USB 2.0 – Ethernet adaptér. Pro virtualizaci byl použit VirtualBox. První rozhraní (označujme jako veth1) bylo na virtuálním počítači

připojeno jako bridge. Tímto způsobem byl virtuálnímu počítači, na kterém probíhal odposlech komunikace, umožněn přístup k internetu. Na virtuálním počítači byl nastaven NAT, který překládal všechny odchozí pakety od Raspberry Pi.

Druhé rozhraní (veth1) bylo připojeno formou „Host-only“ – tedy bylo spojeno pouze s počítačem. Tím byla zajištěna konektivita i v případě potřeby odpojení Raspberry Pi od internetu.

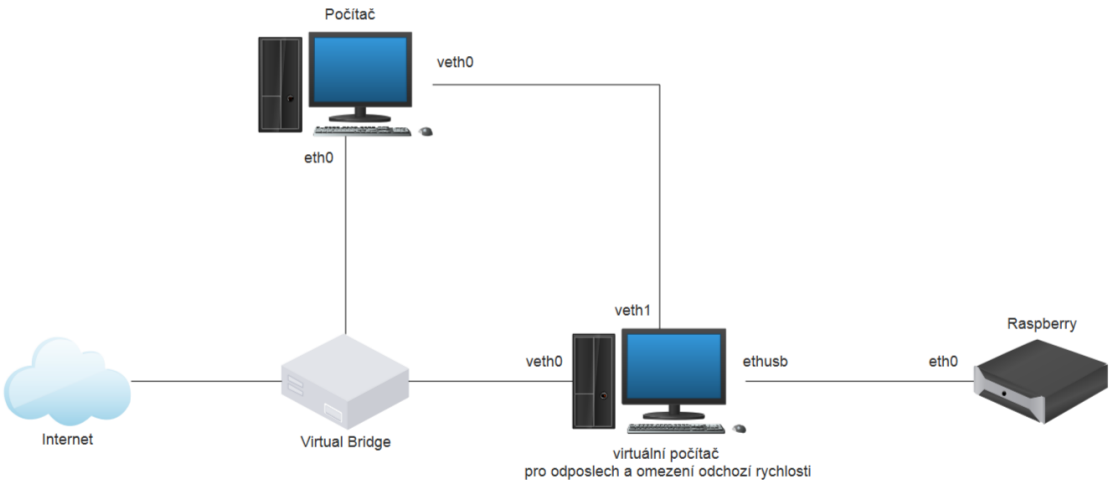
Samotné připojení bylo potřeba také omezit, pro případ že by mělo dojít ke zneužití



Obrázek 6: Nastavení adaptéru veth1 ve VirtualBoxu

Raspberry Pi například k útokům typu DDoS. Omezení bylo řešeno limitem maximální odchozí rychlosti na adaptéru eth2 pomocí nástroje *tc*. Tou byla limitovaná odchozí rychlost na 2 Mbit/s. Při omezování rychlosti bylo mimo jiné zjištěno, že vlivem způsobu zapojení byla rychlost limitovaná i bez použití tc na přibližně 10 Mbit/s. To bylo způsobeno připojením USB 2.0 Ethernet adaptéru do virtuální počítače. VirtualBox má totiž svobodné ovladače pouze pro USB 1.1. A právě teoretická rychlost USB 1.1 je 12 Mbit/s.

Obrázek 7: Fyzické zapojení systému



Obrázek 8: Logické zapojení systému

3.2.2 Možnosti emulace jiných procesorových architektur

Dle zadaných cílů práce byla prozkoumána možnost emulace jiných procesorových architektur. K tomu byl zvolen nástroj QEMU. Mezi hlavní důvody pro volbu tohoto nástroje je výborná integrace v Linuxovém prostředí, dostupnost v balíčkovacích systémech, rychlost a schopnost emulovat pestrou škálu různých procesorových architektur. K instalaci na Raspberry Pi stačí spustit následující příkaz:

```
sudo apt-get install qemu
```

Tímto způsobem byla zajištěna podpora různých architektur na testovacím systému. Nevýhody popsané v kapitole 3.1.1 Virtuální počítač byly v tomto případě převáženy možností analyzovat větší množství a spektrum vzorků malware.

3.2.3 Získání vzorků

Pro analýzu bylo zapotřebí získat vzorky malware, které by bylo možné analyzovat. Hlavním kritériem pro získané vzorky bylo jejich stáří. Respektive, aktuálnost. Čím novější, tím lepší. Včasnou analýzou nových vzorků se tak zvyšovala šance na zachycení komunikace mezi infikovaných strojem a řídícím serverem. Jako zdroj vzorků byl proto vybrán projekt HaaS (viz. kapitola 2.3 Honepot as a Service), protože umožnil poskytnout velké množství vzorků. Všechny vzorky byly nasbírané během března 2018 a jejich analýza probíhala v měsících březen–duben 2018.

3.3 Dynamická analýza

V této kapitole je popsána příprava vzorků a jejich spuštění pro získání dat k další analýze.

3.3.1 Předvýběr vzorků

Během analýzy několika prvních vzorků se ukázalo, že ne všechny vzorky se na testovacím prostředí spustí. Při bližším zkoumání bylo zjištěno, že se dochází k chybám při spuštění malware. Nejčastěji se jednalo o chybu „Bus error“, která se projevovala při spouštění binárních souborů architektury Intel x86. V takovém případě se na výstupu objevilo pouze „Bus error“ a proces byl ukončen. Pro rychlejší zpracování funkčních vzorků tak bylo potřeba nefunkční vzorky vyloučit. Vyloučení jednotlivých vzorků probíhalo tak, že se vzorek nahrál do připraveného prostředí a spustil. Po 10 vteřinách od spuštění byly zkontrolovány výstupy a u této chyby došlo k vyřazení vzorku a opakování cyklu. V případě, že byl vzorek úspěšně spuštěn, byl předán k další analýze. Následně bylo prostřední restartováno a zkontrolováno. Spuštění malware v této fázi bylo prováděno bez přístupu k internetu, plně automaticky a bez dalšího dohledu.

3.3.2 Postup

Pro získání dat k síťové analýze byl zvolen následující postup: na čerstvě nastartované testovací prostředí na Raspberry Pi byl nahrán vzorek malware pomocí nástroje *scp*. Následně byl pomocí *chmod* nastaven bit pro spuštění nahraného vzorku, který byl hned nato spuštěn. Malware i všechny uvedené příkazy byly na připraveném prostředí spouštěny s oprávněním root. Hlavním důvodem byla co možná největší volnost pro malware. Síťový provoz byl zaznamenán a zároveň zobrazován kombinací nástrojů *tcpdump* a *tshark* (textová varianta známého nástroje pro síťovou analýzu Wireshark). Po první minutě byla zaznamenána aktivní síťová spojení a vytížení procesoru následujícím příkazem:

```
ssh raspi "netstat -tunp & cat /proc/loadavg"
```

Vytížení procesoru bylo sledováno pro případnou možnost detekovat snahu o těžbu virtuálních měn. Tento příkaz byl opět proveden za další minutu a pár vteřin. Vícero nástrojů dostupných online omezuje čas spuštění právě na dobu dvě minuty nebo kratší, proto byl čas

analýzy pro jednotlivé vzorky prodloužen pro případ, že by se malware snažil vyhnout právě těmto typům analýzy.

Po uplynutí nastaveného času bylo zařízení restartováno:

```
ssh raspi "reboot"
```

A po jeho úspěšném startu byl ověřen stav oddílu systému souborů obsahující zavaděč na případné změny:

```
ssh raspi "sha256sum /dev/mmcblk0p1"
```

Pokud byl výstupem výše uvedené hashovací funknce jiný řetězec než „efcede046081781 3b7bdd7ba3981bb52c7cf6d35abc9b672f8c0a02796f6837b“ bylo zřejmé, že došlo ke změně. K tomu, ale během celého výzkumu nedošlo. Kořenový systém souborů byl testován stejně, ale na jiném stroji. Vzhledem k jeho délce (cca 20 minut) a zavedeným opatřením (viz. 3.2 Příprava vlastního řešení) bylo rozhodnuto provádět test pouze jednou denně.

Po kontrole systému bylo prostředí připraveno ke spuštění dalšího vzorku.

3.3.3 Automatizace

Vzhledem k vhodně použitým technologiím byla automatizace celého procesu relativně přímočará. Stačilo zřetězit použité příkazy do skriptu a následně jej spustit. Největší podíl na časové úspoře neslo vyřazení nefunkčních vzorků. Nicméně pro omezení vzniku škodlivých aktivit směřovaných do internetu (například skenování portů) bylo zapotřebí operátora (osoby), který probíhající síťovou aktivitu monitoroval a byl připraven zasáhnout, pokud by to situace vyžadovala. Avšak díky automatizaci, nebylo třeba se plně soustředit na celý průběh procesu, ale pouze na jeho kritické sekce. Mezi ty patřil moment, kdy se malware spustil svojí komunikaci směrem do internetu. Během doby, kdy docházelo k navrácení zařízení do původního stavu a přípravě spuštění nového vzorku, tak není třeba pozornosti lidské obsluhy.

4 Získaná data

Z projektu HaaS bylo za měsíc březen 2018 získáno celkem 4,3 GB dat (počítáno po deduplikaci a zabalení). V získaných datech bylo celkem 2240 různých souborů.

4.1 Prvotní analýza

Po získání surových dat zachycených honeypoty je třeba nejprve data probrat a očistit. Ne všechny soubory jsou malware.

Největší soubor zabíral 1,8 GB a podle nástroje *file* se jednalo o „DOS/MBR boot sector“ diskový oddíl. Při bližší analýze nástrojem *strings* byl nalezen řetězec „Linux Mint 18.3 Cinnamon 64-bit“. Na základě těchto informací by se dalo odhadovat, že si buď útočník snažil zazálohovat část svého systému, nebo pouze zkoušel, kolik se mu podaří nahrát na honeypot, anebo se mohl snažit přepsat spouštěcí část disku a získat tak systém trvale pod svou kontrolu.

Druhý největší soubor s velikostí 290 MB byl stejného typu, na jeho začátku se však vyskytovaly jiné řetězce, a to „Debian 9.3.0 amd64“. Poté následovaly ještě dva soubory různých typů s velikostí 125 a 100 MB. Zbytek již byly soubory nejrůznějších typů s velikostí menší než 5 MB.

Z výše uvedených statistik byl vyjmut jeden soubor, který byl šifrovaně předán Národnímu úřadu pro kybernetickou a informační bezpečnost. Zvláštní byl v tom, že jakožto soubor obsahující pouze text měl velikost 17 MB. Při dalším zkoumání bylo zjištěno, že obsahuje 1 115 213 řádků s různými doménami. Celkem 945 z nich patřilo pod českou národní doménu .CZ a obsahovaly mimo jiné domény či poddomény Ministerstva zahraničních věcí, Ministerstva životního prostředí, Ministerstva zemědělství nebo Státního ústavu pro kontrolu léčiv.

Tyto tři soubory byly odebrány z následující analýzy.

4.2 Rozdělení vzorků

Pro další zpracování bylo zbylých 2237 souborů rozděleno podle typu. K testování byly vybrány pouze binárních spustitelné soubory, protože u drtivě většiny skriptů (tj. ostatních

spustitelných souborů) je jejich účel snadno rozpoznatelný už jen povrchní statickou analýzou. To však u binárních souborů není možné. Proto je právě dynamická analýza vhodným nástrojem pro jejich rozbor.

Typ souboru:	#
ELF 32-bit LSB executable, Intel 80386	538
Bourne-Again shell script	358
ELF 32-bit LSB executable, MIPS	349
ELF 32-bit MSB executable, MIPS	346
ELF 32-bit LSB executable, ARM	298
ASCII text	238
Perl script text executable	26
data	16
HTML document	10
ELF 64-bit LSB executable, x86-64	10
Ostatní	49

Tabulka 3: Zastoupení jednotlivých typů souborů

4.3 Rozdělení testovaných vzorků

K dynamické analýze byly tedy vybrány jen vzorky následujících typů:

- ELF 32-bit LSB executable, Intel 80386
- ELF 32-bit LSB executable, MIPS
- ELF 32-bit MSB executable, MIPS
- ELF 32-bit LSB executable, ARM
- ELF 64-bit LSB executable, x86-64

Celkem se tedy jedná 1 541 vzorků 5 různých procesorových architektur. Bohužel, jak již bylo zmíněno v kapitole 3.3.1 Předvýběr vzorků, kde je popsána příprava vzorků a jejich spuštění pro získání dat k další analýze**,** ne všechny se podařilo úspěšně spustit. Celkově bylo analyzováno 409 binárních vzorků a jejich rozdělení dle architektur je následující:

Typ souboru:	**počet**
ELF 32-bit LSB executable, Intel 80386	199
ELF 32-bit LSB executable, MIPS	122
ELF 32-bit LSB executable, ARM	78
ELF 64-bit LSB executable, x86-64	10

Tabulka 4: Zastoupení jednotlivých typů souborů

Z dat tak jasně vyplývá, že se nepodařilo spustit ani jeden vzorek architektury MIPS s opačnou endianitou. V ostatních případech by se výsledek dal přiřknout snaze malware vyhnout se běhu v emulovaném prostředí. Nižší počet by se tím vysvětlil i v zdánlivě nativní architektuře ARM – což je architektura použitého Raspberry Pi. Problém zde ale spočívá v drobných odlišnostech jednotlivých pod-architektur těchto procesorů. Raspberry Pi B+ totiž nemá plnou podporu standardu ARMv7, někde označovanou jako „armhf“ (ARM hard float) což je hardwarové rozšíření pro podporu operací s plovoucí desetinou čárkou. Proto nelze na Raspberry Pi B+ provozovat standardní distribuci OS Debian, ale využívá se zde vlastní odvozené distribuce Raspbian, která je sestavena právě pro Raspberry s maximálním využitím ostatních rozšíření [7]. Pravděpodobně z toho důvodu byly binární soubory, jinak zahrnuté ve skupině ARM spuštěny v emulátoru. QEMU totiž všechny nenativní binární soubory automaticky spustí na základě informací v jejich hlavičce. To mohlo vést k chybě při spuštění (viz. 16), ovšem pravděpodobnější je spíše chyba v malware.

4.4 Kontrolní test

Vzhledem k faktu, že se nepodařilo spustit přibližně tři čtvrtiny vzorků, byl proveden test funkčnosti malware na klasickém PC se 64bitovým procesorem. Opakovatelné spouštění pokusu bylo v tomto případě zajištěno balíkem „overlayroot“, který zaručí, že kořenový filesystém bude read-only, ale bude překryt vrstvou „tmpfs“. Tato skutečnost pak umožní malware zapisovat data. Nevýhody popsané v kapitole číslo 3.1.2 samozřejmě platí, nicméně stroj je již vyřazen a nikdy se tak na něj nedostanou citlivá data. Jako provozní nevýhody se projevily příliš dlouhá doba restartu, a tak dlouho trvající kontrola disku, že se ani nevyplatilo jej kontrolovat. Nakonec byla provedena pouze jednou alespoň pro jistotu, že za celý průběh analýzy nedošlo k jeho změně.

Typ souboru:	#
ELF 32-bit LSB executable, MIPS	349
ELF 32-bit MSB executable, MIPS	346
ELF 32-bit LSB executable, ARM	302
ELF 32-bit LSB executable, Intel 80386	51
ELF 64-bit LSB executable, x86-64	10

Tabulka 5: Zastoupení úspěšně spuštěných vzorků na kontrolním prostředí

Jak je vidět z tabulky číslo 5, výsledky jsou poněkud překvapivé. Počet vzorků úspěšně spuštěných na nativní architektuře je paradoxně ještě menší než v případě Raspberry Pi. Nejčastější chyba při pokusu o spuštění malware byla „Bus error“. Chybová hlášení bylo shodné s tím, které se objevovalo při spuštění na původním prostředí. Příčina této chyby je neznámá a nezbývá než spekulovat, zda malware nepočítal s nějakým konkrétním typem hardwaru či verzí jádra operačního systému.

Dle dokumentace se jedná v případě „Bus Error“ nejčastěji o chybu čtení disku, což je z principu vyloučeno, neexistující fyzická adresa (konkrétní typ HW) či nezarovnaná adresa při přístupu do paměti. Z toho by se dalo usuzovat, že se jedná o chybu programátora.

Nutno podotknout, že v případě architektury MIPS byla kontrolní architektura podstatně úspěšnější, a to i v případě vzorků s opačnou endianitou. Paradoxem na závěr je fakt, že se na kontrolní architektuře podařilo spustit více vzorků architektury ARM než na Raspberry Pi.

5 Analýza

Po získání kompletního vzorku dat byla provedena statická analýza nástrojem SSDeep.

```
ssdeep -prg .
```

Ten analyzované nástroje na základě jejich podobnost rozdělil celkem do 14 skupin. Tyto skupiny obsahovaly celkem 383 souborů, což znamená, že celkem 27 vzorků nebylo možné nikam přiřadit.

5.1 Pozorované jevy

Při dynamické analýze bylo pozorováno několik zajímavých jevů. Asi nejzajímavějším byl pokus o DDoS útok, který vlastně nebyl útokem. Jednalo se o pokus zjistit, zdali napadený stroj má možnost podvrhovat zdrojovou IP adresu. Od DDoS útoku se tak tento incident lišil faktem, že pokyn k zaslání paketů přišel z Command and Control serveru, na který tyto pakety pak také mířily (viz. Obrázek 9: Detekce schopnosti podvrhovat zdrojové IP adresy) Schopnost počítače odesílat pakety s libovolnou adresou je typicky limitována buď síťovým umístěním (například za NAT), nebo implementací BCP 38 (Best Current Practice) na straně poskytovatele připojení k internetu (ISP). Právě BCP 38 (někde označované jako RFC 2827) je velmi dobrým nástrojem určeným pro redukci útoků typu Denial of Service. Z implementačního hlediska se vlastně jedná o pravidlo pro firewall, které zahodí všechny odchozí pakety, jejichž IP adresa z dané zdrojové sítě nepochází [8].

Dalším zajímavým jevem byl také pravděpodobný TCP SYN sken. Častým úkolem síťového malware bývá také hledání dalších zranitelných zařízení. V několika případech tak byl během analýzy zaznamenán jev, který vypadal jako sken náhodných IP adres a cílových portů. Později se však ukázalo, že ani toto prvotní podezření nebylo správné.

10.0.3.100	51260	107.160.40.49	3308	TCP	51260 → 3308 [SYN] Seq=
107.160.40.49	3308	10.0.3.100	51260	TCP	3308 → 51260 [SYN, ACK
10.0.3.100	51260	107.160.40.49	3308	TCP	51260 → 3308 [ACK] Seq=
10.0.3.100	51260	107.160.40.49	3308	TCP	51260 → 3308 [PSH, ACK
107.160.40.49	3308	10.0.3.100	51260	TCP	3308 → 51260 [ACK] Seq=
10.0.3.100	51260	107.160.40.49	3308	TCP	51260 → 3308 [PSH, ACK
107.160.40.49	3308	10.0.3.100	51260	TCP	3308 → 51260 [ACK] Seq=
107.160.40.49	3308	10.0.3.100	51260	TCP	3308 → 51260 [PSH, ACK
10.0.3.100	51260	107.160.40.49	3308	TCP	51260 → 3308 [ACK] Seq=
10.0.3.99	45570	107.160.40.49	3308	UDP	45570 → 3308 Len=44
10.0.3.101	15876	107.160.40.49	3308	UDP	15876 → 3308 Len=44
10.0.3.97	2820	107.160.40.49	3308	UDP	2820 → 3308 Len=44
10.0.3.103	23814	107.160.40.49	3308	UDP	23814 → 3308 Len=44
10.0.3.93	48390	107.160.40.49	3308	UDP	48390 → 3308 Len=44
10.0.3.107	39690	107.160.40.49	3308	UDP	39690 → 3308 Len=44
10.0.3.85	8460	107.160.40.49	3308	UDP	8460 → 3308 Len=44
10.0.3.115	5907	107.160.40.49	3308	UDP	5907 → 3308 Len=44
10.0.3.69	59670	107.160.40.49	3308	UDP	59670 → 3308 Len=44
10.0.3.131	3876	107.160.40.49	3308	UDP	3876 → 3308 Len=44
10.0.3.37	31020	107.160.40.49	3308	UDP	31020 → 3308 Len=44
10.0.3.163	65349	107.160.40.49	3308	UDP	65349 → 3308 Len=44
10.0.2.229	39255	107.160.40.49	3308	UDP	39255 → 3308 Len=44
10.0.3.227	57225	107.160.40.49	3308	UDP	57225 → 3308 Len=44
10.0.2.101	55725	107.160.40.49	3308	UDP	55725 → 3308 Len=44
10.0.4.99	40721	107.160.40.49	3308	UDP	40721 → 3308 Len=44
10.0.1.101	22874	107.160.40.49	3308	UDP	22874 → 3308 Len=44
10.0.5.99	7969	107.160.40.49	3308	UDP	7969 → 3308 Len=44
9.255.255.101	22963	107.160.40.49	3308	UDP	22963 → 3308 Len=44
10.0.7.99	8000	107.160.40.49	3308	UDP	8000 → 3308 Len=44
9.255.251.101	22885	107.160.40.49	3308	UDP	22885 → 3308 Len=44
10.0.11.99	8062	107.160.40.49	3308	UDP	8062 → 3308 Len=44
9.255.243.101	22985	107.160.40.49	3308	UDP	22985 → 3308 Len=44
10.0.19.99	8186	107.160.40.49	3308	UDP	8186 → 3308 Len=44
9.255.227.101	22929	107.160.40.49	3308	UDP	22929 → 3308 Len=44
10.0.35.99	8178	107.160.40.49	3308	UDP	8178 → 3308 Len=44
9.255.195.101	22817	107.160.40.49	3308	UDP	22817 → 3308 Len=44
10.0.67.99	8162	107.160.40.49	3308	UDP	8162 → 3308 Len=44
9.255.131.101	22849	107.160.40.49	3308	UDP	22849 → 3308 Len=44
10.0.131.99	8130	107.160.40.49	3308	UDP	8130 → 3308 Len=44
9.255.3.101	22913	107.160.40.49	3308	UDP	22913 → 3308 Len=44
10.1.3.99	8066	107.160.40.49	3308	UDP	8066 → 3308 Len=44

Obrázek 9: Detekce schopnosti podvrhovat zdrojové IP adresy

5.2 Vzorky zaznamenaných komunikací

5.2.1 Malware BitCoinMiner-AY [PUP]

Právě při bližší analýze pravděpodobného TCP skenu bylo zjištěno, že vzorek, u kterého toto chování bylo pozorováno, patří do jedné ze skupin malware detekované pomocí SSDeep. Při analýze síťového provozu ostatních vzorků bylo zjištěno, že všechny se snaží připojit ke stejným cílům. Navíc tak docházelo v naprosto shodném pořadí. Při zadání hashe do služby VirusTotal bylo zjištěno, že daný vzorek

(SHA256 455aea7f3dae5e531453103860e9f09960168a7bdd230 423cbf4c9d93fd88d20)

je detekován antivirem Avast jako BitCoinMiner-AY [PUP] s celkovou detekcí 32/60. Ze 6 vzorků stejné skupiny byl nalezen jeden, který nikdy nebyl na službě VirusTotal testován (SHA256 46e6891b93672a5d47ede67046870ea1437b7a75c648d14bb4b5f2f5 7364aa4d). Po jeho nahrání bylo zjištěno, že ho detekuje celkem 25 antivirů, což je v průměru přibližně o 7 méně než ostatní vzorky. Dá se tedy odhadovat, že snaha tvůrců malware snížit procentuální detekci souboru napříč antivirovými produkty má alespoň drobné úspěchy.

Protože byl vzorek na službě VirusTotal označen jako miner, síťová spojení, o které se pokoušel, tedy spíše mohly vést na tzv. pooly, do kterých se sdružují zařízení se záměrem těžit virtuální měnu.

IP	**port**
45.77.55.161	3333
58.99.32.135	4545
113.59.33.59	1488
124.251.33.242	3027
125.211.202.186	3142
163.17.30.212	8525
198.2.199.236	8635
218.211.90.199	8434
221.204.214.158	7635
222.43.116.233	2258

Tabulka 6: Seznam síťových soketů, ke kterým se připojoval malware BitCoinMiner-AY [PUP]

5.2.2 Malware ELF:Elknot-AE [Trj]

Během sledování síťového provozu byly zaznamenány DNS dotazy na doménu **ddos518cc.top**. Při bližším zkoumání s pomocí SSDeep bylo zjištěno, že se jedná o celkem 18 vzorků, které podle klasifikace SSDeep patří do stejné kategorie, přičemž 7 z nich nebylo doposud nahráno na VirusTotal. Po nahrání byly detekovány pod stejným názvem jako ostatní.

Jaku už název napovídá, jedná se o trojan. Každá varianta měla snahu se připojit ke svému Command and Control serveru, který byl ale téměř pokaždé jiný. U celkem 3 vzorků bylo spojení úspěšné a byla zaznamenána komunikace s řídícím serverem. Ve všech případech docházelo k trvalé výměně dat, které ale byly po celou dobu záznamu konstantní. Jedinou výjimku tvořil začátek spojení, při kterém posílal trojan na server kromě ostatních dat

následující řetězce: „-== Love AV ==-“ a „Linux 4.9.59+.1:G2.40“. Krom úspěšných byla zaznamenána dvě částečně úspěšná spojení, při kterých klient odeslal úvodní data s řetězci, ale nepřišla mu odpověď. Část druhého řetězce byla tvořena z verze jádra napadeného systému. Přesný výstup nástroje *uname -rs* byl „Linux 4.9.59+“.

Ze záznamu síťového provozu byly identifikovány tyto indikátory kompromitace:

Domény
ddos518cc.top
lsfzgzs.f3322.net
s2010218.f3322.net

Tabulka 7: Seznam domén zachycených analýzou síťového provozu ELF:Elknot-AE [Trj]

IP adresy
43.252.231.202
61.147.112.10
101.254.149.193
118.184.32.55
122.193.64.147
123.129.217.153
123.249.13.32
222.186.58.135

Tabulka 8: Seznam zachycených IP adres ELF:Elknot-AE [Trj]

Tyto získané indikátory kompromitace y byly následně porovnány s databází Passive DNS poskytovanou rakouským CERT týmem. Z výsledků bylo patrno, že některé z výše uvedených síťových koncových bodů byly použity i pro jiné útoky.

Například IP adresa 43.252.231.202 byla pravděpodobně použita pro phishing. V roce 2015 na ně bylo přeloženo několik domén obsahující „apple“ ve svém názvu:

Phishingové domény
apple-fxors.com
mail.apple-fxors.com
www.apple-fxors.com
www.apple-safrs.com
apple-fxors.com

Tabulka 9: Phishingové domény získané z databáze Passive DNS

Mezi dalšími záznamy se v Passive DNS systému objevila například IP adresa 122.193.64.147, na kterou v roce 2017 odkazoval A záznam domény **www.pay128.com**. Nalezená doména **s2010218.f3322.net** podle záznamů odkazovala v roce 2016 na IP adresu 115.239.248.234. Zatímco v době provádění analýzy odkazovala na zaznamenanou IP adresu 122.193.64.147. Právě při úspěšném spojení na tuto IP adresu se podařilo zaznamenat komunikaci malware s DNS serverem.

S pomocí nástroje *whois* se nepodařilo najít žádnou významnou spojitost mezi získanými IP adresami. Jejich vlastníci byli většinou poskytovatelé hostingových služeb ve východní Asii.

5.2.3 Malware ELF:Aesddos-H [Trj]

Tato skupina osmi vzorků byla identifikovaná nástrojem SSDeep se ve všech případech připojovala na port 48080 a ve všech 8 případech se jednalo o architekturu malware kompilovaného pro architekturu Intel 80386.

IP adresy
42.51.194.26
42.51.45.185
43.224.249.71
118.184.32.55
121.201.125.213
121.201.127.27
222.186.34.102

Tabulka 10: Seznam zachycených IP adres ELF:Aesddos-H [Trj]

Při porovnávání záznamů z Passive DNS byla nalezena souvislost se skupinou vzorků 5.2.2 ELF:Elknot-AE [Trj]. Na IP adresu 42.51.194.26 ukazoval v dubnu 2018 A záznam domény **3124166750.f3322.net**, u které se shodovala doména druhého řádu **f3322.net**

s doménou zachycenou v komunikaci dříve analyzovaného vzorku. Stejně tak IP adresa 43.224.249.71 byla použita jako adresa C&C serveru u předchozí skupiny. Na IP adresu 121.201.125.213 byly systémem Passive DNS detekované překlady domény st-barths.xyz v únoru 2018. Účel této domény nebyl dále zkoumán, nicméně podporuje často zmiňovanou statistiku .XYZ TLD, která ukazuje na fakt, že tato doména má velmi špatnou reputaci [10]. Poslední zkoumanou IP adresou v systému Passive DNS byla 222.186.34.102. Na tu se od září 2017 do ledna 2018 odkazovala doména **longxin99.cn** a během ledna také **loocoo.f3322.net**. Doména **loocoo.f3322.net** je tak dalším pojítkem s malware Malware ELF:Elknot-AE [Trj].

5.2.4 Malware ELF:Elknot-BT [Cryp]

Tato skupina dvou unikátních vzorků, byla podle služby VirusTotal příbuzná k Malware ELF:Elknot-AE [Trj]. Z těchto dvou vzorků byla k dipozici pouze jedna kompletní komunikace, během které ale nedošlo ke zneužití napadeného stroje. V obou případech se jednalo o architekturu malware kompilovaného pro architekturu Intel 80386 a v obou případech se malware připojoval na TCP port 10991.

IP adresy
222.73.129.173
23.234.29.116

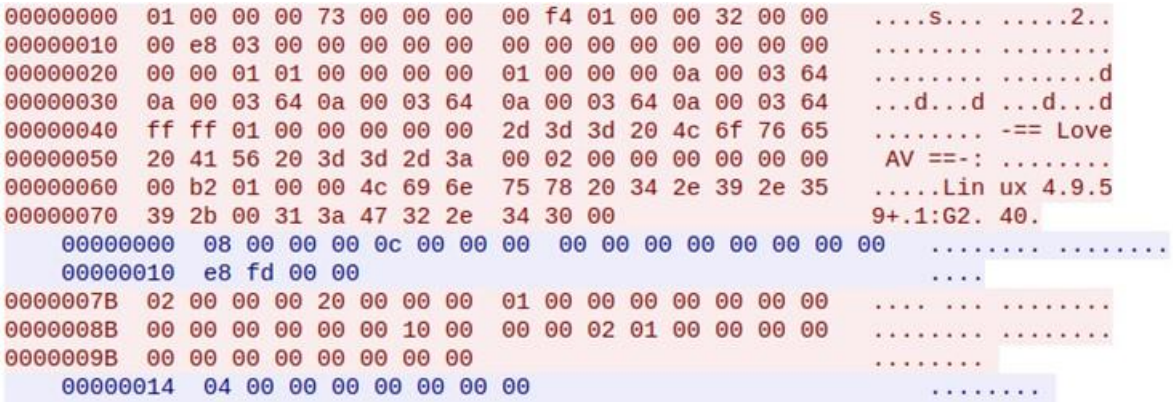
Tabulka 11: Seznam zachycených IP adres ELF:Elknot-BT [Cryp]

Při zkoumání IP adresy 23.234.29.116 v systému Passive DNS byl objeven seznam domén, které byly během roku 2018 překládány právě na tuto IP adresu.

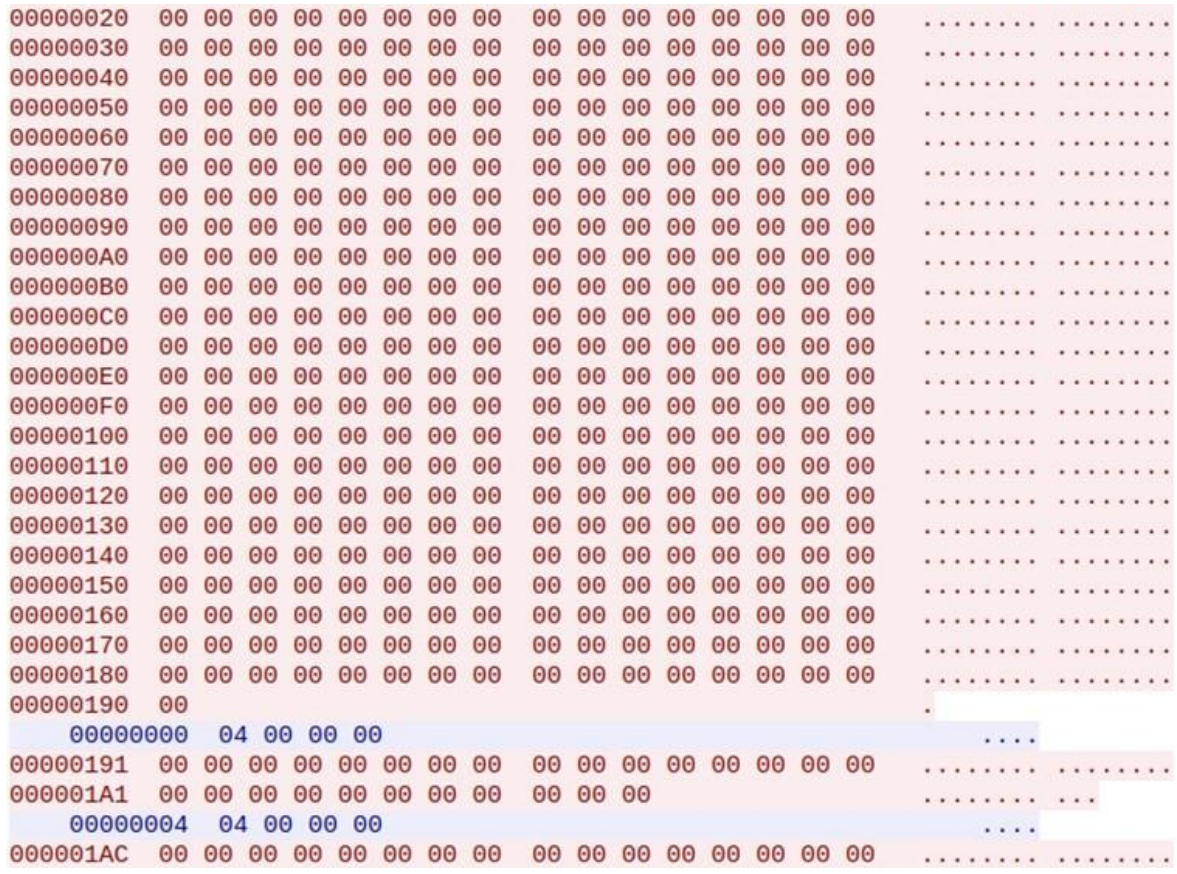
Domény:
a3zzy.top
cling.club
deedf.top
dzq520.top
huyun3.top
jies.top
jldhxy.top
s8q.top

Tabulka 12: Domény získané z databáze Passive DNS pro IP adresu 23.234.29.116

Žádná z těchto domén se neshodovala s již dříve objevenou. Jediná spojitost s Malware ELF:Elknot-AE [Trj] byla pouze v tom, že malware odesílal na C&C server řetězec obsahující řetězec nesoucí verzi jádra operačního systému „Linux 4.9.59+“, ale ani pozice tohoto řetězce od začátku dat neodpovídala dříve zaznamenané pozici.



Obrázek 11: Záznam komunikace s C&C serverem u ELF:Elknot-AE [Trj]



Obrázek 10: Záznam komunikace s C&C serverem u ELF:Elknot-BT [Cryp]

Na základě síťové analýzy se tedy vůbec nejedná o příbuzné vzorky, a také SSDeep tyto vzorky rozdělil do různých skupin. lze tak usoudit především dle obsahu přenášené komunikace. V obou je sice vidět verze linuxového jádra, ale zatímco v prvním případě se přenáší další data v druhém případě se nic jiného nepřenáší. Pozice řetězce představujícího verzi jádra se v komunikaci taktéž liší.

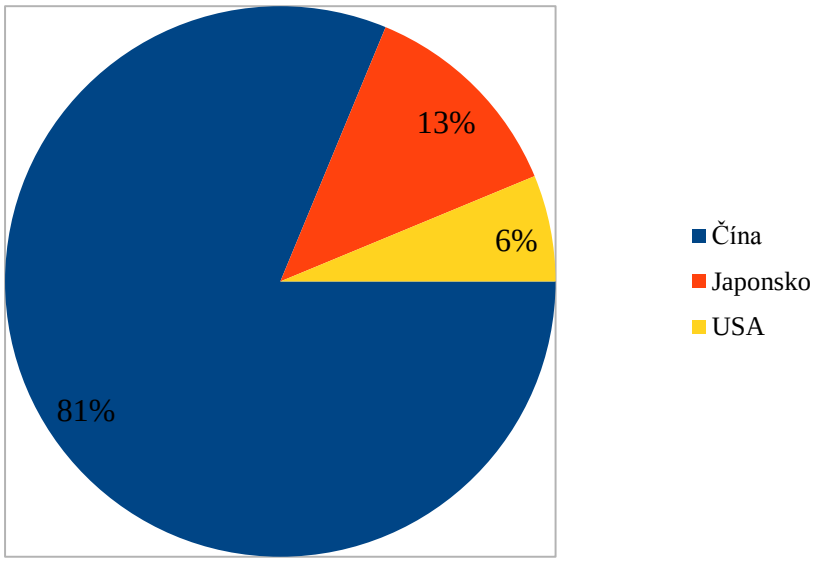
Na adresu 222.73.129.173 odkazovala v roce 2018 pouze doména jzanz.com.

5.3 Získané IP adresy C&C serverů

V rámci analýzy vzorků bylo identifikováno několik IP adres C&C serverů. Nabízí se tedy možnost tyto IP adresy na firewallu úplně blokovat, případě alespoň generovat upozornění, pokud by ke spojení na některé z uvedených IP adres došlo. V průběhu času samozřejmě může dojít k tomu, že se využití IP změní na něco užitečného, proto je zapotřebí dobré dokumentace, aby v případě problémů spojených se síťovou konektivitou bylo možné rychle vyloučit či potvrdit tuto možnost.

IP	**Země**
23.234.29.116	USA
42.51.194.26	Čína
42.51.45.185	Čína
43.224.249.71	Japonsko
43.252.231.202	Japonsko
61.147.112.10	Čína
101.254.149.193	Čína
118.184.32.55	Čína
121.201.125.213	Čína
121.201.127.27	Čína
122.193.64.147	Čína
123.129.217.153	Čína
123.249.13.32	Čína
222.186.34.102	Čína
222.186.58.135	Čína
222.73.129.173	Čína

Tabulka 13: Seznam získaných IP adres a jejich původu



Obrázek 12: Poměr umístění C&C serverů dle zemí

Na základě seznamu získaných IP adres byl sestaven graf zobrazující počet umístění objevených C&C serverů v jednotlivých zemích. Graf tak slouží pouze pro přehlednost. Pro případné závěry by bylo potřeba mít delší seznam získaných IP adres.

6 Diskuze

Sběr indikátorů kompromitace proběhl úspěšně. Fakt, že se některé vzorky nepodařilo spustit ničemu nevadí. Jak je vidět v analýze pomocí nástroje SSDeep (v kapitole 5 Analýza) malware se často vyskytuje ve velmi podobných mutacích a není tedy zapotřebí analyzovat každý vzorek. Je to též prakticky nemožné, protože nelze zajistit, aby honeypoty zachytily všechny vzorky malware, které se zrovna vyskytují.

Podařilo se prokázat, že navrhované řešení funguje dostatečně bezpečně a lze jej bez potíží používat i na dále.

Jak se ukázalo, lidský faktor v tomto případě úplně vyloučit nelze, protože ne každá IP adresa, kam se malware připojuje je C&C serverem, ale může se jednat o například o server sdružující stroje těžící virtuální měny. Zde pak záleží na požadavcích na kvalitu vstupních dat. Zatímco u běžných soukromých firem by bylo zablokování těžařských serverů spíše výhodou, v případě firem specializujících se na těžbu virtuálních měn by mohlo mít nasazení takových pravidel fatální následky.

Druhým důvodem, proč nelze tuto činnost automatizovat plně, je potřeba včasného zastavení odchozího kybernetického útoku jako například DDoS.

7 Závěr

Zisk vzorků malware proběhl úspěšně a výrazně překročil rozsah zadání. To se nakonec ukázalo jako příhodné, protože ne všechny vzorky bylo možné spustit. Bylo rovněž dokázáno, že chyby při spuštění malware nebyly způsobeny nevhodným zvolením prostředí. Ostatně navrhované řešení je na běhovém prostředí nezávislé a lze ho snadno aplikovat na libovolnou jinou architekturu dle potřeb analytika.

Zvolené běhové prostředí pro opakované spouštění malware se ukázalo být velmi efektivním a levným řešením a představuje tak velmi vhodný způsob, jak podobnou dynamickou analýzu malware realizovat. Zvláště pak provedená automatizace sběru síťového provozu významně usnadnila celý proces potřebný k získání síťových indikátorů kompromitace. Připravené řešení se osvědčilo a během celého pokusu nedošlo k pozměnění kořenového systému souborů ani bootloaderu malware.

V rámci porovnání získaných indikátorů kompromitace ů se službou VirusTotal došlo k nahrání do té doby neidentifikovaných vzorků, což přidá antivirovým společnostem možnost tyto vzorky analyzovat a zdokonalit tak pravidla pro jejich detekci. Tato skutečnost poukázala na fakt, že by bylo vhodné získané vzorky binárních souborů zasílat službě VirusTotal automaticky.

Obohacení získaných IP adres o data z Passive DNS rovněž přineslo zajímavé výsledky a umožnilo nalézt spojitost mezi dvěma zdánlivě odlišnými vzorky malware. Bylo tak prokázáno, že v obou případech byla využita stejná infrastruktura řídících serverů.

Předání dat získaných v rámci této diplomové práce zainteresovaným stranám (Národní úřad pro kybernetickou a informační bezpečnost, projekt Turris a projekt PROKI) proběhlo úspěšně. Pouze pro projekt PROKI neměla získaná data žádný přínos, protože získané IP adresy nepatřili pod Českou republiku.

Výsledky z této práce byly prezentovány na konferenci/výstavě Secutech 2018 na Taiwanu, v závěrečné zprávě projektu HaaS a vyšly také v druhém čísle recenzovaného časopisu DSM v roce 2018 s velmi kladnou zpětnou vazbou.

Řešení se ukázalo jako použitelné a bude využito k dalším analýzám.

8 Definice pojmů a zkratek

ARM	Procesorová architektura
Bootloader	Malý program zodpovědný za zavedení a spuštění operačního systému.
Botnet	Botnetem se nazývá skupina Zombie (botů) připojující se k C&C
C&C	Command and Control server – řídící počítač botnetu. Občas zkracováno na C2.
CERT	Computer emergency response team – z dnešního hlediska se jedná o ekvivalent CSIRT týmu s rozdílem, že CERT je registrovanou ochrannou známkou vlastněnou univerzitou Carnegie Mellon.
CSIRT	Computer Security Incident Response Team – tým lidí, jehož úkolem je řešit bezpečností incidenty.
DDoS	Distributed Denial of Service – útok, jehož cílem je znepřístupnění cílové infrastruktury, kterého se z pohledu cílové infrastruktury účastní více zařízení.
DoS	Denial of Service – útok jehož cílem je znepřístupnění cílové infrastruktury.
HaaS	Honeypot as a Service – viz. kapitola 6 Honepot as a Service.
IOC	Indicators of Compromise – viz kapitola 8 Indikátory kompromitace.
IRC protokol	Internet Releay Chat Protocol – Protokol sloužící k textové komunikaci.
MIPS	Procesorová architektura
NAT	Network address translation – překlad síťových adres.
P2P sítě	Peer to peer sítě – takové sítě, které na rozdíl od client-server nemají centrální bod. Všichni účastníci jsou si rovni a neexistuje zde typicky takový prvek, jehož výpadkem by zkolabovala celá síť.
Passive DNS	Obecně se jedná o systém umožňující dohledání zaznamenaných DNS dotazů, čímž velmi napomáhá při řešení kybernetických incidentů. Vzhledem k faktu, že DNS záznamy nejsou statické, je tak možné zjistit, na jakou IP adresu bylo doménové jméno historicky přeloženo. Konkrétně se pak v rozsahu této práce jedná o neveřejný systém provozovaný rakouským týmem CERT.at.
PROKI	Predikce a ochrana před kybernetickými incidenty je projekt sdružení CZ.NIC pro sběr a sdílení informací o kybernetických nebezpečích. Více na https://csirt.cz/page/3586/proki/
RFC	Request for comment – sada standardů.
SHA	Hashovací algoritmus.
Turris	Projekt sdružení CZ.NIC s cílem vytvořit bezpečný open-source router. Více na https://www.turris.cz/
Zombie	Infikovaný počítač pod kontrolou útočníka. Typicky ovládaný prostřednictvím C&C serveru.

9 Seznam obrázků

10 Seznam tabulek

11 Použité zdroje

1: Raygoza, Daniel. Automated Malware Similarity Analysis. [Online] 2009 http://www.blackhat.com/presentations/bh-usa-09/RAYGOZA/BHUSA09-Raygoza-MalwareSimAnalysis-PAPER.pdf

2: Hořejšek, Michal. CZ.NIC spouští HaaS: honeypot as a service. [Online] 2018 https://www.root.cz/clanky/cz-nic-spousti-haas-honeypot-as-a-service/

3: Lock, Hun-Ya. Using IOC (Indicators of Compromise) in Malware Forensics. [Online] 2013 https://www.sans.org/reading-room/whitepapers/forensics/ioc-indicators-compromise-malware-forensics-34200

4: Security Update for Windows Hyper-V. [Online] 2016 https://docs.microsoft.com/en-us/security-updates/SecurityBulletins/2016/ms16-045

5: Keragala, Dilshan.Detecting Malware and Sandbox Evasion Techniques. [Online] 2016 https://www.sans.org/reading-room/whitepapers/forensics/detecting-malware-sandbox-evasion-techniques-3666

6: Puodzius, Cassius.UEFI malware: how to exploit a false sense of security. [Online] https://www.welivesecurity.com/2017/10/19/malware-firmware-exploit-sense-security/

7: Can I put Debian on my Raspberry Pi?. [Online] 2016 https://wiki.debian.org/RaspberryPi

8: Ferguson, P.Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. [Online] 2000 https://tools.ietf.org/html/bcp38

12 Přílohy

12.1 Seznam kryptografických otisků analyzovaného a podrobně zkoumaného malware

Pozn.: Otisky jsou generované nástrojem sha256sum. Jsou zde uvedeny pro jednoznačnou identifikaci vzorků a pro jejich případné dohledání.

Malware BitCoinMiner-AY [PUP]

939e9597bced04c62eb80ba48166b80d36cf27ed96cbea2e47f084158389dd6a

241262527d74f553518ac363ef32390c781bcaf28a7a30c8c0263543a5126792

df4e5961c5aa6a974c13a9313c1cd9104d26cb265b12fe5b76abed934a9dc908

60ca9318b716ea48f8034171eee0c35bfb8d4e49053c592996829212e80f279f

46e6891b93672a5d47ede67046870ea1437b7a75c648d14bb4b5f2f57364aa4d

455aea7f3dae5e531453103860e9f09960168a7bdd230423cbf4c9d93fd88d20

Malware ELF:Elknot-AE [Trj]

f1c4dbfbfeb8bfe0cfd4a7be32c956841dd0964353b0aa7f57c75752e071fef1

d9d0aad54e9d9c6c2699f54495be95494ef8ce9af21c0694fbe8dab63d471950

69fa3f67a04cf6e1277233e62336058624bd9e86b6dd20ac0e6afe6a0b656267

5d1d59da125af31eafa5e47519f010c65546edcdb1d201027028ab98b16cb7d8

1fbe991d9cbffefaae9057763748987def5247b91d39588ad1adb3ed629c4485

d7a2109dbbd7e77ab4823debc94ab79fdbc3258e175d8d2949ffb707954723db

cf26a45031face3997e08f3f73aa80ebc698e056609969a07de8c614d3b88bee

9ce16a5c0214fea482a1bb64365fae740f33df1126d4f8c6ff7ce5d7a8279f4a

8d07a6671508882ae6e88da9a0e1d8ea8dccb3d93ed7082447a55d59a54e0521

4c8567eb7eea9543dc6ca547255cb0eea9d6cd1af6dfb157713323049f594b64

45b3a512f7694c42377d9e173e755e11096532258f954f2e6f565ada82dd45d6

41cee2745feb202f317a83bdb262dc503605758ab68f58f9aff31f29429ba0eb

31bb62317ad2ed735155dba83d7ec24d2390e832cf0c668abd31bd58dea11ab6

2b8d2f4af668819674884af2ae9135fe0b90326b0f478c92ff5601c8739fdcf7

15c846a83fff865c3d412cb7591cbbe5685c266013159524d8953db47bb00c55

08d44093b6a7afd0c3a35f09123ad35045dc808ad2c91380941fb79c2a9a985d

05dfe14e93f226fec17e3e565e592bad561db83ec39d7c560cd5488db38147bf

0463517eb99ff02e401e2afab1fc2d5f73e6a2f1efd60ca32c30e0b3f1a8836a

Malware ELF:Aesddos-H [Trj]

c6979a457c105f8ddb3d73ab4fb2c8930aba700fb8ef1f12aa3f06e1487d1e61

fa12d3741f7b3ef6e7df282f0c4f2e1900e0e421a085a78a2c06b13db391a6df

ec5a6f4a1948b64bab84494eee6b22bd309761ba457fa76115e6e3e3df90da45

d1fd6ffca9d43add5a7c0232c3316d3850bcbb8eb0ffcb64c0c988e0837bae04

904c4d085b5be5154e88feb7c2d8e3cd957424e960ab315fee98a95b763616bd

7b3368c569c4a6324737782e620a004c93205eab99e8b15d87a3c2e42a4bda8c

611aa2e044a560a73ee2d2b1b92a4ee4b6d3ce5c7e138e836cafab5f6f52391c

30e04acca10f729fd5060978509371400c66bc459e4f6faae1f6c9fd72f2aebf

Malware ELF:Elknot-BT [Cryp]

dec3b78a9898c7937268d5afa79e83a51ae009e075ae4faeac6af782804cea34

37f43bbc323abb2460a4bf23ee356ebee8d0ab6807db472d633ee18919341361

12.2 Zaznamenané komunikace

Příklad snahy Malware BitCoinMiner-AY [PUP] o navázání komunikace – neúspěšně.

0.014826	10.0.3.100	43358	163.17.30.212	8525	TCP	43358 → 8525 [SYN] Seq=0
5.010215	10.0.3.100	40634	113.59.33.59	1488	TCP	40634 → 1488 [SYN] Seq=0
10.018990	10.0.3.100	38348	58.99.32.135	4545	TCP	38348 → 4545 [SYN] Seq=0
15.027854	10.0.3.100	44496	125.211.202.186	3142	TCP	44496 → 3142 [SYN] Seq=0
15.392474	125.211.202.186	3142	10.0.3.100	44496	TCP	3142 → 44496 [RST, ACK] S
15.418570	10.0.3.100	50240	222.43.116.233	2258	TCP	50240 → 2258 [SYN] Seq=0
20.426500	10.0.3.100	34884	221.204.214.158	7635	TCP	34884 → 7635 [SYN] Seq=0
20.763874	221.204.214.158	7635	10.0.3.100	34884	TCP	7635 → 34884 [RST, ACK] S
20.784589	10.0.3.100	55652	218.211.90.199	8434	TCP	55652 → 8434 [SYN] Seq=0
25.792899	10.0.3.100	35204	198.2.199.236	8635	TCP	35204 → 8635 [SYN] Seq=0
25.948643	198.2.199.236	8635	10.0.3.100	35204	TCP	8635 → 35204 [RST, ACK] S
25.963300	10.0.3.100	58536	124.251.33.242	3027	TCP	58536 → 3027 [SYN] Seq=0
31.028404	10.0.3.100	59226	10.0.3.3	53	DNS	Standard query 0x7756 A s
31.542849	10.0.3.3	53	10.0.3.100	59226	DNS	Standard query response 0
32.600959	10.0.3.100	43376	163.17.30.212	8525	TCP	43376 → 8525 [SYN] Seq=0
37.617547	10.0.3.100	40652	113.59.33.59	1488	TCP	40652 → 1488 [SYN] Seq=0
42.618137	10.0.3.100	38366	58.99.32.135	4545	TCP	38366 → 4545 [SYN] Seq=0
47.626797	10.0.3.100	44514	125.211.202.186	3142	TCP	44514 → 3142 [SYN] Seq=0
47.965610	125.211.202.186	3142	10.0.3.100	44514	TCP	3142 → 44514 [RST, ACK] S
47.977344	10.0.3.100	50258	222.43.116.233	2258	TCP	50258 → 2258 [SYN] Seq=0
52.985304	10.0.3.100	34902	221.204.214.158	7635	TCP	34902 → 7635 [SYN] Seq=0
53.334210	221.204.214.158	7635	10.0.3.100	34902	TCP	7635 → 34902 [RST, ACK] S
53.348092	10.0.3.100	55670	218.211.90.199	8434	TCP	55670 → 8434 [SYN] Seq=0
58.356920	10.0.3.100	35222	198.2.199.236	8635	TCP	35222 → 8635 [SYN] Seq=0
58.511132	198.2.199.236	8635	10.0.3.100	35222	TCP	8635 → 35222 [RST, ACK] S
58.527266	10.0.3.100	58554	124.251.33.242	3027	TCP	58554 → 3027 [SYN] Seq=0
58.880488	10.0.3.2	58554	10.0.3.100	3027	ICMP	Destination unreachable (
58.897991	10.0.3.100	50518	10.0.3.3	53	DNS	Standard query 0xf6cc A s
58.899384	10.0.3.3	53	10.0.3.100	50518	DNS	Standard query response 0
59.908363	10.0.3.100	43394	163.17.30.212	8525	TCP	43394 → 8525 [SYN] Seq=0
64.916650	10.0.3.100	40670	113.59.33.59	1488	TCP	40670 → 1488 [SYN] Seq=0
69.925127	10.0.3.100	38384	58.99.32.135	4545	TCP	38384 → 4545 [SYN] Seq=0
74.934198	10.0.3.100	44532	125.211.202.186	3142	TCP	44532 → 3142 [SYN] Seq=0
75.279874	125.211.202.186	3142	10.0.3.100	44532	TCP	3142 → 44532 [RST, ACK] S
75.287472	10.0.3.100	50276	222.43.116.233	2258	TCP	50276 → 2258 [SYN] Seq=0
80.295169	10.0.3.100	34920	221.204.214.158	7635	TCP	34920 → 7635 [SYN] Seq=0
80.651049	221.204.214.158	7635	10.0.3.100	34920	TCP	7635 → 34920 [RST, ACK] S
80.660709	10.0.3.100	55688	218.211.90.199	8434	TCP	55688 → 8434 [SYN] Seq=0
85.668969	10.0.3.100	35240	198.2.199.236	8635	TCP	35240 → 8635 [SYN] Seq=0
85.823068	198.2.199.236	8635	10.0.3.100	35240	TCP	8635 → 35240 [RST, ACK] S
85.843128	10.0.3.100	58572	124.251.33.242	3027	TCP	58572 → 3027 [SYN] Seq=0
90.850649	10.0.3.100	34618	10.0.3.3	53	DNS	Standard query 0x3c14 A s
90.852547	10.0.3.3	53	10.0.3.100	34618	DNS	Standard query response 0
91.863350	10.0.3.100	43412	163.17.30.212	8525	TCP	43412 → 8525 [SYN] Seq=0
93.432029	10.0.3.2	58572	10.0.3.100	3027	ICMP	Destination unreachable (
96.871348	10.0.3.100	40688	113.59.33.59	1488	TCP	40688 → 1488 [SYN] Seq=0

Příklad snahy Malware ELF:Elknot-AE [Trj] o navázání komunikace – neúspěšně.

```
0.003380   10.0.3.100 43302 10.0.3.3     53 DNS Standard query 0xb6ec A www.weipai87.cn 75
1.120843     10.0.3.3 53 10.0.3.100   43302 DNS Standard query response 0xb6ec A www.weipai87.cn A 123.58.2.230 91
1.217772   10.0.3.100 45822 123.58.2.230 25000 TCP 45822 → 25000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
1.684556 123.58.2.230 25000 10.0.3.100   45822 TCP 25000 → 45822 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 60
1.716082   10.0.3.100 45824 123.58.2.230 25000 TCP 45824 → 25000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
2.140500 123.58.2.230 25000 10.0.3.100   45824 TCP 25000 → 45824 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 60
2.153612   10.0.3.100 45826 123.58.2.230 25000 TCP 45826 → 25000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
3.590309 123.58.2.230 25000 10.0.3.100   45826 TCP 25000 → 45826 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 60
3.607061   10.0.3.100 45828 123.58.2.230 25000 TCP 45828 → 25000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
4.048753 123.58.2.230 25000 10.0.3.100   45828 TCP 25000 → 45828 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 60
4.069101   10.0.3.100 45830 123.58.2.230 25000 TCP 45830 → 25000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
4.516606 123.58.2.230 25000 10.0.3.100   45830 TCP 25000 → 45830 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 60
4.524603   10.0.3.100 45832 123.58.2.230 25000 TCP 45832 → 25000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
4.942233 123.58.2.230 25000 10.0.3.100   45832 TCP 25000 → 45832 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 60
4.953721   10.0.3.100 45834 123.58.2.230 25000 TCP 45834 → 25000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
5.388365 123.58.2.230 25000 10.0.3.100   45834 TCP 25000 → 45834 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 60
5.393547   10.0.3.100 45836 123.58.2.230 25000 TCP 45836 → 25000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
5.820303 123.58.2.230 25000 10.0.3.100   45836 TCP 25000 → 45836 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 60
5.839119   10.0.3.100 45838 123.58.2.230 25000 TCP 45838 → 25000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
```

Další neúspěšný příklad:

```
0.011847   10.0.3.100 44172 139.196.38.5 7288 TCP 44172 → 7288 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
```

Příklad úspěšné komunikace:

```
0.356493 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 60
0.376674   10.0.3.100 33544 123.129.217.153 25000 TCP 33544 → 25000 [PSH, ACK] Seq=1 Ack=1 Win=29200 Len=123 177
0.376837 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [ACK] Seq=1 Ack=124 Win=65535 Len=0 60
0.719815 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [PSH, ACK] Seq=1 Ack=124 Win=65535 Len=20 74
0.763134   10.0.3.100 33544 123.129.217.153 25000 TCP 33544 → 25000 [ACK] Seq=124 Ack=21 Win=29200 Len=0 60
0.773504   10.0.3.100 33544 123.129.217.153 25000 TCP 33544 → 25000 [PSH, ACK] Seq=124 Ack=21 Win=29200 Len=40 94
0.773838 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [ACK] Seq=21 Ack=164 Win=65535 Len=0 60
1.425832 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [PSH, ACK] Seq=21 Ack=164 Win=65535 Len=8 62
1.428792   10.0.3.100 33544 123.129.217.153 25000 TCP 33544 → 25000 [ACK] Seq=164 Ack=29 Win=29200 Len=0 60
1.439717   10.0.3.100 33544 123.129.217.153 25000 TCP 33544 → 25000 [PSH, ACK] Seq=164 Ack=29 Win=29200 Len=40 94
1.440169 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [ACK] Seq=29 Ack=204 Win=65535 Len=0 60
2.425828 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [PSH, ACK] Seq=29 Ack=204 Win=65535 Len=8 62
2.428367   10.0.3.100 33544 123.129.217.153 25000 TCP 33544 → 25000 [PSH, ACK] Seq=204 Ack=37 Win=29200 Len=40 94
2.428596 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [ACK] Seq=37 Ack=244 Win=65535 Len=0 60
3.426133 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [PSH, ACK] Seq=37 Ack=244 Win=65535 Len=8 62
3.442846   10.0.3.100 33544 123.129.217.153 25000 TCP 33544 → 25000 [PSH, ACK] Seq=244 Ack=45 Win=29200 Len=40 94
3.443440 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [ACK] Seq=45 Ack=284 Win=65535 Len=0 60
4.425959 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [PSH, ACK] Seq=45 Ack=284 Win=65535 Len=8 62
4.437187   10.0.3.100 33544 123.129.217.153 25000 TCP 33544 → 25000 [PSH, ACK] Seq=284 Ack=53 Win=29200 Len=40 94
4.437342 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [ACK] Seq=53 Ack=324 Win=65535 Len=0 60
5.426000 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [PSH, ACK] Seq=53 Ack=324 Win=65535 Len=8 62
5.432365   10.0.3.100 33544 123.129.217.153 25000 TCP 33544 → 25000 [PSH, ACK] Seq=324 Ack=61 Win=29200 Len=40 94
5.432674 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [ACK] Seq=61 Ack=364 Win=65535 Len=0 60
6.426029 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [PSH, ACK] Seq=61 Ack=364 Win=65535 Len=8 62
6.429520   10.0.3.100 33544 123.129.217.153 25000 TCP 33544 → 25000 [PSH, ACK] Seq=364 Ack=69 Win=29200 Len=40 94
6.429769 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [ACK] Seq=69 Ack=404 Win=65535 Len=0 60
7.426086 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [PSH, ACK] Seq=69 Ack=404 Win=65535 Len=8 62
7.430078   10.0.3.100 33544 123.129.217.153 25000 TCP 33544 → 25000 [PSH, ACK] Seq=404 Ack=77 Win=29200 Len=40 94
7.430529 123.129.217.153 25000 10.0.3.100   33544 TCP 25000 → 33544 [ACK] Seq=77 Ack=444 Win=65535 Len=0 60
```

Obsah úspěšné komunikace (červeně klient, modře C&C server):

```
00000000  01 00 00 00 73 00 00 00  00 f4 01 00 00 32 00 00   ....s... .....2..
00000010  00 e8 03 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000020  00 00 01 01 00 00 00 00  01 00 00 00 0a 00 03 64   ........ .......d
00000030  0a 00 03 64 0a 00 03 64  0a 00 03 64 0a 00 03 64   ...d...d ...d...d
00000040  ff ff 01 00 00 00 00 00  2d 3d 3d 20 4c 6f 76 65   ........ -== Love
00000050  20 41 56 20 3d 3d 2d 3a  00 02 00 00 00 00 00 00    AV ==-: ........
00000060  00 b2 01 00 00 4c 69 6e  75 78 20 34 2e 39 2e 35   .....Lin ux 4.9.5
00000070  39 2b 00 31 3a 47 32 2e  34 30 00                  9+.1:G2. 40.
    00000000  08 00 00 00 0c 00 00 00  00 00 00 00 00 00 00 00   ........ ........
    00000010  e8 fd 00 00                                       ....
0000007B  02 00 00 00 20 00 00 00  01 00 00 00 00 00 00 00   .... ... ........
0000008B  00 00 00 00 00 00 10 00  00 00 02 01 00 00 00 00   ........ ........
0000009B  00 00 00 00 00 00 00 00                           ........
    00000014  04 00 00 00 00 00 00 00                           ........
000000A3  02 00 00 00 20 00 00 00  01 00 00 00 00 00 00 00   .... ... ........
000000B3  00 00 00 00 00 00 10 00  00 00 02 01 3a 00 00 00   ........ ....:...
000000C3  62 02 00 00 00 00 00 00                          b.......
    0000001C  04 00 00 00 00 00 00 00                           ........
000000CB  02 00 00 00 20 00 00 00  01 00 00 00 00 00 00 00   .... ... ........
000000DB  00 00 00 00 00 00 10 00  00 00 02 01 05 00 00 00   ........ ........
000000EB  06 01 00 00 00 00 00 00                           ........
    00000024  04 00 00 00 00 00 00 00                           ........
000000F3  02 00 00 00 20 00 00 00  01 00 00 00 00 00 00 00   .... ... ........
00000103  00 00 00 00 00 00 10 00  00 00 02 01 02 00 00 00   ........ ........
00000113  c7 00 00 00 00 00 00 00                           ........
    0000002C  04 00 00 00 00 00 00 00                           ........
0000011B  02 00 00 00 20 00 00 00  01 00 00 00 00 00 00 00   .... ... ........
0000012B  00 00 00 00 00 00 10 00  00 00 02 01 01 00 00 00   ........ ........
0000013B  c6 00 00 00 00 00 00 00                           ........
    00000034  04 00 00 00 00 00 00 00                           ........
00000143  02 00 00 00 20 00 00 00  01 00 00 00 00 00 00 00   .... ... ........
00000153  00 00 00 00 00 00 10 00  00 00 02 01 02 00 00 00   ........ ........
00000163  c6 00 00 00 00 00 00 00                           ........
    0000003C  04 00 00 00 00 00 00 00                           ........
0000016B  02 00 00 00 20 00 00 00  01 00 00 00 00 00 00 00   .... ... ........
0000017B  00 00 00 00 00 00 10 00  00 00 02 01 01 00 00 00   ........ ........
0000018B  c6 00 00 00 00 00 00 00                           ........
    00000044  04 00 00 00 00 00 00 00                           ........
```

Příklad úspěšné komunikace Malware ELF:Aesddos-H [Trj]:

```
 0.010139   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
 0.343530 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 6
 0.345212   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [ACK] Seq=1 Ack=1 Win=29200 Len=0 60
 0.371544   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [PSH, ACK] Seq=1 Ack=1 Win=29200 Len=1024 1078
 0.371893 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [ACK] Seq=1 Ack=1025 Win=65535 Len=0 60
 0.752130   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [PSH, ACK] Seq=1025 Ack=1 Win=29200 Len=20 74
 0.752668 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [ACK] Seq=1 Ack=1045 Win=65535 Len=0 60
 2.775030   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [PSH, ACK] Seq=1045 Ack=1 Win=29200 Len=20 74
 2.775463 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [ACK] Seq=1 Ack=1065 Win=65535 Len=0 60
 4.327268 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [PSH, ACK] Seq=1 Ack=1065 Win=65535 Len=413 467
 4.340474   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [ACK] Seq=1065 Ack=414 Win=30016 Len=0 60
 4.792462   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [PSH, ACK] Seq=1065 Ack=414 Win=30016 Len=20 74
 4.793311 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [ACK] Seq=414 Ack=1085 Win=65535 Len=0 60
 5.470874 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [PSH, ACK] Seq=414 Ack=1085 Win=65535 Len=2 60
 5.476849   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [ACK] Seq=1085 Ack=416 Win=30016 Len=0 60
 6.816771   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [PSH, ACK] Seq=1085 Ack=416 Win=30016 Len=20 74
 6.817197 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [ACK] Seq=416 Ack=1105 Win=65535 Len=0 60
 8.824936   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [PSH, ACK] Seq=1105 Ack=416 Win=30016 Len=20 74
 8.825401 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [ACK] Seq=416 Ack=1125 Win=65535 Len=0 60
10.469078 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [PSH, ACK] Seq=416 Ack=1125 Win=65535 Len=2 60
10.470606   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [ACK] Seq=1125 Ack=418 Win=30016 Len=0 60
10.840442   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [PSH, ACK] Seq=1125 Ack=418 Win=30016 Len=20 74
10.840618 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [ACK] Seq=418 Ack=1145 Win=65535 Len=0 60
12.856709   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [PSH, ACK] Seq=1145 Ack=418 Win=30016 Len=20 74
12.857288 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [ACK] Seq=418 Ack=1165 Win=65535 Len=0 60
14.873516   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [PSH, ACK] Seq=1165 Ack=418 Win=30016 Len=20 74
14.874571 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [ACK] Seq=418 Ack=1185 Win=65535 Len=0 60
15.485885 118.184.32.55 48080 10.0.3.100   39566 TCP 48080 → 39566 [PSH, ACK] Seq=418 Ack=1185 Win=65535 Len=2 60
15.493115   10.0.3.100 39566 118.184.32.55 48080 TCP 39566 → 48080 [ACK] Seq=1185 Ack=420 Win=30016 Len=0 60
```

Obsah úspěšné komunikace:

```
00000000  56 45 52 53 4f 4e 45 58  3a 4c 69 6e 75 78 2d 34   VERSONEX :Linux-4
00000010  2e 39 2e 35 39 2b 7c 30  7c 30 20 4d 48 7a 7c 34   .9.59+|0 |0 MHz|4
00000020  33 34 4d 42 7c 31 32 32  4d 42 7c 48 61 63 6b 65   34MB|122 MB|Hacke
00000030  72 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   r....... ........
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

-zkráceno-

```
000003E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000003F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000400  49 4e 46 4f 3a 30 2e 30  25 7c 30 2e 30 36 20 4d   INFO:0.0 %|0.06 M
00000410  62 70 73 00                                       bps.
00000414  49 4e 46 4f 3a 30 2e 38  25 7c 30 2e 30 39 20 4d   INFO:0.8 %|0.09 M
00000424  62 70 73 00                                       bps.
    00000000  07 00 00 00 80 5e 12 00  7c ec 0e 02 02 37 b6 71   .....^.. |....7.q
    00000010  00 00 00 00 e8 71 cd 00  00 00 00 00 e8 71 cd 00   .....q.. .....q..
    00000020  00 00 93 7c 01 00 00 00  08 eb 0e 02 08 eb 0e 02   ...|.... ........
    00000030  64 eb 0e 02 3e d9 95 7c  d8 ea 0e 02 01 00 00 00   d...>..| ........
    00000040  25 9e 95 7c d0 73 d1 00  b8 eb 0e 02 ad 9d 95 7c   %..|.s.. .......|
    00000050  78 07 d1 00 c9 9d 95 7c  41 00 00 00 d8 73 d1 00   x......| A....s..
    00000060  b4 5f 12 00 78 70 00 00  f8 23 15 00 20 01 00 00   ._..xp.. .#.. ...
    00000070  fe ae 00 7c 00 00 d1 00  04 e9 0e 02 63 6f 64 65   ...|.... ....code
    00000080  20 eb 0e 02 01 00 00 00  25 9e 95 7c e0 0f be 00    ....... %..|....
    00000090  00 ec 0e 02 ad 9d 95 7c  48 07 d1 00 c9 9d 95 7c   .......| H......|
```

Příklad úspěšné komunikace Malware ELF:Elknot-BT [Cryp]:

```
0.007082   10.0.3.100 45402 23.234.29.116 10991 TCP 45402 → 10991 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1
0.171586 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
0.186962   10.0.3.100 45402 23.234.29.116 10991 TCP 45402 → 10991 [PSH, ACK] Seq=1 Ack=1 Win=29200 Len=401 455
0.187404 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [ACK] Seq=1 Ack=402 Win=65535 Len=0 60
0.408936 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [PSH, ACK] Seq=1 Ack=402 Win=65535 Len=4 60
0.423199   10.0.3.100 45402 23.234.29.116 10991 TCP 45402 → 10991 [PSH, ACK] Seq=402 Ack=5 Win=29200 Len=27 81
0.423783 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [ACK] Seq=5 Ack=429 Win=65535 Len=0 60
1.909841 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [PSH, ACK] Seq=5 Ack=429 Win=65535 Len=4 60
1.924648   10.0.3.100 45402 23.234.29.116 10991 TCP 45402 → 10991 [PSH, ACK] Seq=429 Ack=9 Win=29200 Len=27 81
1.924993 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [ACK] Seq=9 Ack=456 Win=65535 Len=0 60
3.409284 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [PSH, ACK] Seq=9 Ack=456 Win=65535 Len=4 60
3.418984   10.0.3.100 45402 23.234.29.116 10991 TCP 45402 → 10991 [PSH, ACK] Seq=456 Ack=13 Win=29200 Len=27 81
3.419528 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [ACK] Seq=13 Ack=483 Win=65535 Len=0 60
4.909425 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [PSH, ACK] Seq=13 Ack=483 Win=65535 Len=4 60
4.917796   10.0.3.100 45402 23.234.29.116 10991 TCP 45402 → 10991 [PSH, ACK] Seq=483 Ack=17 Win=29200 Len=27 81
4.918105 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [ACK] Seq=17 Ack=510 Win=65535 Len=0 60
6.409501 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [PSH, ACK] Seq=17 Ack=510 Win=65535 Len=4 60
6.416251   10.0.3.100 45402 23.234.29.116 10991 TCP 45402 → 10991 [PSH, ACK] Seq=510 Ack=21 Win=29200 Len=27 81
6.417185 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [ACK] Seq=21 Ack=537 Win=65535 Len=0 60
7.910106 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [PSH, ACK] Seq=21 Ack=537 Win=65535 Len=4 60
7.911946   10.0.3.100 45402 23.234.29.116 10991 TCP 45402 → 10991 [PSH, ACK] Seq=537 Ack=25 Win=29200 Len=27 81
7.912841 23.234.29.116 10991 10.0.3.100   45402 TCP 10991 → 45402 [ACK] Seq=25 Ack=564 Win=65535 Len=0 60
```

Obsah úspěšné komunikace

```
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000010  00 4c 69 6e 75 78 20 34  2e 39 2e 35 39 2b 00 00   .Linux 4 .9.59+..
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
```

-zkráceno-

```
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000190  00                                                .
    00000000  04 00 00 00                                     ....
00000191  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000001A1  00 00 00 00 00 00 00 00  00 00 00                 ........ ...
    00000004  04 00 00 00                                     ....
000001AC  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000001BC  00 00 00 00 00 00 00 00  00 00 00                 ........ ...
```

```
000001E2  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
000001F2  00 00 00 01 00 00 00 00  00 00 00                 ........ ...
    00000010  04 00 00 00                                     ....
```

```
00000269  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00000279  00 00 00 02 00 00 00 00  00 00 00                 ........ ...
    00000024  04 00 00 00                                     ....
```

12.3 Vnitřní smyčka skriptu pro spuštění a zaznamenání chování malware

```
#!/bin/bash

cd $sourcepath
for malware in $(ls -p | grep -v / | grep -v log)
do
          echo "$(date +%X) Preparing malware sample $samplenum"
          echo $(date +%X)" Sample architecture: "$(file -b $malware | \
         sed -e 's|.*executable, ||g' | sed -e 's/,.*//g')

          scp $malware raspi:/tmp
          type=$(file -b "$malware" | sed -e "s/,.*//g" | sed -e "s/ /_/g" | sed -e "s|/||g")
          workingDir="${savetopath}${samplenum}-$type"
          mkdir $workingDir
          cp $sourcepath/$malware $workingDir
          ssh kali "tcpdump -U -s0 -i eth3 -w - 'not (tcp port 22 and \
            host 10.0.3.100 and host 10.0.3.17)' 2>/dev/null" | \
         tee $workingDir/dump.pcap > /tmp/tshark 2>/dev/null &
          wpid=$!
          tpid=$(jobs -p)

          echo $(date +%X)' All set. Detonating now!'
          ssh raspi "chmod +x /tmp/$malware && /tmp/$malware" 2>  $workingDir/error.log &
          sleep 5
          skip=0
          rm $malware
          sleep 55
          ssh raspi "netstat -tunp & cat /proc/loadavg" > $workingDir/netstat.txt

          sleep 61
          echo $(date +%X)' Going for reboot..'

          ssh raspi "netstat -tunp & cat /proc/loadavg" > $workingDir/netstat2.txt
2>/dev/null
          ssh raspi "reboot" 2>/dev/null &

          ssh kali 'while [[ "$(ifconfig ethusb | head -n 1 | grep RUNNING)" ]]; \
                    do sleep 0.2; \
                   done'
          echo -e $(date +%X)' \e[1;32mReboot command succesfull\e[0m'
          kill $tpid $wpid
          wait $tpid $wpid 2>/dev/null
          echo $(date +%X)' Waiting for boot to finish...'
          ssh kali 'while [[ ! "$(ping 10.0.3.100 -c1 | grep 'from')" ]]; \
                    do sleep 0.2; \
                   done'
          sleep 2
          hash=$(ssh raspi "sha256sum /dev/mmcblk0p1")
          if [[ "$hash" == \
         'efcede0460817813b7bdd7ba3981bb52c7cf6d35abc9b672f8c0a02796f6837b  /dev/mmcblk0p1' \
         ]]; then
                    echo -e $(date +%X)' \e[1;32mBoot OK\e[0m'
```

```
          else
                    echo -e $(date +%X)' \e[1;31m!!!!!!!!!!!!  Boot Corrupted
!!!!!!!!!!!!!\e[0m'
                    read
          fi
        # ((samplenum++)) nelze použít - value too great for base (error token is "008")
          samplenum=$(( 10#$samplenum + 1 ))
          samplenum=$(printf "%03d" $(( 10#$samplenum )) )
done
```