

Implementace aktivních protiopatření pro honeypoty

Implementation of Active Countermeasures for Honeypots

Bc. Jan Bobák

Diplomová práce
2014



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Jan Bobák**
Osobní číslo: **A12455**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **kombinovaná**

Téma práce: **Implementace aktivních protiopatření pro honeypoty**

Zásady pro vypracování:

1. Navrhněte možnosti nasazení prvků pro detekci a lokalizaci útočníků v IT infrastruktuře s využitím honeypotů.
2. Implementujte firewall a provedte vhodnou konfiguraci.
3. Charakterizujte varianty útoků.
4. Analyzujte možnosti získání informací o útoku a provedte test.
5. Vytvořte protiopatření a popište možnosti odvrácení útoku.
6. Ověřte zda navržené systémy fungují.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Linux: dokumentační projekt. 3. aktualiz. vyd. Brno: Computer Press, 2003, xviii, 1001 s. ISBN 8072267612.**
2. **Bezpečností linuxová distribuce s vestavěným honeypotem. Zlín, 2013. Diplomová práce. Univerzita Tomáše Bati. Vedoucí práce Ing. David Malaník, Ph.D.**
3. **LINUXEXPRES: Správa linuxového serveru [online]. 2014 [cit. 2014-02-02]. ISSN 1801-3996. Dostupné z: <http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru>**
4. **BeEF: The browser exploitation framework project [online]. [cit. 2014-02-03]. Dostupné z: <http://beefproject.com/>**
5. **KRČMÁŘ, Petr. Linux: tipy a triky pro bezpečnost. 1. vyd. Praha: Grada, 2004. ISBN 8024708124.**
6. **SPITZNER, Lance. Honeypots: Definitions and Value of Honeypots. Honeypots: Tracking Hackers [online]. 2003 [cit. 2014-02-03]. Dostupné z: <http://www.tracking-hackers.com/papers/honeypots.html>**
7. **MEČÍŘ, Michal. Použití honeypotů pro monitorování útoků. Brno, 2009. Bakalářská práce. Masarykova univerzita. Vedoucí práce Mgr. Daniel Kouřil.**
8. **HATCH, Brian, George KURTZ a James LEE. Linux hackerské útoky: bezpečnost Linuxu – tajemství a řešení. Praha: SoftPress, 2002, 576 s. ISBN 8086497178.**

Vedoucí diplomové práce:

Ing. David Malaník, Ph.D.

Ústav informatiky a umělé inteligence

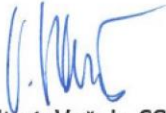
Datum zadání diplomové práce:

21. února 2014

Termín odevzdání diplomové práce:

20. května 2014

Ve Zlíně dne 21. února 2014


prof. Ing. Vladimír Vašek, CSc.
děkan




doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Diplomová práce se zabývá implementací aktivního protiopatření do honeypotu. V teoretické části se práce zabývá návrhem firewallu, popisuje různé formy útoku na server a uvádí jednotlivé druhy útočníků, kteří jsou za tyto útoky z velké části zodpovědní. Teoretická část se dále zabývá základním popisem honeypotů a vhodnosti jejich použití v praxi. Na základě návrhu a vzhledem k požadavkům, je vytvořen s použitím nástroje iptables firewall, jež je nasazen mezi jednotlivé segmenty sítě. Dále jsou do honeypotu implementovány nástroje, které umožní snadnější vystopování útočníka a zároveň znemožní jeho další přístup. Popis procesu tvorby je uveden v praktické části. V závěru je proveden test na odolnost proti průniku a kontrola shromážděných dat.

Klíčová slova: Linux, honeypot, firewall, iptables, Debian, VMware, BeEF

ABSTRACT

This thesis deals with the implementation of active countermeasures to the honeypot. In the theoretical part of the thesis deals with the design of the firewall, describes the various forms of attack on the server and lists the various types of attackers who are responsible for those attacks. The theoretical part also deals with the basic description of honeypots and their suitability in practice. The proposal with the basic requirements is created by iptables firewall tool, which is then attached to each network segment. In addition to honeypots are implemented tools that allow for easier tracking down the attacker and also prevents its further access. Description of the process of creation is given in the practical part. At the end is made the test for resistance to penetration and for control of the data collected.

Keywords: Linux, honeypot, firewall, iptables, Debian, VMware, BeEF

Rád bych poděkoval Ing. Davidu Malaníkovi, PhD. za vedení práce, cenné rady a vstřícnost při konzultacích.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 HONEYPOTY	11
1.1 ROZDĚLENÍ HONEYPOTŮ	11
1.1.1 Podle míry interakce	11
1.1.2 Podle účelu využití	12
1.2 UMÍSTĚNÍ HONEYPOTŮ	12
2 BEZPEČNOST LINUXOVÝCH SERVERŮ	14
2.1 TYPY ÚTOKU	14
2.1.1 Fyzické vniknutí	14
2.1.2 Vnější útok	14
2.1.3 Vnitřní útok	15
2.1.4 Cílené a automatizované útoky	16
2.2 METODY ÚTOKŮ	16
2.2.1 Odepření služby	16
2.2.2 Převzetí totožnosti	19
2.2.3 Útoky na webové aplikace	21
2.3 HACKEŘI	23
2.3.1 Typy útočníků	23
2.3.2 Motivace hackerů	25
3 BEEF (BROWSER EXPLOITATION FRAMEWORK)	27
4 FIREWALL	30
4.1 DEFINICE FIREWALLU	30
4.2 POLITIKY FIREWALLU	30
4.3 TYPY FIREWALLU	30
4.4 NÁSTROJE PRO FILTROVÁNÍ PAKETŮ	31
4.5 IPTABLES	32
4.6 NÁVRH FIREWALLU	35
4.6.1 Typ distribuce a její důležité vlastnosti	35
4.6.2 Požadavky na firewall	36
4.6.3 Umístění firewallu v síti	38
II PRAKTICKÁ ČÁST	39
5 NAsAZENÍ FIREWALLU	40
5.1 VOLBA PRACOVNÍHO PROSTŘEDÍ	40
5.2 VOLBA JÁDRA A INSTALACE	42
5.3 ZABEZPEČENÍ FIREWALLU	42
5.3.1 Aktualizace	42
5.3.2 Kontrola síťových služeb	43
5.3.3 Autentizace	44
5.3.4 Vzdálený přístup – SSH	45

5.4	KONFIGURACE SÍŤOVÝCH ROZHRANÍ.....	47
5.5	SESTAVENÍ SKRIPTU IPTABLES.....	47
5.5.1	Parametry jádra	47
5.5.2	Moduly jádra	48
5.5.3	Deklarace proměnných.....	48
5.5.4	Zavedení pravidel iptables	49
5.5.5	Integrace firewallu do systému	53
5.6	TESTOVÁNÍ KONFIGURACE FIREWALLU	54
5.6.1	Testování, zda zařízení komunikují skrz firewall a připojení k internetu.....	54
5.6.2	Testování ping of death.....	56
6	HONEYPOT A IMPLEMENTACE BROWSER EXPLOITATION FRAMEWORKU	57
6.1	METODIKA	57
6.2	HONEYPOT	58
6.2.1	Výběr distribuce a honeypotu	58
6.2.2	Instalace distribuce ANANSI.....	59
6.2.3	Logování událostí.....	59
6.3	BEEF	60
6.3.1	Instalace.....	60
6.3.2	Spuštění a konfigurace	62
6.3.3	Hlavní nabídka	63
6.3.4	Útoky pomocí BeEF.....	65
6.3.4.1	Konfigurace modulů	65
6.3.4.2	Výběr modulů	68
7	ODVRÁCENÍ ÚTOKU.....	70
8	TEST ÚTOKU NA HONEYPOT	72
9	ZÍSKANÉ INFORMACE.....	75
9.1	PŘÍCHOD ÚTOČNÍKA	75
9.2	VÝSLEDKY NASAZENÝCH MODULŮ	76
9.3	DÍLČÍ ZÁVĚR TESTOVÁNÍ AKTIVNÍHO PROTIOPATŘENÍ BEEF.....	78
	ZÁVĚR	79
	CONCLUSION	80
	SEZNAM POUŽITÉ LITERATURY.....	81
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	82
	SEZNAM OBRÁZKŮ	84
	SEZNAM TABULEK.....	85
	SEZNAM PŘÍLOH.....	86

ÚVOD

S vývojem nových technologií přichází i nové zbraně pro Internetový svět. Ty neustále ohrožují společnost, a to s čím dál větší intenzitou a nemilosrdnou chutí vše ovládat nebo ničit. Abychom byli připraveni na takovou míru útoků, musíme se podílet na vývoji obranných systémů, jenž tyto zbraně zničí dříve, než ovládnou celý svět a stanou se tak nezastavitelnými.

Ochrana před různými druhy útoku zvyšuje bezpečnost na Internetu. Avšak samotný antivirus nebo včasné instalace aktualizací nejsou mnohdy bohužel dostačující. A proto je vhodné do každé sítě implementovat správně nastavený firewall, který tato rizika ještě více minimalizuje. Tento filtr zkontroluje informace přicházející z vnější sítě a buď jim přístup povolí nebo je zamítne. To se děje dle konfigurace pravidel administrátorem, jenž takto činí individuálně, přímo na míru konkrétní sítě.

Jako další bezpečnostní prvek, a to nejen v podnikových sítích, lze zařadit do sítě tzv. honeypot. Jedná se o informační systém, jehož smyslem je lákat potencionální narušitele. Má taktéž za úkol shromažďovat informace o jejich činech a sledovat stopy. To vše probíhá efektivně a automatizovaně s dalšími možnostmi využití, jako je reporting, testování či samotná analýza dat.

Nejen malware, jako jsou viry a červy, znepříjemňují uživatelům Internetu jejich životy, ale hlavně samotní lidé (se zlými úmysly), mohou způsobit velké škody. Zkušení hackeři jsou největší hrozbou Internetu a společnost musí být připravena čelit jejich útoku. Proto je vhodné soustředit se nejen na bezpečnost, ale zároveň implementovat do sítě další opatření, která pomohou například útočníka lokalizovat nebo získat o případném napadení ještě více informací.

Výše popsanými aspekty se zabývá tato práce. Jejími dílčími cíli jsou důvody použití honeypotu, integrace aktivních protiopatření, implementace firewallu a výsledek shromažďování informací v praxi.

I. TEORETICKÁ ČÁST

1 HONEYPOTY

Obecně lze honeypoty definovat jako prvky v síti nebo v počítačovém systému, jež mohou fungovat jako dedikované servery a shromažďovat informace o nežádoucích narušitelích. V zásadě se jedná o informační systém, který detekuje neoprávněný průnik či pokus o útok.

Funguje jako prostředek, jež láká útočníky na své simulované služby a zaznamenává útok a jeho rozsah. Jelikož se běžným způsobem na síti nezobrazuje, nesměřuje na něj žádný provoz, a lze tak vycházet z faktu, že každá interakce s tímto prvkem, je s největší pravděpodobností nebezpečná aktivita a jedná se o neoprávněný přístup. V zásadě nedokáže nahradit tradiční bezpečnostní systém, ale díky své funkci ho kvalitně doplní.

1.1 Rozdělení honeypotů

Základní členění je nutné specifikovat zejména pro zjednodušení výběru honeypotu pro specifické nasazení v síti. Převážně se tedy dělí dle míry interakce s útočníkem a dle účelu použití.

1.1.1 Podle míry interakce

Vzhledem k velkému rozsahu možnosti využití honeypotů je obtížné získat přehled nad všemi druhy honeypotu, a proto se rozdělují do dvou obecných kategorií dle interakce s útočníkem: nízká-interakce a vysoká-interakce. Interakce definuje úroveň aktivity, jakou povolí honeypot útočnickovi.

Honeypoty s nízkou mírou interakce - systémy, které dokážou emulovat konkrétní protokoly na úrovni transportní vrstvy modelu TCP/IP (Transmission Control Protocol/Internet Protocol) a vytvoří dojem, že se útočník připojuje ke skutečnému systému. Předností honeypotu s nízkou interakcí je jeho jednoduchost v nasazení a údržba s minimálním rizikem reálných škod. Výhodou také je, že emulované služby snižují dopad napadení a útočník nemá nikdy přístup k operačnímu systému. Naopak nevýhodami mohou být horší detekce novějších typů útoků a také snadnější odhalení zkušeným útočníkem. Vždy záleží na nastavení dané služby a její omezení. Například služba FTP (File Transfer Protocol), která naslouchá na portu 21, může jen emulovat FTP přihlášení nebo podporovat řadu dalších příkazů.

Honeypoty s vysokou mírou interakce – skutečné systémy a aplikace, které mohou obsahovat reálné zranitelnosti. Jedná se o komplexní řešení, jež dokáže zachytit rozsáhlé množství informací a zvládne vytvářet předpoklady o chování útočníka při jeho dalších pokusech. Obecně platí, že honeypoty s vysokou mírou interakce mohou disponovat možnostmi jako honeypoty s nízkou mírou interakce, avšak umí toho mnohem více. Mají například schopnost detekovat nové typy útoků. Zachycují více informací, včetně nových nástrojů, komunikace nebo stisknutých kláves. Nevýhodou je ale zvýšené riziko, protože se jedná o reálný systém. A s tím se zvyšuje i údržba a složitost instalace nebo nasazení.[6]

1.1.2 Podle účelu využití

Podle účelu využití se dělí honeypoty na produkční a výzkumné.

Produkční – cílem těchto honeypotů (vystupujících jako prvky, jež zajišťují vyšší bezpečnost sítě) zpravidla bývá, odvést útočníka od reálných systémů a tím odvrátit hrozící nebezpečí. Jsou tedy umístěny do vnitřní sítě a zaměřují se na detekci nežádoucích vlivů z vnějšího prostředí. Většinou se jedná o hotová řešení, jež jsou jednodušší na nasazení a údržbu. Z toho důvodu se v produkční oblasti používají spíše honeypoty s nižší mírou interakce splňující tyto požadavky.

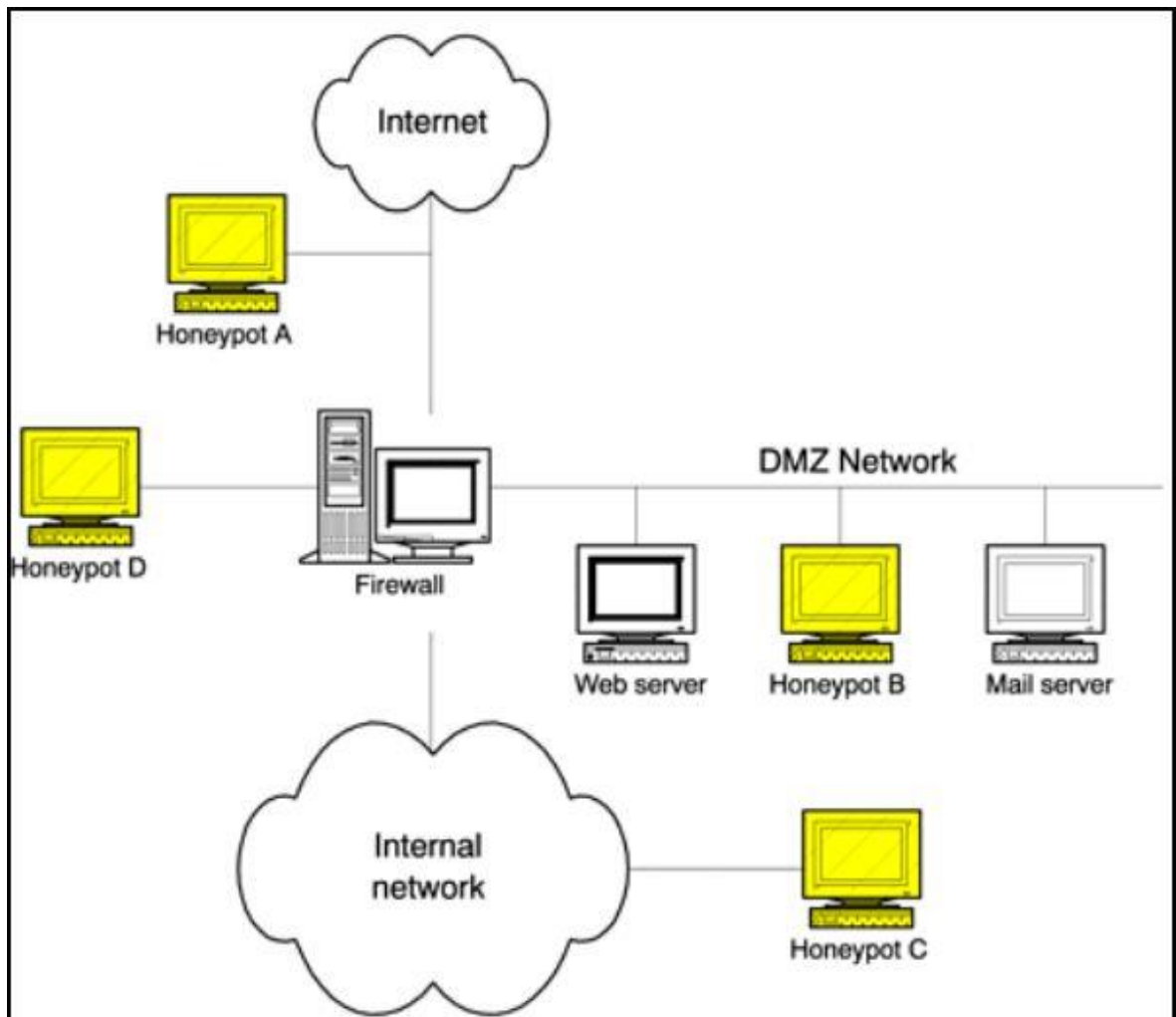
Výzkumné – pro získávání informací a následnou analýzu jsou vhodnější spíše výzkumné honeypoty, a to se střední až vysokou mírou interakce. Zaznamenávají více dat oproti produkčním a dokážou i analyzovat chování útočníka. Ať už se jedná o škodlivý malware nebo reverzní inženýrství, jež odhaluje princip obranného programu a zkouší vytvořit jeho fungující kopii. Na základě této analýzy se dále navrhnou náležitá bezpečnostní opatření.[7]

1.2 Umístění Honeypotů

Honeypot lze umístit v podstatě ve čtyřech hlavních lokacích. První z možností umístění je ve vnitřní síti LAN (Local Area Network) – za firewallem, druhá ve vnější síti před firewallem, třetí v tzv. Demilitarizované zóně – DMZ a konečně čtvrtá v Internetu. Všechny možnosti umístění znázorňuje *Obr. 1*. Můžeme použít najednou i více variant honeypotů pro přesnější sběr dat o průnicích a jejich detekcích.

Umístění na vnější straně může pouze zvýšit riziko. Narušitel by tak při interakci s honeypotem zvýšil povědomí o organizaci a to by mohlo přivábit více útočníků. Někteří odborníci naopak tvrdí, že by ho v případě napadení mohli identifikovat jako nasazený

honeypot, čímž by útočníka vyplašili. Samozřejmě je i možnost, že narušitel nepozná, co je platný systém a co honeypotu. Nicméně existují pochybnosti, které tvrdí, že například díky využití neznáme chyby v zabezpečení, může použít honeypot k útoku na ostatní relevantní účastníky sítě. Obecně ovšem platí, že tato lákadla, používaná pro prevenci proti útoku v organizaci, mají největší hodnotu v nasazení za obranu perimetru nebo ve vnitřních sítích.



Obr. 1 Umístění honeypotu [6]

Pokud je tedy kupříkladu honeypot umístěn ve vnitřní síti, je toto umístění realizováno za firewall či směrovač a pokud k němu chce útočník přistupovat, musí být nejprve schopen zvládnout všechny překážky, číhající na hraničních síťových prvcích. Ovšem v případě, že dojde ke kompromitaci honeypotu, hrozí i riziko napadení vnitřní sítě.[6]

2 BEZPEČNOST LINUXOVÝCH SERVERŮ

Následující kapitola demonstruje základní popis útoků na severy, které mohou nastat, a taktéž probírá problematiku hackerů, kteří tyto útoky provádějí.

2.1 Typy útoku

Z hlediska směru útoku lze uvažovat o třech hlavních typech útoků, a to: fyzickým přístupem k serverům (počítačům), připojením v síti, například prostřednictvím sítě LAN nebo nejčastěji používaném připojením přes Internet.

2.1.1 Fyzické vniknutí

V případě velkých společností (kam se lze dostat bez použití identifikačních čipů, kde nemají k dispozici přístupové či docházkové systémy, případně ostrahu v budově) je prakticky velmi jednoduché prostě vejít do firmy, posadit se k náhodnému klientskému počítači na síti a začít si připravovat systém pro budoucí vzdálené vniknutí. Dokonce i v malých společnostech není žádný problém si obléknout kombinézu a v přestrojení za technika kabelové společnosti, vniknout do serverové místnosti. Pak jen už stačí vytvořit nový uživatelský účet s právy administrátora a tento pak využít k páčání neřestí.

Takové přímé vniknutí může zrealizovat i hacker, který je už ve společnosti zaměstnancem. V ten moment je to samozřejmě problém, jelikož takový útočník již má platný účet na síti a zná všechny informace o síti a mnohdy i metody zabezpečení. Většinou se může jednat o špiona nebo frustrovaného zaměstnance, který do jisté chvíle zůstává bez povšimnutí jako časovaná bomba. Tuto hrozbu lze samozřejmě řešit kvalitním sledováním systému. Což s největší pravděpodobností nezabrání odcizení nebo zničení dat, ale lze tak zjistit, jak přesně ke krádeži nebo zničení došlo a z kterého účtu byl tento útok veden s následným obviněním. Nejúčinnější metoda proti takovému útoku je ovšem oznámení, že je sledováno vše, co jde dovnitř i ven z firmy. Poté by potenciální útočník své pokusy o hackování anuloval v momentě, kdy by dostal strach z možného odhalení.[11]

2.1.2 Vnější útok

Z hlediska obrany proti vnějšímu útoku by měly stačit zásadní bezpečnostní mechanismy běžících služeb. Avšak v tomto případě by musel být dokonalý administrátor, který by zamezil nestandardnímu chování jednotlivých služeb. Ale také v případě

neomylného a všeznalého správce není možné, vyloučit řadu chyb různých programů a bezpečnostních děr v systému. Bohužel v reálném světě jsou využívány další aplikace nebo dodatečné moduly, které tyto chyby rovněž obsahují. Příkladem může být webový server, který většinou funguje spíše jako proxy, což znamená, že přijme požadavek a předá jej interpretu příslušného programovacího jazyky (Java, PHP, atd.), v němž běží aplikace, která přijatý požadavek zpracuje. V daný moment tyto útoky nemusí být už jen na služby - jako je třeba SSH (Secure Shell) ale i na aplikace, které jsou přístupné prostřednictvím příslušné služby. Ty jsou v lepším případě buď konfigurovány administrátory anebo v horším případě, jako je třeba webhosting, samotnými uživateli. V ten moment ovšem nastává problém, jelikož ne každý uživatel dohlíží na aktuálnost softwaru a zajištění bezpečnostní díry. Dochází tak ke vzniku zastaralých programů, které mohou představovat pro daný server reálnou hrozbu.[3]

2.1.3 Vnitřní útok

Za vnitřní útok lze považovat jakýkoliv útok, který není vnější. Zaměstnanec firmy nebo zákazník, například hádáním hesla, uhodne přihlašovací údaje a získá práva „roota“ nebo získá přístup s využitím bezpečnostní trhliny v některé z běžících aplikací na serveru. V ten moment je server prakticky v moci útočníka a ten si s ním může libovolně modifikovat soubory systému dle své vůle. Svě počínání může i šikovně zamaskovat, třeba podvržením nástrojů jako je: ls, ps nebo i bash a zajistit, že některé soubory budou i účtům s právem roota uschovány.

Druhým nejhorším případem je zpravidla i přístup k uživatelskému Shellu bez práv roota, díky kterému lze taktéž vyvolat ničení některých služeb nebo pokračovat v průzkumu pro získání přístupu k citlivým údajům. Ve většině situací bude tedy i uživatelský účet dostačující pro hledání různých zranitelností a k neoprávněnému vstupu na zakázaná místa. K dalšímu útoku by mohl posloužit i přístup k interpretu programovacího jazyka. Ten mnohdy postačí k postupu někam dál, byť omezeně například k databázi nebo administrační části webového rozhraní webové aplikace. V daný moment, následkem porušení bezpečnosti služby, vystaví útoku i samotný server.

Obecně by se dalo říct, že oproti vnějším útokům jsou vnitřní daleko závažnější, jelikož ochránit vnitřní systém je mnohem složitější. Pokud se útočník dostane na server, všechny informace o systému, nainstalované nástroje, umístěná data, budou mu prakticky

k dispozici a to s přihlédnutím na fakt, že má možnost doinstalovat si další nástroje, které následně k cílenému útoku využije.

2.1.4 Cílené a automatizované útoky

Dnešní často nasazované hrozby na Internetu jsou roboti, kteří se automatizovaně snaží hledat běžící služby, skenovat sítě a zjišťovat další informace, podle toho jak byly naprogramováni. Poté se pak pokouší třeba prostřednictvím nějaké zranitelnosti na webovém serveru, získat přístup k Shellu nebo začlenit počítač do botnetu, což jim umožní například využít tento infikovaný systém, jako nástroj k rozesílání spamů nebo uskladňování nějakého nelegálního materiálu. Proti těmto relativně jednoduchým programům postačí běžná ochrana, jako je pravidelná aktualizace systému nebo dobře nastavená politika přístupových hesel.

Větší riziko tvoří cílené útoky, které se snaží získat přístup na konkrétní IP adresu nebo zahltit přímo nějakou službu požadavky, a to proto, že za tímto útokem s největší pravděpodobností stojí určitá osoba. Zde už může nastat větší problém, pokud útočník využije některé zranitelnosti a získá práva roota.[3]

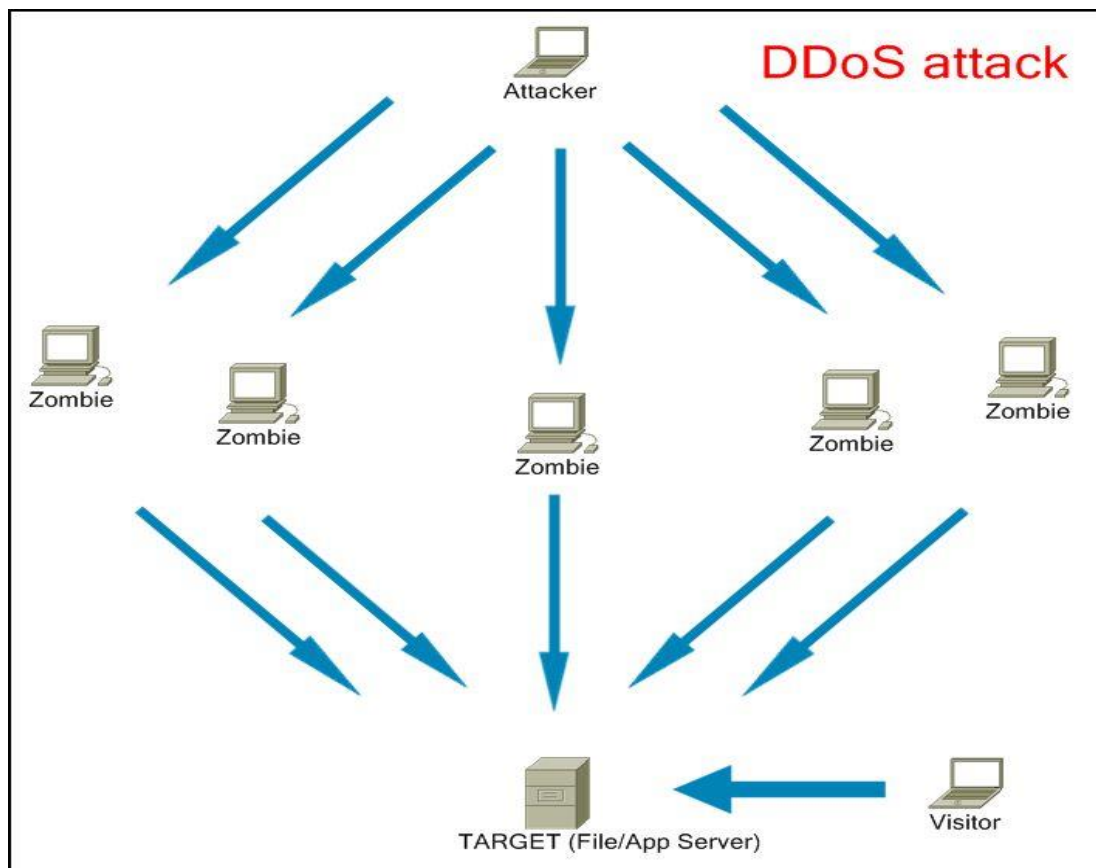
2.2 Metody útoků

2.2.1 Odepření služby

Odepření služby je bezesporu jedním z nejjednodušších a nejčastěji realizovaných útoků na síťové služby, které může způsobit nejen přerušení služby, ale i celkově znemožnit fungování serveru. Takové jednání je často využíváno pro vyvolání nepříjemnosti různým organizacím, mnohdy i většího rozsahu. Následky mohou působit několik minut, ale i hodin nebo dnů a mohou mít velký vliv na stabilitu sítě a integritu dat. K napadení počítače nebo jím poskytovaných služeb lze použít celou řadu metod prostřednictvím nejrozšířenějšího protokolu TCP/IP. Mezi útoky typu odepření služeb patří: ping of Death, útoky SYN, zaplavení ICMP (Internet Control Message Protocol), útoky na konkrétní služby, přesměrování DNS (Domain Name Service), nebo přesměrování toku dat v síti. Podrobnější vysvětlení bylo provedeno níže:

Ping Of Death – Historicky nejstarším a nestrašidelnějším útokem na síťové vrstvě, je útok „Ozvěna smrti“. Speciálně navržený ICMP paket s větší než maximální možnou velikostí, může způsobit havárii implementace TCP/IP vytvořené chybnou alokací paměti.

Ochranou proti úplnému zhroucení je konfigurace firewallu, ve kterém bude implicitně nastavena kontrola ICMP paketů nebo jejich úplné blokování. Při přesné specifikaci daného typu ICMP paketu – echo request, je výhodné tento paket povolit, avšak eliminovat možnosti zneužití přidáním limitů, které například zahodí paket, který je větší, než stanovené maximum.



Obr. 2 DDoS útok

Útok SYN a zaplavení ICMP – dalším typem útoku pro přetížení cílových služeb mohou být pokusy s požadavky na informace nebo o připojení s mírnou úpravou úvodního paketu, kterému je nastaven bit SYN. Zapracování falešné zprávy s příznakem SYN, zabere mnohem více času procesoru a operační paměti u příjemce, který tyto informace zaznamenává a následně vyhradí paměť pro údaje o připojení. Když je nadále přijímáno velké množství těchto paketů, nejsou již díky spotřebě veškeré kapacity paměti, přijaty pakety od legitimních uživatelů.

Velmi podobná situace může nastat i zaplavením ICMP paketů – typu request, jež jsou nepřetržitě zasílány na cílový počítač a ten stejně jako v předchozím případě, není schopen zaznamenávat běžný síťový provoz. Jako ochrana i v tomto případě poslouží opět

konfigurace pravidel ve stavovém firewallu, jež znemožní přijímat extrémně často vysílané pakety s příznakem SYN nebo zahodí nezvykle velké objemy ICMP paketů.

Útok na konkrétní služby – velmi častým důvodem útoku na server bývá i zájem na deaktivaci některé služby, kterou počítač připojený k síti podporuje a převzetí totožnosti této služby. Cílem může být kterákoliv služba, kterou server poskytuje. Proto je vhodné nepoužívané nebo nebezpečné služby blokovat již na firewallu a zamezit k nim takto útočnickovi přístup. Uživatel přistupuje ke službě pomocí konkrétního portu a každá služba očekává, že bude přijímat data ve specifickém formátu. Tedy data přijatá na portu DNS budou v jiném formátu než data odeslána na port SSH.

Přesměrování DNS – hacker může pomocí speciálních technik odposlouchávání sítě sledovat počítač, který poskytuje službu DNS a zjistit, jakou sekvenci pro generování rekurzivních dotazů do DNS využívá. Poté již stačí zfalšovat odpověď na jeden z dotazů do DNS, jenž bude na základě příkazů ve zprávě přesměrovávat na jiný počítač, pod kontrolou útočníka. Tímto lze dosáhnout i převzetí totožnosti jiného počítače. Zvláště nebezpečné riziko nastane, pokud hacker umístí na server relevantní IP adresy a klientské počítače k nim budou přistupovat jako k platným adresám, při překladu doménových jmen.

Přesměrování toku dat v síti – důležitým prvkem v síti je směrovač, který pomocí routovacích tabulek směřuje tok informací po síti a ven z ní. Pokud se ovšem podaří hackerovi získat nad routerem kontrolu a pozměnit jeho směrovací tabulky, dokáže tím síť izolovat a nasměrovat provoz sítě požadovanou cestou. Pro zachování funkce sítě i přes možná zpomalení nebo omezení se musí směrovač přizpůsobit aktuálním podmínkám v síti a nadále si vyměňovat informace mezi ostatními směrovači a komunikovat s jinými směrovači i mimo síť. To ovšem podléhá povolení správce. Posílají si i vzájemné aktualizace o svých tabulkách směrování, pomocí směrovacích protokolů, jako je RIP (Routing Information Protocol), OSPF (Open Shortest Path First) a BGP (Border Gateway Protocol). Nejméně spolehlivý RIP, využívaný k aktualizaci síťových informací, není opatřen funkcí autentizace, a proto může bez větších problémů změnit hacker nastavení směrovače tak, aby posílal pakety do nasměrované sítě nebo odepíral požadavky na některé služby. Další zmíněné protokoly jsou v používání směrovacích tabulek při aktualizaci opatrnější. Přesměrování lze provést i pomocí speciálně modifikovaných ICMP *Redirect* paketů tvořících dojem, že jsou zaslány na nesprávný směrovač a že je vhodnější, použít lepší nebo efektivnější cestu, k cílovému počítači. Avšak falšovat pakety

je nesnadné, protože musejí vypadat, jakoby vycházeli z nejbližší postaveného směrovače.[11]

2.2.2 Převzetí totožnosti

Ve chvíli, kdy hacker pořád nezískal přístup k počítačům v síti, je na řadě převzetí totožnosti jiného počítače. Tento druh útoku je poněkud složitější než zneužití protokolu, kde je napadena chyba v některé z veřejných služeb, za účelem získání většího přístupu, než by bylo běžným způsobem možné. V tomto případě se útočník pokusí napadnout počítač, o kterém zjistí (třeba odposloucháváním síťového provozu) dostatek informací, aby jej zneužil jako prostředníka. Může použít i odepření služby, což využije k oslabení sítě a způsobit další nepříjemnosti, které mu ještě usnadní prolomení přihlašovacích údajů. Poté se prostřednictvím tohoto počítače pokusí o další přístupy, na další počítače a další služby, které hodlá převzít. Mezi dané strategie patří například útoky prostřednictvím přímého směrování nebo převzetí totožnosti služeb, podrobněji dále rozepsány:

Útoky prostřednictvím přímého směrování – při problémech v síti lze použít jako diagnostický nástroj přímé směrování, které je výhodné zejména pro obcházení problémů v sítích. Bohužel je poměrně velmi často hackery zneužíván, a proto by tato možnost měla být v naší síti implicitně zakázána. Tzn. konfigurace firewallu musí být opatřena blokadou všech paketů TCP/IP s nastaveným přímým směrováním, pocházejících z Internetu. Při komunikaci serveru s ověřeným počítačem, se může hacker napojit a vložit mezi tyto prostředky falešnou zprávu, třeba pro provedení změny hesla. Pokud je administrátor v daný moment napojen na server (například přes službu telnet) a útočník vyšle zmíněný příkaz, může server zareagovat odpojením administrátora a otevřením komunikační cesty pro hackera. Útok přímého směrování lze využít i pro získání totožnosti k serveru DNS. Uživatelé v síti, kteří přistupují na DNS server, jež je v moci hackera a používají jej pro překlad doménových adres na IP adresy, budou přesměrováváni na počítače nastavené hackerem, jenž může sloužit i k zachycení uživatelských hesel.

Převzetí totožnosti služeb DHCP, DNS a WINS – při převzetí totožnosti služby DHCP (Dynamic Host Configuration Protocol), od které získávají informace o síťové konfiguraci klienti sítě, hrozí riziko, že hacker přesměruje tyto počítače na kterýkoliv nepřátelský server. Pokud by převzal hacker totožnost serveru WINS (Windows Internet Naming Service), bude pravděpodobně posílat ostatním počítačům neplatné IP adresy v NetBIOS a

při převzetí totožnosti serveru DNS (Domain Name System), pokud není poskytován od ISP (Internet Service Provider), zase neplatné IP adresy doménových názvů.

K přístupu na tyto servery musí útočník nejprve získat identitu některého z počítačů v síti a s ním pak útočit na konkrétní server pomocí odepření služby. Většinou tak přeposílá veškerou komunikaci do externí sítě, kde je zařízení, které nahradí odcizený počítač. Proto je vhodné sledovat síťový provoz a v případě, že je nalezena služba běžící na neautorizovaném počítači, je nutné tento server neprodleně odstavit.

Převzetí totožnosti serveru a zcizení hesla – počítač může být oklamán i zasláním šifrovaných přihlašovacích údajů, odposlechnutých v komunikaci. Tomu nevádí, že je v šifrované podobě, jelikož tento počítač stejně očekává, že je heslo zašifrované. Obrana proti tomuto typu útoku je použití mechanismu „výzva a odezva“, kdy heslo není nikdy přenášeno. Klientův systém přihlašování započne zasláním zprávy na server, který odpoví odesláním jedinečné kombinace čísel a oba účastníci komunikace toto číslo zašifrují uživatelským heslem, načež klient pošle zprávu v podobě vykonané šifry zpět na server. Pokud jsou výsledky shodné, server nabude dojmu, že klient má správné heslo a že použil stejný klíč. Výhodou je, že tímto postupem nelze využít odposlechu k odcizení hesla.

Využití prostředníka – útok z pozice prostředníka, je postaven na principu využití počítače umístěného mezi dvěma stanicemi. Opět tedy klientem a serverem, mezi nimiž probíhá komunikace. Pokud ji hacker bude odposlouchávat, odcizí totožnost a bude přeposílat dotazy na vybraný hostitelský počítač. Klient si v ten moment bude myslet, že komunikuje se serverem, server naopak, že komunikuje s klientem a mezitím do komunikace vstoupí hacker a bude bez povšimnutí měnit veškerý provoz mezi nimi.

Tento typ útoku je vhodné použít, pokud se administrátor připojuje prostřednictvím telnetu na server. Hackerův počítač by si nechal přeposlat pověření přihlášení pro přístup k serveru a poté stáhnul ze serveru soubor s heslem, místo jiného úkonu, který chce administrátor na serveru vykonat.

V Internetu se proti tomuto typu útoku lze velmi obtížně bránit. Naneštěstí pro hackery, je provedení útoku z pozice prostředníka vcelku těžké a uplatněná opatření proti převzetí totožnosti nebo odepření služby postačí i pro ochranu proti metodě využití prostředníka. Nicméně, zásadním pravidlem při přihlašování z účtu administrátora je, nikdy se nepřipojovat přes nezabezpečenou síť. Vždy je vhodné komunikaci zašifrovat pomocí autentizačních balíčků od třetích dodavatelů (např. SecureID od Secure Dynamic),

díky čemuž je dosaženo zcela bezpečné komunikační propojení v síti TCP/IP a značně ztížena cesta útočníka do prostředí naší sítě.[11]

2.2.3 Útoky na webové aplikace

V této části jsou uvedeny nejčastější metody útoku na webové aplikace. Jsou to zejména SQL (Structured Query Language) Injection, XSS (Cross Site Scripting) a krádež session.

Injection a SQL Injection – tento typ útoku je vhodný, pokud jsou od uživatele při sestavování dotazů nevhodně ošetřeny vstupy do jiných systémů. Když je použita obsluha parametrů při tvorbě SQL dotazů ve formě: *String query = "select * from tabulka where field = '"+parametr+'";* pak stačí umístit místo hodnoty „parametr“ např. hodnotu *"or 1=1 or field "* a v ten moment lze přečíst všechna data v databázi, případně i z jiných tabulek, díky dotazům „*join*“. Tak jak je jednoduchý útok, je i obrana. Dotaz SQL nesmí být sestavován pomocí skládání řetězců, ale např. v případě programovacího jazyku Java lze použít tzv. „pozicových“ parametrů, které se postarají o správné obalení parametrů a příčinou toho, bude mnohem bezpečnější kód bez jednoduchého dotazování. Výsledek by tedy v tomto případě byl následující:

- *String query = "select * from tabulka where field = '"+parametr+'";*
- *Query q = session.createQuery(query);*
- *q.setString("p_field", param);*

Krádež session – relace mezi klientem a serverem, díky které si lze vyměňovat pakety, by se dala označit jako „*session*“. V Protokolu HTTP (Hypertext Transfer Protocol), se nejčastěji předává jako proměnná v URL (Uniform Resource Locator) cílové stránky nebo jako tzv. „*cookie*“, což je zpráva, odeslaná prohlížeči a následně uložena na počítači s výčtem předvoleb apod., která slouží k rozlišování uživatelů.

Mechanismus je možné napadnout a získat identifikátor session, na jehož základě lze relevantně vystupovat jako přihlášený uživatel. Proto je vhodné tyto identifikátory náhodně generovat s rovnoměrným rozložením čísel a využívat i kontrolu IP adres jednotlivých uživatelů. Problém může nastat v případě, kdy bude chtít přihlášený uživatel zůstat v této relaci třeba několik dnů nebo týdnů a nebude chtít zadávat neustále přihlašovací údaje. Vzhledem k dynamičnosti IP adres poskytovaných od většiny ISP, je tak pravděpodobné, že se tato adresa, bude často měnit. Proto je vhodné nastavit správně

omezení cookie na dané doméně a kontrolovat dobrou platnost IP adresy. Při ukládání identifikace do URL se musí ohlídat, aby nedošlo ke změně refereru. Zakázat načítání obrázků z nedůvěryhodných serverů a při odkazování, provést přesměrování uživatele.[13]

Cross Site Scripting (XSS) – pokud programátor nevykoná dobrou práci nebo je statická webová stránka tvořena neznalým uživatelem, může obsahovat spoustu zranitelností, ve kterých lze velmi jednoduše falšovat původ obsahu nebo díky kterým vede cesta k získání cookies. V dnešní době jsou moderní prohlížeče schopny čelit chybám v kódu a zotavit se z ní. Avšak způsob zotavování není ve standardu zcela definován a může být různými prohlížeči interpretován stejně, s touto chybou. Pro tento případ jsou k dispozici knihovny, které dokážou kód zkontrolovat a očistit od nepovolených značek.

Vždy po nahrání webových stránek, je vhodné otestovat vstup od uživatele, zda neobsahuje nějakou XSS chybu. Kontrola by měla sestávat minimálně ze dvou částí: doplnění chybějících značek v HTML kódu a odstranění nepovolených značek.[13]

Jeden z nejvíce populárních způsobů a také i někdy z nejsnadněji detekovatelných, je použití párových HTML tagů `<SCRIPT>`, vložených do webové stránky. Kód je možné vkládat jak do těla stránky, tak do hlavičky. Takto vložený skript se provede ihned, jakmile je webových prohlížečem načten. Tedy v praxi začne se zobrazovat obsah webové stránky, jenž je umístěn před tagem `<SCRIPT>`, následně se zobrazování stránky pozastaví a provede se kód JavaScriptu. Poté se načte zbývající část webové stránky umístěné za ukončovacím tagem `</SCRIPT>`. Ukázka použití:

- Externí skript: `<SCRIPT SRC=//ukazka.com/xss.js></SCRIPT>`
- Vložený skript: `<SCRIPT> alert("XSS"); </SCRIPT>`

Pro kontrolu XSS zranitelností poslouží webový scanner, který prochází celý web a automaticky kontroluje zranitelnosti XSS, SQL injection, google hacking nebo další. Například online aplikace *Web Vulnerability Scanner*¹ od společnosti Acunetix umožňuje provést tuto kontrolu šitou na míru za poplatek. Jsou dostupné i scannery² zdarma.

¹ <https://www.acunetix.com/vulnerability-scanner/>

² <http://xss-scanner.com/>

V typickém XSS útoku hacker infikuje legitimní webové stránky škodlivým skriptem na straně klienta. Když uživatel navštíví tuto webovou stránku, skript stáhne do svého prohlížeče a v ten moment je ohrožen. Podrobněji rozebráno v kapitole 3 BeEF (Browser Exploitation Framework)

2.3 HACKEŘI

Jelikož je důležité pro správnou obranu poznat nepřítele, jsou popisovány typy útočníků a jejich motivy pro útok.

2.3.1 Typy útočníků

Různé zdroje literatury zařazují tyto typy útočníků pod hlavní termín „hacker“ členící se v zásadě dle motivace útoku. Tento pojem lze definovat následovně: „jednotlivec či skupina, s úmyslem napadnout nějaký systém nebo síť, s cílem jej poškodit, získat cenné data nebo jej zneužít jako prostředníka k dalšímu použití při větších útocích. Pro síťové útočníky existují různá označení, jako je „*Script kiddies*“, *odborníci na zabezpečení* nebo *průmysloví špioni* apod. Podrobnosti o jednotlivých typech jsou podrobně popsána na následujících řádcích:

Script kiddies (*skriptovací děti*) – jedná se o nejrozšířenější a nejobvyklejší typ útočníků, jelikož tito (často mladí a nezkušení) lidé bez znalostí, hledají a využívají již vytvořené skripty a nástroje zkušenými hackery, kterými se pak snaží někoho obtěžovat nebo páchat škody. Většinou používají počítače doma nebo v prostředí školních lavic a dost často se snaží ohromit své vrstevníky, svým rádobý neotřelým činem. Jsou to konzumenti již vytvořených nástrojů, návodů a v okouzlení hrdinskými skutky ostatních, se je snaží napodobovat v jejich světě fantazie a neomezení. Obvykle hackují především proto, aby získali něco zadarmo, většinou hudbu nebo programy. Tento typ není v komunitě lidí zcela oblíben, jelikož skript kiddies často útočí a zlomyslně bez uvážení ničí data, a to častěji než ostatní skupiny. Dokonce i mezi profesionály postrádají uznání.

Dospělí skript kiddies – pokud byli v době studia či v zaměstnání příliš zaměřováni na útoky a výpočetní techniku, mohlo se stát, že tito ještě mladí skript kiddies nedodělali školu nebo z nějakého ještě horšího důvodu, přišli o zaměstnání. Obvykle si udržují práci na poloviční úvazek, kde si vydělají na nájem. Oproti mladší generaci jsou ovšem již schopni tvořit ve svém volném čase nástroje a skripty používané při různých útocích. Většinou se nejedná o žádné zločince. Jejich úmysl bývá spíše pirátství a zaměřují se na

software nebo různá média, která „odblokovávají“ od ochranných prvků pro komerční využití. Důsledkem toho mnohdy bývají provokace proti vládě nebo obrovským společnostem, se kterými tato skupina nesouhlasí. Tvoří desetinu komunity hackerů.

Ideologičtí hackeri – v posledních letech se tento ideologický hacking změnil na informační válku, a proto byl zařazen do těchto kategorií. Jedná se o vyjadřování souhlasu či nesouhlasu k různým politickým otázkám, jako je nacionalismus nebo ochrana životního prostředí. Často bývají znehodnoceny různým způsobem stránky odpůrců nebo jsou proti nim podnikány útoky, jejichž důsledkem je odepření služby. Obvykle jsou prezentovány médií a takto (za nejisté podpory vlády) je nelze na základě místních zákonů postihnout.

Hackeri (zločinci) – tato skupina se označuje jako „černé klobouky na straně zla“. Jedná se o komunitu, která na základě početných znalostí a zkušeností se zabezpečením, útočí na plánované cíle s jasným záměrem. Obvykle se jedná o loupež či nelegitimní převod peněz z jednoho účtu na druhý do ciziny nebo jiný druh finančního zisku. Zpravidla jsou tito útočníci zřídka odhaleni, jelikož jejich inteligence i možnosti jsou natolik propracované, že se dokážou dost často schovat do ústraní a zamést za sebou všechny stopy. O svých schopnostech se veřejně nezmiňují a jsou tak nesdílní už od povahy, a proto je možnost nalezení takového člověka ještě těžší.

Průmysloví špioni – špioni zpravodajských služeb podnikající útoky proti konkurentům. Jedinci, objevující se ve státní správě, kteří mají mnohdy obrovský přístup k informacím, jsou za jejich mnohdy nízké platy často demotivováni k loajální práci a přejdou na stranu společnosti, která jim za vyzrazení těchto tajemství dobře zaplatí. Tato „loajalita“ se prodává na černém trhu a je prodávána za co největší nabídku.

Odborníci na zabezpečení – tito lidé se nachází na nejnižší úrovni hrozby, protože kdyby se stali hrozbou, v ten moment by byli označeni za hackery na straně zla. Jejich schopnosti umožňují neustále vytvářet záplaty na ochranu systému a testovat jejich funkčnost, jež pomáhá zdokonalit kvalitu zabezpečení uživatelů. Obvykle jsou tito profesionálové nájímáni velkými společnostmi, kde zkoumají trhliny v jejich systému a dostávají více peněz, než kdyby se snažili o útoky za hranicí zákona. Mnohdy i bývalí hackeri, které lze odlišit od hackera jen jeho tvůrčí činností, kterou vykonává. Řídí se zásadou, že jediné bezpečné prostředí je takové, které je dobře otestované na možnost selhání z důvodu zabezpečení.[11]

2.3.2 Motivace hackerů

Důvodů, proč se hackeři na svůj cíl zaměřují s cílem zisku či nějaké škody, může být více, avšak tak jako jakýkoliv jiný trestný čin, je takové jednání nelegální. Jako hlavní důvody motivace jsou uvedeny: proslulost, ničení, politika, finanční zisk nebo touha po vědomostech. Podrobnější popis byl rozebrán níže:

Proslulost – nejčastějším důvodem pro páčání útoku, je zviditelnění se v komunitě hackerů. Tento faktor je proslulý u skupiny *skript kiddies* nebo jiných méně zkušených „*crackerů*“. Snaží se tedy stát známými spácháním nějakého nelegálního útoku ve větší míře, která bude mediálně známá a proslulá, čímž se pak chlubí mezi sebou a navzájem si tak dokazují větší oblíbenost mezi svými vrstevníky. Prakticky, se jedná o studenty nebo mladší lidi s počítačem s připojením k Internetu, s větším množstvím volného času, ze kterých se časem stanou profesionálové a dost často se ocitnou v části rozhodování, zdali se bude jejich cesta ubírat na stranu zla nebo naopak dobra.

Zlomyslnost nebo ničení – ať už je cílem jen obtěžování nebo cílená destrukce, vždy takto útočí člověk na základě zlomyslného jednání. Jedná se o lidi, kteří mají radost z poškozování cizích věcí nebo ničení majetku. Dalo by se to přirovnat k lidem, co bez důvodu házejí kameny na projíždějící vlak nebo rozbíjejí lavičky na ulici. Může to být příčina jejich frustrace či hněvu a tímto způsobem pouze tento problém vypouští na povrch. Takový útočník obvykle maže systémové soubory nebo rovnou ničí data na disku. Dalším typem člověka by mohl být nespokojený zaměstnanec, který je, například po propuštění se schopen vyrovnávat s tímto problémem velmi destruktivním způsobem a zničí třeba server nebo kompletně vymaže data společnosti. Což je (samozřejmě díky dnešním pracovním smlouvám a ochrany zabezpečení citlivých dat) mnohdy problém se zákonem.

Politický motiv – pokud má motivace politický charakter, může být potencialem cílem útoku například vláda. Takové útoky mohou být jak z jiných zemí, tak od demonstrujících a nespokojených obyvatel. Jako příklad by se dalo uvést napadení vládních serverů v několika menších zemích v Evropě. Tento útok však nebyl úspěšný, díky selhání komunikačního systému Země. Jako další příklad, by se daly zmínit i další státy, jako je Izrael nebo Palestina, kteří mezi sebou vedou válku v kyberprostoru již řadu let. Cílem útoků hackerů se stala i Čína a v čase Internetu tyto útoky budou i nadále pokračovat.

Finanční zisk – většina dnešních obchodů se provádí přes Internet a i to nahrává těmto „králům Internetu“ k připlezení se do takového obchodu, s cílem získávat opakovaně nemalou finanční částku. Nové technologie a standardy, které se starají o bezpečnost těchto transakcí, stále nejsou na takové úrovni, aby zamezili crackerovi, udělat si z něj zlatý důl. Neustálý pohyb financí na Internetu dovoluje počítačovým kriminálíkům vyloupit banku s počítačem v ruce, což je znatelně jednodušší a bezpečnější varianta než se zbraní v ruce. V dnešním světě je velmi běžná situace odcizení peněz z bankovního účtu na účet v cizině. Takto - zřídka kdy lokalizovatelný útočník - pak uniká se sumou peněz bez postihu a nebojí se znovu útočit. Tyto nezjistitelné činy jsou po celém světě na denním měřítku.

V některých případech jsou využity osoby, které se nacházejí v konkurenční společnosti, pro odcizení důvěrných dat nebo obchodních tajemství, na základě kterých jsou pak schopni konkurenta předstihnout a vydělat tak peníze. Nejčastěji se tak děje v podobě pirátství, kdy zevnitř společnosti někdo zcizí nově vydaný software. Ten je dále distribuován do prostředí internetového „warezu“ (pirátský software) a za několik málo hodin je tento program vyneseno do celého světa.

Při použití webových obchodů jsou transakce zabezpečeny s využitím protokolu SSL (Secure Socket Layer), který při přenášení informace mezi zákazníkem a obchodem data šifruje. Avšak tyto citlivá data jsou dále ukládána společnostmi různými způsoby a ty nejsou zrovna vždy v bezpečí. V posledních letech bylo mnoho případů, kdy byla odcizena databáze kreditních karet, a to i v nezašifrované podobě, což samozřejmě způsobuje nemalý problém majitelům těchto karet. Někteří lidé jsou ochotni za čísla kreditních karet dobře zaplatit, a tak je páchání těchto podvodů vcelku lukrativní obor i pro hackery.

Vědomosti – dalším motivem, proč provádět útoky, je málo uváděný důvod – získání znalostí. Touha po informacích žene mnoho hackerů kupředu, a to třeba jen proto, aby zjistili, jak funguje určitý systém. Jejich cílem tedy není odcizení dat nebo zničení, jako v předchozích případech, ale pouhé vzrušení ze hry. V tomto případě se tak hackeři, kteří se díky svému abstraktnímu myšlení a trpělivosti dostanou do jádra systému, stávají lepšími správci sítě nebo programátory. Avšak bohužel jsou i tací, kteří takto nabitě znalostí zúročí v dalším plánovaném a větším útoku a zásluhou otevřeného sdílení těchto informací na Internetu, problém ještě prohlubují.[12]

3 BEEF (BROWSER EXPLOITATION FRAMEWORK)

Browser Exploitation Framework (BeEF) je penetrační nástroj ze sady tzv. „backdoorů“, pro testování bezpečnosti webového prohlížeče, napsaný v programovacím jazyce Ruby a navržený tak, aby bezpečnostní slabina prohlížeče, sloužila zároveň i k provádění útoku na jiný prohlížeč. V podstatě dokáže navázat oboustrannou komunikaci mezi útočником a webovými prohlížeči, avšak má v tomto směru více využití. Dokáže zranitelné prohlížeče proměnit v tzv. „zombie“ a poté s nimi dle modulárních dovedností libovolně manipulovat.

Je postaven na architektuře klient-server. Skládá se tedy ze serverové aplikace, která tyto zombie spravuje a JavaScriptu, běžícího v prohlížeči cílových hostitelů. Tradičně je JavaScript „vstříkovan“ útočником do HTML kódu, a to buď prostřednictvím útoku jako XSS nebo SQL Injection. Kromě těchto běžných metod, existuje i třetí, méně častý přístup vstříkování na webové stránky – přes MITM³ (Man in The Middle). Jakmile se tento proces zapracuje, odešle zprávu BeEF serveru a v ten moment je již připraven zpracovávat příkazy ve formě funkcí JavaScriptu, jež byly odeslané ze serveru na klienta. Příkazy zaslané do prohlížeče se spustí pomocí modulů běžících v odvětví BeEF serveru. Tyto moduly posílají příkazy, které provádějí vše, od snímání otisků prohlížeče a pluginů, po povolení útočnikovovi vstupovat na server proxy webového provozu přes prohlížeč.

BeEF je aktivně vyvíjen jeho vývojáři, kteří mají v plánu začlenit mnoho nových funkcí. Vytvářejí další moduly, které slouží k plnění úkolů, jako je: logování stisknutých kláves, síťové skenování portů, detekce nástroje Tor⁴, využití dalších rozšíření Mozilly nebo detekce funkcí prohlížeče apod.

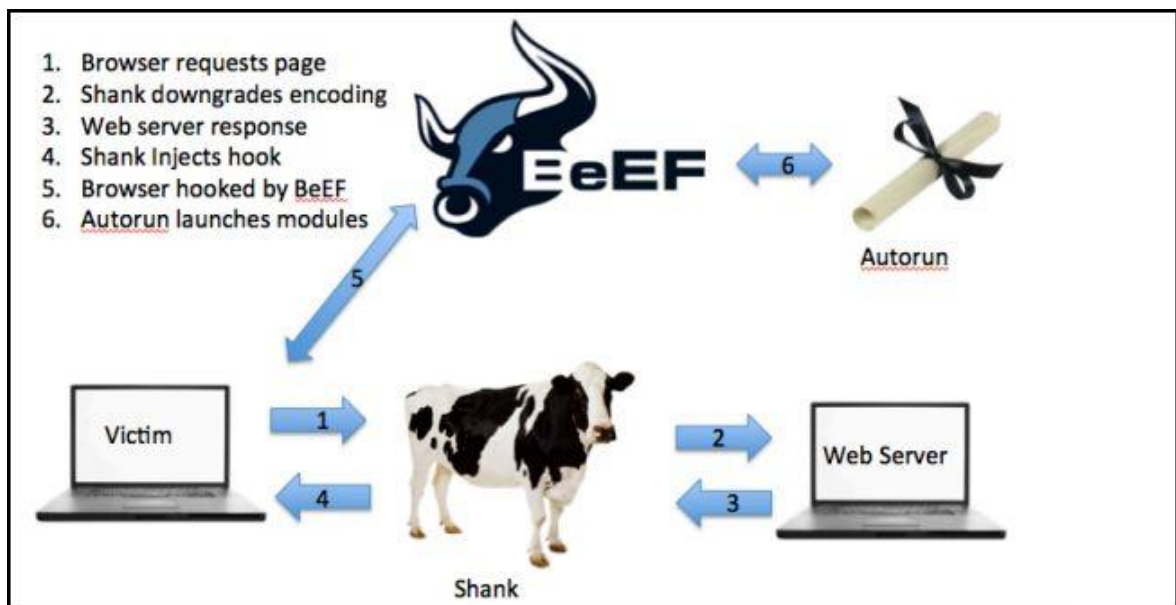
³ MITM – třída útoku, který umožňuje vložení sebe samého do středu mezi komunikační tok mezi dvěma nebo více stranami. Lez použit celou řadu technologií, včetně, ARP (Address Resolution Protocol), DHCP nebo 802.11 (standart bezdrátové sítě).

⁴ Tor - aplikace, která umožňuje uživatelům anonymně přistupovat k různým internetovým stránkám na Internetu, čemuž napomáhá model klient-server, kdy uživatel využívá pouze klientskou část a jeho datový tok vstupuje nejprve na servery Toru a až poté k cílovému počítači, což zhoršuje možnost vysledování stop činnosti uživatele.

Jakmile je kód JavaScriptu zpracován prohlížečem, započne dotazování mezi prohlížečem a BeEF serverem pomocí BeEF „hook“ (háčku). Je-li komunikace stabilizována, shromáždí se některé základní informace o prohlížeči do lokální databáze a prohlížeč se automaticky přidá do seznamu „zombies“ nebo kontrolovaných prohlížečů. BeEF pak udržuje tyto háčky a čeká na zaslání modulů útočníkem k jednotlivým „zombies“.

Tato metoda funguje spolehlivě pro menší množství „zombies“. Nicméně, s větším počtem se stává ruční spouštění požadovaných modulů na každém z nich velmi zdoluhavým procesem. Pak je možnost nastavit použitím části BeEF tzv. „REST API“ autorun skript, který bude pouštět řadu modulů (například nově vybrané cíle) automaticky.

Pomocí automatického spouštění skriptu může útočník udělat hromadně kroky, které zjistí najednou více informací a shromáždí je do databáze serveru. Následující obrázek (Obr. 3), zobrazuje proces vstupu oběti (Victim) na webovou stránku a po cestě zpět je infikována Beef agentem. Následná komunikace s BeEF již spustí automaticky skript autorun a ten provede potřebné operace.



Obr. 3 BeEF[4]

Základní postup útoku je následující:

1. Záškodník provede útok na webovou stránku v naději, že najde nějakou náchylnou na XSS.

2. Když takovou stránku nalezne, využije její zranitelnost XSS a infikuje ji BeEF agentem.
3. Útočník vyčkává na otevření infikované stránky návštěvníkem.
4. Jakmile je webová stránka zobrazena návštěvníkem, BeEF agent na to upozorní (v tento moment může útočník převzít kontrolu na návštěvníky přes BeEF)
5. Útočník „postrčí“ návštěvníky prohlížeče k načtení stránky s klientem nebo použijte pokročilé triky sociálního inženýrství k donucení návštěvníka ke stažení a instalaci nežádoucích souborů, pluginů nebo rozšíření.
6. Oběť je nakažena a připravena k dalším útokům.

Největším problémem je fakt, že při využívání XSS, běží aplikace jen, když má oběť stránku otevřenou. Většina uživatelů si ovšem stránku po nějaké době budou chtít zavřít a přejít na jinou nebo zavřít dříve, než je vůbec šance zahájit útok. Proto byly vymyšleny různé metody, které se touto problematikou zabývají.

Jako nejpoblárnější z nich, se údajně osvědčilo nasadit na stránky obrázky nebo videa s „lehkou tematikou“, jež udrží mladé a muže, takřkajíc „na svém místě“. Dále lze využít metodu „plýtvání časem“ hru, dlouho načítající video nebo otevření nové záložky s BeEF háčkem v novém okně, čehož si většina lidí ani nevšimne.

Další metodou, byla navrhována technologie „pop-under“, jež zahrnuje dvě velmi jednoduché technologie JavaScriptu. Nejdříve otevře v novém okně reklamu a poté ji ukryje na zadní část monitoru. Většina novějších prohlížečů umožňuje zobrazit nové okno pouze tehdy, pokud bylo voláno například kliknutím myši jako výsledek interakce s uživatelem obslužnou rutinou události. Ostatní neinteraktivní události jsou v novém okně blokovány. K tomu, abychom předešli tomuto omezení, poslouží právě připojení posluchače událostí, přímo do dokumentu nebo těla dokumentu. To umožní ulovit všechna události kliknutí myši, které nebyly zabráněny ostatními subjekty zpracovávajících událostí, kliknutí myši nebo otevření okna, aniž by byla blokována. Jako příklad poslouží výběr textu uživatelem, který spustí popisovač připojený k dokumentu a okno se objeví za použití výše uvedeného kódu. Takových technik, jak obejít tento tzv. „únos“ kliknutím myši, je více.

Jak lze vidět, metod, jak udržet uživatele na stránce, je celá řada a vzhledem k tomu, že člověk je tvor velmi tvořivý, tak se jistě další brzy najdou.[4]

4 FIREWALL

4.1 Definice firewallu

Označení „firewall“ by se dalo obecně přeložit jako „protipožární zeď“, avšak vzhledem k faktu, že se tento název v češtině neujal, se dodnes používá spíše původní anglický výraz.

V zásadě se jedná o zařízení zapojené mezi veřejnou a privátní sítí, které rozhoduje o tom, jaký síťový provoz bude propouštěn a jaký blokován. Ve své podstatě tedy zastává funkci jednoho z nejdůležitějších bezpečnostních prvků zabezpečení počítače nebo sítě.

Důvodů, proč se taková komponenta nebo více komponent s tímto obranným systémem mezi různé sítě nasazuje, může být více, avšak primárním důvodem je ochrana uživatele od možných hrozeb ve veřejné síti. Rizika, která s sebou přináší připojení na Internet, mohou být různá. Ať už se jedná o možnost zcizení dat uložených na počítači nebo zneužití prostředků v síti. Stejně tak lze nastavit opatření proti porušení bezpečnostní politiky, jako je používání zakázaných služeb ve vnitřní síti, apod.[1]

4.2 Politiky firewallu

Pro potřeby své sítě a jejích účastníků musí mít každý firewall vytvořenou politiku. Zpravidla se používá ke dvěma účelům. Udržet lidi (útočníky) venku nebo naopak lidi (zaměstnance) uvnitř. Základní metodika sestavování bezpečnostní politiky je poměrně jednoduchá a skládá se z následujících pravidel:

1. Definice služeb, které firewall bude nabízet.
2. Výpis uživatelů, pro které jsou služby určeny.
3. Ustanovení, jaké služby potřebují jaké skupiny uživatelů.
4. Zajištění bezpečnostních služeb pro každou skupinu služeb.
5. Prohlášení za nežádoucí všech ostatních metod přístupu.

4.3 Typy firewallu

Firewally lze členit na dva základní typy vzhledem k implicitnímu postoji filtrování sítě: filtrovací firewally a proxy servery. Oba typy se postupem let vyvíjely a vylepšovaly.

Filtrovací firewally – fungují na bázi filtračních pravidel, která rozhodují o tom, zdali příchozí IP datagramy mohou být propuštěny do sítě nebo zahozeny či odmítnuty. K rozhodování o zahození datagramu poslouží celá řada nadefinovaných kritérií, jako jsou například typ protokolu, číslo portu, zdrojová a cílová adresa datagramu. Veškerý proces se provádí na síťové úrovni, a proto se filtrování stará pouze o spojení mezi aplikacemi a ne o aplikace samotné. Výhodou takového firewallu je jeho rychlost, jednoduchost a nízká cena.

Proxy server – systém nebo aplikace, která působí jako prostředník pro požadavky klientů, kteří hledají prostředky z jiných serverů. Klient se připojí k proxy serveru a požádá některé služby, jako je připojení, soubor, webové stránky nebo jiné zdroje dostupné z jiného serveru a proxy server vyhodnotí žádost jako způsob, jak zjednodušit a kontrolovat jejich složitost. Jedná se o nejrozšířenější formu přístupu k řešení aplikačních bran na aplikační úrovni. Proxy neboli zástupce, není nic jiného než proces, který běží na firewallu a zastupuje uživatele při procházení webových stránek. Usnadňuje tím přístup k Internetu a poskytuje anonymitu.[1]

4.4 Nástroje pro filtrování paketů

Nástroje pro filtrování paketů se s vývojem jádra vyvíjely a vylepšovaly. První variantou tohoto nástroje používaného jádrem pro přijetí nebo zahození paketů IP, byl *ipfwadm*. Fungoval s jádrem ve verzích 1.2 až 2.1 a byl založen na systému BSD (Berkeley Software Distribution) *ipfw*. V případě vlastnictví staršího systému lze inovovat na novější verzi, jež zajistí použití novějšího nástroje *ipchains*, který sloužil jako náhrada od jádra 2.1.102. Mimo stejné funkce předchozího nástroje, umožňoval srozumitelnější syntaxi a mechanismus spojování více sad pravidel tzv. „řetězování“. Dalšími výhodami je například schopnost sledování paketů přicházejících na vysokorychlostní síťová rozhraní pro 32bitové čítače nebo adekvátní práce s fragmentací IP. Pozitivním přínosem byla i nově nabitá podpora pro další protokoly transportní vrstvy (jiné než UDP - User Datagram Protocol a TCP) a zadávání inverzních pravidel.[10]

S novými distribucemi Linuxu založených na jádře 2.4 přichází balíček *Netfilter*, jehož nedílnou součástí je uživatelský nástroj *iptables*. Syntaxe je podobná staršímu nástroji *ipchains*, avšak rozdíly přichází v mnoha důležitých oblastech:

- při výpisu řetězce lze nulovat pouze tento řetězec pomocí volby – Z

- čítače zásad jsou nulovány nulováním vestavěných řetězců
- změna velikosti písma u jednotlivých názvů parametrů (INPUT, OUTPUT, FORWARD) z malých písmen na velká
- počet znaků pro názvy řetězců povolen až na hodnotu 31
- změna příznaku pro hledání paketů IP s nastaveným bitem SYN na - syn
- příznaky rozhraní se nyní rozdělují na příchozí a odchozí. V řetězcích input a forward se použije parametr – i a v řetězcích output nebo forward pracuje deskriptor rozhraní – o.
- ipchains MASQ změněno na MASQUERADE

Více o iptables v následující kapitole.

4.5 Iptables

Hierarchie pravidel zavedených v iptables je nesmírně důležitá. Pokud paket splňuje definované podmínky prvního možného pravidla pro vstup do sítě, nebude již kontrolován dalšími pravidly a bude buďto povolen nebo zahozen. Proto je nezbytné si pro konstrukci firewallu popsat základní syntaxe a vysvětlit běžně používané parametry iptables:

syntaxe: *iptables [tabulka] [akce] [chain] [ip_část] [match] [target] [target_info]*

Tabulka – první částí základní syntaxe jsou tabulky a ty jsou přítomny ve třech provedeních: filter, nat a mangle. Někteří autoři uvádí i čtvrtý typ raw. V případě, že není tabulka specifikována, je defaultně použita výchozí bulka filter. To je vhodné zejména pro základní filtrování nebo logování. Nat se využívá pro přepis adres, port forwarding případně tvorby maškarády⁵. Mangle slouží k zpracování hlaviček paketů a raw pro nastavování výjimek. Přítomnost jednotlivých druhů tabulky je závislá na konfiguraci jádra a na nastavení jaderných modulů. Výběr tabulky lze učinit přepínačem – t (table).[9]

Akce – výpis příkazů, přičemž nejpoužívanější jsou první čtyři: append, policy, list a flush. Pravidlo se zapisuje s pomlčkou před písmenem, např. iptables – A.

⁵ Maškaráda je proces, díky kterému počítače v privátní síti mohou přistupovat k službám Internetu. Podrobněji v kapitole 5 Nasazení FIREWALLu

- *A (append)* – na konec řetězce přidá další pravidlo
- *P (policy)* – nastaví výchozí politiku
- *L (list)* – volba provede výpis všech pravidel v řetězci
- *F (flush)* – vyprázdní všechna pravidla v řetězci
- *D (delete)* – vymaže pravidlo. Lze ho zadat ve tvaru, ve kterém bylo přidáno nebo pomocí indexu pravidla, který je možné získat zadáním rozšířené volby – *lin (line numbers)*
- *R (replace)* – náhrada indexu pravidla dalším pravidlem
- *I (insert)* – na začátek řetězce přidá nové pravidlo
- *N (new chain)* – tvorba nového vlastního řetězce
- *E (rename chain)* – přejmenuje vlastní řetězec
- *X (delete chain)* – smaže vlastní řetězec[9]

Chain – Každá tabulka se dále skládá z několika sad pravidel, označujících se jako „řetězce“ nebo „řetězy“. Pojmenovávají se podle způsobu, jakým jsou zpracovávány. Výčet základních řetězců tabulek filter a nat. Mangle obsahuje všech pět položek zmíněných tabulek (viz Obr. 4).

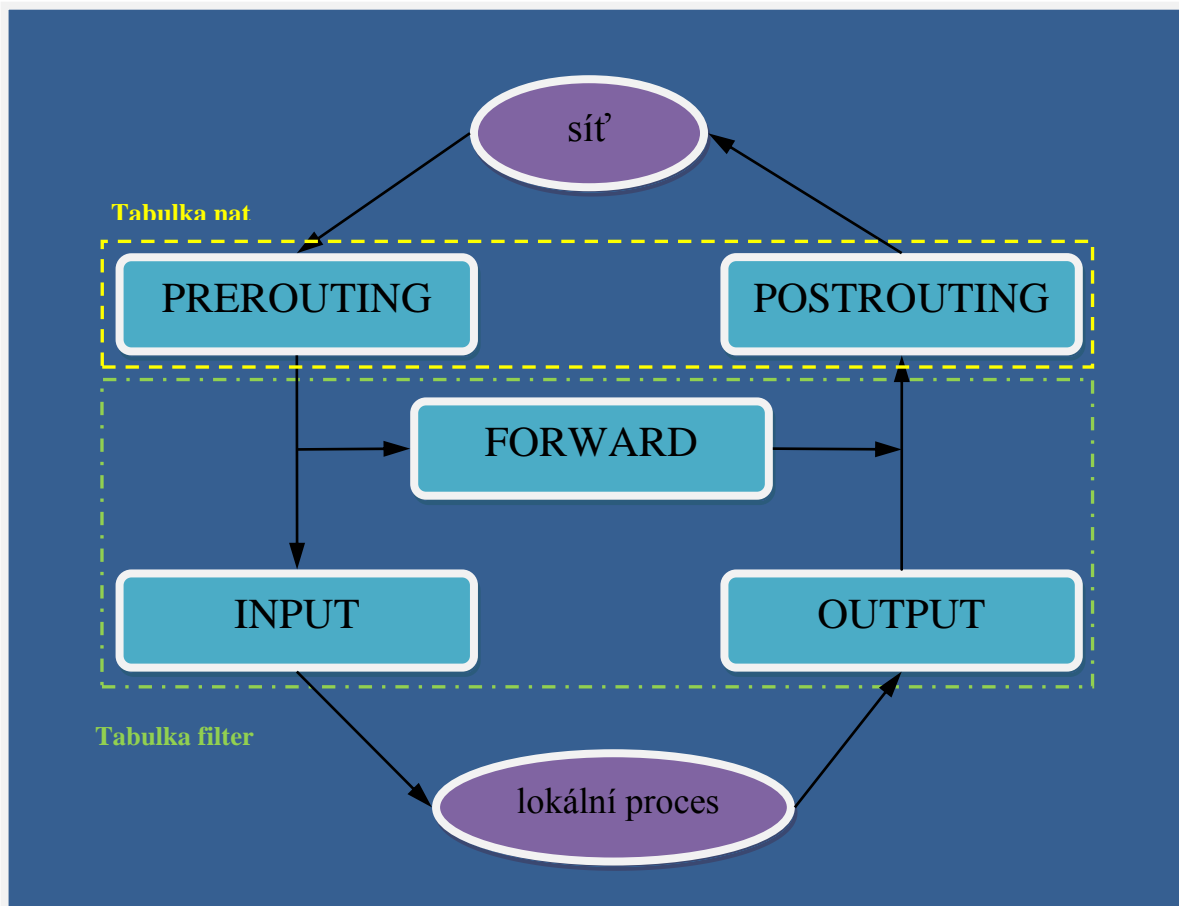
Tabulka filter:

- **INPUT** – pravidla vstupního řetězce se aplikují pro pakety směřující do počítače.
- **OUTPUT** – pravidla výstupního řetězce jsou použita pro odchozí pakety.
- **FORWARD** – řetězec, který se stará o data určené pro jiný počítač a přeposílá je mezi jednotlivými sítěmi. Tato data neprochází INPUTem ani OUTPUTem. Funkci je nutno explicitně povolit v jádře (dále se problémem zabývá kapitola 5 Zavedení pravidel iptables).[3]

Tabulka nat:

- **PREROUTING** – řetězec vhodný pro modifikaci cílové cesty příchozích paketů (DNAT). Hodící se pro port forwarding⁶ do neveřejné místní sítě.
- **POSTROUTING** – řetězec, který se používá pro změnu zdrojové adresy (SNAT), masquerade (maškaráda).

⁶ Přesměrování portů je využíváno v případě, že se uživatel potřebuje připojit na port pomoci směrovače na soukromé adrese v lokální síti. Směrovač musí disponovat funkcí podpory překladu síťových adres.



Obr. 4 Blokové schéma průchodu paketu firewallem

Parametry – při formulaci pravidel síťové činnosti lze použít parametry pro specifikaci pravidel (symbol “!” vykřičník znázorňuje negaci). Zde je výpis těch nejběžnějších:

- p [!] (protokol) – výběr z protokolů, např. ‘tcp’, ‘icmp’, ‘tcp’ a ‘all’, eventuelně z */etc/protkols*. V případě vynechání volby bude defaultně nastaveno na ‘all’.
- s [!] (source) – specifikace zdrojové adresy, která se má hledat. Adresou může být jméno počítače v síti nebo jen jeho IP. Masku lze zadat ve zkrácené formě, jež vyjadřuje počet jedniček na její levé straně nebo pomocí čtyř desítkových číslic oddělených tečkami. Zápis 255.255.255.0 je tedy ekvivalentní zápisu IP s maskou 24.
- d [!] (destination) – specifikace cíle, která se má hledat
- i [!] (--in-interface) – název rozhraní, skrze které zpráva dorazila (určeno jen pro pakety, které se testují v řetězech FORWARD, OUTPUT a POSTROUTING). Pokud bude volba vynechána, použijí se všechna rozhraní.

- o [!] (--out-interface) – název rozhraní, ze kterého se bude zpráva odesílat (určeno jen pro pakety, které se testují v řetězech FORWARD, OUTPUT a POSTROUTING). Pokud bude volba vynechána, použijí se všechna rozhraní.
- j (jump) – provede se v případě, že paket splní podmínky daného pravidla a paket bude odeslán na cílovou adresu této volby.
- g (goto) – oproti jump odkáže na uživatelem definovaný řetěz, kde bude zpracování pokračovat.[10]

Cíl – vyjadřuje životnost paketu v síti, na základě provedené akce parametrem – j.

Konkrétní akce jsou následující:

- **ACCEPT:** paket je akceptován a projde filtrem
- **REJECT:** paket je zamítnut a zahozen. Zdrojový počítač bude o akci informován.
- **DROP:** paket je rovněž zahozen, ale tentokrát bez odeslání informace počítači.[9]

4.6 Návrh firewallu

Při návrhu firewallu je potřeba zohlednit prostředí, do kterého má být nasazen. Dále finanční možnosti a v neposlední řadě výběr operačního systému, který by měl být pro tento účel vytvořen. V následujících kapitolách byl rozebrán důvod výběru linuxové distribuce a její vlastnosti, požadavky na firewall a schéma umístění v síti.

4.6.1 Typ distribuce a její důležité vlastnosti

Rozhodujícím krokem ve stavbě firewallu je výběr vhodné distribuce, která nejlépe odpovídá požadavkům pro zabezpečení a poskytuje možnosti dalšího rozšíření. Programy obsažené v jednotlivých distribucích se víceméně shodují, mohou se však lišit v rychlosti vydávání balíčků a pravidelných aktualizací, které jsou pro opravu bezpečnostních chyb klíčové.

Jednou z nejrozsáhlejších distribucí Linuxu, na jejímž udržování se podílí skupina několik set dobrovolných vývojářů, je jednoznačně verze Debian. Její síla spočívá ve velmi spolehlivém systému testování, jež umožňuje neustále vydávat stabilní verze. Na těchto verzích se pracuje zhruba dva roky, a proto jsou předurčeny i pro použití na serverových

stanicích. Díky podpoře⁷ centra pro informace o zabezpečení a kvalitnímu balíčkovacímu systému, se zařazuje mezi nejbezpečnější distribuce vůbec.[10]

4.6.2 Požadavky na firewall

Aby nasazení firewallu splňovalo svůj účel a sloužilo k ochraně sítě, je nutné při jeho návrhu zavést z hlediska bezpečnosti několik základních pravidel:

- vyčlenit si pro firewall samostatný počítač – vzhledem k nízkému nároku na výkon, postačí i starší počítač, doplněný o síťové karty. Mnohdy poslouží i virtualizační systém, jenž ovšem vyžaduje vyšší paměti RAM (Random Acces Memory) v hostitelském počítači a základní desku, která emulace operačního systému podporuje. Některé desky ještě vyžadují povolení této funkce v BIOS (Basic Input Output System).
- ochrana celé sítě – nejvhodnější umístění počítačů v síti je za firewall. Nenechat tedy žádnou stanici (volně dostupnou k Internetu) bez ochrany. Celý systém umístění jednotlivých klientských stanic i serverů v síti tak dosáhne koncept vyšší bezpečnosti a kompletně ztíží útočníkovi, na základě účelně sestavených pravidel, průchod do sítě.
- poskytování služeb – nemělo by být poskytováno více služeb, než uživatelé skutečně potřebují nebo jsou určeny pro chod počítače. Deaktivace nepotřebných služeb je tak dalším krokem k lepšímu zabezpečení celé sítě.
- defaultně zahazovat všechny pakety – možnosti nastavení výchozí politiky pro průchod sítí lze nastavit na povolení všech příchozích i odchozích paketů. Obecně se ovšem doporučuje obrácená varianta, a to řídit se pravidlem: „co není vysloveně povoleno, je zakázáno“.[5]
- povolení ICMP⁸ paketů – z praktických zkušeností o testování komunikace mezi jednotlivými stanicemi nebo směrovači, je vhodné zavést povolení ICMP paketů

⁷ <http://www.debian.org/security/>

⁸ ICMP (Internet Control Message Protocol) – protokol ze sad protokolů standartu TCP/IP. Servisní pakety určené pro přenos chybových a diagnostických zpráv.[9]

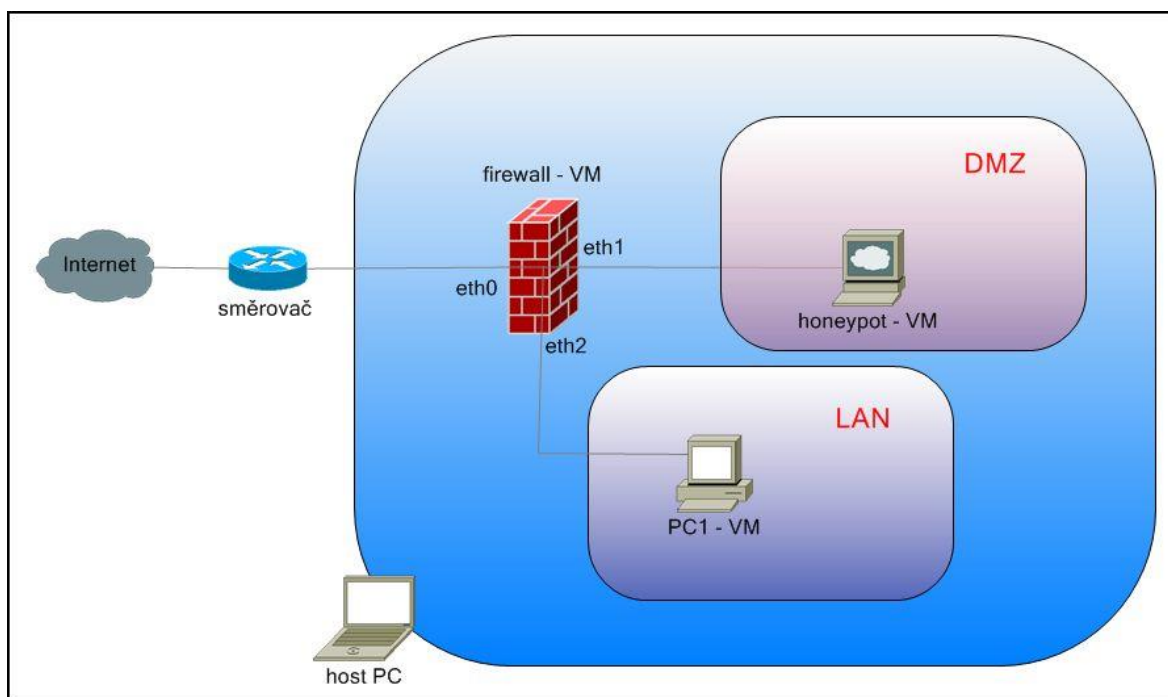
pro zjištění stavu IP a řídicích zpráv. Pro správnou funkčnost je potřeba přinejmenším propouštět čtyři důležité ICMP zprávy, a to jsou typy: 0 – „echo reply“, 3 – „destination unreachable“, 8 – „echo request“ a 11 – „time exceeded“, a to z toho důvodu, že používají nezbytné programy jako ping nebo traceroute.

- logování – pro výpis informací o provedených akcích jednotlivých pravidel, se zavádí parametr LOG, který lze libovolně nastavit a specifikovat k požadované situaci. Umožňuje i další rozšíření o limit pro frekvenci zapisování logu nebo vkládání vlastních poznámek.[9]
- Použití NAT (Směrování) – jedno z nejpoužívanějších pravidel pro zamaskování IP adresy, které by měl firewall poskytovat. Při opuštění podnikové sítě se maskuje za jednu veřejnou IP adresu určenou pro přístup uživatelů umístěných ve vnitřní síti k Internetu.
- Stav spojení – povolení příchozích paketů jen na navázaná spojení a sledování jejich stavů. Ty se monitorují pomocí modulu sledování stavů, v němž se paket může nacházet. Tyto stavy jsou: NEW, ESTABLISHED, RELATED a INVALID.
- Bezpečnost SSH – aby mohly určené prvky ve vnitřní síti přistupovat k firewallu pomocí služby SSH, musí být tato služba implicitně povolena a patřičně nakonfigurována pro vzdálená připojení. Předpokládá se šifrovaná komunikace a zákaz přímého přístupu pro uživatele root. Měla by být splněna posílení možnost autentizace pomocí využití autentizačních klíčů.[3]
- Ochrana před DoS (Denial of Service) útoky – proti zahlcení četným množstvím požadavků a následného pádu systému nebo přetížení počítače, lze použít účinnou ochranu omezení počtu nově příchozích spojení. Následkem takového útoku bývá často vnucení opakovaného resetu konkrétního počítače nebo narušení či zpomalení komunikace mezi účastníky sítě.
- Ochrana proti podvržení IP adres – pro přístup k službám vyhrazených jen určitým počítačům, může útočník použít fintu maskování IP adresy. To by mohlo vést k dojmu, že se jedná o účastníka vnitřního prostoru, následně k oklamání firewallu a umožnění přístupu do soukromé sítě.[5]

4.6.3 Umístění firewallu v síti

Pokud jsou v počítačové síti poskytovány různé služby typu email nebo DNS, mohou se, v případě umístění těchto služeb v privátní síti, vystavovat zvýšenému riziku napadení. Aby jakýkoliv útok – i s třeba s pomocí velkého objemu dat – neměl takový dopad i na vnitřní síť, je vhodné používat tradiční architekturu s demilitarizovanou zónou. Zahlcení firewallu neustálými požadavky může výrazně ohrozit i pravděpodobnost průniku do lokální sítě, a proto je vhodné tyto sítě oddělit.

Následující schéma (Obr. 5), zobrazuje umístění firewallu v konkrétní síti. Základem této topologie je firewall, demilitarizovaná (DMZ), privátní síť a veřejná síť s připojením k Internetu. Brána firewall je umístěna mezi těmito sítěmi a disponuje třemi rozhraními, které tyto sítě fyzicky oddělují. Díky tomuto konceptu oddělení, nastává izolace serveru umístěného v DMZ od privátní sítě. V případě narušení bezpečnosti ze strany serveru, dosáhne bezpečnost vyššího stupně kontroly a zabrání pokusu o vstup nechtěným uživatelům do privátní sítě. Na firewallu musí být striktně definovaná pravidla, jenž zajistí plynulý průchod mezi jednotlivými rozhraními a překlad portů služeb na serveru na cílovou adresu a daný port.



Obr. 5 Umístění firewallu v síti

II. PRAKTICKÁ ČÁST

5 NASAZENÍ FIREWALLU

Zařazení firewallu do sítě hraje podstatnou roli v bezpečnosti sítě. Avšak, aby firewall plnil svou funkci, je třeba správně zvolit topologii sítě v závislosti na finančních možnostech, poskytnutých službách a vytíženosti serverů. Vhodná topologie pro naše účely je nasazení jednoho firewallu se třemi síťovými kartami oddělující vnější síť, DMZ a LAN. Demilitarizovaná zóna je tvořena jako samostatná větev, jež bude vycházet z firewallu.

Praktické sestavení firewallu vychází zejména z uvedených požadavků v kapitole 4.6.2 Požadavky na firewall a výběru nástroje pro filtrování paketů, jenž je uveden v kapitole 4.5 Iptables. Na tomto základě, byla stanovena tato metodika sestavení firewallu:

1. Volba pracovního prostředí
2. Volba jádra a instalace
3. Zabezpečení firewallu
4. Konfigurace sítě
5. Zavedení pravidel iptables
6. Testování konfigurace firewallu

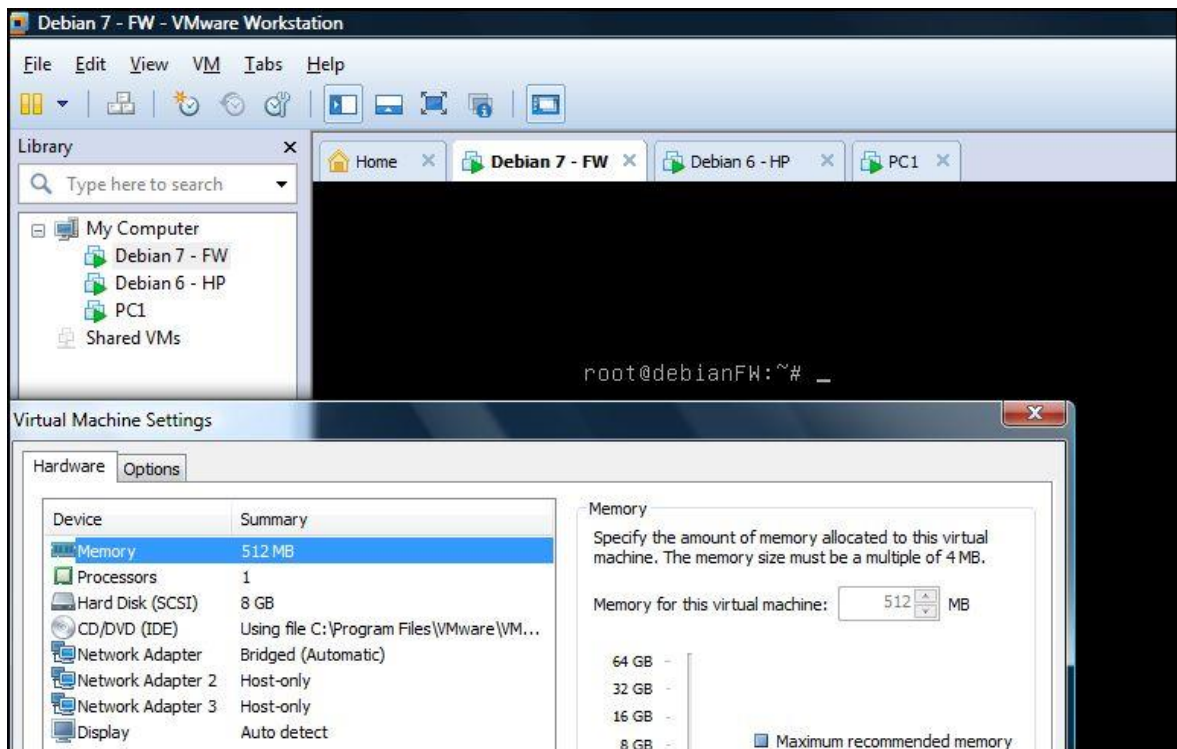
5.1 Volba pracovního prostředí

Jako pracovní prostředí pro tvorbu firewallu byl vybrán virtuální počítač realizován aplikací VMware Workstation⁹ verze 10. Jedná se o emulátor, pod kterým lze vytvářet virtuální komponenty a následně provést instalaci operačního systému. To vše lze realizovat za běhu Windows na hostitelském počítači. Takto je možné vytvořit více operačních systémů, provozovat mezi nimi virtuální síť, monitorovat funkčnost sítě či testovat viry bez ohrožení. Mezi další výhody této aplikace pro simulace operačních systémů přes virtuální prostředí patří i tyto:

- efektivní tvorba počítačových sítí
- jednoduché uživatelské prostředí

⁹ <http://www.vmware.com/cz/products/workstation>

- možnost připojení dalšího hardware, jako jsou síťová rozhraní nebo CD mechanika pro instalaci systému ze souboru s ISO obrazem
- možnost provozovat i náročnější grafické aplikace, díky instalaci speciálních balíčků VMware Tools, která jsou dostupná pro většinu operačních systémů
- zálohování vytvořených virtuálních stanic vykopírováním celé složky, obsahující simulovaný systém, na externí medium a funkční opětovné obnovení na jiném hostitelském počítači v plném rozsahu
- uložení aktuálního stavu systému bez nutnosti vypínání
- vzdálené připojení k pracovní stanici z jiného počítače nebo i chytrého telefonu



Obr. 6 Uživatelské prostředí a nastavení hardwarových komponent

Při tvorbě virtuálního stroje bylo ponecháno defaultní nastavení základních hardwarových komponent (RAM 512MB, 1x CPU - Central Processing Unit a 8GB HDD) a k prvnímu síťovému adaptéru byla přidána další dvě rozhraní. Jejich konfigurace byla provedena podle našich požadavků tak, že u prvního bylo zvoleno připojení k síťovému mostu, díky čemuž přijímá nastavení od reálného DHCP serveru na směrovači. Další dva byly nastaveny na „Host-only“, jenž umožňuje vytvořit izolované virtuální síť.

Uživatelské prostředí virtuálního prostředí a hardwarové nastavení komponent, je zobrazeno na předchozím obrázku (Obr. 6)

5.2 Volba jádra a instalace

Pro potřeby firewallu byl nainstalován pouze nutný základ operačního systému a standartní systémové nástroje. Konkrétně tedy Debian (7.4.0i386-netinst) o velikosti instalačního média do 300MB. Výhodou této instalace je, že si lze zvolit pouze aplikace, které jsou potřeba, a menu není plné nepoužívaných programů. Nutností je vlastnit iso obraz, který je samozřejmě volně ke stažení na oficiálních stránkách Debianu¹⁰ a v době instalace funkční připojení k internetu. Samotná instalace proběhla poměrně snadno, a není tedy nutné ji zde podrobně rozepisovat.

5.3 Zabezpečení Firewallu

Pro bezpečnou síť je žádoucí tento firewall patřičně zabezpečit, tedy nastavit aktualizování nainstalovaných balíčků (případně notifikaci o dostupných aktualizacích), nasadit silné heslo pro přístup do systému, vypnout nebezpečné služby a také nastavit vzdálený přístup přes SSH. Uvedenou problematikou se zabývá tahle kapitola.

5.3.1 Aktualizace

Pro možnost aktualizace systému, aplikací a dalšího rozšíření je nutné upravit zdroje softwaru. Ty jsou uloženy v souboru `/etc/apt/sources.list`. Změna byla provedena editací souboru `sudo nano /etc/apt/sources.list` a vložení následujících řádků:

```
deb http://ftp.debian.org/debian/ wheezy main contrib non-free
deb-src http://ftp.debian.org/debian/ wheezy main contrib non-free
deb http://security.debian.org/ wheezy/updates main contrib non-free
deb-src http://security.debian.org/ wheezy/updates main contrib non-free
```

Samotná aktualizace je následně jednoduše proveditelná s použitím dvou příkazů: `sudo aptitude update` a `sudo aptitude upgrade` (případně `safe-upgrade` nebo `full-upgrade`). Pro

¹⁰ <https://www.debian.org/CD/netinst/>

zautomatizování kontrol nových aktualizací byl doinstalován program apticron, který zjistí dostupnost nových aktualizací a notifikuje o nich na zadaný email. Instalace proběhla příkazem `sudo aptitude install apticron` a konfigurace se provedla modifikací souboru `apticron.conf` v adresáři `/etc/apticron/` a to následovně:

- `EMAIL="admin.it@gmail.com"` – zprávy se budou odesílat na zadaný e-mailový účet
- `NOTIFY_NEW="0"` – vypnutí upozorňování na dostupné aktualizace pro balíčky, jež nebyly nainstalovány
- `CUSTOM_SUBJECT="apticron aktualizace"` – umístí do předmětu přijaté zprávy zadaný řetězec. Tato volba zjednoduší následné vyhledávání obdržených notifikací v emailovém klientovi.
- `CUSTOM_FROM="spravce.it@gmail.com"` – změna defaultního odesílatele na relevantní emailovou adresu.

5.3.2 Kontrola síťových služeb

Vzhledem k tomu, že firewall bez správného ošetření bezpečnosti a vypnutí nepotřebných služeb, neplní dostatečně svou funkci, musí být tyto rizika eliminována, správným nastavením. K zjištění nepotřebných služeb posloužily příkazy:

- `netstat -atup`
Přepínač plní tuto funkci: a – zobrazí všechny běžící služby, t - protokol TCP, u - protokol UDP, p – název programu na daném portu.
- `ps -ax`
Dodá informaci o procesech. Přepínače ax zajistí vypsání všech procesů běžících v systému.
- `lsof -P -i -n -s TCP:LISTEN`
Příkaz zobrazí všechny otevřené soubory v síťové komunikaci a kontroluje otevřené porty na hostu.

Byly vypnuty služby `exec`, `talk`, `finger`, `exim` a `telnet`. Služby poté byly testovány použitím stejných příkazů a již žádné nenaslouchaly. Vzhledem k definování IP adres manuálně, není nutný ani `dhclient`.

5.3.3 Autentizace

Autentizace jednotlivých uživatelů je řešena pomocí modulu PAM (Pluggable Authentication Modules), jež umožňuje poměrně snadno modularizovat autentizační procedury. Ten se skládá ze dvou součástí: První jsou sady knihoven (modulů), které se mohou dynamicky připojovat a typicky se instalují do složky *usr/lib/security* (případně do */lib/security*). Další důležité součásti, na nichž tento systém běží, jsou konfigurační soubory, a ty jsou defaultně instalovány do */etc/pam.d/*.

Plná kontrola nad tímto systémem je spjata s konfiguračním souborem */etc/pam.d/login*, kam se umístí informace o druhu ověřování, kterým se mají přihlašující uživatelé podrobit, než mohou přistupovat k jejich shellu.

Heslová politika pro přístup k systému byla zvolena klasickým, a to následovně: Heslo musí být opatřeno minimálně 15 znaky skládající se z minimálně jednoho velkého a jednoho malého písmena, jednu číslici a jeden znak. Pro zadání nového hesla má uživatel pouze tři pokusy.

Aby byla naše navrhnutá heslová politika zavedena do systému, bylo potřeba doinstalovat modul *pam-passwdqc*. Ten se nachází v balíčku *libpam-passwdqc*. Instalace byla provedena příkazem *aptitude install libpam-passwdqc*. Následná konfigurace souboru */etc/pam.d/common-password* zařadila tento modul k použití a zavedla patřičná oprávnění přidáním následujícího řádku:

```
Password required pam_passwdqc.so min=disabled,disabled,disabled, 15, 12
enforce=everyone retry=3
```

Význam parametrů:

- *min=disabled,disabled,disabled, 15, 12* – nastavuje heslo složené alespoň ze čtyř typů znaků. Číslo 15 určuje délku hesla a 12 délku náhodně vygenerované heslové fráze *passphrase*.
- *enforce=everyone* – heslová politika vynucena pro všechny uživatele
- *retry=3* – umožňuje uživatelům tři pokusy zadání správně sestaveného hesla dle zadaných kritérií

Podrobnou dokumentaci lze nalézt v */usr/share/doc/libpam-passwdqc*.

5.3.4 Vzdálený přístup – SSH

Pro vzdálený přístup k firewallu byl doinstalován nástroj OpenSSH příkazem `aptitude install openssh-server`. Poté byl konfigurován hlavní konfigurační soubor daemonu OpenSSH, jenž je umístěn v `/etc/ssh/sshd_config`, a to následovně:

- `IgnoreRhosts yes` – udává, že nebude použito výhradně ověřování `rhosts`
- `RhostsAuthentication no` – udává, zda je ověřování `rhosts` dostačující pro ověřování uživatelů
- `PermitRootLogin no` – zakazuje přímé přihlášení pomocí uživatele `root`
- `RSA Authentication yes` – povoluje ověřování RSA
- `StrictModes yes` – zakazuje přístup pro zápis do domovských adresářů ostatním uživatelům
- `AuthorizedKeysFile %h/.ssh/authorized_keys` – implicitně nastaví uložení klíčů
- `PasswordAuthentication no` – nastavuje metodu ověřování pouze na povolení RSA klíčů a zakazuje použít ověřování heslem

Výše vypsané zákazy a omezení jsou dalším přínosem pro zabezpečení komunikačního protokolu `ssh`. Avšak aby mohl honeypot volně přistupovat k firewallu a přidávat další pravidla do `iptables`, bude se zabezpečeně připojovat přes `ssh` bez potřeby zadávání hesla. Tuto potřebu lze vykonat pomocí metody autentizace RSA klíčů.

```
root@debianhp:~# ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/root/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_dsa.
Your public key has been saved in /root/.ssh/id_dsa.pub.
The key fingerprint is:
12:05:a8:dc:13:25:e1:cd:58:a2:ce:6c:17:7e:a8:9c root@debianhp
The key's randomart image is:
+--[ DSA 1024]-----+
  |          .          |
  |   ooB .           |
  |  ..o+ .+         |
  | +0.00 .          |
  | = +.o S          |
  | 0 + . .          |
  | E                 |
  |-----+
root@debianhp:~# ssh-copy-id -i .ssh/id_dsa.pub 192.168.0.1
```

Obr. 7 Výstup programu `ssh-keygen`

Protokol ssh využívá při přístupu na vzdálený stroj asymetrické kryptografie. Autentizaci lze provést pomocí dvojici vygenerovaných klíčů. Ty musí být vytvořeny na každé klientské stanici zvlášť. K vytvoření těchto klíčů posloužil program *ssh-keygen*. Příkazem *ssh-keygen -t dsa* byla spuštěna na honeypotu tvorba soukromého a privátního klíče. Při tomto procesu bylo nutné zadat i tzv. passphrase, což je heslo sloužící k zašifrování uloženého soukromého klíče. Oba klíče se uložily do vybraných souborů. Privátní do */root/.ssh/id_dsa* a veřejný do */root/.ssh/id_dsa.pub*. Veřejný klíč se nakopíroval na stroj s firewallem příkazem *ssh-copy-id -i .ssh/id_dsa.pub 192.168.0.1*. Celý postup i s vygenerovaným klíčem je znázorněn na Obr. 7.

```
KeyChain 2.6.8; http://www.gentoo.org/projeck/keychain/
Copyright 2002-2004 Gentoo Foundation; Distributed under the GPL

* Found existing ssh-agent (1258)
* Known ssh key: /root/.ssh/id_rsa

root@debianhp:~# ssh 192.168.0.1
Linux debianFW 3.2.0-4-686-pae #1 SMP Debian 3.2.54-2 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr  6 08:36:21 2014
root@debianFW:~# _
```

Obr. 8 Nástroj keychain

Avšak vzhledem k tomu, že bylo potřeba nastavit, aby se mohl honeypot automaticky přihlašovat zabezpečeně na firewall, musel se využít nástroj *keychain*. Využití tohoto nástroje pro automatické přihlašování bez hesla, přináší své výhody v tom, že si sám vyžádá hesla ke klíčům, o které se má starat a následně v případě potřeby, spustí i *ssh-agenta*. Aby bylo zavedeno jeho spuštění před každým startem systému, postačilo do souboru *\$HOME/.bashrc* zapsat následující příkazy:

- */usr/bin/keychain \$HOME/.ssh/id_rsa*
- *Source \$HOME/.keychain/\$HOSTNAME-sh*

Po odhlášení a opětovném přihlášení byl proveden test spojení přes ssh. Nejprve se nástroj zeptal na heslo a na povolení k přidělení pro *keychain*. Po úspěšném zadání bylo zařízení přihlášeno. Další logování již proběhlo bez hesla (viz Obr. 8)

5.4 Konfigurace síťových rozhraní

V současných distribucích je nastavení rozhraní relativně jednoduchá záležitost. Jelikož jádro již obsahuje všechny potřebné ovladače, které automaticky aktivuje vzhledem k zjištěným připojeným zařízením, stačí jen vyplnit potřebné údaje. Aby byla nastavena síťová rozhraní dle navrhovaného schématu, byl proveden zápis IP adres staticky. Eth1 bylo nakonfigurováno pouze pro honeypot a eth2 pro LAN se stejným rozsahem sítě. Nastavení jednotlivých rozhraní demonstruje následující tabulka.

Tab. 1 – nastavení rozhraní firewallu

Rozhraní	IP adresa	Brána	Popis směru
Eth0	10.0.0.1	10.0.0.254 (router)	WAN
Eth1	192.168.0.1	/	HONEYPOT
Eth2	192.168.2.1	/	LAN

V Linuxu se nastavuje statická IP adresa úpravou souboru *interfaces* v adresáři */etc/network*. Po této úpravě je důležité restartovat síťovou službu, aby se projevil změny, pomocí příkazu */etc/init.d/networking restart*. Po restartu sítě se mi nepodařilo inicializovat správné nastavení sítě, avšak restart systému uvedl vše do pořádku a plně funkčnosti. Kontrolu jsem vykonal příkazem *ifconfig / less*.

5.5 Sestavení skriptu iptables

5.5.1 Parametry jádra

Nastavením jednotlivých procesů jádra lze upravit chování netfilteru nebo dosáhnout vyšší úrovně zabezpečení. Konfiguraci lze provést zápisem hodnoty do souboru v procesu v */proc*.

Podvržení falešné IP adresy útočníkem je jedna z metod, kterou se snaží útočník firewall oklamat, aby se mohl dostat ke službám, které jsou obvykle vyhrazeny jen pro určité počítače z vnitřního adresního prostoru. Pro kontrolu paketů s obsahem zfalšovaných adres, jenž přichází na jednotlivá rozhraní, posloužilo zapnutí služby *rp_filtru* jednoduchým příkazem: *echo "1" > /proc/sys/net/ipv4/conf/eth0/rp_filter*.

Metoda TCP SYN cookies rychleji detekuje útoky typu SYN flood. Aktivace se provedla příkazem: *echo "1" > /proc/sys/net/ipv4/tcp_syncookies*.

Povolení přeposílání paketů z jednoho rozhraní na druhé, umožňuje příkaz: *echo "1" > /proc/sys/net/ipv4/ip_forward*

Aktivace funkce *rp_filtr* proti falšování IP adres byl zavedena do všech rozhraní pomocí smyčky: *for IFACE in /proc/sys/net/ipv4/conf/*/rp_filter; do echo "1" > \${IFACE} done*

5.5.2 Moduly jádra

V této části se nahrávají vestavěné moduly jádra do konfigurace firewallu. Pro náš účel byly potřeba moduly např. pro inicializaci modulů, logování, zjištění stavu nebo omezení. Výpis všech nahrených modulů:

- */sbin/depmod -a*
- */sbin/modprobe ip_tables*
- */sbin/modprobe ip_conntrack*
- */sbin/modprobe iptables_filter*
- */sbin/modprobe iptables_mangle*
- */sbin/modprobe iptables_nat*
- */sbin/modprobe ipt_LOG*
- */sbin/modprobe ipt_state*
- */sbin/modprobe ipt_limit*

5.5.3 Deklarace proměnných

Vzhledem k velkému počtu pravidel ve skriptu iptables bylo vhodné, nadefinovat si některé proměnné pro zkrácení délky řetězce. Tato operace usnadní následné vypisování jednotlivých pravidel a znemožní i různé druhy překlepů, jež se mohou objevit při psaní adres nebo názvů rozhraní. Pokud se změní adresa některého rozhraní nebo počítače v síti, stačí přepsat danou proměnnou a úprava se projeví ve všech pravidlech, v nichž se tato konstanta nachází. Níže uvedené varianty konstant defínují cesty k iptables, adresy a názvy rozhraní u firewallu:

- *IPT="/sbin/iptables"*

- *LO_NIC*="lo"
- *LO_IP*="127.0.0.1"
- *NET_NIC*="eth0"
- *NET_IP*="10.0.0.1"
- *DMZ_NIC*="eth1"
- *DMZ_IP*="192.168.0.1"
- *DMZ_HP*="192.168.0.100"
- *LAN_NIC*="eth2"
- *LAN_IP*="192.168.2.1"

5.5.4 Zavedení pravidel iptables

Pravidla lze zadávat přes příkazový řádek s okamžitou funkčností. Avšak z důvodu možné ztráty po restartu je vhodnější zapisovat do skriptu a následně nastavit jeho automatické spouštění po restartu systému. Do souboru */root/rules.sh* byla zapsána pravidla potřebná k provozování firewallu.

Zrušení všech existujících pravidel a vytvořených řetězců:

- *\$IPT -F*
- *\$IPT -X*
- *\$IPT -t nat -F*
- *\$IPT -t filter -F*
- *\$IPT -t mangle -f*

Implicitní politika nastavena na „zahazuj vše, co není povoleno“, pro tabulku filter:

- *\$IPT -P INPUT DROP*
- *\$IPT -P OUTPUT DROP*
- *\$IPT -P FORWARD DROP*

Nejdříve byla povolena veškerá příchozí komunikace, pro virtuální rozhraní „loopback“. To je použitelné zejména pro funkčnost meziprocesové komunikace různých programů a služeb:

- *\$IPT -A INPUT -p ALL -i \$LO_NIC -j ACCEPT*

Firewall považuje vnitřní síť LAN za důvěryhodnější oproti DMZ a vnější síti NET. A tak pro tato rozhraní povoluje všechna ICMP příchozí pakety a vybrané porty, avšak s tím rozdílem, že se pro síť DMZ povolí přístup pouze pro spojení, jenž byla zahájena na firewallu. Pro honeypot je nutné, aby mohl při zapisování nových pravidel, používat ssh, a proto bylo zavedeno pravidlo, pouze pro přístup k tomuto portu, a to oběma sítím.

- *\$IPT -A INPUT -p ICMP -i \$LAN_NIC -j ACCEPT*
- *\$IPT -A INPUT -p TCP -i \$LAN_NIC --dport 22 -j ACCEPT*
- *\$IPT -A INPUT -p ICMP -i \$DMZ_NIC -j ACCEPT*
- *\$IPT -A INPUT -p TCP -i \$DMZ_NIC --dport 22 -j ACCEPT*
- *\$IPT -A INPUT -p ALL -d \$DMZ_IP -m state --state ESTABLISHED,RELATED -j ACCEPT*

V síti NET se kromě povolení příchozích paketů pro již existujících spojení a přístupu na port ssh nepovolují již všechny ICMP pakety, ale pouze ty servisní pakety, jež jsou důležité pro služby ping nebo traceroute. Navíc zde byla implementována u paketu typu „echo regist - 8“ ochrana proti DoS útoku „Ping of Death“. Usměrnění příchozích paketů na jeden za sekundu, a to maximálně pět po sobě jdoucích paketů, lze nastavit díky podmínkám modulu limit. Jako další znepríjemnění pro útočníka posloužilo omezení velikosti maximální délky paketu v rozmezí 28-92 bytů, což reálně dovoluje přijmout paket s velikostí 64bytů. Důvodem je, že 20 bytů zahrnuje záhlaví a 8 bytů je určeno pro adresy dané cesty, tedy zdrojovou a cílovou.

- *\$IPT -A INPUT -p TCP -i \$NET_NIC --dport 22 -j ACCEPT*
- *\$IPT -A INPUT -p ALL -i \$NET_NIC -m state --state ESTABLISHED,RELATED -j ACCEPT*
- *\$IPT -A INPUT -p ICMP -i \$NET_NIC --icmp-type 0 -j ACCEPT*
- *\$IPT -A INPUT -p ICMP -i \$NET_NIC --icmp-type 3 -j ACCEPT*
- *\$IPT -A INPUT -p ICMP -i \$NET_NIC --icmp-type 8 -m limit --limit 1/s --limit-burst 5 -m length --length 28:92 -j ACCEPT*
- *\$IPT -A INPUT -p ICMP -i \$NET_NIC --icmp-type 11 -j ACCEPT*

Vylepšená obrana proti útoku hrubou silou na SSH z vnější sítě navazuje na zabezpečení SSH v kapitole 5.3.4 Vzdálený přístup – SSH. První pravidlo vytvoří možnost spojení na port 22 naslouchající na firewallu a zároveň v případě, že je toto první příchozí, je mu přiřazeno označení SSH_HOST. Pokud se tedy někdo pokusí přihlásit 4x za minutu nebo více, je paket zalogován a je mu přiřazen daný prefix. Doplnění modul recent zaznamenává zdrojovou adresu a jeho vzdálenost, což je vhodné když se tento záškodník pokouší s podvrženou adresou danou službu omezit. Poslední pravidlo je terminující.

- *\$IPT -A INPUT -i \$NET_NIC -p TCP --dport 22 -m state --state NEW -m recent --set --name SSH_HOST -j ACCEPT*
- *\$IPT -A INPUT -i \$NET_NIC -p TCP --dport 22 -m recent --update --seconds 60 --hitcount 4 --rttl --name SSH_HOST -j LOG --log-prefix "utok_ssh_brute_force"*
- *\$IPT -A INPUT -i \$NET_NIC -p TCP --dport 22 -m recent --update --seconds 60 --hitcount 4 --rttl --name SSH_HOST -j DROP*

Pokud se paket pokouší navázat spojení bez příznaku SYN, bude zahozen.

- *\$IPT -A INPUT -p tcp --tcp-flags SYN,ACK SYN,ACK -m state --state NEW -j REJECT --reject-with tcp-reset*
- *\$IPT -A INPUT -p tcp ! --syn -m state --state NEW -j DROP*

Nakonec pravidel bloku s řetězcí INPUT bylo přidáno logování příchozích zamítnutých paketů s prefixem, jež poslouží pro lepší orientaci při čtení logu a řešení problémů.

- *\$IPT -A INPUT -m limit --limit 12/h -j LOG --log-prefix "zakázané pakety INPUT"*

Odchozí spojení byla pro všechna rozhraní povolena, jen pakety do sítě NET mohou odcházet na vybrané porty. Konec bloku opět loguje nevyhovující pakety jako u pravidel řetězce INPUT.

- *\$IPT -A OUTPUT -s \$LO_IP -p ALL -j ACCEPT*
- *\$IPT -A OUTPUT -o \$LAN_NIC -j ACCEPT*
- *\$IPT -A OUTPUT -o \$DMZ_NIC -j ACCEPT*
- *\$IPT -A OUTPUT -p icmp -j ACCEPT*
- *\$IPT -A OUTPUT -o \$NET_NIC -p TCP --sport 22 -j ACCEPT #SSH*

- *\$IPT -A OUTPUT -o \$NET_NIC -p UDP --dport 53 -j ACCEPT #DNS UDP*
- *\$IPT -A OUTPUT -o \$NET_NIC -p TCP --dport 53 -j ACCEPT #DNS TCP*
- *\$IPT -A OUTPUT -o \$NET_NIC -p TCP --dport 80 -j ACCEPT #WWW*
- *\$IPT -A OUTPUT -o \$NET_NIC -p TCP --dport 443 -j ACCEPT #HTTPS*
- *\$IPT -A OUTPUT -m limit --limit 12/h -j LOG --log-prefix "zakázané pakety OUTPUT"*

Abychom docílili žádané komunikace oběma směry mezi jednotlivými sítěmi s různým stupněm důvěryhodnosti, musí být nadefinována pravidla i v řetězci FORWARD. Díky tomu mohou tyto sítě komunikovat na základě udělených oprávnění. Směrování paketů vstupujících na rozhraní LAN_NIC není nijak omezeno. Povolen je i směr z DMZ do NET. Avšak vstup do sítě LAN je možný pouze na základě spojení navázaných zevnitř, a proto by neměla být ze strany DMZ možnost, například „pingnout“ na zařízení ve vnitřní síti.

- *\$IPT -A FORWARD -p ALL -i \$LAN_NIC -j ACCEPT*
- *\$IPT -A FORWARD -p ALL -i \$DMZ_NIC -o \$LAN_NIC -m state --state ESTABLISHED,RELATED -j ACCEPT*
- *\$IPT -A FORWARD -p ALL -i \$DMZ_NIC -o \$NET_NIC -j ACCEPT*

Na rozhraní NET je ještě přidáno pravidlo pro pakety mířící k službám v honeypotu. Pro vyšší přehlednost byl vytvořen nový řetězec „*forward_dmz*“, který definuje stav NEW a umožňuje povolit přístup k povoleným portům v tomto směru.

- *\$IPT -A FORWARD -i \$NET_NIC -o \$LAN_NIC -m state --state ESTABLISHED,RELATED -j ACCEPT*
- *\$IPT -A FORWARD -i \$NET_NIC -o \$DMZ_NIC -m state --state ESTABLISHED,RELATED -j ACCEPT*
- *\$IPT -N forward_dmz*
- *\$IPT -A FORWARD -i \$NET_NIC -o \$DMZ_NIC -m state --state NEW -j forward_dmz*
- *\$IPT -A forward_dmz -p ICMP -j ACCEPT*
- *\$IPT -A forward_dmz -p TCP --dport 80 -j ACCEPT*

- *\$IPT -A forward_dmz -p UDP --dport 443 -j ACCEPT*
- *\$IPT -A forward_dmz -p TCP --dport 443 -j ACCEPT*

Logování přeposílaných paketů.

- *\$IPT -A FORWARD -m limit --limit 12/h -j LOG --log-prefix "zakázané pakety FORWARD"*

Další pravidlo je vytvořeno za účelem překládání adres podle určeného vzoru příjemce pro odchozí pakety do sítě Internetu. Pravidlo funguje tak, že pokud paket opouští firewall přes rozhraní NET, tak se jeho původní adresa nahradí za adresu v rozsahu této sítě. Aby pakety věděli, jakou cestou se mají dostat zpět k odesílateli, provede se automaticky v tabulce mapování.

- *\$IPT -t nat -A POSTROUTING -o \$NET_NIC -j MASQUERADE*

Posledním blokem pravidel bylo umožněno zařízením v sítích NET a LAN komunikovat s honeypotem na vybraných portech.

- *\$IPT -t nat -A PREROUTING -p TCP -i \$NET_NIC --dport 80 -j DNAT --to-destination \$DMZ_HP*
- *\$IPT -t nat -A PREROUTING -p TCP -i \$NET_NIC --dport 443 -j DNAT --to-destination \$DMZ_HP*
- *\$IPT -t nat -A PREROUTING -p TCP -i \$LAN_NIC --dport 80 -j DNAT --to-destination \$DMZ_HP*
- *\$IPT -t nat -A PREROUTING -p TCP -i \$LAN_NIC --dport 443 -j DNAT --to-destination \$DMZ_HP*

5.5.5 Integrace firewallu do systému

Iptables i jeho předchůdci řeší potřebu vrácení mnoha pravidel do jádra při každém restartu systému. Jelikož jádro nemá žádné uložisko nezávislé na napájení, je potřeba znovu inicializovat vytvořené zásady filtrování paketů po každém nastartování systému. Proto musí být pravidla umístěny do souboru skriptu a poté do */etc/network/if-pre-up.d/*.

Následující blok zobrazuje příkazy skriptu *iptablesload_skript*, jenž vytvoří soubor */etc/network/if-pre-up.d/iptablesload* a začlení do něj pomocí příkazu *echo* níže uvedené

řádky. Jejich funkce spočívá v provedení načtení konfigurace ze souboru *iptables.rules.sh*, nastavení iptables a zapnutí funkce pro přeposílání paketů, kterou je nutné po restartu rovněž nastartovat. Poslední řádkem se už jen přidala práva souboru pro spuštění.

- `#!/bin/sh`
- `/sbin/iptables-restore < /etc/iptables.rules`
- `echo "1" > /proc/sys/net/ipv4/ip_forward`
- `chmod +x /etc/network/if-pre-up.d/iptablesload.sh`

Konečnou, avšak velmi podstatnou částí, bylo vsunutí příkazu *iptables-save > /etc/iptables.rules* do souboru pravidel *rules.sh*. Díky tomuto zásahu, se po zahájení operací ve skriptu, vytvoří soubor */etc/iptables.rules* generovaný systémem a z něj pak bude iptables načítat při spouštění systému.

Výstupem zpracování firewallu jsou dva skripty *iptablesload_skript.sh* a *rules.sh*.

5.6 Testování konfigurace firewallu

Po navržení požadované konfigurace firewallu je potřeba otestovat, zda se nastavení skutečně chová dle navržených pravidel. Pro naše nastavení je nutné zejména ověřit, zda mezi jednotlivými sítěmi probíhá komunikace a že neprobíhá mezi prvky, u kterých probíhat naopak nemá. Testování lze provádět pomocí dalšího skriptu, ale pro naše účely zcela postačí test pomocí nástrojů ping a traceroute. Nakonec byl pokus o útok pomocí zahlcení pakety tzv „ping of death“.

5.6.1 Testování, zda zařízení komunikují skrz firewall a připojení k internetu

Pro větší přehled při testování komunikace nástrojem ping byl výsledek odezvy jednotlivých zařízení v různých segmentech sítě zaveden do tabulky *Tab. 2 – testování komunikace*. Díky tomu lze přehledně vidět komunikaci mezi nakonfigurovanými rozhraními jednotlivých zařízení v různých segmentech sítě. U Firewallu byla testována všechna rozhraní pro každý segment sítě, ke kterému náleží. Tedy např. pro PC1 v síti LAN s IP adresou 192.168.2.10, byl testován ping na firewallu na rozhraní eth2 s IP adresou 192.168.2.1. Jak lze z tabulky vyčíst, dle nastavení iptables je komunikace dle našich požadavků. Honeypot ze sítě DMZ se v případě narušení nedostane do sítě LAN, což je v tomto případě naprosto žádoucí. Dále bylo záměrně nastaveno, aby nekomunikoval

HOST v síti NET s počítači v síti LAN. Zároveň ani s honeypotem kvůli bezpečnosti. Honeypot se naopak na HOSTa dostane.

Tab. 2 – testování komunikace

Zařízení	IP adresa zařízení	Směr dotazu		Ping
Firewall (Eth0)	10.0.0.1	NET	10.0.0.254	OK
Firewall (Eth1)	192.168.0.1	HONEYPOT	192.168.0.100	OK
Firewall (Eth2)	192.168.2.1	PC1	192.168.2.10	OK
Honeypot v DMZ	192.168.0.100	Firewall	192.168.0.1	OK
		PC1	192.168.2.10	KO
		NET	10.0.0.254	OK
		HOST	10.0.0.10	OK
		WWW	77.75.72.3	OK
PC1 v LAN	192.168.2.10	Firewall	192.168.2.1	OK
		HONEYPOT	192.168.0.100	OK
		NET	10.0.0.254	OK
		HOST	10.0.0.10	OK
		WWW	77.75.72.3	OK
HOST v NET	10.0.0.10	Firewall	10.0.0.1	OK (pouze eth0)
		PC1	192.168.2.10	KO
		NET	10.0.0.254	OK
		HONEYPOT	192.168.0.100	KO
		WWW	77.75.72.3	OK

Pro připojení k Internetu se testovalo dotazem ping na adresu „Seznamu“, tedy 77.75.72.3. Aby fungoval dotaz ve smyslu doménové adresy www.seznam.cz, bylo potřeba upravit ještě konfigurační soubor etc/resolv.conf konkrétně tedy na *nameserver 10.0.0.254*, což nastavilo překlad adres při přístupu k Internetu z LAN.

V další části probíhal test spojení trasováním pomocí jednoduchého programu traceroute. Díky tomu lze zjistit, přes které stroje komunikace prochází. Následný obrázek dokazuje, že komunikace z PC1 v LANu při vyslání paketu na adresu 192.168.0.100, kde leží honeypot, prochází firewallem (viz Obr. 9).

```
root@bt:~# traceroute 192.168.0.100
traceroute to 192.168.0.100 (192.168.0.100), 30 hops max, 60 byte packets
 1 192.168.2.1 (192.168.2.1) 0.607 ms 0.518 ms 0.496 ms
 2 192.168.0.100 (192.168.0.100) 16.291 ms 16.244 ms 16.191 ms
```

Obr. 9 Traceroute

5.6.2 Testování ping of death

Testováno pomocí dvou konzolí příkazového řádku, z nichž při dotazu na příkaz `ping 10.0.0.1` byla na první zaznamenána běžná odpověď, avšak na druhém „Vypršel časový limit žádosti“.

Další test měl vyzkoušet omezení velikosti přijatých bytů. Zatímco první příkaz `ping 10.0.0.1 -t -l 64` zaznamenal pozitivní odpověď, u příkazu s vyšším počtem bytů se tak již nekonalo a rozhraní neodpovídalo. Test demonstruje *Obr. 10*.

```
C:\Users\fujitsu>ping 10.0.0.1 -t -l 64
Příkaz PING na 10.0.0.1 - 64 bajtů dat:
Odpověď od 10.0.0.1: bajty=64 čas < 1ms TTL=64
Odpověď od 10.0.0.1: bajty=64 čas < 1ms TTL=64

Statistika ping pro 10.0.0.1:
Pakety: Odeslané = 2, Přijaté = 2, Ztracené = 0 (ztráta 0%),
Přibližná doba do přijetí odezvy v milisekundách:
    Minimum = 0ms, Maximum = 0ms, Průměr = 0ms
Control-C
^C
C:\Users\fujitsu>ping 10.0.0.1 -t -l 65
Příkaz PING na 10.0.0.1 - 65 bajtů dat:
Vypršel časový limit žádosti.
Vypršel časový limit žádosti.
```

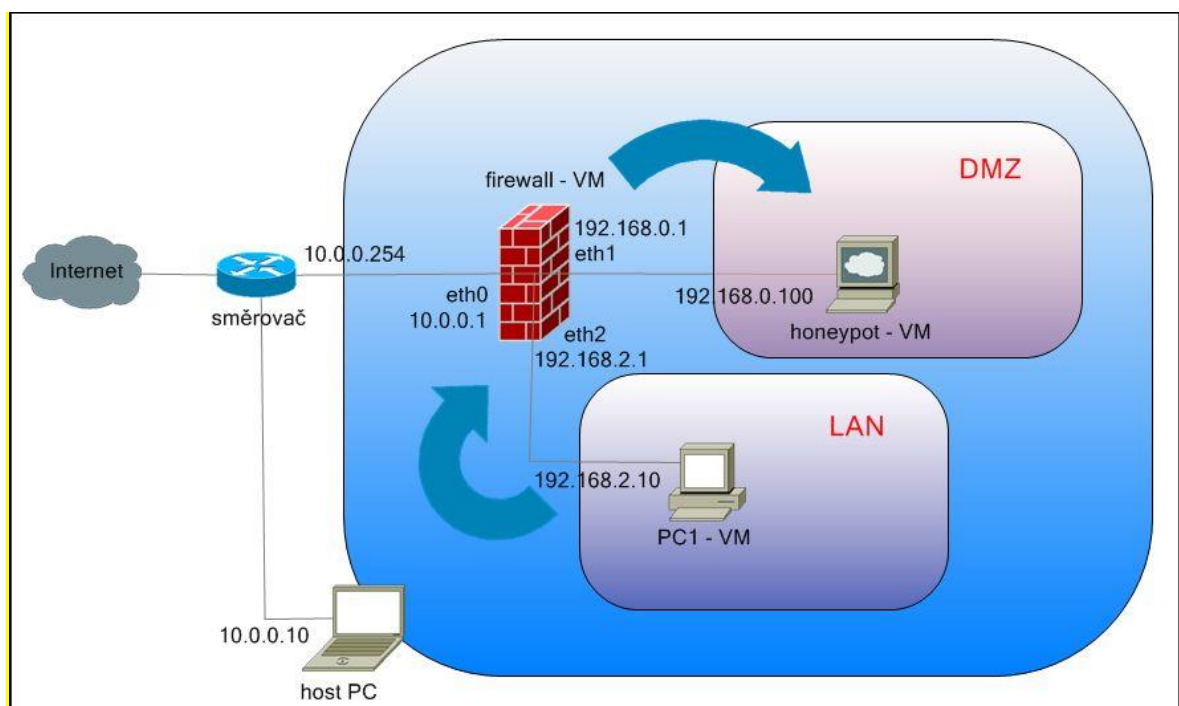
Obr. 10 Test ping of death

6 HONEYPOT A IMPLEMENTACE BROWSER EXPLOITATION FRAMEWORKU

6.1 Metodika

Základem obrany proti útočníkům je použití vhodných bezpečnostních prvků v síti a jejich správná konfigurace. Včasné odhalení vetřelce je důležité zejména z důvodu možnosti dalšího napadení, které může vést k narušení soukromí uživatele a odcizení citlivých dat. Proto je žádoucí nasadit do této sítě honeypot, který bude detekovat narušitele a do něj implementovat opatření, jež zajistí informace o útočnickovi.

Honeypot bude umístěn do demilitarizované sítě a bude kooperovat s firewallem. Pokud honeypot detekuje nějaký vstup, o němž lze předpokládat, že je nelegální, tak s pomocí aktivního protiopatření zjistí dostupné informace a zamezí tomuto narušiteli možnost dalšího vstupu, novým pravidlem na firewallu.



Obr. 11 Topologie sítě s IP adresami

K přístupu na honeypot vedou prakticky dvě cesty. Jedna ze sítě LAN a druhá z vnější sítě. Obě tyto cesty procházejí přes firewall, na němž jsou nastaveny pravidla pro pakety, pocházejících z jednotlivých segmentů sítě a jejich rozhraní. Popis sítě a IP adresy konkrétních prvků v této topologii jsou znázorněny na Obr. 11. Šipky definují možný směr útoku.

6.2 Honeypot

Honeypot je svým nastavením snadnější kořist pro vetřelce než ostré produkční systémy, ale je upraven menšími systémovými modifikacemi tak, aby veškerá aktivita byla zaznamenána a pro pozdější vystopování útoku. Měl by se jevit jako reálný systém a měl by se mu se všemi funkcemi přibližovat. Útočník nesmí mít pocit, že je sledován.

Hlavní myšlenka spočívá v tom, že vetřelec pronikne do systému a snaží se o vykonání nějaké operace. Honeypot tomu přihlíží a nechává jej do systému vstupovat opakovaně, přičemž během těchto návštěv jsou informace shromážděny a zaznamenávány.

Musí být pro vetřelce zajímavý. Ten musí mít pocit, že našel ten správný server, plný hodnotných dat. Měl by obsahovat nějaké falešné data, účty, které by měly být nasimulované. Je žádoucí, aby útočník na něm strávil co nejvíce času, aby si prošel systémem, a přitom nám poskytl co nejvíce informací o svých úmyslech.

Bezpečnost honeypotů musí být neustále sledována a ošetřena proti zneužití, které by se mohlo vymknout kontrole. Proto musí být dobře oddělen od lokální sítě, aby jej útočník nemohl použít jako výchozí bod pro další útoky, anebo sběr ostrých dat.

6.2.1 Výběr distribuce a honeypotu

K tomuto účelu posloužila nově vytvořená distribuce ANANSI s vestavěným honeypotem, použitelným jak pro produkční, tak pro výzkumné účely. Zvolená distribuce s honeypotem Dionaea vychází již z hotového řešení a je vhodná pro použití v této práci, jelikož splňuje následující požadavky:

- Operační systém Linux, kvůli snazší integraci a stability s licencí opensource.
- Možnost vzdáleného přístupu SSH v šifrované podobě.
- Grafická nadstavba GUI není potřebná.
- Emulace služeb, avšak v tomto případě budou firewallem zakázány.
- Logování (ukládání vzorků malware, textový záznam o útocích do databáze a upozorňování na mail).
- Možnost další úprav a rozšíření.
- Možnost konfigurace systému.

6.2.2 Instalace distribuce ANANSI

Distribuce byla nainstalována opět jako virtuální zařízení, pomocí vizualizačního softwaru VMware Workstation. Vzhledem k tomu, že operační systém nedisponoval grafickým prostředím, bylo dostačující použití velikosti 512MB paměti RAM a 1 procesor. Ostatní hardware jako tiskárny nebo zvukové karty byly odebrány, jelikož v tomto systému nebudou potřeba.

Umístění tohoto počítače bylo realizováno dle schématu topologie sítě *Obr. 11 Topologie sítě s IP adresami za firewall do demilitarizované sítě.*

Konfigurace síťových zařízení proběhla, jako v předchozím případě u firewallu, manuálně. O instalaci a konfiguraci se už není potřeba podrobně zmiňovat, jelikož tato otázka byla rozebrána v praktické části při nasazení firewallu v kapitolách 5.1 Volba pracovního prostředí a 5.4 Konfigurace síťových rozhraní. Vhodné je ovšem zmínit, že síťový adaptér byl nastaven jako Host-only, aby byla umožněna komunikace s firewallem ve stejném segmentu.

6.2.3 Logování událostí

Důležitou funkcí, kterou tento honeypot disponuje, je logování důležitých událostí a je vhodné ji v této části zmínit. Mezi původní vlastnosti softwaru Dionaea patří zaznamenávání informací o jednotlivých útocích, které byly na honeypot prováděny. Jsou zaznamenávány do souboru *logsql.lite*. Následně je umístěn vzorek staženého malware do adresáře */opt/dionaea/var/dionaea/binaries* a ten je možné dále zasílat na průzkum nebo z něj vycházet při tvorbě bezpečnostních aplikací.

Dionaea si vede log, v němž je možné spatřit podrobné informace o jejím běhu */opt/dionaea/var/log/dionaea.log*. V případě, že nastanou nějaké chyby při tomto běhu aplikace, jsou zalogovány do souboru */opt/dionaea/var/log/dionaea-errors.log*. Další systémové události již zaznamenává pomocí vestavěného nástroje rsyslog do adresáře */var/log*.

Pokud je v systému detekována nově vzniklá událost, odesílá se na interní mailbox, pomocí e-mailové komunikace. Takto je zobrazena notifikace nové zprávy při každém přihlášení uživatele do systému.

6.3 BEEF

6.3.1 Instalace

Instalace byla provedena manuálně vcelku složitým způsobem podle oficiálních stránek projektu BeEF pomocí následujících příkazů:

- *sudo apt-get update*

Nejprve bylo nutné aktualizovat systém a až poté nainstalovat potřebné závislosti. Již zde ovšem nastal problém odmítnutím dostupnosti. Byla provedena kontrola přístupu na Internet a dále souboru */etc/apt/sources.list*, ve kterém se nenacházely potřebné odkazy. Soubor byl tedy aktualizován o dva zmíněné odkazy:

- *deb http://ftp.cz.debian.org/debian/ squeeze main contrib non-free*
- *deb-src http://ftp.cz.debian.org/debian/ squeeze main contrib non-free*
- *sudo apt-get install curl git ruby build-essential libsqlite3-ruby libsqlite3-dev libssl-dev*

Po aktualizaci byla spuštěna instalace potřebných balíčků z repositáře.

- *sudo curl https://raw.githubusercontent.com/wayneeseguin/rvm/master/binscripts/rvm-installer | bash -s stable*

Nyní je pomocí příkazu *curl* stáhnout instalátor nástroje RVM (Ruby Version Manager), který umožní instalaci interpretů Ruby a přepínání mezi nimi. Tento skript se spustí s parametrem *stable* a nainstaluje danou verzi RVM.

- *source /home/[your_username]/.rvm/scripts/rvm* konkrétně tedy – *source /etc/profile.d/rvm.sh*

Provede se import příkazem *source* do současné relace bashe, což umožní používat RVM.

- *rvm pkg install zlib --verify-downloads 1*

S použitím *rvm*, instalace balíčku *zlib*.

- *rvm install ruby-1.9.3-p545*

Stažení a instalace dané verze ruby hlásilo chybu: „*curl: (7) could not connect to host*“ „*There was an error while trying to resolve rubygems version for 'latest'*“.

Halting the installation“. Pro nápravu musely být, dle oficiálních stránek, použity následující příkazy:

- *rm fix-permissions*
- *rm remove 1.9.3*
- *rm install 1.9.3 --rubygems 2.2.2*
- *rm use 1.9.3*

Nastavení stažené verze jako defaultní.

- *gem install bundler*

Instalace nástroje bundler byla taktéž chybová, neboť se instalátor nespustil a zobrazil hlášku:

ERROR: Could not find a valid gem 'bundler' (>=0), here is why:

Unable to download data from http://rubygems.org/ - Errno:ETIMEDOUT:

Connection timed out – connect(2) (https://api.rubygems.org/latest_specs.4.8.gz)

Poté byla testována funkčnost komunikace se zdrojovým serverem, příkazem:

- *curl -ILf https://s3.amazonaws.com*

Další zpráva vykazovala chyby s certifikáty SSL při připojování k serveru a dále se zjistilo, že je nekorektně nastaven datum. Byl aktualizován pomocí programu *ntpdate* a zkontolován příkazy:

- *apt-get install ntpdate*
- *ntpdate -v -b 0.it.pool.ntp.org*
- *date +%F*

Poté byl čas již v pořádku, byla provedena znovu aktualizace systému a následná reinstalace již umožnila v pořádku spustit další příkazy, které stáhly BeEF a nainstalovaly *bundler*:

- *git clone git://github.com/beefproject/beef.git*
- *cd beef*
- *bundle install*
- *ruby beef*

6.3.2 Spuštění a konfigurace

Poslední zmiňovaný příkaz framework nastartoval. Nicméně další spuštění lze učinit i příkazem `./beef`. (viz Obr. 12).

```
root@debianhp:~/beef# ruby beef
[17:58:07] [*] Bind socket [imapeudora1] listening on [0.0.0.0:2000].
[17:58:08] [*] Browser Exploitation Framework (BeEF) 0.4.5.1-alpha
[17:58:08] |   Twit: @beefproject
[17:58:08] |   Site: http://beefproject.com
[17:58:08] |   Blog: http://blog.beefproject.com
[17:58:08] |_  Wiki: https://github.com/beefproject/beef/wiki
[17:58:08] [*] Project Creator: Wade Alcorn (@WadeAlcorn)
[17:58:10] [*] BeEF is loading. Wait a few seconds...
[17:58:26] [*] 10 extensions enabled.
[17:58:26] [*] 202 modules enabled.
[17:58:26] [*] 2 network interfaces were detected.
[17:58:26] [+] running on network interface: 127.0.0.1
[17:58:26] |   Hook URL: http://127.0.0.1:3000/hook.js
[17:58:26] |_  UI URL:   http://127.0.0.1:3000/ui/panel
[17:58:26] [+] running on network interface: 192.168.0.100
[17:58:26] |   Hook URL: http://192.168.0.100:3000/hook.js
[17:58:26] |_  UI URL:   http://192.168.0.100:3000/ui/panel
[17:58:26] [*] RESTful API key: 1fcaf1e716f922104fd88aa0e0a260116532513c
[17:58:27] [*] HTTP Proxy: http://127.0.0.1:6789
[17:58:27] [*] BeEF server started (press control+c to stop)
```

Obr. 12 Start BeEF

Obrázek znázorňuje, že BeEF je spuštěn na dvou různých rozhraních, localhostu a na vnějším eth0, které bude zobrazeno návštěvníkovi této adresy na daném portu (v tomto případě 3000). Všechna tato nastavení a další, jsou nastavitelná pomocí souboru `config.yaml`, který je k nalezení v kořenovém adresáři programu. Dále je zde zobrazen odkaz na ovládací panel uživatelského rozhraní `http://192.168.0.100:3000/ui/panel`, na který by měl být přístup z libovolného počítače, na stejné lokální síti. Tento port musí být ovšem povolen na firewallu, což pro nastavení z lokální sítě je v tuto chvíli dostupné díky aktivnímu firewallu dle nastavených pravidel v souboru `rules.sh`.

Po zadání odkazu na lokálním počítači byla zobrazena již stránka BeEFu a k přístupu do hlavní nabídky posloužily přihlašovací údaje ve formě: `beef:beef`.

Samotné nastavení probíhá konfigurací hlavního souboru `./beef/config.yaml`. Nejdříve je nutná změna přihlašovacích údajů:

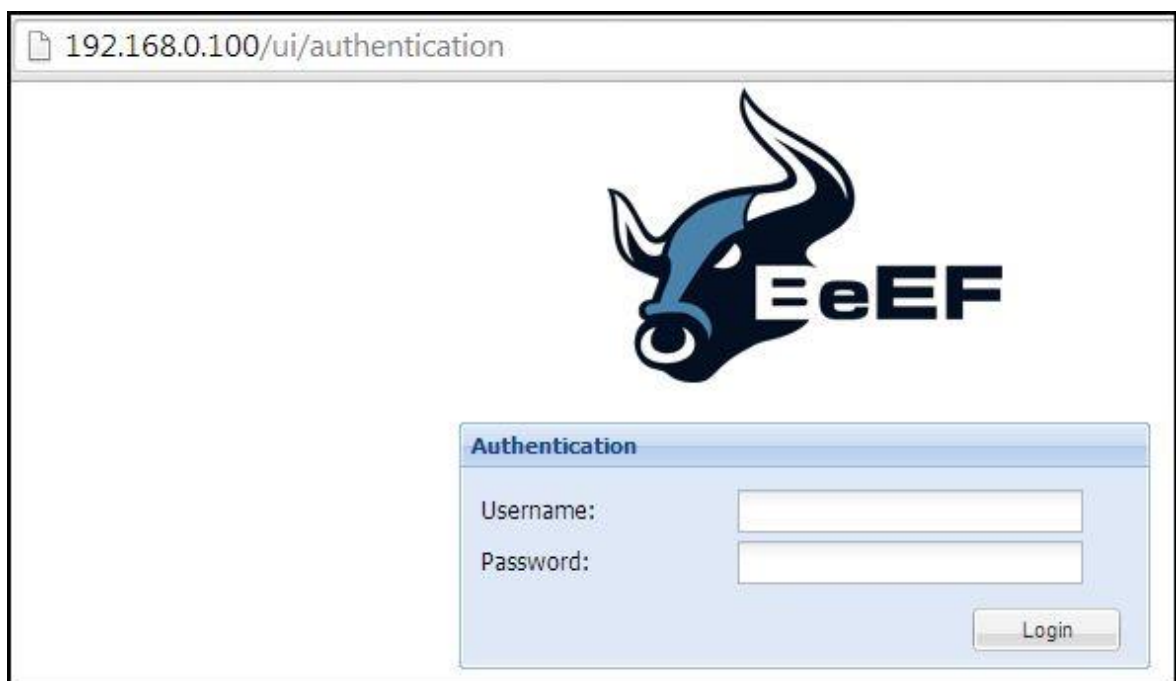
- user: “non-beef“
- passwd: “non-beef“

Dále změna portu “3000“ na “80“, aby měl útočník pocit, že navštěvuje webovou stránku spuštěnou na webové aplikaci.

V konfiguračním souboru `/opt/dionaea/etc/dionaea/dionaea.conf` honeypotu dionaea, byla vypnuta emulace služby http, která byla přenechána aplikaci BeEF. V sekci services byla odebrána „http“ služba. Následně tedy vypadal výpis services v souboru takto:

```
services = {  
  
    serve = ["https", "tftp", "ftp", "mirror", "smb", "epmap", "sip", "mssql"]  
  
}
```

Vložení `http://192.168.0.100/ui/panel` do vyhledavače, se otevře přihlašovací tabulka k aplikaci BeEF na honeypotu. (Obr. 13)



Obr. 13 Přihlášení BeEF

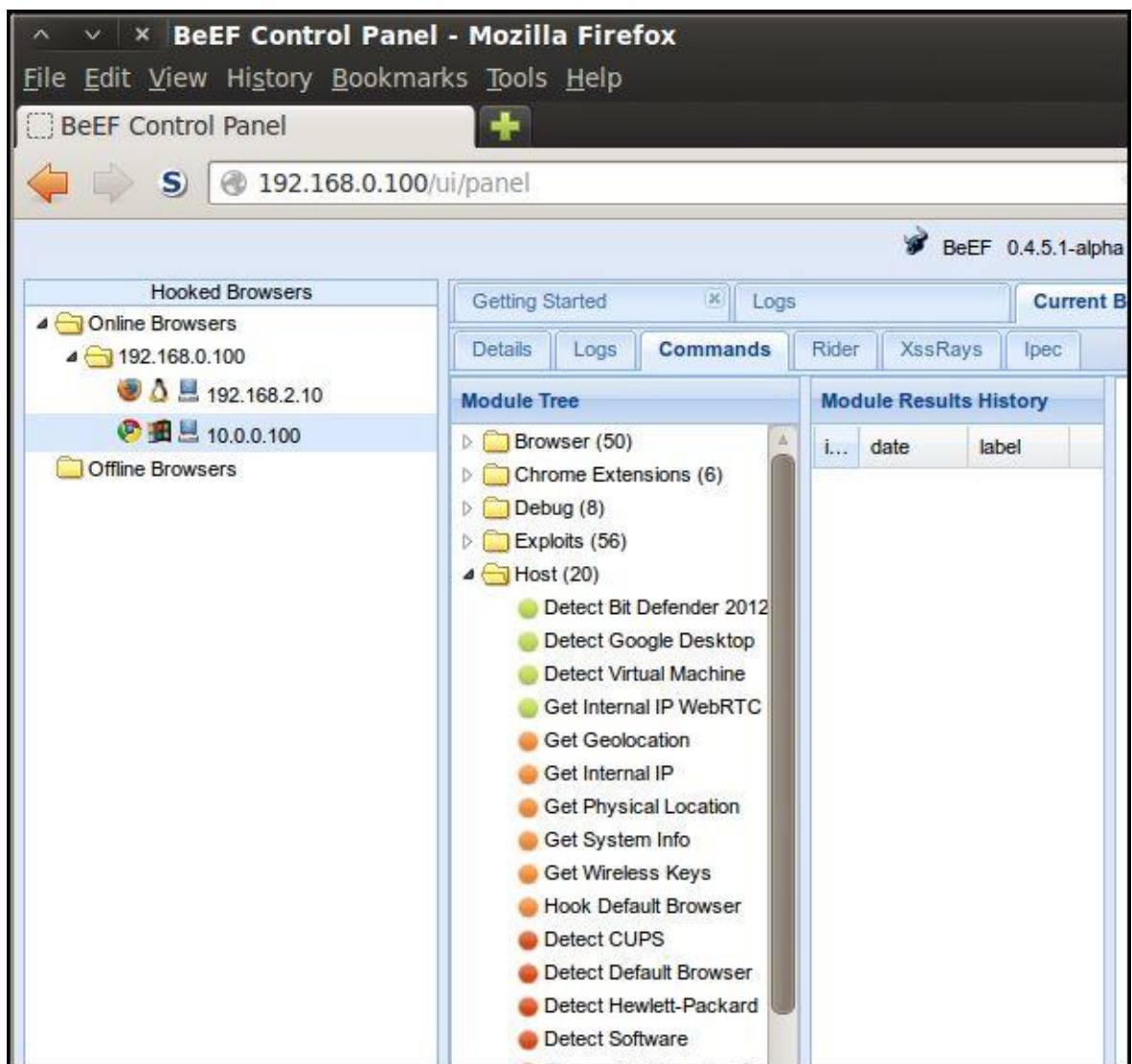
6.3.3 Hlavní nabídka

Po úspěšně provedené autentizaci je zobrazen hlavní panel aplikace. V levé části administračního rozhraní se zobrazí napadení uživatelé (zombie). Ti se dále dělí na „Online“ a „Offline“ dle aktuálního stavu užití prohlížeče. V pravé části je pak zobrazen log aktivity. Po kliknutí na konkrétní zombie, se zobrazí panel s dostupnými informacemi o tomto počítači. Prostřední panel zároveň ukazuje různé útočené funkce, které lze proti uživateli použít. Po přechodu do sekce „Commands“ lze vidět širokou škálu modulů. BeEF je schopen zjistit individuálně pro každý počítač, které moduly budou pracovat ve

využívaném prohlížeči. Proto tyto moduly označuje čtyřmi barevnými ikonami a každá z nich má svůj důležitý význam:

- Zelená – modul bude pro tento cíl fungovat a neměl by být uživatelem vidět.
- Oranžová – modul bude pro tento cíl fungovat, ale uživatel jej může zaznamenat. Vhodné použití pro moduly typu sociálního inženýrství.
- Bílá – modul ještě není pro tento cíl ověřen.
- Červená - modul nebude vůči tomuto cíli fungovat.

Hlavní nabídka i s aktivními zombies je zobrazena na *Obr. 14*.



Obr. 14 BeEF – kontrolní panel

6.3.4 Útoky pomocí BeEF

Jak již bylo zmíněno v teoretické části kapitola 3 BeEF (Browser Exploitation Framework), aby bylo možné zaútočit na prohlížeč, musí být nejprve realizován vstup na webové stránky, kde běží BeEF s JavaScriptem. Existuje celá řada způsobů, jak to udělat, avšak nejjednodušší je vložit na webové stránky tento skript, kde je umístěna IP adresa serveru a port, na kterém BeEF naslouchá:

```
<script src="http://192.168.0.100:80/hook.js" type="text/javascript"> </script>
```

Pro naše účely postačí i demo stránka aplikace, v níž je skript integrován a není potřeba ji měnit. Pak už stačí nějakým způsobem dostat uživatele na napadené stránky. Může být použita technika „man in the middle“ nebo sociální inženýrství, jako jsou telefonní hovory, emaily, nebo odkazy na společenských sítích. Dále pak lze použít celou řadu útoků pro zjištění informací o uživateli, a to pomocí funkcí samotné aplikace, kterých je obrovské množství nebo i s využitím dalších programů, jako je třeba známý Metasploit Framework.

Pokud se útočník připojí na server, bylo by vhodné použít sadu příkazů, které o něm zjistí dostupné informace, a to okamžitě po jeho příchodu, neboť server může rychle opustit. Příkazy lze zadávat ručně, ale mnohem lepší je implementovat do BeEFu nástroj, který po příchodu narušitele vykoná příkazy za nás. Jedná se o tzv. *autoran*, který se přidáním hodnotou *true* začlení do skriptu. a jeho použití přináší mimo jiné tyto výhody:

- když se prohlížeč stane zombie (vstoupí na napadené webové stránky), modul se automaticky spustí proti němu a vykoná svou akci, pro kterou byl naprogramován
- umožňuje spustit více modulů najednou
- moduly mohou být spuštěny chytře na základě informací o prohlížeči
- zahájí shromažďování informací okamžitě
- útok může být použit pro více cílených útoků

6.3.4.1 Konfigurace modulů

Každý modul se skládá ze tří základních souborů: *config.yaml*, *command.js* a *module.rb*.

config.yaml – konfigurační soubor popisující vlastnosti modulu. Obsahuje základní informace, jako je: název modulu, popis funkce modulu, kategorie modulu a prohlížeče, na které lze cíleně útočit.

Pro automatický start modulu musí být začleněna funkce *autorun* s hodnotou „*true*“. Níže lze vidět ukázkou zkonfigurovaného souboru v modulu „*Geolocation*“, jeden z automaticky nasazených modulů proti útočníkovi na server. V něm jsou obsaženy všechny zmíněné informace. Tento modul by měl fungovat na všech verzích internetových prohlížečů, což nastavuje hodnota „*All*“, u informace o cíli v příkazu „*working*“. V opačném případě, může být nastaveno „*not_working*“ nebo „*user_notify*“. Což znamená, že při použití tohoto modulu může být uživatel notifikován s nějakým vstupem na obrazovce a vyžaduje jeho potvrzení pro úspěšné dokončení vykonávané akce.

Tyto hodnoty jsou po nastavení zobrazeny v grafickém rozhraní jako barevné ikony, jež byly popisovány v kapitole 6.3.3 Hlavní nabídka Další možnosti definování cíle se zapisují písmeny jako: „*F*“ - Mozilla Firefox, „*O*“ – Opera, „*S*“ – Safari, „*IE*“ – Internet Explorer a „*C*“ – Google Chrome.

Soubor `config.yaml` u modulu „*Geolocation*“ tedy vypadá následovně:

```
# phonegap
```

- *beef:*
- *module:*
- *phonegap_geo_locate:*
 - *enable: true*
 - *category: "Phonegap"*
 - *name: "Geolocation"*
 - *description: "Geo locate your victim."*
 - *autorun: true*
 - *target:*
 - *working: ["All"]*

module.rb – další soubor umožní zařadit modul do rozhraní aplikace BeEF. Skládá se většinou z metod, které jsou konfigurována uživatelem. Například metoda *self.options*,

vrací matici, jež definuje data navržená pro uživatele, jako jsou velikosti použitých oken, jejich typy nebo název.

Metoda *post_execute* ukládá informace shromážděné pomocí skriptu v seznamu informací o napadeném prohlížeči. Soubor *module.rb* u modulu „Geolocation“ je zadán takto:

- *class Phonegap_geo_locate < BeEF::Core::Command*
- *def post_execute*
- *content = {}*
- *content['result'] = @datastore['result']*
- *save content*
- *end*
- *end*

command.js – posledním povinným souborem v modulu je tento JavaScript tzv. „payload“. Jedná se o důležitou část modulu, která by měla být zahrnuta ve funkci nazývané „*beef.execute*“. Kromě toho jsou zde možné další uživatelské úpravy. Tento příkaz by měl být použit pro návrat informací do BeEF kontroleru:

- *beef.net.send("<%= @command_url %>", <%= @command_id %>, "data");*

Dále je zobrazena část souboru opět u modulu „Geolocation“, která udává ve funkci sběru dat poptávané příkazy, jež vedou k dosažení žádaných dat o útočníkovi i s odkazem na webovou stránku a mapou na google.com:

- *beef.execute(function() {*
- *var onSuccess = function(position) {*
- *result =*
- *'Latitude: ' + position.coords.latitude + '\n' +*
- *'Longitude: ' + position.coords.longitude + '\n' +*
- *'Altitude: ' + position.coords.altitude + '\n' +*
- *'Accuracy: ' + position.coords.accuracy + '\n' +*

- `'Altitude Accuracy: ' + position.coords.altitudeAccuracy + '\n' +`
- `map = 'Map url: http://maps.google.com/?ll=' +`
- `position.coords.latitude + ',' + position.coords.longitude;`

6.3.4.2 Výběr modulů

Pro účely této práce bylo potřeba vybrat vhodné moduly, které pomohou získat co možná nejvíce informací vedoucím k dopadení nebo alespoň k lokalizaci útočníka, který neoprávněně vstoupil na honeypot. Moduly je vhodné volit tak, aby po opakujícím se vstupu bylo možné lépe určit totožnost narušitele, a proto dále v této práci bude demonstrován výběr modulů a důvody tohoto výběru:

browser_fingerprinting – popis identifikátorů generovaný z informací získaných z jednoho daného zařízení, které mohou být použity k identifikaci pouze pro tento jediný přístroj. V tomto případě pro prohlížeč, který je právě využíván ke sledování naše webové stránky, ověří otisk jeho verze kontrolou přítomnosti snímků prohlížeče.

get_system_info – první z nasazených modulů má za úkol získat celou řadu informací o hostitelském systému a použitém hardwaru, na kterém systém běží. Poskytuje údaje o počtu jader procesoru, velikosti použité a celkové paměti či režimy zobrazení obrazovky. Definuje použitý systém, jeho verzi i architekturu. Dále nabídne podrobnosti o aplikaci Java nebo jména síťových rozhraní a IP adresy. Korektní průběh modulu vyžaduje Java Applet a potvrzení od uživatele.

geolocation – velmi důležitý modul, jež se snaží o vyhledání geografické polohy cílového objektu. Výstupem je odkaz na webovou stránku pomocí google.maps. První spuštění vypsal chybu: *[INIT] Geolocation failed - Could not find MaxMind GeoIP database '/opt/GeoIP/GeoLiteCity.dat'*. Aplikace nabádala ke stažení z odkazu <http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz>. Oprava se tedy provedla následujícími příkazy, kterými byla stažena potřebná databáze, poté extrahována a následně přesunuta na požadovanou pozici v adresáři GeoIP:

- `wget -N http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz`
- `gunzip GeoLiteCity.dat.gz`
- `mv GeoLiteCity.dat /opt/GeoIP/`

Poté již proběhlo spuštění modulu v požadované funkci.

detect_popup_blocker - pop-up vyskakovací okna umožňují jednoduše zobrazit reklamu nebo použití nežádoucích a nekonečně otravných upoutávek. Tu lze vepsat do HTML kódu pomocí JavaScriptu a nastavit dle libosti tvůrce. Zjištění, zda má návštěvník tato okna povolena, vede k možnostem dalšího útoku a k vidině méně zabezpečeného prohlížeče či nezkušeného uživatele. Modul vyžaduje akci uživatele. Bez tohoto potvrzení je útočník upozorněn výsledkem „false“.

detect_toolbars – modul sloužící k detekci nástrojových lišt pro webové prohlížeče, jež činí jeho funkce a služby dostupnějšími. Většina běžně dostupných „toolbarů“ jsou zdarma a jejich základem je vesměs vyhledávání na Internetu, a to včetně obrázků nebo skupin.

Avšak jeho předností mohou být i jiné funkce jako je překlad slov mezi jazyky, kontrola pravopisu ve formulářích nebo blokování vyskakovacích oken. Má ovšem i stinné stránky. Může kupříkladu obsahovat infekci, která způsobí počítači velké nepříjemnosti. Mezi ně lze zařadit zobrazování nechtěných reklam na obrazovce, sledování historie počítače, ba dokonce odcizení osobních údajů.

Modul standardně detekuje nástrojové lišty od společnosti Bing, Google a Ask, ale lze přidat do skriptu i další často používané definice „toolbarů“. Díky takto získané informaci, lze lépe detekovat prohlížeč nepřítele, případně jej přeměřovat na škodlivé stránky s dalšími útočnými prvky.

detect_soc_nets – registrovaní členové mohou využívat dnes bezesporu nejpoužívanější služby Internetu, což jsou sociální sítě. Ty umožňují uživatelům vytvořit osobní profil, sdílet informace, fotografie a komunikovat mezi sebou.

Modul zobrazí používané sociální sítě uživatele. Je nastaven pro detekci těch nejčastějších jako je Facebook, Gmail či Twitter. Pokud uživatel síť využívá, může na něj být podniknut další útok v podobě získání přihlašovacích údajů ke zjištění totožnosti či přístupu na jeho osobní účet.

pretty_theft – po předchozím zjištění využívání služeb sociálních sítí je vhodné otestovat na uživateli modul, který zobrazí dialogové okno napodobující Facebook nebo Linked. Pokud bude zadáno do přihlašovacího pole jméno a heslo, získáváme okamžitě přístup k jeho osobnímu profilu na vybrané sociální síti.

detect_tor – modul zjistí, zda uživatel v současné době používá Tor. Jedná se o software využívaný k ukrytí identity uživatele při pohybu na Internetu.

7 ODVRÁCENÍ ÚTOKU

Když útočník navštíví na honeypotu webovou stránku, zobrazí se mu stránka nastražená aplikací BeEF. Ta bude sloužit jako prostředek k vykonání nastavených akcí aktivního protiopatření a poté bude útočník odstřižen. Vzhledem k tomu, že jeho IP adresa bude zachycena a odeslána na firewall s novým pravidlem, již pro něj nebude možné z této IP adresy k honeypotu nikterak přistupovat. Postupů, jak tento proces zrealizovat, je jistě mnoho, avšak v tomto případě byla provedena realizace tohoto úkolu takto:

Nejdříve byl přidán do operačního systému malý nástroj, který umožňuje spravovat více sezení a běží dál i v okamžiku, kdy je uživatel od počítače odpojen. Umí tedy spouštět virtuální obrazovky a jejich výstup ukládat do souboru. To je pro tento účel velmi důležité, neboť to povede k získání IP adres nového návštěvníka.

Byla tedy provedena instalace aplikace jménem *Screen* známým příkazem: `apt-get install screen`. Poté byla spuštěna nová relace obrazovky s názvem *beef*: `screen -S beef -L`. Tímto se zahájil záznam do souboru `/root/screenlog.0`.

Problémem bylo vytřídění nových IP adres každého návštěvníka a zároveň odstranění nežádoucích znaků. K tomu dopomohly příkazy *grep*, *awk* a *sed*. *Grep* načte ze standardního vstupu textová data a na základě nastavených kritérií vypíše požadované informace na zadaný výstup. Zde vykonává výpis řádků, které obsahují řetězec „New Hooked Browser“, jelikož takto si BeEF označuje všechny nové návštěvníky jeho stránky. *Awk* se postaral o výpis pouze šestého sloupce, jenž obsahuje řetězec s IP adresou a dalšími nežádoucími znaky, jako jsou znaky „*ip*“ nebo čárka. Výpis vypadal takto: „*ip:10.0.0.11,*“.

Nakonec byl použit editor *sed* sloužící mimo jiné k procházení vstupního souboru a k jeho modifikaci vzhledem k nastavení textových transformací. Tímto odstranil poslední znaky a do textového souboru *ip.txt*, byly zavedeny pouze IP adresy uloženy do sloupce. To vše uskutečnil následující řádek:

```
cat /root/screenlog.0 | grep "New Hooked Browser" | awk '{print $6}' | sed 's/ip://g' | sed 's/,//g' > ip.txt
```

V další části skriptu se odehrává získávání adres ze souboru *ip.txt* a odesílání nového pravidla nástroje *iptables* na firewall přes SSH. K tomu je nutné neomezené spojení s firewallem, které bylo řešeno v kapitole 5.3.4 Vzdálený přístup – SSH, a proto se touto problematikou již není nutno zabývat.

Řádek obsahuje standardní příkaz pro předání programu většího množství parametrů, v tomto momentě pole IP adres, uložených v souboru *ip.txt* a řetězec *iptables*, jež nastaví implicitně firewallu zákaz přístupu na rozraní *eth1* a port 80, konkrétnímu uživateli. Celý příkaz bude vypadat takto:

```
xargs -a ip.txt -I '{}' ssh 192.168.0.1 'iptables -I FORWARD -o eth1 -s {} -p tcp --
destination-port 80 -j DROP'
```

Kvůli zautomatizování celého procesu, je potřeba výše uvedený skript zadat do *cronu*, který toto načasované spouštění zajistí. Jeho úlohy jsou definované v souboru *crontab* a pro editaci slouží příkaz *crontab -e*. Spouštění souboru */root/scripts/ip.sh* v pravidelných intervalech bylo umožněno, přidáním tohoto řádku:

```
*/10 * * * * /root/scripts/ip.sh
```

Jednotlivé hvězdičky reprezentují časový údaj, který je zadáván číselnou hodnotou. První místo patří minutám, a proto zápis **/10* spouští skript každých deset minut.

Program *sleep* pozastaví vykonávání skriptu po dobu 5 minut a poté mu umožní pokračovat ve stanovených instrukcích. Hodnota je zadána v sekundách. Příkaz *sleep 300* byl vložen mezi příkaz pro výpis IP adres a odeslání pravidla na firewall, aby bylo možné vykonat akce jednotlivých modulů BeEFu a získat tak informace o útočnickovi.

Po konfiguraci vypadal proces spouštění honeypotu takto:

Nejdříve byl vykonán skript *run_dionaea* umístěný v adresáři */usr/bin/honeypot*. Poté se zadal příkaz *screen -S beef -L* k vytvoření nové relace aplikace *screen* pod názvem „beef“ se záznamem do souboru */root/screenlog.0*. Dále byl zapnut samotný BeEF příkazem *./beef -x*, s přepínačem *x*, který vymazal již vytvořená spojení mezi předchozími návštěvníky při testování aplikace a vyčkával na nově přichozí útočníky v aktivním režimu.

8 TEST ÚTOKU NA HONEYPOT

Testování probíhalo standardním způsobem. Nejdříve se zjišťovaly dostupné IP adresy v lokálním rozsahu, otevřené porty emulující běžící služby a poté byl zadán odkaz na webovou stránku běžící na honeypotu.

K tomuto účelu byly vytvořeny dva virtuální počítače s různými operačními systémy i prohlížeči. Počítače byly označeny jako PC1 a PC2. K ilustraci poslouží následující tabulka popisující konfigurace testovaných PC.

Tab. 3 – testované počítače

Počítač	IP adresa	Sít'	Operační systém	Prohlížeč
PC1	10.0.0.11	NET vnější síť	Windows 7 32 – bit	Internet Explorer
PC2	192.168.2.10	LAN vnitřní síť	Linux – BackTrack 5r3	Mozilla Firefox

1. Skenování IP adres a otevřených portů.

- a) V PC1 byl použit program Advanced Port scanner od společnosti Famatech. Ten umožňuje rychle a efektivně skenovat rozsah IP adres v lokální síti a to včetně otevřených portů.

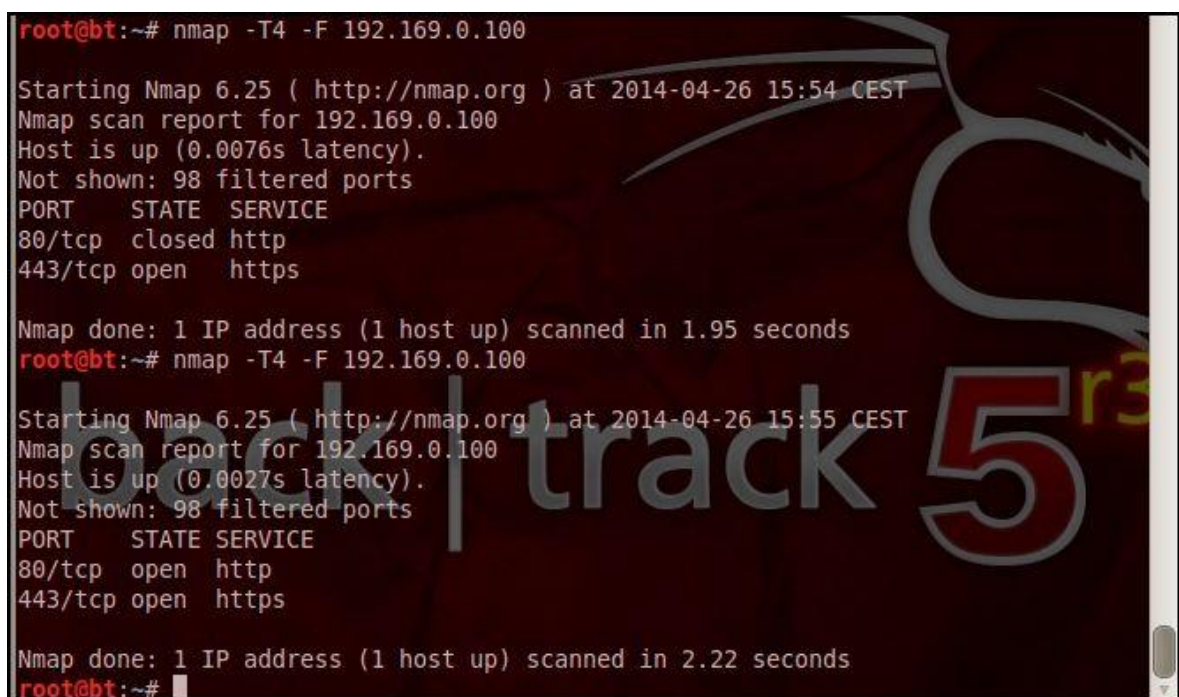
Po oskenování standardního rozsahu v lokální síti 192.168.0.1 až 192.168.0.255 – byl zjištěn počítač v síti s IP adresou 192.168.0.100 a s otevřeným portem 80 (testování bez spuštěné služby honeypotu). (viz Obr. 15)



Obr. 15 Skenování sítě a portů PC1

- b) V PC2 proběhlo rychlé skenování programem *nmap* (profilem Quick scan) na otevřené porty, a to s tím rozdílem, že v druhé fázi je spuštěn framework . (viz *Obr. 16*)

Výsledek ukázal, že firewall otevírá pouze porty 80 a 443, jak bylo nastaveno pomocí nástroje iptables. Bez filtrování firewallu jsou emulovány i ostatní služby na jiných portech, jako je FTP nebo DNS. V této práci je ovšem otevření těchto portů nežádoucí, a proto byly firewallem uzavřeny. To činí tuto síť ještě bezpečnější a minimalizuje možnosti dalšího útoku.



```
root@bt:~# nmap -T4 -F 192.169.0.100
Starting Nmap 6.25 ( http://nmap.org ) at 2014-04-26 15:54 CEST
Nmap scan report for 192.169.0.100
Host is up (0.0076s latency).
Not shown: 98 filtered ports
PORT      STATE SERVICE
80/tcp    closed http
443/tcp   open  https

Nmap done: 1 IP address (1 host up) scanned in 1.95 seconds
root@bt:~# nmap -T4 -F 192.169.0.100
Starting Nmap 6.25 ( http://nmap.org ) at 2014-04-26 15:55 CEST
Nmap scan report for 192.169.0.100
Host is up (0.0027s latency).
Not shown: 98 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https

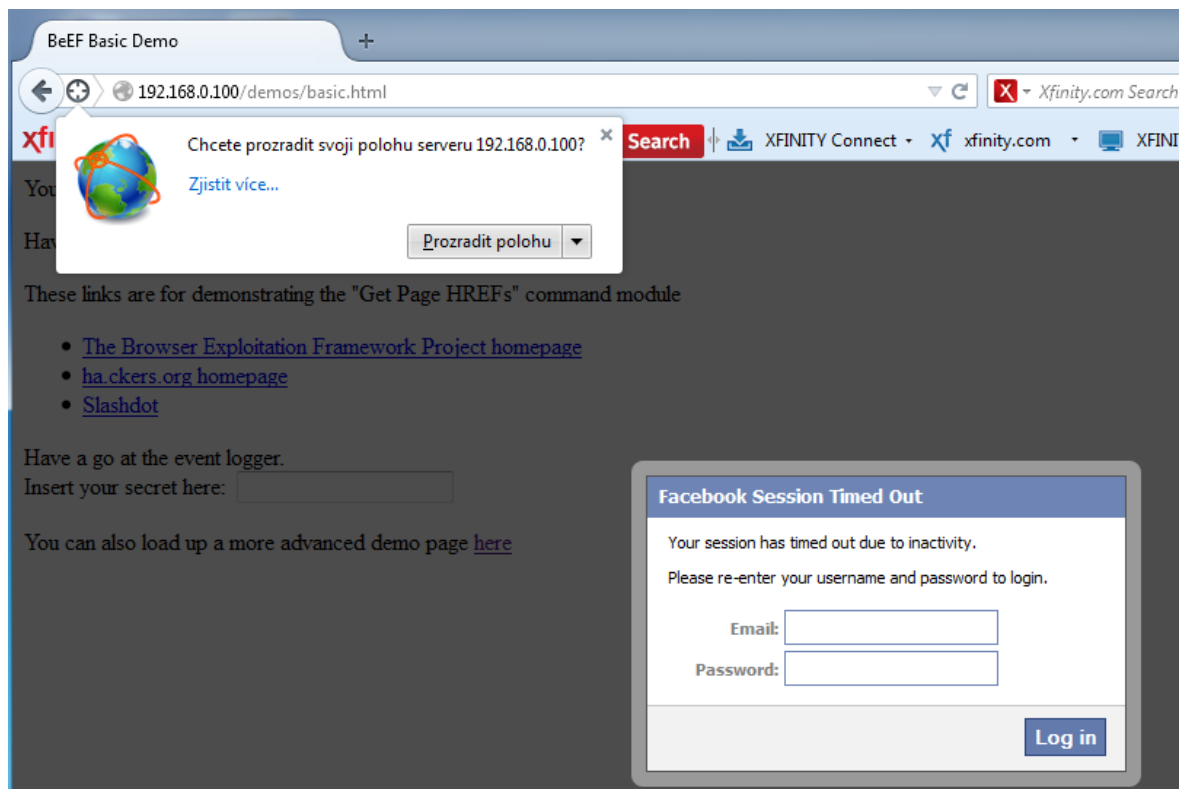
Nmap done: 1 IP address (1 host up) scanned in 2.22 seconds
root@bt:~# █
```

Obr. 16 Skenování sítě a portů PC2

2. Zadání adresy serveru honeypotu do prohlížeče.

V okamžiku zadání adresy honeypotu do prohlížeče byla načtena webová stránka aktivního protiopatření BeEF. Poté se zobrazilo upozornění, zdali je uživatel ochoten prozradit svoji polohu serveru 192.168.0.100 a přihlašovací okno k facebooku. Vše je zobrazeno na *Obr. 17*.

Během následujících deseti minut ztratili oba prohlížeče z obou sítí se serverem spojení a to již nebylo možné žádným způsobem navázat. Opětovný průzkum příkazem *nmap* zobrazil port 80 jako „*filtered*“.



Obr. 17 Stránka zobrazená útočníkem po přístupu na server

9 ZÍSKANÉ INFORMACE

9.1 Příklad útoku

V okamžiku, kdy se připojil útočník na webovou stránku, byly proti němu automaticky spuštěny nakonfigurované moduly v podobě sady příkazů, jak dokazuje Obr. 18.

```
[15:17:04] [*] BeEF server started (press control+c to stop)
[15:17:06] [*] New Hooked Browser [id:1, ip:192.168.2.10, type:FF-14, os:Linux], hooked domain [192.168.0.10:80]
[15:17:07] [*] New Hooked Browser [id:2, ip:10.0.0.11, type:IE-11, os:Windows 7], hooked domain [192.168.0.100:80]
[15:17:10] [*] Hooked browser [id:1, ip:192.168.2.10] has been sent instructions from command module [id:1, name:'Detect Social Networks']
[15:17:10] [*] Hooked browser [id:1, ip:192.168.2.10] has been sent instructions from command module [id:2, name:'Detect Tor']
[15:17:10] [*] Hooked browser [id:1, ip:192.168.2.10] has been sent instructions from command module [id:3, name:'Geolocation']
[15:17:10] [*] Hooked browser [id:1, ip:192.168.2.10] has been sent instructions from command module [id:4, name:'Fingerprint Browser (PoC)']
[15:17:10] [*] Hooked browser [id:1, ip:192.168.2.10] has been sent instructions from command module [id:5, name:'Detect Toolbars']
[15:17:10] [*] Hooked browser [id:1, ip:192.168.2.10] has been sent instructions from command module [id:6, name:'Detect Popup Blocker']
[15:17:10] [*] Hooked browser [id:1, ip:192.168.2.10] has been sent instructions from command module [id:7, name:'Pretty Theft']
[15:17:11] [*] Hooked browser [id:2, ip:10.0.0.11] has been sent instructions from command module [id:8, name:'Detect Social Networks']
[15:17:11] [*] Hooked browser [id:2, ip:10.0.0.11] has been sent instructions from command module [id:9, name:'Detect Tor']
[15:17:11] [*] Hooked browser [id:2, ip:10.0.0.11] has been sent instructions from command module [id:10, name:'Geolocation']
[15:17:11] [*] Hooked browser [id:2, ip:10.0.0.11] has been sent instructions from command module [id:11, name:'Fingerprint Browser (PoC)']
[15:17:11] [*] Hooked browser [id:2, ip:10.0.0.11] has been sent instructions from command module [id:12, name:'Detect Toolbars']
[15:17:11] [*] Hooked browser [id:2, ip:10.0.0.11] has been sent instructions from command module [id:13, name:'Detect Popup Blocker']
[15:17:15] [*] Hooked browser [id:1, ip:192.168.2.10] has executed instructions from command module [id:6, name:'Detect Popup Blocker']
```

Obr. 18 Reakce aktivního protiopatření na návštěvu útočníka

Log zobrazuje nově detekované návštěvníky stránky s jejich IP adresou, typem i verzí prohlížeče a operačním systémem. Poté jsou spuštěny nadefinované moduly s parametrem *autorun = true*.

Získané informace ukládá BeEF do svého graficky zpracovaného a uživatelsky přívětivého rozhraní v záložce *details*. Zde lze ihned po navázání spojení zkoumat spoustu informací o návštěvníkovi (viz Obr. 19). Nejenže byly zachyceny správné údaje o webovém prohlížeči, jako je jeho jméno, verze, jazyk, rozlišení okna, ale i informace o systému, na kterém prohlížeč běží, včetně bitové architektury, ba dokonce rozlišení obrazovky. V části komponenty prohlížeče jsou výsledky o přítomnosti dalších aplikací nebo rozšíření: Flash, VBScript, Silverlight, QuickTime RealPlayer, apod.

Detekce prohlížeče, komponent i operačního systému proběhla u PC1 i PC2 korektně dle očekávání.

Getting Started		Logs	Current Browser
Details		Logs	Commands
Rider		XssRays	Ipec
[-] Category: Browser (7 Items)			
Browser Name:	Firefox		Initialization
Browser Version:	14		Initialization
Browser UA String:	Mozilla/5.0 (X11; Linux i686; rv:14.0) Gecko/20100101 Firefox/14.0.1		Initialization
Browser Language:	en-US		Initialization
Browser Platform:	Linux i686		Initialization
Browser Plugins:	[]		Initialization
Window Size:	Width: 1064, Height: 673		Initialization
[-] Category: Browser Components (14 Items)			
[-] Category: Hooked Page (5 Items)			
[-] Category: Host (7 Items)			
Date:	Sat May 10 2014 12:11:06 GMT+0200 (CEST)		Initialization
Operating System:	Linux		Initialization
Hardware:	Virtual Machine		Initialization
CPU:	32-bit		Initialization
Default Browser:	Unknown		Initialization
Screen Size:	Width: 1076, Height: 845, Colour Depth: 24		Initialization
Touch Screen:	No		Initialization

Obr. 19 Záložka details ve frameworku

9.2 Výsledky nasazených modulů

Získané informace nasazených modulů lze zobrazit v záložce *commands* v jednotlivých kategoriích po kliknutí na daný modul. Výsledky dopadly následovně:

browser_fingerprinting – modul správně identifikoval oba prohlížeče.

- PC1 – *data: browser_type=Internet Explorer&browser_version=7+,10+*
- PC2 – *data: browser_type=Firefox&browser_version=1+,4-5,4+,10+,13+*

get_system_info – modul požádal o potvrzení na hostitelské stanici v podobě vyskakující okna v prohlížeči. V závislosti na potvrzení/nepotvrzení, se vykonala příslušná akce.

- PC1 - potvrzen. Vzhledem k rozsáhlosti uvedena pouze část výsledku:
 - *data: system_info=Available processors (cores): 1*
 - *Maximum memory (bytes): 259522560*

- *Free memory (bytes): 12922584*
- *Total memory (bytes): 16252928*
- *Default Screen: \Display0*
- *Display0 Mode: 1076x845 32bit @ 60Hertz*
- *OS Name: Windows 7*
- *OS Version: 6.1*
- *OS Architecture: x86*
- *Browser Name: sun.plugin*
- *Browser Version: 1.1*
- *Java Vendor: Oracle Corporation*
- *Java Version: 1.7.0_55*
- *Java Specification Version: 1.7*
- *Java VM Version: 24.55-b03*
- *Host Name: localhost*
- *Host Address: 127.0.0.1*
- PC2 – nepotvrzeno: bez výsledku.

geolocation – modul rovněž žádá o povolení získání polohy návštěvníka serveru. Výsledkem jsou údaje o lokaci, kde se útočník nachází.

- PC1 – potvrzeno. Výsledkem bylo:
data: result=Latitude: 50.083302 Longitude: 14.4667 Altitude: null Accuracy: 14000 Altitude Accuracy: null Heading: null Speed: null Timestamp: Sat May 10 2014 11:54:35 GMT+0200 (W. Europe Daylight Time) Map url: http://maps.google.com/?ll=50.083302,14.4667
- PC2 – nepotvrzeno: bez výsledku.

detect_popup_blocker – opět může být vyslaná žádost o reakci uživatele, zdali povolí vyskakující okna.

- PC1 – potvrzeno: *data: popup_blocker_enabled=true*

- PC2 – nepotvrzeno: *data: popup_blocker_enabled=false*

detect_toolbars

- PC1 – *data: toolbars= no toolbars detected*
- PC2 – *data: toolbars= no toolbars detected*

detect_soc_nets – v PC1 bylo provedeno v rámci testu přihlášení k Gmailu i službě Facebook, a to bylo zároveň i detekováno. V PC2 byl proces modulu bez výstupu, a proto se ani v dalším kroku nezjistily žádné přihlašovací údaje.

- PC1 – *data: gmail=User is authenticated to GMail&twitter=User is not authenticated to Twitter (response:timeout)&facebook=User is authenticated to Facebook*
- PC2 – neúspěšné

pretty_theft – na základě úspěšné detekce bude spuštěn příslušný modul s přihlašovacím oknem k vyzvání zadání přihlašovacích údajů k sociální síti Facebook. Výsledek dle zadání uživatele.

- PC1 – *data: answer=muj_email:heslo1234*
- PC2 – neúspěšné

detect_tor – Tor nebyl ani v jednom případě detekován, a to i přesto, že z PC2 byl přístup na stránku proveden přímo z něj.

- PC1 – *data: result=Browser is not behind Tor*
- PC2 – *data: result=Browser timed out. Cannot determine if browser is behind Tor*

9.3 Dílčí závěr testování aktivního protiopatření BeEF

Testování funkčnosti nástroje BeEF bylo v podstatě ve větší míře úspěšné a neodhalilo žádné známky nestandardních reakcí na příchod útočnicka. Kdykoliv byl detekován nějaký vstup, okamžitě se spustily jednotlivé moduly a výsledky zaznamenaly do logu ve webovém rozhraní.

Z výsledků modulů je patrné, že reakce prohlížečů jsou individuální dle nastavení a interakce uživatele a širokou škálu informací lze získat i bez povšimnutí útočnicka o tom, že jeho prohlížeč byl napaden.

ZÁVĚR

Hlavním cílem této práce bylo implementovat aktivní protiopatření do honeypotu za účelem detekovat a lokalizovat útočníka a získat o jeho přístupu co možná nejvíce informací. Poté dochází k odvrácení útoku a eliminaci možnosti opakovaného připojení.

Jako aktivní protiopatření byl vybrán Framework BeEF, jenž je využíván pro penetrační testování webového prohlížeče a zkoumání jeho závažných zranitelností, které mohou vést i k ovládnutí celého systému. Průběh instalace a konfigurace byl demonstrován v praktické části. Zde je obsažen i výpis vybraných modulů, u kterých je zadána vykonávaná funkce a důvod použití. O jejich automatické spouštění se postarala modifikace konfiguračního skriptu, jež je v této části podrobně rozebrána.

Dále byl popsán postup tvorby firewallu, který zahrnuje jeho instalaci, konfiguraci síťových rozhraní, zabezpečení linuxového systému, proces sestavení skriptu pomocí nástroje iptables a následné otestování funkčnosti. Firewall byl nastaven tak, aby byla možná komunikace mezi jednotlivými sítěmi. Povoluje jen vybrané porty pro konkrétní rozhraní a je připraven přijímat další pravidla pro blokování vybraných uživatelů dané sítě.

Dílním cílem této práce bylo charakterizovat varianty útoků na servery, jenž je uvedeno v teoretické části. Podstatná je i zmínka o těch, kteří za těmito útoky stojí, a proto byly popsány jednotlivé typy útočníků a jejich nejčastější důvody pro páčání těchto činů.

Dalším dílním cílem bylo provést test, zda navržené systémy fungují a realizované řešení otestovat na simulaci útoku na honeypotu. Tento test je názorně proveden a zdokumentován rovněž v praktické části. Nakonec jsou analyzována nashromážděná data. Vytyčené cíle práce tedy považuji za splněné.

Největší přínos implementace aktivních protiopatření do honeypotu je získání nezbytných informací o útočnickovi, vedoucí k analýze jeho totožnosti a zamezení dalších útoků. Zpracované metody tvorby firewallu a použití vybraných protiopatření mohou být přínosem pro uživatele nebo administrátory, kteří mají za úkol ochránit webový server proti různým útokům pocházejícím z Internetu, ale i vnitřní sítě. Práce také přispívá k prohloubení znalostí o linuxových systémech a o tvorbě zabezpečovacích a detekčních prvků v síti.

CONCLUSION

The main objective of this work was to implement active countermeasures to honeypotu in order to detect and locate attacker and get his access information as much as possible. Then occurs to avert an attack and eliminate the possibility of reconnection.

Active countermeasures has been selected as the Framework, which is BeEF used for penetration testing of a Web browser, and exploring its serious vulnerabilities which may lead to the domination of the entire system. The progress of the installation and configuration was demonstrated in a practical part. Here is a listing of the selected modules, which is specified by the function and reason for using. Their automatic configuration script allows modifications, which is discussed in detail in this section.

Also describes the procedure of making the firewall that includes its installation, configuration of network interfaces, security, Linux system, the build process script using iptables, and a subsequent test in functionality. The Firewall has been configured to allow communication between networks. To allow only the selected ports for a specific interface, and is ready to receive the next rule to block selected users of the network.

Part of the aim of this work was to characterize the variation of attacks on servers, which is mentioned in the theoretical part. The material is also a mention of those who stand behind those attacks, and were therefore described the different types of invaders and their most common reasons for committing these offences.

Another aim was to test whether the proposed systems work and realized the solution test to simulate an attack on honeypotu. This test is clearly executed and documented also in the practical part. Finally, the collected data are analyzed. Therefore, I consider the work objectives to be fulfilled.

The biggest benefit of implementation of active countermeasures to honeypotu is to obtain the necessary information on the attacker, leading to an analysis of his identity and prevent further attacks. Methods of handled of the firewall and use selected countermeasures, can be a boon for users or administrators who are tasked to protect the Web server against different attacks originating from the Internet, but also to the internal network. Work also contributes to deepening the knowledge on Linux systems and on the development of security and detection of the elements in the network.

SEZNAM POUŽITÉ LITERATURY

- [1] *Linux: dokumentační projekt*. 3. aktualiz. vyd. Brno: Computer Press, 2003, xviii, 1001 s. ISBN 8072267612.
- [2] *Bezpečností linuxová distribuce s vestavěným honeypotem*. Zlín, 2013. Diplomová práce. Univerzita Tomáše Bati. Vedoucí práce Ing. David Malaník, Ph.D.
- [3] *LINUXEXPRES: Správa linuxového serveru* [online]. 2014 [cit. 2014-02-02]. ISSN 1801-3996. Dostupné z: <http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru>
- [4] *BeEF: The browser exploitation framework project* [online]. [cit. 2014-02-03]. Dostupné z: <http://beefproject.com/>
- [5] KRČMÁŘ, Petr. *Linux: tipy a triky pro bezpečnost*. 1. vyd. Praha: Grada, 2004. ISBN 8024708124.
- [6] SPITZNER, Lancze. Honeypots: Definitions and Value of Honeypots. *Honeypots: Tracking Hackers* [online]. 2003 [cit. 2014-02-03]. Dostupné z: <http://www.tracking-hackers.com/papers/honeypots.html>
- [7] MEČÍŘ, Michal. *Použití honeypotů pro monitorování útoků*. Brno, 2009. Bakalářská práce. Masarykova univerzita. Vedoucí práce Mgr. Daniel Kouřil.
- [8] HATCH, Brian, George KURTZ a James LEE. *Linux hackerské útoky: bezpečnost Linuxu - tajemství a řešení*. Praha: SoftPress, 2002, 576 s. ISBN 8086497178.
- [9] Root.cz: informace nejen ze světa Linuxu [online]. 1998 [cit. 2014-03-25]. ISSN 1212-8309. Dostupné z: <http://www.root.cz/clanky/vse-o-iptables-uvod/>
- [10] HONTAÑÓN, Ramón J. *Linux: praktická bezpečnost*. Grada, 2003, 438 s. ISBN 80-247-0652-0.
- [11] STREBE, Matthew a Charles PERKINS. *Firewally a proxy-servery*. Vyd. 1. Brno: Computer Press, 2003, xxi, 450 s. ISBN 80-722-6983-6.
- [12] ANONYMOUS. *Maximální bezpečnost*. 4. vyd. Praha: Softpress, c2004, 2 sv. (440, 544 s.). ISBN 80-864-9765-8.
- [13] Zdroják: *Bezpečnost na webu – přehled útoků na webové aplikace* [online]. 2014 [cit. 2014-04-21]. ISSN 1803-5620. Dostupné z: <http://www.zdrojak.cz/clanky/prehled-utoku-na-webove-aplikace/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ARP	Address Resolution Protocol
BEEF	Browser Exploitation Framework
BGP	Border Gateway Protocol
BIOS	Basic Input Output System
BSD	Berkeley Software Distribution
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DMZ	Demilitarizovaná zóna
DNS	Domain Name Service
DoS	Denial of Service
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ISP	Internet Service Provider
LAN	Local Area Network
MITM	Man in The Middle
OSPF	Open Shortest Path First
PHP	Hypertext Preprocessor
RAM	Random Acces Memory
RIP	Routing Information Protocol
RVM	Ruby Version Manager
SQL	Structured Query Language
SSH	Secure Shell

SSL	Secure Socket Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
XSS	Cross Site Scripting
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WINS	Windows Internet Naming Service
XSS	Cross Site Scripting

SEZNAM OBRÁZKŮ

<i>Obr. 1 Umístění honeypotu [6]</i>	13
<i>Obr. 2 DDoS útok</i>	17
<i>Obr. 3 BeEF[4]</i>	28
<i>Obr. 4 Blokové schéma průchodu paketu firewallem</i>	34
<i>Obr. 5 Umístění firewallu v síti</i>	38
<i>Obr. 6 Uživatelské prostředí a nastavení hardwarových komponent</i>	41
<i>Obr. 7 Výstup programu ssh-keygen</i>	45
<i>Obr. 8 Nástroj keychain</i>	46
<i>Obr. 9 Traceroute</i>	55
<i>Obr. 10 Test ping of death</i>	56
<i>Obr. 11 Topologie sítě s IP adresami</i>	57
<i>Obr. 12 Start BeEF</i>	62
<i>Obr. 13 Přihlášení BeEF</i>	63
<i>Obr. 14 BeEF – kontrolní panel</i>	64
<i>Obr. 15 Skenování sítě a portů PC1</i>	72
<i>Obr. 16 Skenování sítě a portů PC2</i>	73
<i>Obr. 17 Stránka zobrazená útočníkem po přístupu na server</i>	74
<i>Obr. 18 Reakce aktivního protiopatření na návštěvu útočníka</i>	75
<i>Obr. 19 Záložka detaily ve frameworku</i>	76

SEZNAM TABULEK

<i>Tab. 1 – nastavení rozhraní firewallu.....</i>	<i>47</i>
<i>Tab. 2 – testování komunikace.....</i>	<i>55</i>
<i>Tab. 3 – testované počítače.....</i>	<i>72</i>

SEZNAM PŘÍLOH

PŘÍLOHA P I: ZDROJOVÉ KÓDY VYTVOŘENÝCH SKRIPTŮ

PŘÍLOHA P I: ZDROJOVÉ KÓDY VYTVOŘENÝCH SKRIPTŮ

Na přiloženém CD-ROM jsou uloženy tyto soubory se zdrojovými kódy jednotlivých skriptů:

ip.sh

iptablesload_skript.sh

rules.sh