VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

OCHRANA PROTI DISTRIBUOVANÝM ÚTOKŮM HRUBOU SILOU

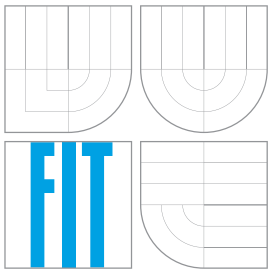
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

JAN RICHTER

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

OCHRANA PROTI DISTRIBUOVANÝM ÚTOKŮM HRUBOU SILOU

DISTRIBUTED BRUTE FORCE ATTACKS PROTECTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — JAN RICHTER
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR LAMPA
SUPERVISOR

BRNO 2010

Abstrakt

Diplomový projekt se zabývá analýzou útoků hrubou silou zaměřených na prolomení autentizace běžných služeb (především ssh) na operačních systémech Linux a xBSD. Jsou zde zkoumány reálné útoky, současné nástroje, možnosti detekce těchto útoků a navrženy nové mechanismy koordinace a vyhodnocování útoků v distribuovaném prostředí. Tyto mechanismy jsou dále implementovány v distribuovaném systému nazvaném DBFAP.

Abstract

This project deals with analysis of brute force attacks focused on breaking authentication of common services (especially ssh) of Linux and xBSD operating systems. It also examines real attacks, actual tools and ways of detection of theese attacks. Finaly there are designed new mechanisms of coordination and evaluation of distributed brute force attacks in distributed environment. These mechanisms are then implemented in distributed system called DBFAP.

Klíčová slova

Internet, útok, ssh, zabezpečení, heslo, hrubá síla, analýza, log, detekce, prevence, Linux, BSD, firewall, MySQL, IDS, IPS, botnet, DBFAP

Keywords

Internet, attack, ssh, security, password, brute force, analysis, log, detection, prevention, Linux, BSD, firewall, MySQL, IDS, IPS, botnet, DBFAP

Citace

Jan Richter: Ochrana proti distribuovaným útokům
hrubou silou, diplomová práce, Brno, FIT VUT v Brně, 2010

Ochrana proti distribuovaným útokům hrubou silou

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Petra Lampy

.....
Jan Richter
24. května 2010

Obsah

Kapitola 1

Úvod

Rychlé rozrůstání Internetu v posledních letech spolu s množstvím nových služeb a informací přináší také velké množství nových nástrah a nebezpečí, která na uživatele v síti čekají, ať už jde o různé formy sociálního inženýrství, viry, červy, trojské koně, podvržené odpovědi DNS serverů, odposlech dat na nezabezpečených sítích nebo útoky hrubou silou. Internet je dnes plný nástrah a bez účinných nástrojů pro detekci a prevenci útoků si uživatel a především administrátor nemůže být ničím jistý.

V této práci se budu zabývat především automatizovanými útoky hrubou silou, které mohou být prováděny náhodně za účelem získání přístupu k dalším počítačům a jejich připojení k botnetům nebo cíleně pro získání přístupu ke konkrétním počítačům a datům. Dále potom rozeberu principy a analýzu těchto útoků a způsoby obrany před tímto druhem útoků.

V další fázi se pokusím na základě informací z předchozí fáze navrhnout efektivnější systém pro detekci a vyhodnocování nejen distribuovaných útoků hrubou silou a pro následnou blokaci těchto útoků. Tento systém by měl pro uchovávání všech potřebných údajů používat relační databázi MySQL.

Navržený systém bude dále implementován a implementace blíže popsána. V závěru této práce bude provedeno testování a ověření funkčnosti a přínosnosti implementovaných postupů.

V celé práci se budu zabývat hlavně útoky na službu ssh, ale většina zde uvedených informací bude platit obecně a bude aplikovatelná také na jiné služby.

Kapitola 2

Ssh

Jednou ze služeb, na které je na unixových systémech nejvíce útočeno hrubou silou, je bezesporu služba SSH, která dnes běží prakticky na všech serverech a také na většině linuxových stanic. Rovněž se často objevuje například na velkém množství domácích routerů a jiných síťových prvků.

Zkratka SSH, neboli *secure shell* (což může být poměrně zavádějící, jelikož se ve skutečnosti o žádný shell nejedná) označuje jak protokol, tak název služby a klienta komunikujícího pomocí tohoto protokolu se serverem.

Protokol SSH je zabezpečenou náhradou dříve používaných protokolů pro přístup ke vzdálenému shellu (například `rsh` nebo `telnet`). Jak již bylo naznačeno v předchozím odstavci, jde o architekturu typu klient/server, server se typicky jmenuje `sshd`, klient potom prostě `ssh`.

Primárním využitím SSH je spouštění příkazů na vzdáleném systému, SSH ovšem umožňuje také zabezpečené kopírování souborů po síti, tunelování TCP portů nebo například tunelování protokolu X11.

Protokol SSH zajišťuje

- Autentizaci spojení - ověří identitu přihlašovaného uživatele, může to být provedeno zadáním hesla nebo třeba ověřením RSA/DSA klíčem.
- Šifrování spojení - všechna přenášená data jsou šifrována a není možné je rozluštit při odposlechu komunikace na síti, v současné době používá implementace OpenSSH šifry 3DES, Blowfish, AES nebo ARC4.
- Integritu spojení - zaručí, že přenášená data dorazí k cíli nezměněna a v případě jejich pozměnění útočníkem je tato skutečnost zjištěna

2.1 Historie

První verze protokolu, SSH-1 byla vyvinuta v roce 1995 na Helsinské technické univerzitě Tatuem Ylönenem poté, co se na univerzitě začali potýkat s problémy odposlechnutých hesel. O program SSH1 začal být obrovský zájem a v témže roce Ylönen založil společnost SSH Communications Security(SCS), jejímž cílem je SSH dále udržovat a vyvíjet.

S rychlým rozšířením SSH-1 bylo objeveno množství problémů, které nebylo možno beze ztráty kompatibility vyřešit a v roce 1996 tedy byla specifikována nová verze protokolu, SSH-2. IETF[1] poté utvořila pracovní skupinu nazvanou SECSH (Secure Shell) pro

[1] Internet Engineering Task Force

standardizaci protokolu a řízení jeho vývoje. V únoru 1997 byl vydán první standardizovaný návrh protokolu a rok poté SCS vydala první softwarovou implementaci SSH-2.

SSH-2 ovšem nebylo mezi uživateli přivítáno příliš vřele, problémem byla omezující komerční licence a odebrání některých užitečných a praktických funkcí a uživatele v přechodu na SSH-2 spatřovali jen málo výhod. Koncem roku 2000 ale SCS uvolnila licenci SSH-2 pro volné použití na operačních systémech Linux, FreeBSD, NetBSD a OpenBSD[2].

Od roku 1999 je zároveň vyvíjen projekt OpenSSH, vyvíjený pod záštitou OpenBSD, který je také dostupný pod OpenBSD licencí. Tato implementace vychází z poslední verze SSH-1, která byla vydána jako open source, 1.2.12. V současné době OpenSSH implementuje jak SSH-1, tak také SSH-2 protokol a od roku 2005 je nejrozšířenější implementací SSH, kterou můžeme nalézt jako výchozí ve velkém množství operačních systémů.

2.2 Konfigurace

Zde bych rád poskytl stručný popis konfigurace SSH démona se zaměřením na direktivy, které nějak souvisí se zabezpečením proti útokům hrubou silou.

Konfigurace SSH démona se obvykle na většině operačních systémů nachází v souboru `/etc/ssh/sshd_config` a může obsahovat mj. některé z následujících direktiv:

PermitRootLogin specifikuje, zda bude povoleno přihlášení uživatele root. Jak je rozebráno v kapitole 4, jedná se o jedno z nejčastěji zkoušených uživatelských jmen a zakázáním přihlášení s tímto uživatelským jménem může být velké množství útoků odraženo.

PasswordAuthentication určuje zda bude povolena autentizace zadáním hesla (v případě nastavení na `no` může být potřeba nastavit ještě direktivu `UsePAM` také na `no`). Po zakázání tohoto typu autentizace bude server vyžadovat autentizaci pomocí RSA/DSA klíčů.

Port umožňuje změnu portu, na kterém bude démon poslouchat. Zvolením nestandardního portu dojde k poměrně spolehlivému odrazení automatizovaných útočníků, ale také vznikne nutnost tento změněný port zadávat všemi uživateli, což může být nežádoucí. Útočník, který bude útočit cíleně na náš systém, ovšem s tímto opatřením žádný problém mít nebude a port, na kterém služba běží, si zjistí například skenováním portů.

ForceCommand umožňuje zadat příkaz, který bude bezprostředně po přihlášení vykonán a poté bude spojení ukončeno.[2] Toto sice není direktiva, která by byla užitečná k zabezpečení SSH démona, který je běžně používán k přihlašování, ale může posloužit k zabezpečení serveru v pozici provizorního honeypotu, kde chceme mít jistotu, že v případě uhodnutí hesla útočník nenapáchá žádnou škodu.

Více ke konfiguraci služby `sshd` je možno nalézt například v manuálových stránkách nebo v literatuře[2].

[2] samozřejmě za předpokladu zadání nějakého vhodného příkazu, např. `echo`, nevhodným příkladem může být `/bin/bash`

2.3 Přihlášení ke službě ssh

Nyní si stručně popíšeme jak probíhá typické přihlášení ke službě ssh, což pro nás bude užitečné v následujících částech projektu.

1. Je provedeno navázání spojení a vyměněny kryptografické informace, tato část sestává z několika dílčích částí.
   (a) Výměna informací o verzi serveru a klienta
   (b) Výměna veřejných klíčů serveru a klienta, klient ověří zda skutečně komunikuje se serverem, se kterým chtěl navázat spojení
   (c) Volba asymetrického kryptografického algoritmu
   (d) Volba symetrického kryptografického algoritmu
   (e) Volba autentizačního algoritmu
   (f) Volba hashovacího algoritmu
   (g) Stanovení sdíleného klíče pro šifrování komunikace (ten bude v průběhu další komunikace v závislosti na konfiguraci průběžně měněn)
2. Navázané spojení je dále šifrováno pomocí symetrické šifry.
3. Klient se autentizuje serveru zvolenou autentizační metodou.

   Pro započetí autentizace klient posílá žádost o spojení `SSH_MSG_USERAUTH_REQUEST`, která obsahuje uživatelské jméno, název služby a název autentizační metody, dále následují případné další parametry metody. V případě, že server danou autentizační metodu zamítne, odpoví zprávou `SSH_MSG_USERAUTH_FAILURE` a seznamem podporovaných autentizačních metod (v tomto seznamu neuvádí metodu `none`, umožňující přihlášení bez autentizace ani v případě, že je na serveru povolena, požadavek na tuto metodu je často používán ke zjištění nabízených autentizačních metod) V případě, že server zvolenou metodu podporuje a metoda je úspěšná, odpoví zprávou `SSH_MSG_USERAUTH_SUCCESS`[10].

   Pro účely této práce budou nejdůležitější následující dvě autentizační metody

   (a) `password` [19] je po metodě `none` nejjednodušší přihlašovací metodou, struktura zaslaného paketu se zprávou `SSH_MSG_USERAUTH_REQUEST` má následující tvar: (datový typ a hodnota)

   ```
   byte      SSH_MSG_USERAUTH_REQUEST
   string    user name
   string    service name
   string    "password"
   boolean   FALSE
   string    plaintext password in ISO-10646 UTF-8 encoding
   ```

   odpovědí mohou být již dříve zmíněné zprávy `SSH_MSG_USERAUTH_FAILURE` a `SSH_MSG_USERAUTH_SUCCESS` nebo zpráva `SSH_MSG_USERAUTH_PASSWD_CHANGEREQ` vyžadující, aby uživatel provedl pro použití této metody změnu svého hesla (důvodem bývá vypršení platnosti starého hesla). Klient na takovou zprávu zareaguje buď použitím jiné metody nebo pošle serveru novou, upravenou žádost o přihlášení ve tvaru:

```
byte      SSH_MSG_USERAUTH_REQUEST
string    user name
string    service name
string    "password"
boolean   TRUE
string    plaintext password in ISO-10646 UTF-8 encoding
string    plaintext new password in ISO-10646 UTF-8 encoding
```

(b) `keyboard-interactive` [6] je novější a obecnější metodou, která vznikla pro usnadnění přidávání nových autentizačních mechanizmů, které díky ní nejsou závislé na nižší autentizační vrstvě. Je takto například umožněno přidání a použití nového PAM modulu bez zásahu do kódu klienta a serveru. Použití této metody je inicializováno opět požadavkem `SSH_MSG_USERAUTH_REQUEST` ve tvaru

```
byte      SSH_MSG_USERAUTH_REQUEST
string    user name (ISO-10646 UTF-8)
string    service name (US-ASCII)
string    "keyboard-interactive" (US-ASCII)
string    language tag (as defined in [RFC-3066])
string    submethods (ISO-10646 UTF-8)
```

Je zde mimo jiné vidět, že přibyla položka pro specifikaci jazyka pro komunikaci v rámci této přihlašovací metody nebo specifikace submetod a již zde není položka obsahující heslo. Server na tuto zprávu odpovídá zasláním paketu se zprávou typu `SSH_MSG_USERAUTH_INFO_REQUEST`.

```
byte      SSH_MSG_USERAUTH_INFO_REQUEST
string    name (ISO-10646 UTF-8)
string    instruction (ISO-10646 UTF-8)
string    language tag (as defined in [RFC-3066])
int       num-prompts
string    prompt[1] (ISO-10646 UTF-8)
boolean   echo[1]
...
string    prompt[num-prompts] (ISO-10646 UTF-8)
boolean   echo[num-prompts]
```

Touto zprávou žádá klienta o zaslání dalších informací, které má možnost specifikovat, typicky zde bude žádat o zadání hesla, ale je zcela v kontrole zvoleného autentizačního mechanismu na straně serveru, jak bude autentizace probíhat.

Klient po obdržení této zprávy zobrazí položku `name` a `instruction`, pokud nejsou prázdné a následně vypisuje všechny položky pole `prompt` a čte odpovědi na tyto dotazy. Příslušná položka z pole `echo` určuje, zda má být uživatelův vstup zobrazován nebo ne (tedy zabrání například zobrazení uživatelem zadávaného hesla na monitoru).

Po obdržení uživatelova vstupu je serveru zaslána zpráva obsahující odpovědi na všechny jeho dotazy ve tvaru

```
byte      SSH_MSG_USERAUTH_INFO_RESPONSE
int       num-responses
```

```
string    response[1] (ISO-10646 UTF-8)
...
string    response[num-responses] (ISO-10646 UTF-8)
```

Server na základě získaných informací reaguje buď zasláním zprávy o úspěchu nebo neúspěchu prováděného přihlášení nebo může zasílat další zprávy typu `SSH_MSG_USERAUTH_INFO_REQUEST` s dalšími dotazy.

Tato autentizační metoda tedy umožňuje značnou variabilitu při vytváření autentizačních mechanismů a díky ní by například bylo možné provádět přihlášení na základě kombinace zadání hesla a vypočítání náhodně vybrané matematické hádanky nebo provedení nějakého Turingova testu.

Další často používanou autentizační metodou je metoda `publickey`, která probíhá pomocí asymetrické kryptografie. Klient serveru pošle svůj veřejný klíč, ten je serverem porovnán s klíčem, který má pro uživatele uložený, pokud souhlasí, server vygeneruje náhodnou zprávu, kterou tímto klíčem zašifruje a pošle ji klientovi, který ji musí za pomoci svého soukromého klíče rozšifrovat a poslat zpátky serveru, čímž se ověří vlastnictví soukromého klíče klientem a autentizace je úspěšná.

4. Klient je připojen a autentizován a může komunikovat se serverem po zabezpečeném spojení.

Více podrobností o průběhu navázání spojení a autentizačních metodách je možno nalézt v použité literatuře[10][16] a RFC 4252[19] a 4256[6].

Kapitola 3

Principy útoků

Způsobů počítačových útoků dnes existuje celá řada, mezi časté techniky patří třeba odposlech komunikace na síti, podvržení informací, různé způsoby zneužití chyb v programech (buffer overflow, sql injection), (D)DoS útoky nebo nejrůznější techniky sociálního inženýrství. Popis těchto technik však překračuje rozsah této práce a vydaly by na několik dalších publikací, zde se dále budeme zabývat pouze útoky hrubou silou.

3.1 Útoky hrubou silou

Útok hrubou silou je velice jednoduchý typ útoků, kdy útočník postupně zkouší podle určitého pravidla zadávat jména a hesla a kontroluje, zda se přihlášení se zvolenou kombinací jména a hesla zdařilo nebo ne.

Jako přihlašovací jména jsou často používány výchozí loginy (root, admin, oracle, postgres...), jména získána dolováním na webu (například z emailových adres) nebo často se vyskytující jména a příjmení (james, john...).

Tyto útoky můžou být podle množiny zkoušených hesel rozděleny do tří skupin:

- Slovníkový útok hrubou silou - útočníci aplikace vychází ze předem připraveného souboru hesel, které postupně zkouší
- Hybridní útok hrubou silou - pro útok je použit slovník jako v předchozí variantě, ale aplikace zkouší každé heslo doplnit dalšími znaky, nejčastěji číslicemi.
- Obecný útok hrubou silou - v tomto typu útoku se slovník nepoužívá, aplikace provádějící útok sama postupně generuje a zkouší všechna hesla složená z předem specifikované množiny znaků.

Tento jednoduchý typ útoku pak může být dále zdokonalován paralelizací. Útočný program může běžet a útočit ve více vláknech a nebo běžet distribuovaně na větším množství počítačů.

3.1.1 Dostupný software

Mezi nejznámější z volně dostupných programů pro lámání hesel hrubou silou patří program `johntheripper`. Tento program sice není vytvořen pro vzdálené přihlašování, ale stojí alespoň za zmínku. Program slouží pro získání hesel uložených v hashované podobě v nějakém dostupném souboru (typicky například `/etc/shaddow`). Jde tedy o nástroj pro lámání hashovacích funkcí hrubou silou.

Dalším, již síťovým, volně dostupným programem je program `brutessh`, tento program je napsán v pythonu, umí pouze slovníkové útoky a může běžet ve vláknech (ve výchozím nastavení spouští 12 vláken). Program sám o sobě nepodporuje distribuované použití, ale s vytvořením vhodné front-end aplikace by neměl být problém ho takto použít. Pro navazování spojení se serverem je v zde používána knihovna `paramiko`, kvůli které potřebuje modul `crypto` pro python.

Použití programu je velice jednoduché, například pro útok na počítač `192.168.2.180` s uživatelským jménem `john` a hesly ze souboru `hesla.txt` stačí použít následující příkaz.

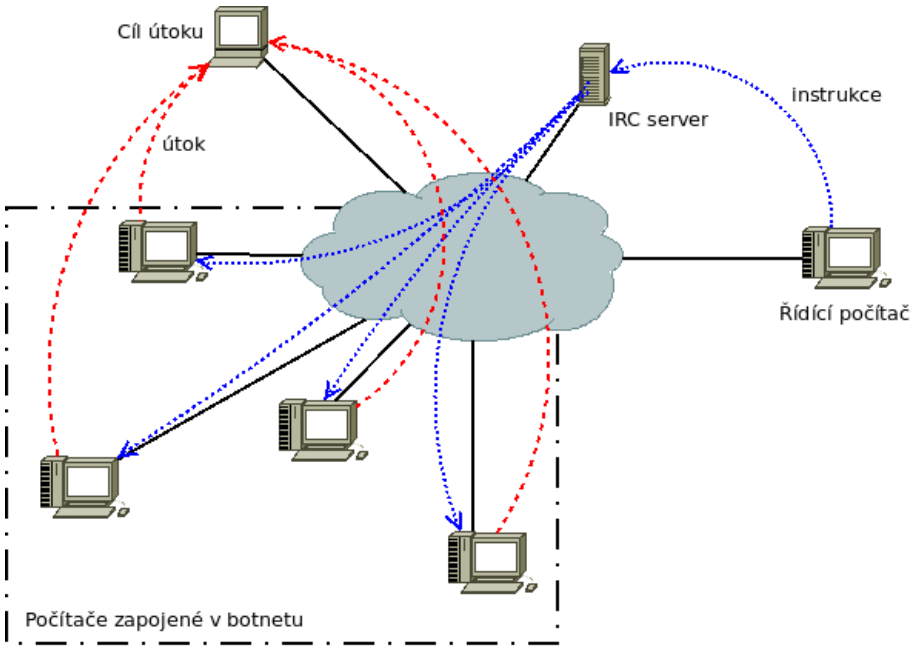
```
python brutessh.py -h 192.168.2.180 -u john -d hesla.txt
```

Autorem `brutessh` je Christian Martorella, pořadatel konferencí FIST (First Improvised Security Testing) zaměřených na bezpečnost počítačových sítí. Stejně jako spousta dalších zajímavých aplikací dostupných na jeho webu `www.edge-security.com`, je tato aplikace určena pro výzkumné účely a testování.

Výše popisovaný program `brutessh` je možno získat z webu jeho autora na adrese `http://www.edge-security.com/brutessh.php`.

3.2 Botnety

Botnet je pojem označující skupinu (síť) počítačů, na kterých běží škodlivý software (často nazýván bot), který umožňuje vzdálené ovládání každého počítače. V této síti dále existují mechanismy pro koordinované spuštění příkazů na jednotlivých počítačích. Základní schéma jednoduchého centralizovaného botnetu je na obrázku 3.1



Obrázek 3.1: Jednoduché schéma botnetu

3.2.1 Vznik botnetů

Připojování nových počítačů do botnetu může probíhat několika způsoby, mezi nejvýznamnější patří

- rozesílání škodlivého software jako SPAM, uživatel musí program spustit,
- vystavování infikovaných souborů na warez serverech,
- využití chyb v aplikacích běžících na cílovém počítači,
- útoky hrubou silou na služby pro vzdálený přístup a poté infikování počítače.

3.2.2 Využití botnetů

Botnety můžou mít pro svého tvůrce různé využití, v konečném důsledku ovšem za vším stojí peníze, botnet může být autorem prodán, pronajímán nebo využíván k jiné nekalé činnosti, která mu bude vynášet peníze. Typickými činnostmi, ke kterým botnety slouží jsou:[13]

- masivní DDoS útoky
- rozesílání SPAMu (jak mailového, tak SPAMování webových formulářů)
- instalace ad-ware/malware na napadené počítače
- získávání důvěrných informací z napadených počítačů a jejich zneužití nebo prodej
- phishing - na napadených hostech je možno editovat pevné DNS záznamy a nebo rozesílat podvržené DNS odpovědi do jejich okolních sítě
- "klikací" podvody - je možno automaticky provádět "klikání" na reklamní bannery

3.2.3 Kontrola nad botnety

Botnety jsou typicky ovládány centralizovaně (jak bylo vidět na schématu botnetu 3.1). Každý aktivní bot se připojí k předem definovanému tzv. command-and-control (CnC) serveru a čeká na příkazy, nejčastěji je to realizováno pomocí protokolů IRC nebo HTTP[7].

Zatímco pomocí IRC je jednotlivým botům aktivně zasíláno co mají provést, pomocí HTTP bývají vystaveny příkazy na předem specifikovaném místě na webu, v nedávné době se stalo populárním k této činnosti používat například sociální síť twitter[14].

Tato základní centralizovaná topologie bývá často rozšířená o více CnC serverů pro případ, že by byl některý server odhalen a zablokován. Samozřejmě je možné postavit botnet na jakékoliv jiné myslitelné topologii, vždy to pak spolu přináší určité výhody i nevýhody v náročnosti správy, údržbě, rychlosti reakce a pod.

Kapitola 4

Analýza reálných útoků

V této části práce se budeme zabývat vybranými výsledky provedených analýz z logů z několika linuxových serverů. Pro analýzu byla použita databáze MySQL, do které byly pomocí sady několika jednoduchých skriptů doplněny informace z logů.

Typické záznamy o neúspěšném pokusu o přihlášení od ssh démona v logu vypadají takto:

```
May  1 22:06:26 kazi sshd[46407]: error: PAM: authentication error for
  illegal user admin from cni1.cbinf.com
May  1 22:06:32 kazi sshd[46418]: error: PAM: authentication error for
  root from cni1.cbinf.com
May  2 00:16:18 kazi sshd[54720]: Invalid user anna from 89.149.208.141
```

Na začátku řádku je uvedeno datum a čas záznamu, hostname počítače, následuje identifikace procesu (název a pid), který záznam zapsal a text záznamu. Zde například jde na druhém řádku o přihlášení uživatele `root` z adresy `cni1.cbinf.com`. Na prvním a třetím řádku jde o přihlášení v systému neexistujících uživatelů `admin` a `anna`.

K analýze použitá databázová tabulka má následující schéma:

```
CREATE TABLE  `dp`.`kazi2` (
  `id` int(11) NOT NULL auto_increment,
  `date` datetime NOT NULL,
  `host` varchar(128) NOT NULL,
  `username` varchar(32) NOT NULL,
  `type` varchar(16) NOT NULL,
  `ip` varchar(15) NOT NULL,
  `country` varchar(50) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

do prvního sloupce se zaznamenává jednoznačné id, následuje datum převedené na MySQL formát data a času, hostname počítače, ze kterého bylo přihlašování prováděno a uživatelské jméno. Sloupec type může nabývat hodnot “valid” nebo “invalid” podle toho, zda uživatel v systému existuje. Do sloupců ip a country pak byly další sadou skriptů zjištěny a doplněny o ip adresy[1] a země, ve kterých se počítače nacházejí.

[1] v logu jsou díky nastavené hodnotě `UseDNS yes` zdrojové ip adresy překládány, což nemusí být vždy výhodou, protože některá doménová jména po určitém čase již nemusí být možné přeložit zpátky

4.1 Statistiky ze serveru kazi

První analýza byla provedena nad logy z fakultního serveru `kazi` za období od 1.1.2008 do 4.10.2009, které obsahovaly celkem 195 474 zaznamenaných chybných pokusů o přihlášení k serveru pomocí SSH.

Server `kazi` slouží jako home a mailserver pro zaměstnance fakulty, existuje na něm tedy spousta uživatelů a v logu se objevují také neplatné pokusy o přihlášení skutečných uživatelů (a to jak se správně uvedeným přihlašovacím jménem a chybou v hesle, tak s překlepy v uživatelském jméně), které je třeba z analýzy odfiltrovat. Stejně tak pravděpodobně neúspěšné pokusy o přihlášení k rootovskému účtu z adresního rozsahu fakulty nebudou útoky, ale jen chyby při zadávání hesla. Celkově tak tedy bylo odstraněno 4 469 záznamů, což tvoří 2,3% ze všech záznamů. Přes 97% pokusů o přihlášení tedy bylo součástí útoků na tento server.

V první části analýzy jsem se zaměřil na frekvenci používaných přihlašovacích jmen a tabulka 4.1 ukazuje 15 nejčastěji použitých loginů.

login	**počet**	**podíl**
root	14 316	7,50 %
admin	11 497	6,02 %
james	5 370	2.81 %
backup	2 658	1,39 %
access	1 121	0,59 %
account	1 075	0,56 %
test	854	0,45 %
albert	821	0,43 %
cacti	816	0,43 %
at	807	0,42 %
asterisk	795	0,42 %
oracle	627	0,33 %
john	531	0,28 %
user	496	0,26 %
robert	490	0,26 %

Tabulka 4.1: Použitá přihlašovací jména na serveru `kazi`

Jak je z tabulky vidět, nejčastěji používaným přihlašovacím jménem je zde "root", což koresponduje také s jinými veřejně dostupnými výsledky analýz útoků hrubou silou[15]. Ale jak uvidíme dále, situace se začíná pomalu měnit.

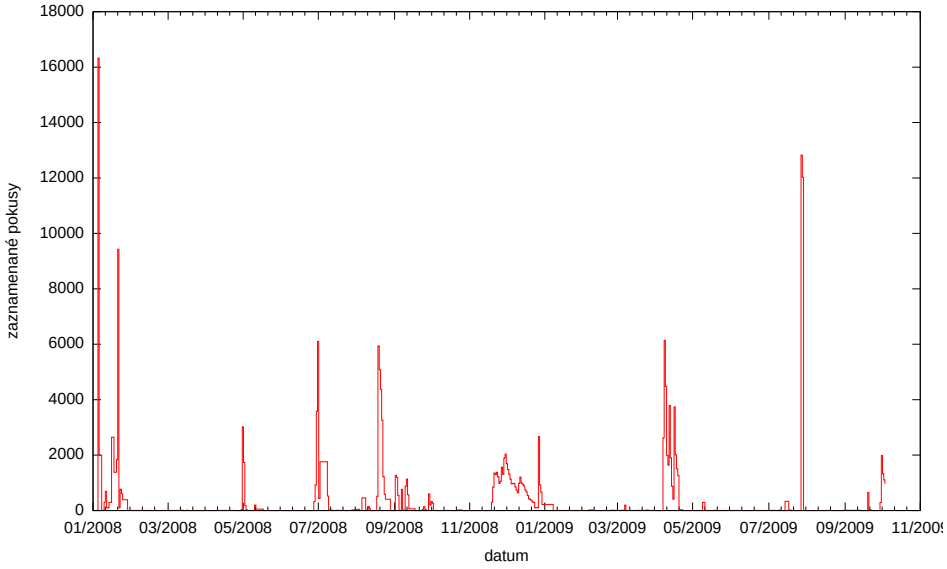
Dalším zajímavým srovnáním je množství pokusů o přihlášení z různých zemí, seznam opět 15-ti nejčastěji se vyskytujících zemí je v tabulce 4.2

Hodnota "None" v tabulce znamená, že zemi nebylo možné rozpoznat.

Pro rychlé vyhledání rozsáhlejších útoků jsem vytvořil histogram zobrazující počet jednotlivých pokusů o přihlášení na dny. Tento histogram je na obrázku 4.1. Je z něj patrné, že útoky probíhaly víceméně nepravidelně a s různou razancí.

země	**počet**	**podíl**	**země**	**počet**	**podíl**
United States	32 975	17.26 %	None	503	13,26 %
Japan	27 366	14.33 %	United States	381	10,04 %
Germany	14 498	7.59 %	Germany	255	6,72 %
Poland	13 296	6.96 %	Netherlands	201	5,30 %
None	9 687	5.07 %	Italy	189	4,98 %
Italy	6 880	3.60 %	Poland	168	4,43 %
Spain	6 849	3.59 %	China	159	4,19 %
Netherlands	5 717	2.99 %	Czech Republic	122	3,22 %
China	5 157	2.70 %	France	110	2,90 %
Russian Federation	4 464	2.34 %	Spain	97	2,56 %
Brazil	3 955	2.07 %	Japan	91	2,40 %
Taiwan, Province of China	3 892	2.04 %	United Kingdom	86	2,27 %
Australia	2 789	1.46 %	Australia	82	2,16 %
Finland	2 746	1.44 %	Russian Federation	77	2,03 %
Korea, Republic of	2 719	1.42 %	outh Africa	57	1,50 %

Tabulka 4.2: země původu útoků na server `kazi`, vlevo všechny pokusy, vpravo dle unikátních ip adres



Obrázek 4.1: Počet neplatných přihlášení k serveru `kazi` v čase

Při analýze logu bylo nalezeno několik typických distribuovaných útoků, které se vyznačovaly

- množstvím ip adres, ze kterých byly zaslány pokusy o přihlášení ve stejném čase (rozdíly max. jednotky minut),
- používáním stejných přihlašovacích jmen nebo jmen abecedně seřazených,
- z každé ip adresy přišel pouze jeden nebo velmi malé množství pokusů o přihlášení,
- v některých útocích se ovšem stejné ip adresy po uplynutí určité doby opět zapojily

Ukázka malé části průběhu jednoho z útoků je v tabulce 4.3.

datum a čas	**login**	**host**	**země**
2008-09-03 09:21:52	student	213.37.70.13	Spain
2008-09-03 09:21:52	student	213.37.70.13	Spain
2008-09-03 09:21:53	student	213.37.70.13	Spain
2008-09-03 09:22:31	student	213.215.39.74	France
2008-09-03 09:22:32	student	213.215.39.74	France
2008-09-03 09:22:33	student	213.215.39.74	France
2008-09-03 09:23:27	student	79.28.101.87	Italy
2008-09-03 09:23:28	student	79.28.101.87	Italy
2008-09-03 09:23:29	student	79.28.101.87	Italy
2008-09-03 09:24:30	student	67.154.137.94	United States
2008-09-03 09:24:31	student	67.154.137.94	United States
2008-09-03 09:24:31	student	67.154.137.94	United States
2008-09-03 09:26:12	student	213.23.193.42	Germany
2008-09-03 09:26:13	student	213.23.193.42	Germany
2008-09-03 09:26:14	student	213.23.193.42	Germany
2008-09-03 09:28:00	student	71.118.8.244	United States
2008-09-03 09:28:02	student	71.118.8.244	United States
2008-09-03 09:28:05	student	71.118.8.244	United States
2008-09-03 09:29:31	student	217.7.238.30	Germany
2008-09-03 09:29:34	student	217.7.238.30	Germany
2008-09-03 09:29:35	student	217.7.238.30	Germany
2008-09-03 09:30:28	student	212.96.170.118	Czech Republic
2008-09-03 09:30:29	student	212.96.170.118	Czech Republic
2008-09-03 09:30:29	student	212.96.170.118	Czech Republic
2008-09-03 09:34:15	student	195.250.188.225	Estonia
2008-09-03 09:34:16	student	195.250.188.225	Estonia
2008-09-03 09:34:18	student	195.250.188.225	Estonia

Tabulka 4.3: ukázka distribuovaného útoku na server `kazi`

Největším zaznamenaným samostatným útokem na server `kazi` byl útok prováděný od 15:32:02 28.7.2009 do 7:54:30 následujícího dne. Během tohoto útoku bylo provedeno celkem 24 847 neplatných pokusů o přihlášení z jediné japonské ip adresy s přibližně 3 s dlouhými rozestupy mezi jednotlivými pokusy.

Největším distribuovaným útokem byl útok v trvající od 4.12.2008 8:58:45 do 19.12.2008 0:21:32. Celkem bylo při tomto útoku provedeno 12 806 neúspěšných pokusů o přihlášení k serveru z 1 429 různých ip adres.

Celkový počet rozpoznaných útoků[2] je 605, což při 190 515 záznamech v logu tvoří v průměru 315 pokusů o přihlášení na jeden útok a průměrný počet zapojených ip adres v takovém útoku je 6,3.

[2] útok byl pokládán za skončený, pokud po posledním neúspěšném přihlášení v sérii došlo k alespoň 30-minutové pauze

Po odstranění 225 útoků, které sestávaly jen z jednoho pokusu o přihlášení se průměrný počet záznamů v jednom útoku zvýší na 501 dílčích pokusu o přihlášení a průměrný počet ip adres použitých v jednom útoku se změní na 10.

4.2 Statistiky z firemních serverů

Další analýza byla provedena nad logy ze dvou serverů firmy C.S.G., `otesanek1` a `otesanek2` umístěných v České republice a na Slovensku a připojených různými ISP. Na těchto serverech na rozdíl od fakultního serveru kazi (4.1) neexistují jiní uživatelé, než `root` a jeden další uživatel s omezenými právy. Pro přihlášení jsou zde dále oprávněnými osobami používány výhradně RSA klíče, všechny zaznamenané neúspěšné pokusy lze tedy pokládat za útoky.

Celkový počet zaznamenaných neplatných přihlášení je u těchto serverů 509 574 pro server `otesanek1` a 37 893 pro server `otesanek2`.

Zpracovávané logy z obou serverů obsahují záznamy SSH démona od 1.1.2009 do 29.11.2009, tedy oproti logům ze serveru `kazi` nepokrývají rok 2008.

login	**počet**	**podíl**	**login**	**počet**	**podíl**
admin	26584	5,22 %	admin	9 168	24,19 %
test	7553	1,48 %	james	5 101	13,46 %
james	5383	1,06 %	root	3 234	8,53 %
root	4960	0,97 %	test	570	1,50 %
oracle	3494	0,69 %	oracle	455	1,20 %
guest	3138	0,62 %	guest	151	0,40 %
user	2670	0,52 %	user	115	0,30 %
a	2186	0,43 %	sales	86	0,23 %
info	1829	0,36 %	staff	75	0,20 %
student	1486	0,29 %	office	74	0,20 %
webmaster	1317	0,26 %	alias	72	0,19 %
testing	1315	0,26 %	recruit	72	0,19 %
tester	1310	0,26 %	PlcmSpIp	60	0,16 %
temp	1302	0,26 %	webmaster	57	0,15 %
support	1281	0,25 %	aaron	51	0,13 %

Tabulka 4.4: Použitá přihlašovací jména na serverech `otesanek1` a `otesanek2`

V tabulce je vidět docela velký nepoměr mezi nejvyššími příčkami, zatímco nejpoužívanější přihlašovací jméno na serveru `otesanek2` pokrývá přes 24% všech případů, na serveru `otesanek1` je tomu u nejčastějšího loginu jen lehce přes 5%.

Tento rozdíl je způsoben pravděpodobně špatně nakonfigurovaným útočícím agentem, v logu serveru `otesanek1` se objevuje obrovské množství velice netypických loginů (např. DMrtHaZq, password1, !@#$%^&*() a pod.), které byly velice pravděpodobně určeny k použití jako heslo. Tuto teorii také podporuje fakt, že naprostá většina těchto netypických loginů byla zkoušená ze stejné ip adresy.

V menší míře se toto objevilo také na serveru `kazi`.

Celkový počet útoků na server `otesanek1` byl 1 222 a s počtem 509 574 záznamů byl průměrný počet pokusů o přihlášení během jednoho útoku 417. Průměrný počet ip adres zúčastněných na jednom takovém útoku je 1,6. Počet "útoků" sestávajících z pouze jednoho samostatného pokusu o přihlášení byl 330. Po jejich odečtení ze statistik se průměrný počet pokusů na útok zvýší na 571 a počet ip adres v průměrném útoku na 2,2.

Na server `otesanek2` bylo provedeno 1 140 útoků, což s počtem 37 983 záznamů v logu dělá průměrný počet 33 pokusů o přihlášení během jednoho útoku s průměrně 1,8 ip adresami použitými v jednom útoku. Po odfiltrování 242 útoků sestávajících z pouze jednoho osamoceného pokusu o přihlášení se průměrný počet přihlášení v jednom útoku zvýší na 42 a počet ip adres figurujících v útoku na 2,3.

Poměrně vysoké počty útoků sestávajících z pouze jednoho neplatného pokusu o přihlášení může být dán použitou metodou pro roztřiďování jednotlivých pokusů, kdy byly útoky rozlišovány tak, že mezi posledním záznamem o neplatném přihlášení staršího útoku a prvním záznamem novějšího útoku uběhlo minimálně 30 minut. Takto mohly být rozděleny některé útoky s hodně dlouhými intervaly mezi jednotlivými pokusy do více útoků.

4.3 Hesla použitá při útocích

Vzhledem ke skutečnosti, že všechny zdroje pro mou analýzu byly získány za období před započetím práce na projektu, neobsahují záznamy o použitých heslech, které ssh démon z bezpečnostních důvodů standardně neukládá. Uvedu zde tedy alespoň pár statistik převzatých z výzkumu provedeného v roce 2008 na Clarkson University[15].

heslo	**podíl**
username	56,9 %
123456	3,6 %
password	1,4 %
test	0,8 %
12345	0,6 %
test123	0,5 %
123	0,5 %
1234	0,5 %
passwd	0,4 %
admin	0,4 %

Tabulka 4.5: nejpoužívanější hesla při útocích hrubou silou[15]

Výsledky analýzy hesel jsou v tabulce 4.5, z tabulky vyplývá, že v naprosté většině případů útočník zkouší heslo stejné jako použité uživatelské jméno, případně je doplní o další znaky a číslice před a nebo za jménem.

4.4 Celkové zhodnocení

Nyní se podíváme na celkové vyhodnocení jednotlivých dílčích částí analýzy.

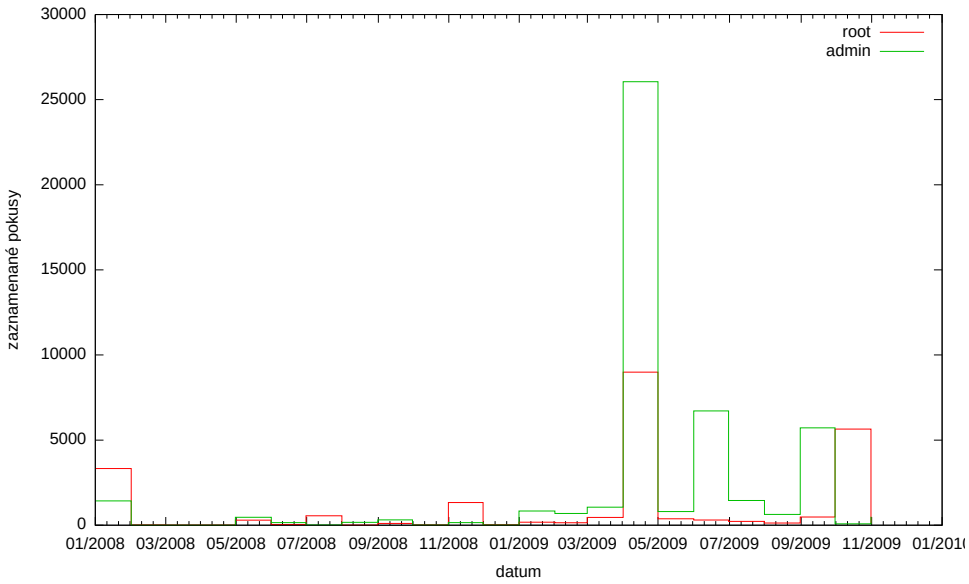
4.4.1 Přihlašovací jména

Ze starších výzkumů[15] vyplývá, že nejčastěji používaným přihlašovacím jménem v útocích hrubou silou je `root`, a to až ve 25,7%, má analýza serveru `kazi` za poslední dva roky ukazuje pouze 7,5% zastoupení tohoto uživatele a analýza za poslední rok (2009) na serverech `otesanek1` a `otesanek2` ukazují další propad podílu pokusů o rootovské přihlášení. Na první příčku se naopak dostává přihlašovací jméno `admin`, které na serveru otesanek2 dosahuje 24,2%.

Tento jev pravděpodobně souvisí s rozšiřováním různých domácích ADSL a Wi-Fi routerů, které obvykle právě uživatelské jméno `admin` nejčastěji používají a rovněž jsou tato zařízení velmi často používána s výchozími hesly, obvykle bývají zapnuta nepřetržitě a málokdy mají dostatečná úložiště pro zaznamenávání a vyhodnocování pokusů o připojení, čímž se stávají velice zajímavým cílem pro hackery.

Dalším důvodem pro tuto změnu by mohla být určitá osvěta mezi administrátory a blokování rootovského přihlášení (viz kapitola 5).

Na obrázku 4.2 je histogram srovnávající počet pokusů o přihlášení uživatelů root a admin po měsících dohromady za všechny servery.



Obrázek 4.2: Srovnání množství přihlášení jako root a admin v čase

Z histogramu je patrné, že pokusů o rootovské přihlášení v čase neubývá, ale výrazně přibývá pokusů o přihlášení jako uživatel `admin`.

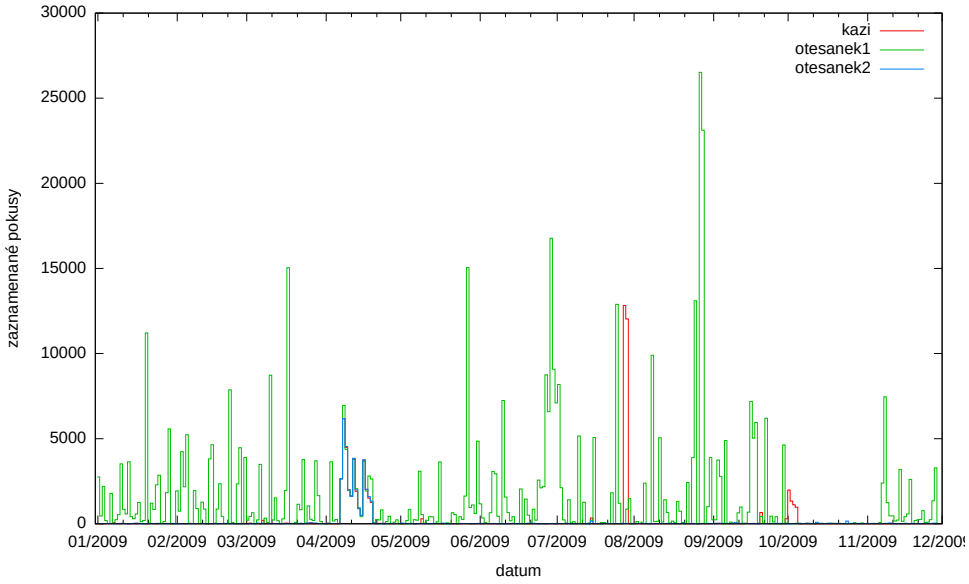
4.4.2 Útočníci

Celkový počet unikátních ip adres útočících na sledované servery byl 6 167, z tohoto 1 030 útočníku bylo zaznamenáno na všech těchto serverech, což tvoří 16,7 %. Při srovnání útoků

na servery `otesanek1` a `otesanek2` bylo zjištěno 1219 shodných útočníků, což tvoří neuvěřitelných 61,8 % všech útočníků na druhý z těchto serverů.

Jelikož každý z těchto serverů je připojen jiným ISP a nemají mezi sebou žádnou další vazbu, lze předpokládat, že většina útoků vedených na tyto servery nebyla přímo cílená na získání přístupu k těmto serverům, ale šlo o globálně prováděné útoky s cílem získání přístupu k jakémukoliv počítači.

Nejzajímavější v tomto ohledu byla série útoků vedených na konci měsíce dubna 2009 na všechny sledované servery, kdy byly dokonce na všech serverech zaznamenána stejná použitá uživatelská jména v přibližně stejném čase s odchylkou okolo půl minuty. Souvislost těchto útoků je dobře viditelná na histogramu 4.3, kde se počtem neplatných přihlášení hodnoty všech sledovaných serverů téměř překrývají.



Obrázek 4.3: Srovnání počtů pokusů o přihlášení na jednotlivých serverech v čase

Největší množství adres ve stejném C subnetu, které byly použity, byl 31, tedy 12% celé této sítě, všechny tyto ip adresy byly použity pouze k jednomu útoku s celkovým množstvím 198 pokusů o přihlášení a pravidelnými 2-sekundovými intervaly mezi jednotlivými pokusy. Evidentně zde tedy musel fungovat nějaký koordinační mechanismus.

Celkové množství nalezených C subentů, ze kterých útočily alespoň 3 ip adresy je 117.

4.4.3 Souhrn nejdůležitějších poznatků

Útoky s nejvýznamnějším počtem jednotlivých pokusů spadaly do následujících kategorií:

- Útok s množstvím pokusů o přihlášení s obvykle několikasekundovými intervaly ze stejné ip
- Útok s mnoha pokusy za sekundu ze stejné ip adresy - tyto můžou, ale nemusí být vždy součástí distribuovaného útoku
- Distribuované útoky z malého množství ip adres, které se cyklicky opakují s několikaminutovými intervaly

- Distribuované útoky z malého množství ip adres v jedné síti, které se nepravidelně střídají, ale obvykle v pravidelných intervalech.
- Distribuované útoky z velkého množství ip adres, kdy každá ip adresa zaútočí typicky ve 2-4 pokusech a pokračuje další útočník.
- Samostatný útok prováděný ze stejné ip adresy, který je po několika týdnech zopakován z jiné ip adresy s naprosto stejnou sérií použitých přihlašovacích jmen.

Kapitola 5

Obrana proti útokům

V boji proti útokům je potřeba útoky nejprve detekovat a vyhodnotit a následně proti nim zvolit vhodná opatření.

5.1 Detekce útoků

Základním předpokladem pro úspěšnou obranu před útoky je jejich detekce, v této části se nebudeme příliš zabývat obecnými technikami detekce útoků v počítačových sítích, ale stručně probereme možnosti a nástroje, které připadají v úvahu při detekci útoků hrubou silou na službu SSH.

5.1.1 Logy démonů jednotlivých služeb

Základním a dnes prakticky jediným používaným způsobem detekce útoků hrubou silou je analýza systémového logu, do kterého běžící služby zapisují kromě jiných informací také záznamy o neúspěšných přihlášeních.

Struktura takovýchto záznamů v logu byla probrána v kapitole 4, zde pouze uvedu několik příkladů rozdílných formátů těchto záznamů, které se v logu mohou objevit.

```
Jan  5 19:46:05 kazi sshd[67344]: error: PAM: authentication error for games
                from c-71-205-237-26.hsd1.mi.comcast.net
Jan  5 19:46:11 kazi sshd[67357]: Invalid user pgsql from 71.205.237.26
Apr  7 04:07:07 glt-otesanek sshd[16939]: pam_unix(sshd:auth): authentication
                failure; logname= uid=0 euid=0 tty=ssh
                ruser= rhost=www.hostproud.com  user=root
Sep 15 07:38:32 kazi sshd[99677]: error: PAM: authentication error for root
                from 94.113.104.47
```

Je vidět, že univerzální detekce pomocí zpracování těchto logů nebude zcela jednoduchá, protože na různých systémech s různou konfigurací se může formát těchto záznamů značně lišit.

Dalším problémem při zpracování logů může být několikanásobné zapsání jednoho samostatného pokusu o přihlášení. Ssh démon může pro autentizaci uživatele použít externí modul (nejčastěji PAM), a ten, v závislosti na konfiguraci, do logu ke stejnému pokusu o přihlášení může zapsat také svůj záznam. Následuje příklad, kdy byl takto jeden neúspěšný pokus o přihlášení zaznamenán hned třikrát.

Nejprve SSH démon zjistil, že uživatel `ftp` neexistuje, zde by správně měl skončit, ale z nějakého důvodu se tak nestalo a byl zavolán příslušný PAM modul, který opět oznámil na dalším řádu neexistenci uživatele `ftp` a následně také oznámí, že přihlášení se nepovedlo.

```
Jan  5 19:47:23 kazi sshd[67543]: Invalid user ftp from 71.205.237.26
Jan  5 19:47:23 kazi sshd[67543]: error: PAM: authentication error for illegal
  user ftp from c-71-205-237-26.hsd1.mi.comcast.net
Jan  5 19:47:23 kazi sshd[67543]: Failed keyboard-interactive/pam for invalid
  user ftp from 71.205.237.26 port 1838 ssh2
```

5.1.2 p0f

Nástroj `p0f` sám o sobě k detekci útoků neposlouží, ale mohl by být zajímavým doplněním k informacím z logů. Tento nástroj umožňuje pasivní fingerprinting vzdáleného systému. Na základě hodnot zaslaných v hlavičkách TCP paketů a drobných odlišností od příslušných RFC je možno specifikovat například vzdálený operační systém a jeho verzi [20][9].

Přidáním výstupu tohoto programu do systémového logu k záznamům o útocích by mohlo být při analýze útoků přínosné[15].

5.1.3 skupiny útočníků

Distribuované útoky lze částečně zmírnit postupným blokováním jednotlivých útočných ip adres, ale aby bylo možno tyto útoky odrážet účinněji, je zapotřebí hledat mezi jednotlivými dílčími útoky nějaký vztah. Jelikož je distribuovaný útok obvykle prováděn nějakým botnetem, je vhodné tyto útoky zaznamenávat a hledat mezi nimi souvislosti, identifikovat související ip adresy a blokovat je v případě opakovaného útoku po celých skupinách, při opakovaném útoku stejného botnetu bude pravděpodobně použita podobná množina ip adres[12].

5.2 Prevence útoků

V této sekci probereme typická doporučení pro obranu před útoky hrubou silou.

5.2.1 Vyžadování silných hesel a jejich kontrola

Velice častým typem útoků hrubou silou je slovníkový útok, proto je krajně nevhodné používání slovníkových hesel, stejně tak používání hesel obsahujících příslušné uživatelské jméno. Vhodným způsobem pro zajištění toho, aby uživatelé nepoužívali příliš slabá hesla, je používání `PAM-cracklib` modulu, který se při změně hesla postará o posouzení síly hesla knihovnou `cracklib` a nedovolí nastavit slabé heslo.

Zábavnou ukázkou toho, jak závažným a rozšířeným problémem je používání slabých hesel, může být následující citace ze serveru lamer.cz

```
<a> normálně teď jsem se přihlašoval na ICQ na Meebo a jak jsem ťukal
    poslední číslici jména, tak jsem místo 3 napsal 2
<a> zadal jsem tam svoje heslo a normálně mne to přihlásilo na účet
    nějaký holky odněkud z Malajsie :-)) měla stejný heslo... měla...
<b> mmnt
<b> co se taky divíš, když máš heslo password?
<b> měl jsi
```

5.2.2 Zákaz přihlášení roota pomocí ssh

Jak bylo potvrzeno v kapitole 4, jedním z vůbec nejčastěji zkoušených uživatelských jmen při útocích hrubou silou je root, což je uživatel existující na každém unixovém systému a má prakticky neomezená práva, díky čemuž se stává snadným a logickým cílem útočníků. Nejjednodušší reakcí je tedy přímé rootovské přihlášení neumožnit (viz část 2.2) a pro přihlášení správce používat jiné přihlašovací jméno, kterému bude umožněno použití příkaz `su`/`sudo`.

5.2.3 Používání nesnadno uhodnutelných přihlašovacích jmen

V případě cíleného útoku například na servery konkrétní společnosti lze často snadno odvodit existující uživatelská jména z veřejně dostupných informací o zaměstnancích společnosti, které pak lze zkoušet při útoku. Je proto vhodné vytvářet pro uživatele nesnadno odvoditelné loginy. Tato praktika ovšem může značně znepříjemnit správu uživatelů, proto se příliš nepoužívá a spíše se dbá na to, aby uživatelé používali silná hesla.

5.2.4 Ssh démon na nestandardních portech

Všechny běžné síťové služby mají přiřazeny standardní čísla portů[1], na kterých jejich démoni naslouchají, u SSH je to port číslo 22. V případě změny čísla portu, na kterém bude SSH démon poslouchat, bude odražena většina automatizovaných útoků, ale bude znesnadněno přihlášení všem regulérním uživatelům, protože budou muset číslo portu znát a zadávat. V případě cíleného útoku ale není pro útočníka problém nejdříve proskenovat všechny porty na serveru a SSH na jiném portu objevit. (Nastavení viz sekce 2.2)

5.2.5 Povolení přihlášení z povolené množiny ip adres

V případě, že nám stačí mít omezený počet míst, ze kterých se na server budou uživatelé připojovat, je možno nakonfigurovat firewall tak, aby povolil připojení pouze z těchto míst. V naprosté většině případů je takovéto omezení ale nežádoucí.

5.2.6 Port-knocking

Další technikou, která by se dala zařadit do kategorie "security-by-obscurity" je port-knocking. Jedná se o nakonfigurování firewallu systému tak, aby defaultně blokoval všechny pokusy o navázání spojení na portu služby SSH a spuštění démona `knockd`. Uživatel, který se k systému bude připojovat poté potřebuje znát čísla portů, na kterých naslouchá `knockd` a ve správném pořadí na ně "zaklepat" zasláním paketů programem `knock`. Poté je na krátký časový interval povoleno z klientovy ip adresy přihlášení.

Tento přístup odrazí jak automatické globální útoky hrubou silou, tak většinu cílených, pokud útočník nemá dostatečné znalosti o prostředí serveru. Cenou za toto opatření je ovšem nepohodlné přihlašování uživatelů.

5.2.7 Přihlášení pomocí RSA/DSA klíčů

Přihlášení použitím asymetrických klíčů je založeno na asymetrické kryptografii. Detaily o principech používaných šifer zde nebudeme probírat, jen stručně popíšu jak toto funguje z uživatelského pohledu.

[1] Do roku 2001 přidělována organizací IANA, nyní ICANN

Uživatel si příkazem `ssh-keygen` vygeneruje dvojici klíčů - privátní a veřejný, ty se podle zadaných parametrů pak vytvoří v adresáři `/.ssh/` pod názvy `id_rsa` a `id_rsa.pub`.[2] Veřejný klíč se vloží do souboru `/.ssh/authorized_keys` na serveru.

Při připojení klienta k serveru je pak ověřena pravost uživatelova privátního klíče a v případě úspěchu je uživatel přihlášen a již není požadováno jeho heslo. Na rozdíl od dříve zmíněných metod, které přihlášení uživatele komplikovaly, tato metoda přihlášení usnadňuje. SSH démon lze poté nakonfigurovat tak, aby přihlášení heslem vůbec neumožňoval a bylo možné se přihlásit jen pomocí klíčů, viz popis konfigurace sshd v části 2.2.

5.2.8 Použití nestandardních autentizačních modulů

Tato metoda je založena na předpokladu, že nástroje, kterými jsou prováděny útoky hrubou silou, používají pro přihlašování existujícího ssh klienta a například pomocí utility `expect` s ním komunikují. Následuje krátký fragment skriptu, který jsem kdysi používal pro automatizované přehrávání konfigurací routerů. Skript se pokusí přihlásit k serveru `$srv_addr` a na výzvu k zadání hesla předá hodnotu `$srv_pass`.

```
#!/usr/bin/expect -f
set timeout -1
spawn ssh $srv_addr
match_max 100
expect {
"*yes/no*"      {       # pokud není server v ~/.ssh/known_hosts
send "yes\n"
expect "*?assword:*"
send "$srv_pass\n"
}

"*?assword:*"   {
send "$srv_pass\n"
}
}
```

Je vidět, že tento skript by příliš nepochodil pokud by dotaz na zadání hesla nesplňoval uvedený výraz, zatímco uživateli by jiný řetězec vyzývající ho k zadání hesla pravděpodobně nijak nevadil, pokud by ho ale nepovažoval za nějaký útok.

Nástroje pracující na o něco nižší vrstvě by se tomuto pravděpodobně vyhnuly a zkoušené heslo by zaslaly ať už by dotaz k zadání hesla byl jakýkoliv, ovšem vypořádat se s tím, že se jich server zeptá kromě hesla například také na to, jak se jmenuje uživatelův křeček, už by jim pravděpodobně způsobilo problémy, zatímco skutečného uživatele by to příliš neomezilo.

Způsobem řešení je tedy vytvoření vlastního PAM modulu, který by pomocí autentizační metody `keyboard-interactive` (viz popis autentizačních metod v části 2.3) požadoval kromě hesla ještě odpověď na nějakou jednoduchou otázku.

[2] samozřejmě v případě RSA klíče, při DSA jsou názvy obdobné

5.2.9 Blokování ip adres, které zaslaly neplatné pokusy o přihlášení

Tento přístup je založen na automatickém prohledávání logů SSH démona a blokování ip adres, které prováděly na počítač útok hrubou silou. Všechny současné nástroje (viz kapitola 6) k tomuto ovšem přistupuji velice primitivním způsobem a nereflektují problém distribuovaných útoků, proto je cílem této práce vytvoření nových postupů právě v této oblasti odrážení útoků.

5.3 Blokování přístupu podle ip adresy

Jak bylo naznačeno v předchozí podkapitole, jednou z technik obrany před útoky hrubou silou, kterou se take dále budeme více zabývat, je blokování ip adres, ze kterých bylo dříve útočeno. Pro samotné zablokování přístupu v prostředí GNU/Linux a na BSD může být použito některého z následujících nástrojů.

5.3.1 iptables

Program `iptables` je určený pro operační systém Linux a slouží ke konfigurování pravidel IPv4 paketového filtru v jádrech v. 2.4 a 2.6. Samotný program umožňuje přidávání, mazání a vypisování jednotlivých pravidel, dále pak například získávání statistik o paketech, které byly dle jednotlivých pravidel zpracovány.

Nejzákladnějšími příkazy, které poslouží k prostému zablokování paketů z ip adresy `a.b.c.d` a jejich opětovnému odblokování jsou

```
$ iptables -A INPUT -s a.b.c.d -j REJECT
$ iptables -D INPUT -s a.b.c.d -j REJECT
```

První příkaz vloží nové pravidlo do tzv. chainu INPUT, ve kterém se jako první hledají pravidla pro příchozí pakety a nařídí, že všechny pakety, které přijdou ze zadané ip adresy, budou zamítnuty. Akce `REJECT` odpoví příslušným ICMP paketem, že spojení nebylo možno navázat. Další možností, která ovšem není v souladu s RFC-1122, je použití akce `DROP`, která paket zahodí a žádnou odpověď odesílateli vracet nebude. Je na uvážení uživatele jak se zachová.

V příkladu jsme nespecifikovali žádný protokol, ani port, proto bude blokována veškerá příchozí komunikace ze zadané ip adresy, podrobnější popis konfigurace je možno nalézt v příslušných manuálových stránkách a literatuře[5].

Pro IPv6 existuje obdoba programu iptables s názvem ip6tables.

5.3.2 ipfw

Program `ipfw` je nativní paketový filtr v prostředí FreeBSD. Základními příkazy pro zablokování a ukončení blokovaání ip adresy `a.b.c.d` je možno použít příkazy

```
$ ipfw add deny tcp from a.b.c.d to any
$ ipfw delete ‘ipfw list | grep -i a.b.c.d | awk ’{print $1;}’‘
```

Zde je možno si všimnout lehkého hacku v druhém příkazu, `ipfw` neumožňuje přímo odebrat pravidlo jeho kompletní specifikací, ale je třeba zadat jeho číslo, to je zde nutné zjistit z výpisu pravidel. Podrobnější informace o `ipfw` lze najít v literatuře [18] a manuálových stránkách.

5.3.3 null route

V případě, že na systému neexistuje žádný paketový filtr, je možno pro zablokování komunikace směrované na nějakou konkrétní ip adresu použít speciální směrovací pravidlo, které nepovede k žádnému cíli. Tento způsob blokování komunikace se nazývá "null routing" nebo také "blackhole filtering". Některé systémy jako Linux, Solaris nebo BSD umožňují nastavení speciálního příznaku pro nastavení takovýchto směrovacích pravidel, na následujícím příkladu je uvedeno jak je možno využít nástroje `route` nebo `ip` na systému Linux a příkaz `route` na systémech Solaris a BSD[3].

```
$ route add -host a.b.c.d reject
$ ip route add blackhole a.b.c.d/32
$ route add -host a.b.c.d 127.0.0.1 -blackhole
```

Na systémech, které tento příznak nemají, je možné jako cílovou adresu použít nějakou, která v sítí neexistuje.

5.3.4 tcp_wrapper

Tcp wrapper je filtr fungující na aplikační úrovni a zpravidla bývá knihovna `libwrap` s kódem tohoto filtru přilinkována ke všem základním démonům na dnešních unixových systémech. Pro konfiguraci se používají dva soubory, `/etc/hosts.allow` a `/etc/hosts.deny`.

První ze souborů obsahuje seznam adres, ze kterých je povoleno se serverem komunikovat, druhý obsahuje seznam adres, ze kterých je komunikace blokována. Obvykle se používá pouze jeden z nich podle toho, zda upřednostňujeme blokování všech a povolení vyjmenovaných klientů či naopak, do druhého z těchto souborů se pak vkládá řádek ve tvaru `ALL: ALL`.

Každý řádek v těchto souborech obsahuje název služby, které se pravidlo týká a adresu, která má být povolena nebo zakázána. Příkladem pro zablokování služby SSH z adresy `a.b.c.d` je řádek v následujícím tvaru:

```
sshd: a.b.c.d
```

Pro zablokování všech služeb je potom možno použít klíčové slovo `ALL`. Více informací opět v literatuře[8].

Kapitola 6

Dostupné nástroje

Pro ochranu před útoky hrubou silou na službu ssh existuje spousta nástrojů, které ve svém principu všechny fungují velice podobně. Pravidelně čtou log ssh démona (viz předchozí kapitola) a hledají v něm opakované chybné pokusy o přihlášení. V případě detekce útoku poté realizují určité kroky k zabránění v dalším přístupu útočníka k počítači. zde uvedu a popíšu tři, dle mého pohledu nejzajímavější, z nich.

6.1 fail2ban

Fail2ban je sada skriptů napsaných v pythonu, může buď periodicky kontrolovat logy služeb (polling) nebo pomocí FAM/Gamin zjišťovat změny a teprve potom logy kontrolovat. Fail2ban má připravené filtry pro hlídání logů spousty služeb a umožňuje snadno pomocí regulárních výrazů přidávat další. Kromě klíčové služby ssh například umí kontrolovat logy poštovního serveru postfix, serveru proftpd, umí pohlídat spoustu nekalých činností nad webovým serverem apache a v neposlední řadě zde jsou také obecné filtry pro PAM.

Možnosti blokace útočných ip adres jsou zde také rozsáhlé, kromě připravených konfigurací pro nastavení firewallu pomocí programů `iptables`, `ipfw` nebo využití tcp wrapperu přidáním adresy útočníka do souboru `/etc/hosts.deny`, lze systém nakonfigurovat pro pro zavolání jakéhokoliv příkazu[4].

Nástroj je k dispozici je na adrese http://fail2ban.org

6.2 sshguard

Sshguard je, jako jeden z mála softwarových nástrojů pro detekci útoků hrubou silou, kompilovaný a je napsaný v jazyce C, díky čemuž slibuje vyšší výkon, než ostatní nástroje. Dále se snaží být velice snadno použitelný a nepoužívá žádný konfigurační soubor, lze pouze nastavit několik základních hodnot jako parametry příkazové řádky.

Program sshguard čte log ze standardního vstupu, proto vyžaduje vytvoření pravidla systémového logeru pro logování přímo na tento vstup. O vše ostatní již se program postará sám, z logu detekuje všechny běžící programy, které zná (kromě `ssh` například `dovecot`, `proftpd` nebo `ftpd`) a všechny začne sledovat. Rovněž detekuje přítomné firewally (spolupracuje s `PF`, `ipfw`, `iptables`, případně zapisuje blokované adresy do`/etc/hosts.deny`).

Za zmínku dále ještě stojí podpora ipv6 a v poslední verzi přidaná podpora whitelistů, kterou ovšem v současném stavu pokládám spíše za nouzové řešení, spočívá v přidání jednoho nebo více parametrů `-w` příkazové řádky, následovaných jednou ip adresou[11].

6.3 denyhosts

Projekt denyhosts je zajímavý tím, že na rozdíl od předešlých nástrojů udržuje centrální celosvětovou databázi identifikovaných útočníků a jednotliví klienti do ní mohou přispívat svými úlovky a také z ní čerpat.

Denyhosts může být spouštěn v určitých intervalech cronem nebo běžet jako démon, kdy je ovšem také výkonná část programu spouštěna v určitých intervalech. Program si zaznamenává poslední přečtenou pozici logu, aby věděl kde začít pří příštím čtení.

Denyhosts nabízí poměrně rozsáhlé možnosti konfigurace, kdy je možno nastavit například rozdílné limitní množství neplatných pokusů o přihlášení pro existující uživatele počítače, pro neexistující uživatele a pro uživatele root.

Již dříve zmíněná vlastnost - distribuce získaných útočných ip adres pomocí centrálního serveru je bezesporu velice zajímavá, ale přináší také rizika podvržených ip adres, v systému neexistuje žádný autentizační mechanismus, který by ovšem stejně neměl velký smysl při dostupných zdrojových kódech. Tento problém je zde částečně řešen použitím dalších dvou konfiguračních direktiv, které umožňují klientovi nastavit minimální počet jiných klientů, kteří musí danou ip adresu nahlásit, aby byla ze serveru zaslána a minimální rozsah časů, kdy byla daná ip adresa nahlášena od různých klientů[17].

Software je k dispozici na adrese http://denyhosts.sourceforge.net

6.4 zhodnocení dostupných nástrojů

Nyní provedeme krátké zhodnocení uvedených nástrojů

6.4.1 Výchozí konfigurace

V tabulce 6.1 je srovnání výchozích konfigurací dříve uvedených nástrojů. Hodnoty na prvním řádku (`findtime`) tabulky specifikují časový interval, ve kterém musí dojít k počtu neplatných pokusů specifikovaném na třetím řádku (`maxretry`), aby došlo k zablokování útočící ip adresy. Druhý řádek (`bantime`) poté specifikuje dobu, po které je přístup pro danou ip adresu opět povolen. Poslední řádek tabulky `polling_interval` pak říká v jakém intervalu daný nástroj přistupuje k systémovému logu, ve kterém hledá nové útoky.

hodnota	**fail2ban**	**sshguard**	**denyhosts**
`findtime`(s)	600	1200	5d/10d/25d
`bantime`(s)	600	420	INF
`maxretry`	5	4	5/5/1
`polling_interval`(s)	1	N/A	30

Tabulka 6.1: výchozí nastavení některých nástrojů pro odrážení útoků hrubou silou

Hodnota `findtime` nástroje sshguard není dokumentována, u programu denyhosts je tato hodnota nastavitelná zvlášť pro pokus o přihlášení reálného uživatele, neexistujícího uživatele a uživatele root. Stejné rozdělení je u tohoto programu také použito u hodnoty `maxretry`.

Nastavení hodnoty `polling_interval` pro fail2ban nelze měnit, ale nástroj podporuje sledování změny logovacího souboru pomocí FAM/Gamin, program sshguard z principu činnosti, popsaného výše, žádný polling nepoužívá.

Z tabulky je patrno, že nástroje fail2ban a sshguard jsou ve výchozím nastavení k útočníkům poměrně hodně mírné, zatímco nástroj denyhosts v některých případech až neskutečně nekompromisní.

6.4.2 Nedostatky současných nástrojů

Hlavními nedostatky v současnosti dostupných nástrojů jsou:

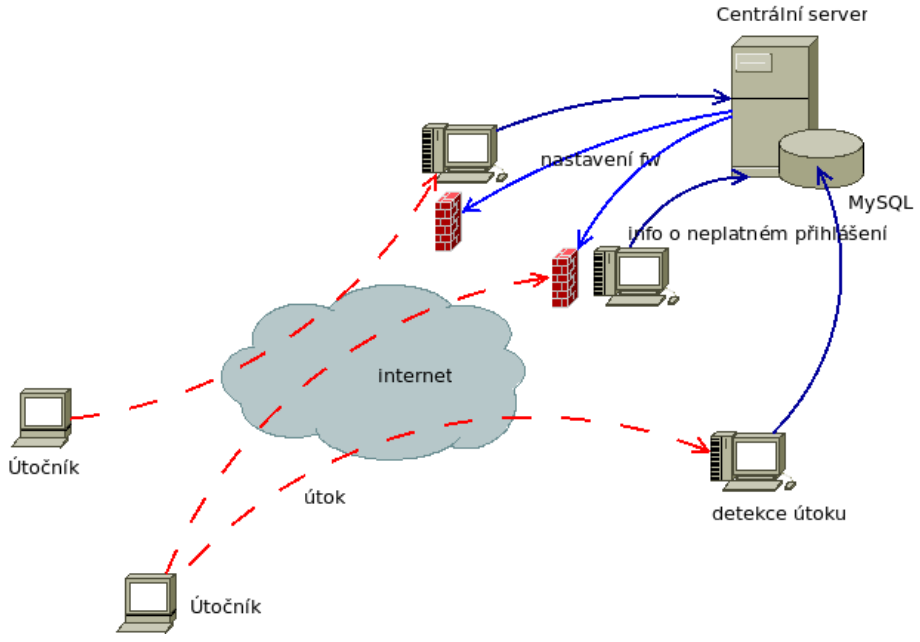
- **kontrola logu po jednotlivých řádcích** - jeden špatný pokus je možno identifikovat jako několik pokusů, pokud do logu zapíše oznámení o neúspěšném přihlášení jak démon služby, tak například PAM,
- **běží pouze na jednom stroji** (některé) - omezené možnosti rozpoznání útoků,
- nebo naopak **běží globálně** (některé) - probíhá celosvětová výměna odhalených útočníků, ale v tomto měřítku nelze účinně uhlídat podvržené ip adresy,
- **neúčinné sdílení informací o útocích** - přestože některé systémy pro detekci a blokování útoků provádějí výměnu informací o zablokovaných adresách, vyhodnocení probíhá vždy pouze na jednotlivých stanicích a nelze proto efektivně odhalit všechny útoky, které by se daly odhalit porovnáváním informací z více stanic vzájemně,
- **detekují útoky pouze na základě ip adresy** - často jsou vedeny distribuované útoky, které ale spojuje použité uživatelské jméno,
- **nespojují útočníky do skupin** - neumožňují tak detekci distribuovaných útoků, vhodným vylepšením by bylo inteligentní vytváření vztahů mezi jednotlivými útočícími ip adresami pro pozdější blokování celých skupin na základě několika málo shodných adres,
- **neperzistentní data o blokovaných útočnících** - po ukončení nástroje jsou informace o útočnících a jejich blokaci zahozeny nebo zůstávají v nějakém logu vytvářeném tímto nástrojem, ale již nikdy nejsou znovu využity.

Kapitola 7

Návrh nástroje pro ochranu před distribuovanými útoky hrubou silou

Mnou vytvářený systém bude systém klientů detekujících útoky s centrálním serverem, který bude vyhodnocovat jednotlivé útoky a na základě definovaných pravidel distribuovat ostatním klientům nová pravidla pro filtrování komunikace.

Na obrázku 7.1 je naznačeno schéma systému. Systém sestává z centrálního serveru s MySQL databázovým serverem a přilehlé počítačové sítě. Na jednotlivých uzlech v síti běží jednoduchý klient, který detekuje nezdařené pokusy o přihlášení, na centrálním serveru potom běží hlavní část celého systému, která zpracovává data od klientů, vyhodnocuje útoky a předává klientům informace o adresách, které mají být zablokovány, případně odblokovány. Klienti využijí dostupných nástrojů (`iptables`) k přizpůsobení firewallu dle instrukcí serveru.



Obrázek 7.1: Schéma systému

7.1 Způsoby detekce a blokace

Detekce útoků by měla probíhat několika způsoby

- několik pokusů (na stejné nebo různá pc) z jedné ip adresy
- několik pokusů v krátkém časovém intervalu se stejným loginem z různých ip adres (případně na různé klienty)
- několik pokusů z různých ip adres ze stejné sítě s C maskou a blokace celé sítě
- opakující se vzorek útočníků ze staršího útoku a zablokování celé skupiny identifikované v tomto předešlém útoku

Pro detekci by měly být shromažďovány rozmanité informace, které napomůžou k dokonalejším možnostem a preciznějšímu nastavení detekce útoků.

- důležitá je rozhodně ip adresa potenciálního útočníka
- použitý login je velice užitečná informace
- informace zda jde o existující nebo neexistující uživatelský účet
- v případě zadání neexistujícího loginu by mohl být užitečným mechanismus zjišťující stupeň podobnosti zadaného loginu s některým z existujících uživatelů
- samozřejmostí je také datum a čas incidentu
- dále při identifikaci útoků pomůže TCP SYN fingerprint získaný nástrojem `p0f`

7.2 Přínos

Tato implementace systému by oproti stávajícím implementacím měla umožnit pružnější zabezpečení středně velkých počítačových sítí. Momentálně existují implementace, které buď chrání jeden samostatný počítač nebo sdílejí informace s celým světem, ale neexistuje žádný mezistupeň[1]. Dále by měla umožnit administrátorovi sítě jednodušší centrální konfiguraci celého systému.

Toto řešení by tedy mělo být přínosem především pro středně velké a velké počítačové sítě.

7.3 Cíle

- Systém by měl implementovat funkcionalitu dostupných nástrojů pro blokování útoků hrubou silou.
- Dále by měl systém vyřešit co nejvíce nedostatků zmíněných v sekci 6.4.
- Projekt by měl být zaměřen především na rozšíření stávajících metod detekce o nové metody detekce distribuovaných útoků.
- Podle poznatků uvedených v části 4.4.3 je také vhodným plánem provádění okamžité blokace ip adresy, ze které došlo k neúspěšnému přihlášení na přiměřeně krátký čas, aby tím nebyl omezen reálný uživatel, který jen špatně zadal heslo.

[1] Nástroj DenyHosts sice počítá s možností použití jiného serveru pro synchronizaci dat o útocích, ale v současnosti není žádný jiný server neexistuje.

7.4 Rizika

Z návrhu systému také vyplývají určitá rizika, na která bude třeba při implementaci brát zřetel a vypořádat se s nimi.

- u velkých sítí s velkým množstvím klientů, na které může být útočeno, bude docházet k poměrně velkým přenosům mezi klienty a centrálním serverem, což by mohlo vyústit až v nechtěný DDoS útok na centrální server klienty.
- při vyhodnocování na serveru budou muset klienti spoléhat na dostupnost serveru a také informacím od serveru plně důvěřovat. Při podvržení dat ze serveru by pak mohl útočník snadno přidávat a odebírat pravidla firewallu na klientech. Při zamýšleném nasazení především na počítačích v samostatných sítích, kde má administrátor všechny síťové prvky pod kontrolou, by toto sice neměl být zásadní problém, ale přesto je třeba s takovou možností počítat.
- při příliš agresivní metodě blokování pokusů o útok systém přijde o cenné informace, které by mohl získat z následujících kroků útoku a tak snižovat úspěšnost detekce budoucích útoků. S tímto problémem by mohli pomoci speciální klienti v roli honeypotů, kteří nebudou provádět žádnou blokaci příchozích spojení.

Kapitola 8

Implementace

Po analýze a návrhu systému pro ochranu počítačové sítě před distribuovanými útoky hrubou silou přichází na řadu implementace Host-based IPS nástroje, který jsem nazval jednoduše DBFAP (Distributed Brute Force Attack Protector). V této kapitole je dále popsán návrh jednotlivých částí systému, struktura databáze, popis konfigurace a také vnitřní logika celého systému.

Jak již bylo v kapitole 7 o návrhu systému popsáno, implementovaný systém sestává ze dvou komunikujících částí, serveru a klienta. Zatímco server je v takovémto systému jeden[1], klientů může být libovolný počet.

Klientská část je aplikace napsaná v jazyce C++. Klient čte systémový log a vždy po nalezení záznamu o chybném přihlášení zapíše do databáze na serveru informace o tomto přihlášení. V pravidelných intervalech potom ze serveru získává seznam adres, které mají být zablokovány.

Serverová část je tvořena pouze MySQL serverem s databází a sadou triggerů, které se starají o celou logiku vyhodnocování útoků z neúspěšných pokusů o přihlášení, o nichž jsou na server zapisovány informace pouze jednoduchými dotazy z klientů. Tímto je do maximální míry omezena komunikace mezi klientem a serverem, potřebná k předání a zpracování události.

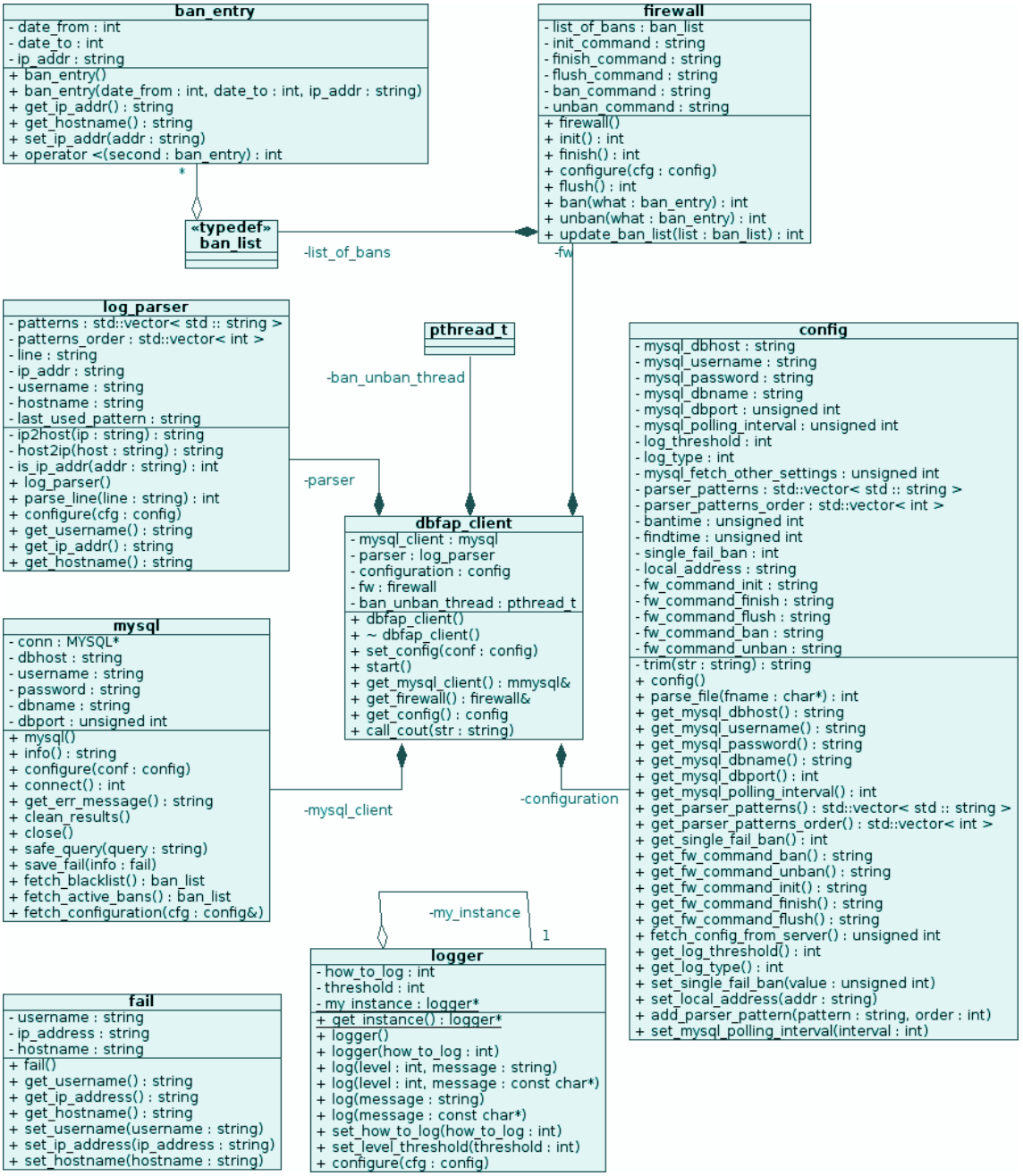
8.1 Popis klienta

Klient DBFAP sestává z několika tříd, znázorněných na diagramu 8.1, které budou dále popsány.

config je třída, jejíž instance se stará o načtení všech hodnot z konfiguračního souboru a z databáze (za pomocí třídy mysql) a poté tyto hodnoty zpřístupňuje všem ostatním třídám pro jejich nakonfigurování.

fail instance třídy fail slouží k uchovávání a předávání informací o jednotlivých neúspěšných pokusech o přihlášení do systému mezi instancí třídy log_parser a mysql.

[1] Bylo by ovšem možno použít například MySQL cluster nebo MySQL replikaci pro zajištění vyšší dostupnosti a/nebo rozdělení zátěže serveru.



Obrázek 8.1: Diagram tříd klientské části aplikace DBFAP

dbfap_client je hlavní, řídící třída, jejíž metoda `start()` obsahuje hlavní smyčku celého programu. Z této metody je také na začátku spuštěno druhé vlákno, které v pravidelných intervalech aktualizuje konfiguraci firewallu. Hlavní vlákno potom na standardním vstupu čte log ssh démona (případně kompletní log systému) a informace získané z těchto logů odesílá na databázový server.

firewall se stará o konfiguraci zvoleného firewallu, obsahuje metody pro inicializaci firewallu pro použití aplikací DBFAP, pro zablokování nebo odblokování konkrétní ip adresy nebo sítě, pro aktualizaci seznamu blokovaných ip adres a pro odstranění všech

změn provedených ve firewallu po ukončení aplikace.

Firewall je touto třídou obsluhován pomocí volání nakonfigurovaných příkazů (viz část o konfiguraci 8.4). Následuje seznam těchto příkazů a příklady konkrétních příkazů pro firewall `iptables`.

- `init_command` je příkaz volaný pro inicializaci firewallu, pro `iptables` je zde použito následující volání:

  ```
  /sbin/iptables -N DBFAP;
  /sbin/iptables -A DBFAP -j RETURN;
  /sbin/iptables -I INPUT -p tcp --dport ssh -j DBFAP
  ```

  První část příkazu vytvoří nový vyhodnocovací řetězec (chain) `DBFAP` ve firewallu `iptables`, další část vytváří jediné pravidlo tohoto řetězce, které říká, že pokud vyhodnocení nově vytvořeného řetězce dospělo až k němu, bude řízení vráceno do volajícího řetězce. Poslední část poté vkládá do základního řetězce `INPUT` pravidlo, které říká, že každý příchozí paket, který směřuje na port ssh, bude zpracován řetězcem `DBFAP`.

  V případě, že není příkaz pro inicializaci definován, předpokládá se, že firewall nepotřebuje žádnou inicializaci.

- `finish_command` je příkaz, který aplikace spouští před svým ukončením. Tento příkaz se postará o vrácení systémového firewallu do původního stavu před spuštěním aplikace. Pro `iptables` je zde použito:

  ```
  /sbin/iptables -D INPUT -p tcp --dport ssh -j DBFAP;
  /sbin/iptables -F DBFAP;
  /sbin/iptables -X DBFAP
  ```

  První z těchto příkazů zruší pravidlo, které směřuje všechny příchozí ssh pakety na náš řetězec `DBFAP`, druhý příkaz odstraní všechny pravidla z tohoto řetězce a poslední příkaz následně tento řetězec odstraní.

  V případě, že není tento příkaz definován, aplikace namísto něj při ukončení zavolá příkaz `flush_command`.

- `flush_command` je příkaz, který odstraní všechny blokující pravidla z firewallu. Pro `iptables`:

  ```
  /sbin/iptables -F DBFAP
  ```

  Tento příkaz již byl použit a popsán jako součást předešlého příkladu.

  V případě, že tento příkaz není definován, je jeho volání nahrazeno postupným voláním příkazu `unban_command` pro jednotlivé adresy, které se aktuálně nachází v banlistu.

- `ban_command` je příkaz, který přidává do firewallu nové pravidlo s adresou k blokaci. Pro iptables:

  ```
  /sbin/iptables -I DBFAP 1 -s ${IP_ADDR} -j DROP
  ```

  Tento příkaz vkládá pravidlo na první místo našeho řetězce `DBFAP` a říká, že pokud je zpracovávaný paket zaslaný z uvedené ip adresy, bude zahozen. Umístění pravidla na začátku je důležité, protože posledním pravidlem tohoto řetězce je pravidlo, které vrací vyhodnocování zpět řetězci `INPUT`.

- `unban_command` je posledním příkazem spouštěným třídou `firewall`, který odstraňuje pravidlo pro blokaci z firewallu, pro `iptables` je používáno:

  ```
  /sbin/iptables -D DBFAP -s ${IP_ADDR} -j DROP
  ```

  Uvedený příkaz odstraní specifikované pravidlo z námi dříve definovaného řetězce `DBFAP`.

logger je třída implementovaná pomocí návrhového vzoru singleton a slouží všem ostatním třídám pro zápis informací do systémového logu. V konfiguraci systému je možno stanovit, jak podrobné výpisy mají být do logu zapsány a zda má být logováno pomocí systémového loggeru nebo na standardní chybový výstup aplikace.

log_parser slouží k parsování logu ssh démona a získávání informací o neúspěšných pokusech o přihlášení k systému. Jednotlivé řádky logu jsou zpracovávány pomocí regulárních výrazů a nejdůležitější úlohu zde plní funkce z knihovny `regex.h`. Jednotlivé regulární výrazy jsou konfigurovatelné a v závislosti na použitém systémovém loggeru, ssh démonu nebo používaných PAM modulech se může potřebná množina těchto výrazů lišit. Pro co možná nejlepší výkon je vhodné ponechat zde pouze výrazy, které mají na konkrétním systému opodstatnění a zároveň parsovat pouze log sh démona, nikoliv log celého systému.

V této třídě je také ošetřen jeden z neduhů většiny současných systémů pro detekci útoků hrubou silou, a to ten, že je při parsování jednoho neplatného pokusu o přihlášení toto přihlášení vyhodnoceno hned jako několik opakovaných pokusů, protože je o něm do logu zaznamenáno více různých řádků. Zde je tento problém řešen tak, že je naposled použitý výraz uchován a pokud další řádek logu vyhoví se stejným výsledkem rozdílnému výrazu, je tento řádek ignorován.

mysql třída mysql slouží pro komunikaci s MySQL databází na centrálním serveru, kromě metod přímo souvisejících s prací s MySQL databází implementuje také přímo metody pro zápis do tabulky `fail` na serveru a pro získávání seznamů ip adres určených k dočasnému zablokování a ke stažení blacklistu.

ban_entry je třída obsahující základní informace o útočníkovi, který má být zablokován, tato třída je v aplikaci často využívána jako součást vektoru `ban_list`.

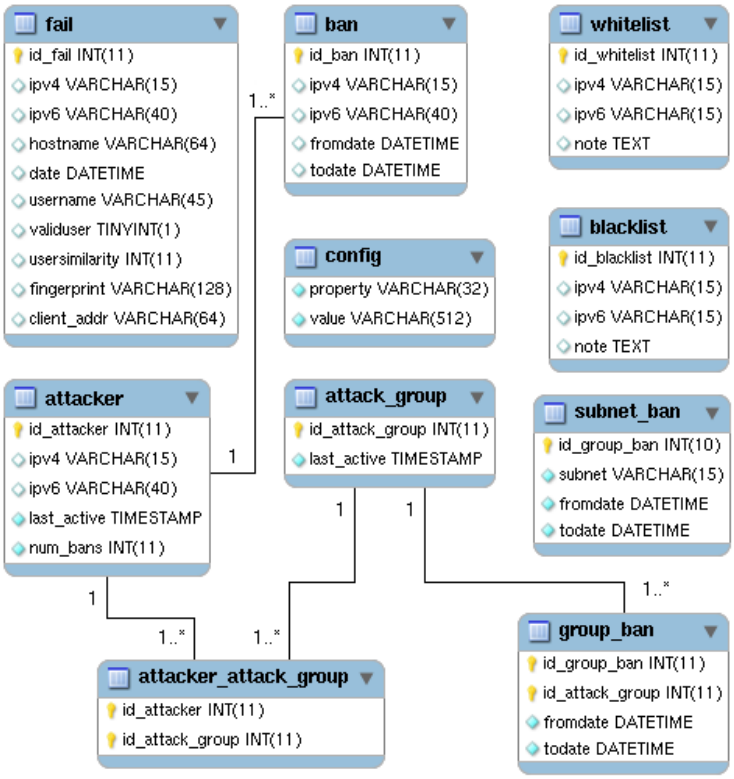
8.2 Popis databáze

Struktura databáze se v průběhu implementace systému mnohokrát měnila až dospěla do svého konečného stavu, který je vidět na obrázku 8.2. Tato sestava tabulek je nejdůležitější součástí systému pro analýzu a záznam informací o provedených útocích.

Jednotlivé tabulky dále stručně popíšu.

config je jednoduchá tabulka obsahující různé konfigurační hodnoty sloužící k vyhodnocování útoků nebo konfiguraci klienta (viz detailnější popis v části 8.4). Tabulka obsahuje dva sloupce, sloupec `property` specifikuje název a sloupec `value` hodnotu konfigurační položky.

fail je jediná tabulka, do které zapisuje klientská aplikace a jsou zde zaznamenávány všechny pokusy o přihlášení ke klientským systémům, které nebyly úspěšné. Trigger nad touto tabulkou provádí základní vyhodnocování a může zapisovat do tabulky



Obrázek 8.2: Návrh MySQL databáze pro systém detekce útoků

ban. Každý záznam obsahuje adresu útočníka, použité uživatelské jméno, datum, kdy byl neplatný pokus zaznamenán a který z klientů zde tento záznam zapsal.

ban je další tabulka na pomyslné cestě záznamů, které v naší databázi vznikají. Do tabulky ban jsou triggerem nad tabulkou fail vkládány záznamy o ip adresách, které mají být klienty zablokovány. Na tuto tabulku jsou navázány další dva triggery, jeden se provádí před vložením záznamu, druhý po vložení, tyto triggery se starají o úpravu délky banu pro opakované útočníky a o generování záznamů do dalších tabulek, především týkajících se rozlišení a blokování skupin útočníků. K jednotlivým banům je v této tabulce zaznamenána ip adresa útočníka, čas začátku a čas konce blokování.

subnet_ban slouží k zaznamenávání blokovaných sítí, které můžou být zablokovány na základě vyhodnocení triggerem tabulky fail. Tabulka obsahuje pro každý záznam adresu blokované sítě a čas začátku a konce blokování této sítě. Jako sítě pro tuto tabulku jsou rozpoznávány pouze sítě s maskou C, implementace jiných než základních masek by zde byla problematická a blokovat sítě s maskou B by již mohlo nepříjemně ovlivnit také oprávněné uživatele.

attacker uchovává informace o každém zaznamenaném útočníkovi a podílí se na spojování útočníků do skupin. Zaznamenána je zde adresa útočníka, čas posledního útoku a celkový počet útoků provedených z této adresy.

attack_group zaznamenává jednotlivé skupiny útočníků. Vždy při zaznamenání útoku je tento nový útok buď přiřazen k poslední skupině v této tabulce nebo je vytvořena

nová skupina, do které je tento útok zařazen. Tabulka obsahuje identifikaci skupiny a čas poslední aktivity v této skupině.

attacker_attack_group je tabulka spojující dvě předešlé tabulky a přiřazující tak jednotlivé útoky do skupin.

group_ban tato tabulka podobně jako již dříve zmíněné tabulky ban nebo subnet_ban slouží k definování banů, v tomto případě jde konkrétně o blokování skupin útočníků. Každý záznam tabulky obsahuje identifikaci skupiny a čas začátku a konce blokování této skupiny.

whitelist je tabulka, do které můžou být uživatelem zadány ip adresy nebo sítě, ze kterých mají být neplatné pokusy o přihlášení ignorovány. Každý záznam obsahuje adresu počítače nebo sítě a případně poznámku popisující proč je zrovna tato adresa v seznamu uvedena. Jelikož prostředky MySQL nepodporují příliš pohodlnou práci s maskami síťových adres, je možno sem zadávat pouze adresy sítí s maskou /16 a /24, což by ovšem mělo stačit pro většinu případů.

blacklist oproti předchozí tabulce obsahuje ip adresy nebo adresy sítí, které mají být zablokovány trvale. V této tabulce se již ovšem může vyskytovat jakákoliv síťová adresa, které je schopen porozumět používaný firewall. Nejlepší možností tedy bude zápis ve formátu A.B.C.D/X.

8.3 Vyhodnocování útoků

Vyhodnocování útoků je prováděno na straně serveru, jelikož zde jsou k tomuto nejlepší předpoklady, server má od svých klientů kompletní seznam všech neplatných pokusů o přihlášení a také veškeré již dříve zpracované výsledky, které si z těchto hodnot spočítal. Pro vyhodnocování útoků zde jsou celkem tři triggery, první z nich je prováděn po vložení záznamu do tabulky `fail`, další před vložením záznamu do tabulky `ban` a poslední z nich po zapsání záznamu do tabulky `ban`. Celkově je pomocí těchto tří triggerů možno provádění všech typů vyhodnocení útoků, které byly popsány v požadavcích na systém DBFAP v části 7.1.

Při neplatném pokusu o přihlášení je klientem proveden zápis do tabulky `fail` a tím je spuštěn první z uvedených triggerů, ten na základě různých testů dále může provést zápis do tabulky `ban`, při čemž jsou spouštěny další triggery nebo do tabulky `subnet_ban`. Triggery v tabulce `ban` dále můžou vkládat záznamy do tabulek `attacker`, `attack_group`, `attacker_attack_group` a `group_ban`.

8.3.1 Jednoduché útoky

Základní způsob vyhodnocení je prováděn na základě opakovaného chybného pokusu o přihlášení ze stejné ip adresy na kteréhokoliv klienta. Tento test je spuštěn vždy při zápisu do tabulky `fail` některým z klientů a jsou kontrolovány starší záznamy v této tabulce v intervalu určeném konfigurací, přitom jsou ignorovány všechny záznamy s ip adresami, které jsou uvedeny ve whitelistu.

8.3.2 Opakované útoky

V případě vyhodnocení útoku z ip adresy, ze které již byl v minulosti nějaký útok evidován, dochází k blokování daného útočníka na delší dobu, než v předchozích případech. Tato doba je určena pomocí hodnoty `reban_coefficient` (viz sekce 8.4 o konfiguraci systému).

8.3.3 Seznamy útočníků

Po každém vyhodnocení incidentu jako útok, je útočník uložen do tabulky `attacker` a přiřazen do skupiny útočníků v tabulce `attack_group`. Útočníci jsou do skupin přiřazováni na základě jednoduchého shlukování, kdy je stanovena hraniční hodnota pro čas, který uběhl mezi dvěma po sobě jdoucími útoky. V případě kratšího rozestupu je útočník přiřazen do aktuální skupiny, při překročení této hodnoty je vytvořena nová skupina a útočník zařazen do této skupiny.

V případě, že bude někdy v budoucnu zaznamenán útok z dostatečného počtu ip adres, které se všechny nacházejí ve stejné skupině útočníků, bude preventivně zablokována celá tato skupina. K zablokování dojde pomocí jiné tabulky, než té, která je určena k blokování jednotlivých útočníků, aby nedocházelo k nechtěnému vytváření nových skupin, které by se pouze nabalovaly na starší skupiny a snižovaly přesnost rozpoznání příštích distribuovaných útoků. Dalším důvodem pro použití nové tabulky je lepší přehled uživatele nad jednotlivými bany v databázi.

8.3.4 Subnety útočníků

Při analýze (viz kapitola 4) bylo odhaleno určité množství útoků, které byly vedeny z více počítačů umístěných ve shodné podsítí s maskou 255.255.255.0. Při neplatných pokusech o přihlášení jsou proto také kontrolovány podsítě s C maskou, ze kterých k těmto incidentům došlo a v případě překročení limitní hodnoty je preventivně zablokována celá tato síť. Pro toto blokování je opět použita jiná tabulka z důvodu přehlednosti a odlišení banů, které jsou skutečně "zasloužené", tedy vyvolané přímým útokem a těch, které mají pouze preventivní charakter.

8.3.5 Útočníci používající stejné uživatelské jméno

Nezanedbatelná část analyzovaných útoků sestávala z pokusů o přihlášení, které byly vedeny z velkého množství ip adres, ale používaly často stejné uživatelské jméno. V systému DBFAP je proto kontrolováno, zda se opakuje stejné přihlašovací jméno a po překročení hraniční hodnoty počtu pokusů ve stanoveném čase, je zablokována každá ip adresa, ze které je proveden neúspěšný pokus o přihlášení s daným uživatelským jménem.

Tento přístup sice pravděpodobně nebude zcela efektivní v případech, kdy je každá ip adresa použita v rámci jednoho útoku pouze jednou, ale poslouží především k vytvoření seznamu útočníků a jejich okamžitému zablokování při příštím útoku.

8.3.6 Kontrola vkládaných banů

Při každém vložení záznamu do tabulek `ban`, `group_ban` nebo `subnet_ban` je zapotřebí kontrolovat zda již takováto blokace v těchto tabulkách není zapsána, protože z důvodu stahování aktuálních seznamů k blokaci v pevných intervalech může dojít k dalšímu ne-

platnému přihlášení na stejném[2] nebo jiném klientovi ve chvíli, kdy již sice měla být daná adresa blokována, ale ještě nebyl seznam blokovaných adres aktualizován. Další vložení by sice nemělo žádný dopad na funkčnost systému, ale v případě použití v rozsáhlých sítích by při některých útocích mohlo docházet ke vzniku velkého množství duplicitních záznamů a opakovanému spouštění triggerů svázaných s těmito tabulkami.

8.4 Konfigurace

Systém je možno konfigurovat na dvou místech, prvním z nich je konfigurační soubor, umístěný na jednotlivých klientech, druhým je tabulka `config` v databázi na serveru.

V konfiguračním souboru klienta (ten je typicky umístěný v `/etc/dbfap/dbfap.cfg`) se nachází povinně konfigurace připojení k MySQL serveru a několik dalších položek.

- `mysql_override_settings` - určuje zda mají být položky, které je možno umisťovat jak do konfiguračního souboru, tak na server (viz následjící odstavec), stahovány ze serveru.
- `mysql_host, mysql_username, mysql_password, myslq_dbname, mysql_port` - názvy těchto položek poměrně jasně vystihují jejich význam, slouží ke konfiguraci připojení k databázovému serveru.
- `log_type` - určuje způsob logování z aplikace, možností je logování úplně zakázat, logovat do souboru, tisknout informace loggeru na standardní výstup nebo poslední dvě možnosti zkombinovat.
- `log_level` - stanovuje úroveň logování, zda budou zaznamenány jen kritické stavy nebo i veškeré ladící informace.
- `fw_command_init, ..._finish, ..._flush, ..._ban, ..._unban` - definuje příkazy pro práci s firewallem, ve výchozím konfiguračním souboru jsou tyto položky nastaveny pro všechny podporované firewally. Ip adresa počítače nebo sítě bude do konfigurovaného příkazu vložena místo symbolu `$DBFAP_ADDR`.

  Konfigurace firewallu tímto způsobem byla zvolena především pro ponechání veškeré moci nad správou firewallu uživateli namísto "zadrátování" několika základních příkazů do kódu aplikace, kdy by již nebylo bez úprav zdrojových kódů a rekompilace možno s tímto cokoliv udělat.

  Pro nezkušeného uživatele by se tento přístup mohl jevit jako příliš náročný na konfiguraci, ale ve výchozím konfiguračním souboru je vše nakonfigurováno tak, aby nemusel nic měnit, případně jen odkomentoval pár řádků pro použití jiného typu firewallu. Vzhledem k pojetí aplikace a určení pro celé počítačové sítě ale bude pravděpodobně cílovým uživatelem aplikace administrátor sítě a ten jistě použitý způsob konfigurace ocení.
- `fw_command_init6, ..._finish6, ..._flush6, ..._ban6, ..._unban6` - podobně jako předešlá skupina nastavuje příkazy pro blokování útočníků, zde jde ale o ipv6 adresy. V případě, že tyto příkazy nejsou nastaveny, budou použity stejné příkazy jako pro ipv4.

[2] v případě, že není povolena funkce `single_fail_ban`, která provádí okamžitou blokaci na klientovi, na kterém došlo k neplatnému přihlášení.

Hodnotami, které mohou být konfigurovány na obou místech, tedy na serveru i klientovi, jsou například regulární výrazy pro rozpoznávání neplatných přihlášení nebo interval pro dotazování.

- `pattern` - regulární výraz sloužící k rozpoznání neplatného pokusu o přihlášení v systémovém logu.
- `pattern2` - stejně jako předešlá položka slouží k rozpoznání neplatného pokusu v logu, ale podřetězce, které jsou při vyhodnocení výrazu vráceny (uživatelské jméno a adresa útočníka) mají obrácené pořadí.
- `mysql_polling_interval` - interval ve kterém bude docházet k dotazování z klientů na seznamy aktuálně blokovaných ip adres.
- `single_fail_ban` - určuje zda má být po každém neúspěšném pokusu o přihlášení zablokována zdrojová ip adresa okamžitě. Dojde k tomu pouze na klientovi, na kterém k neúspěšnému přihlášení došlo a na náhodnou dobu, která je maximálně rovna nastavení `mysql_polling_interval`.

  Povolení této možnosti povede na systémech používajících pro službu ssh autentizační metodu `keyboard-interactive` (viz autentizační metody ssh v části 2.3) v případě nastavení jedné z hodnot `pattern` na `[Ii]nvalid user (.*) from (.*)`, k chování, kdy při přihlášení s neexistujícím uživatelským jménem dojde k zablokování dříve, než bude možno zkusit zadat heslo. Což může, ale nemusí být žádoucí chování.

Ukázkový konfigurační soubor je možno nalézt v přílohách.

Konfigurační hodnoty týkající se vyhodnocování útoků jsou umístěny pouze v databázi, protože k vyhodnocování dochází pomocí databázových triggerů a na klientech by tedy neměly smysl.

Hodnotami používanými při vyhodnocování jsou tyto:

- `findtime` - určuje v jak dlouhém časovém intervalu budou hledány výskyty neúspěšných přihlášení potenciálního útočníka při vyhodnocování útoku.
- `maxretry` - stanovuje kolikrát se musí při vyhodnocení útoku ip adresa útočníka vyskytnout v prohledávaném intervalu, aby byly tyto pokusy o přihlášení vyhodnoceny jako útok.
- `bantime` - nastavuje na jak dlouho má být zablokována útočníkova ip adresa při prvním vyhodnocení této adresy jako útočné.
- `reban_maxretry` - určuje kolik neúspěšných pokusů o přihlášení způsobí zablokování útočníka, který již někdy v minlosti blokován byl.
- `reban_coefficient` - určuje jak má být prodloužena doba `bantime` pro útočníka, který již byl dříve blokován kvůli pozitivně vyhodnocenému útoku. Doba `bantime` je zde násobena hodnotou `reban_coefficient` pro každý dříve nalezený výskyt útočníka v historii. V případě použití `reban_coefficient` 2 a hodnotě `bantime` 1h tak bude druhý pokus postihnut blokací na 2h, třetí pokus bude zablokován na 4h atd.
- `attack_group_threshold` - je hodnota určující interval, po jehož uplynutí se budou zdrojové adresy nově vyhodnocených útoků přiřazovat do nové skupiny útočníků, v případě že v průběhu tohoto intervalu žádný nový útok nebyl vyhodnocen.

- `attack_group_min_hosts` - určuje kolik útočníků příslušejících do některé z dříve zaznamenaných skupin útočníků musí být znovu rozpoznáno jako útočníci, aby došlo k zablokování celé této skupiny.
- `attack_group_bantime` - určuje časový interval po který dojde k zablokování celé skupiny útočníků.
- `username_maxretry` - určuje počet neplatných přihlášení se stejným uživatelským jménem za dobu `findtime`, který je třeba provést, aby došlo k zablokování všech zúčastněných ip adres, ze kterých bylo útočeno.
- `username_bantime` - nastavuje čas, po který budou blokováni útočníci, kteří byli odhaleni na základě použití stejného uživatelského jména.
- `subnet_maxretry` - určuje počet útoků ze stejného C subnetu, po kterém dojde k zablokování celého tohoto subnetu.
- `subnet_bantime` - nastavuje čas, po který bude blokován celý subnet.

8.4.1 Výchozí hodnoty konfigurace

Jako výchozí hodnoty pro jednotlivé položky v konfiguraci byly zvoleny hodnoty odvozené na základě statistik reálných útoků, jejichž analýza je rozebírána v kapitole 4 a také na základě testování, které je popsáno v kapitole 9.1 za účelem odražení co nejvyššího množství útoků a zároveň udržení rozumného výkonu aplikace.

Pro nastavení základní délky blokace útočící ip adresy se ukázalo vhodné použití hodnot mezi 30-ti a 60-ti minutami s prodlužováním na dvojnásobek předešlé délky v případě, že útočník svůj útok opakuje. Jako výchozí nastavení hodnoty `bantime` bylo tedy stanoveno 3600 sekund a pro `reban_coefficient` hodnota 2.

Vhodným časovým intervalem pro vyhledávání souvisejících neúspěšných pokusu o přihlášení se ukázalo předešlých 30 minut před posledním pokusem, ve většině pozorovaných případů se další pokus odehraje do tří minut po předešlém a intervaly nad půl hodiny se téměř nevyskytují a delší nastavení by již mohlo nepříjemně omezit oprávněné uživatele, kteří se opakovaně přepsali v hesle. Výchozí hodnota pro nastavení `findtime` byla tedy stanovena na 1800 sekund.

Hodnota `maxretry`, která stanovuje počet přihlášení, po kterém již bude útočník zablokován byla ve výchozí konfiguraci nastavena na 3, což pokládám za rozumný počet pokusů pro zadání hesla oprávněným uživatelem tak, aby nebyl zablokován. Analýza logů ze serveru kazi ostatně ukazuje, že bezprostředně po sobě se téměř neopakovalo neúspěšné přihlášení jednoho konkrétního oprávněného uživatele více, než dvakrát.

Při blokování célé sítě útočníků bylo jako vhodné výchozí nastavení pro minimální počet adres z dané sítě, hodnotu `subnet_maxretry` stanoveno číslo 4 a základní doba pro zablokování takto detekované sítě, `subnet_bantime` je 1800 sekund. Zde je nutno poznamenat, že rozsáhlé sítě, ze kterých je předpokládáno časté přihlašování oprávněných uživatelů, by měly být uvedeny ve whitelistu, který umožňuje zadání jednotlivých ip adres nebo adres sítí.

Při blokování na základě použití stejného uživatelského jména ve více neúspěšných pokusech v sérii pocházejících z různých ip adres, byly jako vhodné zvoleny výchozí hodnoty stejné jako v předešlém odstavci, tedy 4 pro `username_maxretry` a 1800 sekund pro položku `username_bantime`.

Výchozí hodnoty pro zpracování útoků skupin byly laděny nejdéle ze všech konfiguračních hodnot a nakonec byly stanoveny na 125 sekund jako hraniční hodnota oddělující dvě skupiny, `attack_group_threshold`. Zde bylo nutno volit kompromis mezi rizikem, že bude velká skupina zbytečně rozdělena na více menších a rizikem, že do skupiny útočníků bude přimíchána ip adresa, která do ní nepatří, čemuž se ovšem zcela vyhnout nelze nikdy. Výchozí čas pro zablokování skupiny, `attack_group_bantime` je 1200 sekund. Poslední z hodnot týkajících se blokování skupin je `attack_group_min_hosts`, určující mezní hodnotu pro zablokování nalezené skupiny, číslo 7 se může zdát poměrně vysoké, ale testování ukázalo, že nižší čísla generují příliš mnoho samostatných banů, což bylo kontraproduktivní a úspěšnost zablokování dalších útoků to nezvedalo dostatečně v porovnání se snížením výkonu vyhodnocování, způsobeném právě tímto množstvím banů.

Interval pro dotazování se klientů na aktuální seznam banů a obsah blacklistu, hodnota `mysql_polling_interval` je s ohledem na použití funkce `single_fail_ban` (ve výchozí konfiguraci aktivní) a přijatelnou dobu reakce celého systému v kompromisu se snahou nezatěžovat příliš databázový server, na 10 sekund.

Kapitola 9

Testování a provoz

Pro provoz aplikace je potřeba splnění následujících předpokladů:

1. na klientech je zapotřebí systémový logger, umožňující zápis logů na standardní vstup aplikace DBFAP, podporován je například `syslog-ng`, ale je možno použít také jakýkoliv jiný logger ve spojení s nástrojem tail, což je také postup, který je díky své univerzálnosti a menším nárokům na ručně prováděnou konfiguraci použit ve výchozím připraveném intiscriptu (ten předpokládá ukládání logu ssh démona do `/var/log/sshd/sshd.log`).

   Pro nastavení logování na standardní vstup programu pomocí `syslog-ng` je třeba do konfiguračního souboru `/etc/syslog-ng/syslog-ng.conf` přidat následující zápis:

   ```
   destination dbfap_app {
       program("/usr/sbin/dbfap /etc/dbfap/dbfap.cfg"); };
   filter f_dbfap { facility(auth, authpriv);  };
   log { source(src); filter(f_dbfap); destination(dbfap_app); };
   ```

   Klíčové slovo `destination` na prvním řádku vytváří cíl s názvem `dbfap_app`, na dalším řádku je vytvořen filtr `f_dbfap`, který vybírá pouze záznamy týkající se přihlašování k systému a poslední řádek říká, že má být logováno vše z výchozího zdroje `src`, co projde filtrem `f_dbfap` do cíle `dbfap_app`, což je naše aplikace. Více informací o konfiguraci loggeru `syslog-ng` je možno získat v manuálu[1].

2. na klientech musí být některý z podporovaných nástrojů pro blokování komunikace:

   - iptables
   - ipfw
   - tcp wrapper (hosts.deny)

   Je ale možno uživatelsky nakonfigurovat jakýkoliv jiný způsob blokace, pokud je tuto blokaci možno provádět shellovými příkazy. Pro tento případ je možno nakonfigurovat příkazy pro inicializaci nástroje, přidání a odebrání blokované adresy, odstranění všech blokovaných adres a ukončení nástroje pro blokaci (viz část 8.4 o konfiguraci systému).

   Další možností je také použití prázdných příkazů pokud klient nemá provádět žádné blokování, ale pouze předávat informace o bezpečnostních incidentech na server. Toto

může být užitečné v případě, že je celý systém nakonfigurován příliš tvrdě a jsou takto na klientech firewallem odfiltrovány takřka všechny prováděné pokusy o přihlášení. V tomto případě by se pak do databáze nedostalo moc nových informací o probíhajících útocích.

3. nezbytné je mít v systému k dispozici klientské MySQL knihovny pro klientskou aplikaci a MySQL server na straně serveru. Jelikož je vyhodnocování realizováno na serveru pomocí databázových triggerů a MySQL podporuje triggery od verze 5.0, je vyžadována verze 5.0 nebo novější.

   Na serveru je třeba vytvořit databázi včetně všech triggerů a vytvořit uživatelský účet, který budou využívat klienti pro přístup k této databázi. Toto je možno povést pomocí připraveného SQL skriptu, který vytvoří databázi nazvanou `dbfap`, uživatelský účet `dbfap` s heslem `dbfap`. Je silně doporučeno heslo k tomuto účtu následně změnit.

9.1 Testování aplikace DBFAP

Testování dílčích částí aplikace bylo prováděno v průběhu celého vývoje a probíhalo na podobné sestavě serverů, jako původní analýza. Většina těchto serverů běžela na Linuxové distribuci Gentoo s jádrem 2.6.31-r10 a ssh verze OpenSSH_5.3p1, dále byl používán virtuální server s FreeBSD 8.0 a verzí ssh OpenSSH_5.2p1.

Ve zbytku této kapitoly bude demonstrováno několik testů hotového systému, prováděných na systémech s Gentoo linuxem, zaměřených na ukázku většiny podstatných funkcí implementovaných v aplikaci.

9.1.1 Test základního a opakovaného zablokování útočníka

Nejzákladnějším testem, jehož provedení bylo naprostou samozřejmostí, je základní zablokování útočníka, který provedl stanovený počet neplatných přihlášení ve stanoveném čase. Stejný útočník poté byl použit opakovaně a sledováno prodloužení doby pro jeho zablokování.

Nejprve se podíváme na zkrácený záznam z logu dvou serverů s názvy mgmb a otesanek, na kterých běží DBFAP klienti.

```
12:42:59 mgmb sshd[16848]: pam_unix(sshd:auth): authentication failure ...
12:43:01 mgmb sshd[16837]: error: PAM: Authentication failure for illegal ...
12:43:01 mgmb sshd[16837]: Failed keyboard-interactive/pam for invalid ...
12:43:01 mgmb dbfap[16695]: fw ban: 147.229.176.19
12:43:08 mgmb dbfap[16695]: fw unban: 147.229.176.19
12:43:19 mgmb sshd[16848]: pam_unix(sshd:auth): authentication failure ...
12:43:22 mgmb sshd[16837]: error: PAM: Authentication failure for illegal ...
12:43:22 mgmb sshd[16837]: Failed keyboard-interactive/pam for invalid ...
12:43:22 mgmb dbfap[16695]: fw ban: 147.229.176.19
12:43:28 mgmb dbfap[16695]: fw unban: 147.229.176.19
12:44:13 otesanek sshd[31985]: Invalid user xricht14 from 147.229.176.19
12:44:13 otesanek dbfap[11696]: fw ban: 147.229.176.19
12:44:18 mgmb dbfap[16695]: fw ban: 147.229.176.19
```

Z logu je možno vyčíst, že byly provedeny celkem tři chybné pokusy o přihlášení k našemu systému ze serveru merlin. První dva pokusy byly provedeny na server mgmb,

který vždy bezprostředně po identifikaci neplatného přihlášení provedl zablokování zdrojové ip adresy po zbytek intervalu `polling_interval` a zapsal o tomto informaci na centrální server. Třetí pokus byl proveden na server otesanek, který se zachoval stejně jako mgmb a zablokoval útočníkovi přístup. Následně přístup zablokoval i server mgmb, protože v databázi vznikl záznam o globálním zablokování ip adresy.

V následujícím výpisu z tabulky `fail` na centrálním serveru (kterým je v tomto případě server otesanek) jsou vidět záznamy, které byly v tabulce vytvořeny na základě událostí popsaných před chvílí.

```
+---------+---------------+---------------------+----------+--------------+
| id_fail | ipv4          | date                | username | client_addr  |
+---------+---------------+---------------------+----------+--------------+
|      12 | 147.229.176.19 | 2010-05-13 12:42:59 | xricht14 | 192.168.10.2 |
|      13 | 147.229.176.19 | 2010-05-13 12:43:19 | xricht14 | 192.168.10.2 |
|      14 | 147.229.176.19 | 2010-05-13 12:44:13 | xricht14 | localhost    |
+---------+---------------+---------------------+----------+--------------+
```

Při porovnání tabulky a předešlého výpisu z logu je možno vypozorovat, že systém správně identifikuje, že šlo vždy pouze o jeden pokus o přihlášení přesto, že o něm jsou na serveru mgmb proveden tři záznamy.

Dále následuje výpis z tabulky `ban`, kde se nachází záznam o zablokování útočné ip adresy, který zde byl vytvořen triggerem na základě dosažení limitního počtu (3) chybných přihlášení z jedné ip adresy ve stanoveném časovém intervalu (10 minut), zaznamenaných v tabulce `fail`.

```
+--------+---------------+---------------------+---------------------+
| id_ban | ipv4          | fromdate            | todate              |
+--------+---------------+---------------------+---------------------+
|      3 | 147.229.176.19 | 2010-05-13 12:44:13 | 2010-05-13 13:44:13 |
+--------+---------------+---------------------+---------------------+
```

Po uplynutí času pro zablokování byl postup zopakován a provedena kontrola nově vzniklého času v tabulce `ban`. Ve výpisu z této tabulky je vidět, že opakovaný ban pro ip adresu serveru merlin s `id_ban` 11 má nastavenu dvojnásobnou dobu trvání, než první ban s `id_ban` 3.

```
+--------+----------------+---------------------+---------------------+
| id_ban | ipv4           | fromdate            | todate              |
+--------+----------------+---------------------+---------------------+
|      3 | 147.229.176.19  | 2010-05-13 12:44:13 | 2010-05-13 13:44:13 |
|     11 | 147.229.176.19  | 2010-05-13 14:07:51 | 2010-05-13 16:07:51 |
+--------+----------------+---------------------+---------------------+
```

9.1.2 Test zablokování podsítě

Dalším provedeným testem je test na blokování podsítě, ze které v krátké době zaútočí dostatečný počet počítačů. Pro test bylo použito několik počítačů v CVT a v následujícím zkráceném výpisu z tabulky `fail` je vidět jak všechny tyto pokusy o přihlášení byly zaznamenány.

```
+---------+-----------------+---------------------+----------+
| id_fail | ipv4            | date                | username |
+---------+-----------------+---------------------+----------+
|      16 | 147.229.176.19  | 2010-05-13 13:20:01 | xricht14 |
|      17 | 147.229.176.14  | 2010-05-13 13:25:09 | xricht14 |
|      18 | 147.229.177.196 | 2010-05-13 13:29:08 | xricht14 |
|      19 | 147.229.177.197 | 2010-05-13 13:30:13 | root     |
|      20 | 147.229.177.198 | 2010-05-13 13:31:17 | root     |
|      21 | 147.229.176.157 | 2010-05-13 13:32:38 | xricht14 |
|      22 | 147.229.177.198 | 2010-05-13 13:32:47 | root     |
|      23 | 147.229.176.42  | 2010-05-13 13:34:41 | xricht14 |
|      24 | 147.229.176.190 | 2010-05-13 13:35:33 | xricht14 |
|      25 | 147.229.176.158 | 2010-05-13 13:36:58 | xricht14 |
+---------+-----------------+---------------------+----------+
```

Záznamy s id 21, 23, 24 a 25 jsou všechny ze sítě `147.229.176.0/24` a jelikož jejich počet v intervalu 600 sekund dosáhl limitního počtu 4, byl přidán záznam do tabulky `subnet_ban`, jak je vidět v následujícím výpisu z této tabulky.

```
+-----------+-----------------+---------------------+---------------------+
| id_sn_ban | subnet          | fromdate            | todate              |
+-----------+-----------------+---------------------+---------------------+
|         1 | 147.229.176.0/24 | 2010-05-13 13:36:58 | 2010-05-13 14:06:58 |
+-----------+-----------------+---------------------+---------------------+
```

9.1.3 Test skupiny počítačů

V následujícím testu bude demonstrováno zablokování skupiny útočníků. Aby bylo možno lépe toto demonstrovat v podmínkách, ve kterých tento test byl prováděn, byla při tomto testu vyřazena funkce blokování podsítě.

Následující výpis ukazuje vybranou skupinu útočníků, kteří byli zaznamenáni v tabulce `ban`.

```
+--------+-----------------+---------------------+---------------------+
| id_ban | ipv4            | fromdate            | todate              |
+--------+-----------------+---------------------+---------------------+
|     12 | 147.229.176.14  | 2010-05-13 15:01:06 | 2010-05-13 17:01:06 |
|     13 | 147.229.176.156 | 2010-05-13 15:01:37 | 2010-05-13 15:31:37 |
|     14 | 147.229.176.157 | 2010-05-13 15:01:53 | 2010-05-13 17:01:53 |
|     15 | 147.229.176.158 | 2010-05-13 15:02:41 | 2010-05-13 17:02:41 |
|     16 | 147.229.177.196 | 2010-05-13 15:03:21 | 2010-05-13 17:03:21 |
|     17 | 147.229.177.198 | 2010-05-13 15:03:34 | 2010-05-13 15:33:34 |
|     18 | 147.229.177.200 | 2010-05-13 15:03:57 | 2010-05-13 15:33:57 |
|     19 | 147.229.177.201 | 2010-05-13 15:04:07 | 2010-05-13 15:34:07 |
|     20 | 147.229.177.202 | 2010-05-13 15:04:55 | 2010-05-13 15:34:55 |
|     21 | 147.229.176.118 | 2010-05-13 15:06:19 | 2010-05-13 15:36:19 |
|     22 | 147.229.176.129 | 2010-05-13 15:06:46 | 2010-05-13 15:36:46 |
+--------+-----------------+---------------------+---------------------+
```

Mezi žádnými po sobě jdoucími útoky není doba delší, než limitních 125 sekund, a proto jsou všechny tyto útoky zařazeny do stejné skupiny, což je možno ověřit spojením tabulek `attacker`, `attack_group` a `attacker_attack_group`.

Po uplynutí času, po který byly zablokováni jednotliví útočníci, byl proveden útok pouze ze tří z nich, jak je vidět ve výpisu z tabulky `ban`.

```
+--------+-----------------+---------------------+---------------------+
| id_ban | ipv4            | fromdate            | todate              |
+--------+-----------------+---------------------+---------------------+
|     23 | 147.229.176.118 | 2010-05-13 16:26:55 | 2010-05-13 18:26:55 |
|     24 | 147.229.177.198 | 2010-05-13 16:28:52 | 2010-05-13 18:28:52 |
|     25 | 147.229.177.196 | 2010-05-13 16:29:42 | 2010-05-13 20:31:33 |
+--------+-----------------+---------------------+---------------------+
```

Důsledkem toho bylo vytvoření záznamu v tabulce `group_ban`, který způsobí zablokování přístupu všem útočníků ze seznamu s id = 8, což odpovídá všem útočníků popsaným v prvním výpisu v této části.

```
+-----------+-------------+---------------------+---------------------+
| id_gr_ban | id_att_group | fromdate            | todate              |
+-----------+-------------+---------------------+---------------------+
|         1 |            8 | 2010-05-13 16:29:42 | 2010-05-13 16:59:42 |
+-----------+-------------+---------------------+---------------------+
```

9.1.4 Test s daty ze serveru kazi

Na závěr jsem pro testování vytvořil prostředí, ve kterém běží aplikace DBFAP a skript, který čte systémový log s dříve analyzovanými záznamy ze serveru kazi postupně nastavuje systémový čas na čas jednotlivých záznamů a tyto záznamy potom předává aplikaci DBFAP ke zpracování. Snažil jsem se tak tímto systémem odsimulovat celé téměř dva roky běhu aplikace s reálnými útoky a nepodařenými přihlášeními, které byly provedeny na server kazi v období od 1. 1. 2008 do 4. 10. 2009.

V tomto testu byl zjišťován především podíl různých strategií pro blokování útoků na reálném provozu a také se jednalo o zátěžový test, který odhalil různá slabá místa v použitých triggerech a pomohl s jejích optimalizací.

Použitá data byla zkoušená na výchozí konfiguraci systému DBFAP a také na upravené verzi, která neobsahovala žádně z pokročilejších metod vyhodnocování a podobala se tedy dříve popisovaným nástrojům *fail2ban* nebo *sshguard*. Ani jedna z těchto dvou testovaných verzí nepoužívala okamžité zablokování ip adresy na klientovi, které analyzované nástroje také nepoužívají a ve vytvořeném prostředí nebylo možné rozumně simulovat.

Obě verze aplikace zaznamenaly celkem 195 323 nezdařilých pokusů o přihlášení.

Základní verze z těchto pokusů o přihlášení postupně vygenerovala 39 272 jednohodinových blokací provozu z útočných ip adres, kterými odrazila 115 060 následujících pokusů.

Rozšířená verze, tedy aplikace DBFAP ve výchozím nastavení z uvedeného množství neplatných přihlášení vytvořila celkem 14 571 pravidel pro blokaci a zablokovala jimi 163 595 dalších pokusů. Kromě toho vzniklo 89 pravidel pro blokaci celých sítí s maskou 255.255.255.0, které pomohly k zablokování 153 dalších útoků a 11 708 dílčích

zákazů celých dříve identifikovaných skupin, které zablokovaly celkem 9 519 dalších pokusů o přihlášení. Dohromady tedy bylo zabráněno 173 271 pokusům o přihlášení, které byly součástí nějakého z útoků.

Tabulka 9.1 shrnuje tyto výsledky.

způsob blokace	**počet útoků**
ip adresa	163 595
skupina	9 519
podsíť	153
celkem	173 271

Tabulka 9.1: Počty zablokovaných útoků pomocí jednotlivých přístupů

Po srovnání obou částí testu tedy vychází, že aplikace DBFAP zvládla v datech z reálného provozu odrazit o 50% útoků více, než klasické postupy používané v současných host-based IPS nástrojích. V případě použití funkce `single_fail_ban` by mohlo dojít k odfiltrování dalších až dvou třetin pokusů o přihlášení, které v testu odfiltrovány nebyly.

Kapitola 10

Závěr

V diplomovém projektu jsem se zabýval analýzou distribuovaných útoků hrubou silou na službu ssh, nástroji pro detekci a ochranu před těmito útoky a implementaci takového nástroje za použití relační databáze MySQL. Přesto že se práce soustředí primárně na službu ssh, popisuje obecný přístup pro ochranu jakékoliv služby před útoky na hesla hrubou silou a vytvořený nástroj je s patřičně upravenou konfigurací rovněž použitelný k ochraně jakékoliv služby.

První část práce se zabývá zpracováním logů ze serverů, které byly vystavovány reálným útokům a analýzou těchto logů. Dále jsou zde rozebrány výsledky analýzy, popis zaznamenaných útoků a popis způsobů, kterými je možné se těmto útokům bránit.

V další části práce jsem analyzoval v současné době používané automatizované nástroje, které se snaží s útoky hrubou silou vypořádat a techniky, které k tomu používají. Jde tedy pouze o jednu konkrétní možnost řešení popsaných v předešlé části, z pohledu správce sítě a uživatelů jde ovšem pravděpodobně o nejschůdnější možnost, protože se vyhýbá různým nepopulárním omezením uživatele a přináší relativně efektivní řešení. V této části jsem také analyzoval typické metody vyhodnocování útoků, používané v těchto nástrojích a hledal jejich slabá místa a způsoby jak se těchto slabin v zbavit.

Na základě informací získaných v prvních dvou částech jsem v následující části práce navrhl nové postupy pro řešení problému distribuovaných útoků hrubou silou a snažil se je postupně implementovat. Vznikla tak distribuovaná aplikace DBFAP, která implementuje velkou část požadavků, které jsem si v předešlé části práce stanovil. Systém běží distribuovaně na centrálním MySQL serveru a libovolném množství připojených klientů, rozhodování o útocích je přeneseno z klientů na centrální server a je takto možno využít mnohem více z potenciálu, který data nasbíraná na klientech nabízejí. Je zde řešeno přiřazování útočníků do skupin a blokace celých opakujících se skupin útočníků, blokování celé sítě útočníků nebo třeba spojování útočníků na základě použití stejného přihlašovacího jména. Oproti původním plánům jsem při implementaci ustoupil od použití nástroje `p0f` pro pasivní získávání TCP fingerprintu pro příchozí spojení, protože se ukázalo, že poměr cena/užitek je příliš nízký (především s ohledem na výkon vyhodnocování pomocí databázových triggerů). Ze stejného důvodu pak nebyly použity ani funkce počítající podobnost neexistujícího loginu s některým z existujících, které se ukázalo jako kontraproduktivní.

Nezbytnou součástí celé práce bylo také testování, kterému jsem se věnoval průběžně po dokončení každé části systému a také ve větší míře na v závěru projektu.

Systém DBFAP je otestovaný a funkční, přesto je zde ale spousta možností dalšího zdokonalování. Aplikace ve výchozí konfiguraci by měla být vhodná k běžnému použití, zkušenému administrátorovi pak jistě nebude dělat problémy si občas upravit některou z

hodnot v konfiguračním souboru nebo v databázi k obrazu svému, ale přesto by bylo jistě vhodné do budoucna k aplikaci vytvořit nějaké pohodlné administrační rozhraní, které by například mohlo poskytovat další možnosti dolování informací z nasbíraných dat a vizualizaci útoků. Mezi další možnosti zdokonalování aplikace by mohlo patřit hlídání počítadel použitého firewallu (pokud daný firewall počítadla poskytuje) a v případě, že narůstají podle nich adekvátně prodlužovat čas blokování příslušných ip adres.

Na straně serveru je stále obrovský prostor pro vytváření nových algoritmů pro vyhodnocování útoků a bylo by možné i zde systém dále zdokonalovat, mezi prvními kandidáty by mohl být například nějaký pokročilejší shlukovací algoritmus pro přiřazování útočníků do skupin a algoritmus, který by průběžně promazával skupiny, které na základě jeho rozhodnutí nemá význam v databázi udržovat, což by mohly být například skupiny obsahující méně útočníků, než je spodní limit pro zablokování skupiny nebo skupiny obsahující pouze útočníky, kteří se nacházejí také v jiných skupinách.

Literatura

[1] BalaBit IT Security Ltd.: The syslog-ng Administrator Guide. [online]navštíveno 10.5.2010.
URL http://www.balabit.com/dl/guides/syslog-ng-v2.0-guide-admin-en.pdf

[2] Barrett, D. J.; Silverman, R. E.: *SSH, The Secure Shell: The Definitive Guide, Second Edition*. O'Reilly Media, 2005, iSBN 978-0-596-00895-6.

[3] Brown, M. A.: Guide to IP Layer Network Administration with Linux. [online]navštíveno 4.1.2010.
URL http://linux-ip.net/html/linux-ip.html#ex-list-route-blackhole

[4] community, F.: Fail2ban:Community Portal.
URL http://fail2ban.org

[5] Csaba, B.: Vše o iptables. [online]navštíveno 4.1.2010.
URL http://www.root.cz/serialy/vse-o-iptables/

[6] Cusack, F.; Forssen, M.: Generic Message Exchange Authentication for the Secure Shell Protocol (SSH). 2006.
URL http://www.ietf.org/rfc/rfc4256.txt

[7] Gu, G.; Zhang, J.; Lee, W.: BotSniffer: Detecting Botnet Command and Control Chanels in Network Traffic.

[8] Kerslager, M.: TCP wrapper. [online]navštíveno 4.1.2010.
URL http://www.pslib.cz/ke/TCP_wrapper

[9] Krčmář, P.: Sbíráme otisky: pasivní p0f. [online] navštíveno 23.12.2009.
URL http://www.root.cz/clanky/sbirame-otisky-pasivni-p0f/

[10] Ledvina, J.: SSH. [online]navštíveno 15.5.2010.
URL http://www.kiv.zcu.cz/~ledvina/vyuka/PDS/PDS-tut/ssh/ssh.doc

[11] Maggi, F.: Sshguard.
URL http://www.sshguard.net

[12] Malecot, E.; Hori, Y.; Sakurai, K.; aj.: (Visually) Tracking Distributed SSH Brute Force Attacks? 2008.

[13] Namestnikov, Y.: The economics of botnets. 2009, [online]navštíveno 10.12.2009.
URL http://www.viruslist.com/en/downloads/pdf/ynam_botnets_0907_en.pdf

[14] Nazario, J.: Twitter-based Botnet Command Channel.
http://asert.arbornetworks.com/2009/08/twitter-based-botnet-command-channel,
[online] navštíveno 23.12.2009.

[15] Owens, J.; Matthews, J.: A Study of Passwords and Methods Used in Brute-Force SSH Attacks.

[16] Red Hat, Inc.: Event Sequence of an SSH Connection. [online]navštíveno 14.5.2010.
URL http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/ref-guide/s1-ssh-conn.html

[17] Schwartz, P.: DenyHosts home page.
URL http://denyhosts.sourceforge.net/index.html

[18] The FreeBSD Documentation Project: IPFW. [online]navštíveno 4.1.2010.
URL http://www.freebsd.org/doc/en/books/handbook/firewalls-ipfw.html

[19] Ylonen, T.; Lonvick, C.: The Secure Shell (SSH) Authentication Protocol. 2006.
URL http://www.ietf.org/rfc/rfc4252.txt

[20] Zalewski, M.: the new p0f. [online] navštíveno 23.12.2009.
URL http://lcamtuf.coredump.cx/p0f.shtml

Příloha A

Obsah CD

Přiložené CD obsahuje zdrojové kódy aplikace DBFAP, konfigurační soubor a skript pro vytvoření databáze. Dále jsou zde instrukce pro instalaci aplikace a ebuild pro snadnou instalaci na linuxovou distribuci Gentoo.

Příloha B

Konfigrační soubor

Vzorový výchozí konfigurační soubor klienta.

```
##
# Distributed Brute Force Attacks Protector
#
# sample configuration file
# Jan Richter, 2010
##

[general]
# sets whether should be configuration fetched from mysql server
mysql_override_settings = 1

# log_type: 0 none, 1 print, 2 log, 3 log + print
log_type = 2
# log_level: 0=emergency .. alert, crit, err, warn, notice, info, .. 7=debug
log_level = 6

[mysql]
# connection to mysql server, most important thing here
mysql_host =  localhost
mysql_username =  dbfap
mysql_password =  dbfap
mysql_dbname =  dbfap
mysql_dbport =  3306
mysql_polling_interfal = 10

[parser]
# defines regexp patterns for log parser, can be overriden from mysql server
# NOTE: some log entries differ in order of username and hostname
# if there is username on first place, use "pattern"
# otherwise use "pattern2" if there is hostname first
pattern = "[Aa]uthentication failure for invalid user (.*) from (.*)"
pattern = "[Aa]uthentication failure for illegal user (.*) from (.*)"
pattern = "[Aa]uthentication failure for (.*) from (.*)"
```

```
pattern = "[Aa]uthentication error for (.*) from (.*)"
pattern = "[Ii]nvalid user (.*) from ([a-zA-Z0-9.-]*)"
pattern = "[Ii]llegal user (.*) from ([a-zA-Z0-9.-]*)"
pattern = "[Uu]ser not known to the underlying authentication module for
    (.*) from (.*)"
pattern = "[Ff]ailed keyboard-interactive/pam for invalid user (.*)
    from (.*) port"
pattern2 = "[Aa]uthentication failure; .* rhost=(.*)  user=(.*)"


[firewall]
# firewall configuration, can be overriden from mysql server
single_fail_ban = 1

#####################
# firewall commands
#####################
#
# here should be specified which firewall commands will be used by DBFAP
#

#
[iptables firewall]
#==================
#
# init command
# this one will block ssh from banned hosts
fw_command_init = /sbin/iptables -N DBFAP; iptables -A DBFAP -j RETURN;
    iptables -I INPUT -p tcp --dport ssh -j DBFAP
fw_command_init6 = /sbin/ip6tables -N DBFAP; iptables -A DBFAP -j RETURN;
    iptables -I INPUT -p tcp --dport ssh -j DBFAP

# alternatively to block all communication attempts from banned hosts,
# NOTE: you should change finish command as well
#fw_command_init = /sbin/iptables -N DBFAP; iptables -A DBFAP -j RETURN;
    iptables -I INPUT -j DBFAP

# finish command (clean everything what we added to firewall)
fw_command_finish = /sbin/iptables -D INPUT -p tcp --dport ssh -j DBFAP;
    iptables -F DBFAP; iptables -X DBFAP
fw_command_finish6 = /sbin/ip6tables -D INPUT -p tcp --dport ssh -j DBFAP;
    ip6tables -F DBFAP; ip6tables -X DBFAP

#fw_command_finish = /sbin/iptables -D INPUT -p tcp -j DBFAP;
    iptables -F DBFAP; iptables -X DBFAP

# flush command
fw_command_flush = /sbin/iptables -F DBFAP
```

```
fw_command_flush6 = /sbin/ip6tables -F DBFAP

# ban command
fw_command_ban = /sbin/iptables -I DBFAP 1 -s $DBFAP_ADDR -j DROP
fw_command_ban6 = /sbin/ip6tables -I DBFAP 1 -s $DBFAP_ADDR -j DROP

# unban command
fw_command_unban = /sbin/iptables -D DBFAP -s $DBFAP_ADDR -j DROP
fw_command_unban6 = /sbin/ip6tables -D DBFAP -s $DBFAP_ADDR -j DROP

#
[ipfw firewall]
#==================
#
#fw_command_init =
#fw_command_finish =
#fw_command_flush =
#fw_command_ban = ipfw add deny tcp from $DBFAP_ADDR to me 22
#fw_command_unban = ipfw delete
    `ipfw list | grep -i $DBFAP_ADDR | awk '{print $1;}'`


#
[tcp wrappers]
#==================
#
#fw_command_init =
#fw_command_finish =
#fw_command_flush =
#fw_command_ban = printf %b "ALL: $DBFAP_ADDR\n" >> /etc/hosts.deny
#fw_command_unban = sed -i.old /ALL:\ $DBFAP_ADDR/d /etc/hosts.deny



#
[null firewall]
#==================
#
# use null firewall if you want to use this host as a kind of honeypot
# this host will report fails to server but will not ban any attacker
#
#fw_command_init =
#fw_command_finish =
#fw_command_flush =
#fw_command_ban =
#fw_command_unban =
```