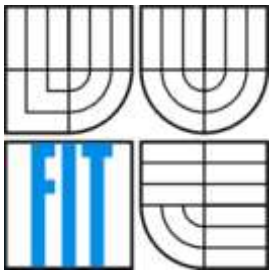
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DETEKCE OPERAČNÍCH SYSTÉMŮ V SÍŤOVÉM PROVOZU POMOCÍ IPFIX

DETECTION OF OPERATION SYSTEMS IN NETWORK TRAFFIC USING IPFIX

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN VYMLÁTIL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR MATOUŠEK, Ph.D.

BRNO 2014

Abstrakt

Tato práce se zabývá problematikou detekce operačních systémů v síťovém provozu pomocí IPFIX. Používá k tomu metodu fingerprintingu, kdy jsme schopni na základě informací z IP a TCP hlavičky určit OS. Kombinace těchto údajů tvoří jedinečný podpis operačního systému. Na základě zjištěných informací je navržen a implementován plugin pro sondu FlowMon. Plugin je testován na živém síťovém provozu a na pcap souborech.

Abstract

This task deal with detection of operation system in network traffic using IPFIX. The main idea of this task is based on the fingeprinting, when we use information from IP and TCP headers to determine operation system. This data represent a unique signature of the operation system. Based on the information a plugin for the FlowMon probe was designed and implemented. Plugin was tested on live network traffic and pcap files.

Klíčová slova

Detekce, operační systém, fingerprinting, FlowMon, IPFIX.

Keywords

Detection, operation system, fingerprinting, FlowMon, IPFIX.

Citace

Vymlátil Martin: Detekce OS v síťovém provozu pomocí IPFIX, bakalářská práce, Brno, FIT VUT v Brně, 2014

Detekce operačních systémů v síťovém provozu pomocí IPFIX

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.

……
Martin Vymlátil
13. května 2014

Poděkování

Chci poděkovat především vedoucímu této bakalářské práce Ing. Petru Matouškovi, Ph.D. za věnovaný čas, cenné rady a připomínky, bez nichž by tato práce vznikala velice těžce. Mé poděkování patří také Mgr. Martinu Elichovi a Ing. Petru Špringlovi ze společnosti INVEA-TECH za spolupráci a rady při návrhu a implementaci. V neposlední řadě patří mé poděkování rodině a přítelkyni za neustálou podporu.

Obsah

1 Úvod

V dnešní době, kdy neustále roste počet operačních systémů, ale také počet chytrých telefonů, není určitě na škodu pro administrátora sítě vědět, kdo a s jakým operačním systémem pracuje. Tuto informaci může využít nejen při rozvoji sítě, ale také při nákupu nového softwaru, při správě počítačů v síti, např. pro nasazení nového OS nebo přechodu na novější verzi, atd.

1.1 Proč detekovat OS?

Důvodů proč detekovat OS, které se vyskytují v naší síti, je několik. Mezi ty nejvýznamnější patří [11]:

- Určení zranitelných cílových hostitelů: díky detekci OS v naší síti můžeme nalézt stroje, které potřebují bezpečnostní záplatu nebo aktualizaci.
- Inventář sítě a podpora: existují i administrativní důvody, proč sledovat OS v síti. Konkrétně např. než obnovíme smlouvu týkající se podpory určitého OS, můžeme zjistit, zda ho někdo vlastně používá. To může být užitečné i při plánování rozpočtu IT oddělení v podniku.
- Detekce neautorizovaných a nebezpečných zařízení: stále větší dostupnost mobilních zařízení a levná síťová zařízení mohou způsobit, že zaměstnanci budou vytvářet vlastní bezdrátové sítě (WAP). Tímto ovšem ohrožují firemní síť a zpřístupňují ji potencionálním útočníkům. Pravidelným skenováním lze zabránit použití takovýchto neautorizovaných a nebezpečných zařízení.
- Sociální inženýrství
- Detekce NAT zařízení

1.2 Cíle a struktura bakalářské práce

Cílem této bakalářské práce je seznámit se s možnostmi detekce operačních systémů na základě síťového provozu a dále za využití získaných informací navrhnout a implementovat plugin pro sondu FlowMon. Sonda FlowMon patří společnosti INVEA-TECH. Na základě spolupráce s touto firmou vznikla tato bakalářská práce. Plugin má za cíl rozšířit funkčnost sondy o klasifikaci operačních systémů, které se budou ze sondy exportovat pomocí protokolu IPFIX. Je nutné tedy rozšířit IPFIX záznam, aby kolektor dokázal zpracovat údaje získané z pluginu.

Kapitola 2 popisuje protokol IPFIX, především jeho architekturu a využití. Následující kapitola 3 pak zkoumá možnosti a omezení detekce OS v síťovém provozu a analyzuje dostupné nástroje zabývající se tímto tématem. V první části kapitoly 4 je popsána sonda FlowMon, její architektura a využití. Hlavní náplní této kapitoly je přiblížit návrh pluginu pro klasifikaci OS. Samotná implementace je předmětem kapitoly 5, jsou zde také popsána možná rozšíření a vylepšení pluginu. V kapitole 6 je popsáno testování pluginu, které probíhalo převážně na pcap souborech. Poslední kapitola je věnována zhodnocení výsledků.

2 Protokol IPFIX

Na provoz v počítačové síti se můžeme dívat jako na velké množství datových toků mezi síťovými prvky. Především pro správu sítě, ale i pro jiné účely, je důležité a užitečné mít informace o těchto tocích. Proto bylo nutné vytvořit jednotný protokol, který reprezentuje tok, dokáže sbírat informace o tocích, exportovat tyto informace a to vše ve specifickém formátu.

Následující kapitola se bude zabývat samotným protokolem IPFIX, jeho architekturou a využitím. Tento protokol používá sonda FlowMon pro distribuci dat v rámci sítě, proto je nutné znát jeho architekturu a také jeho vlastnosti a možnosti.

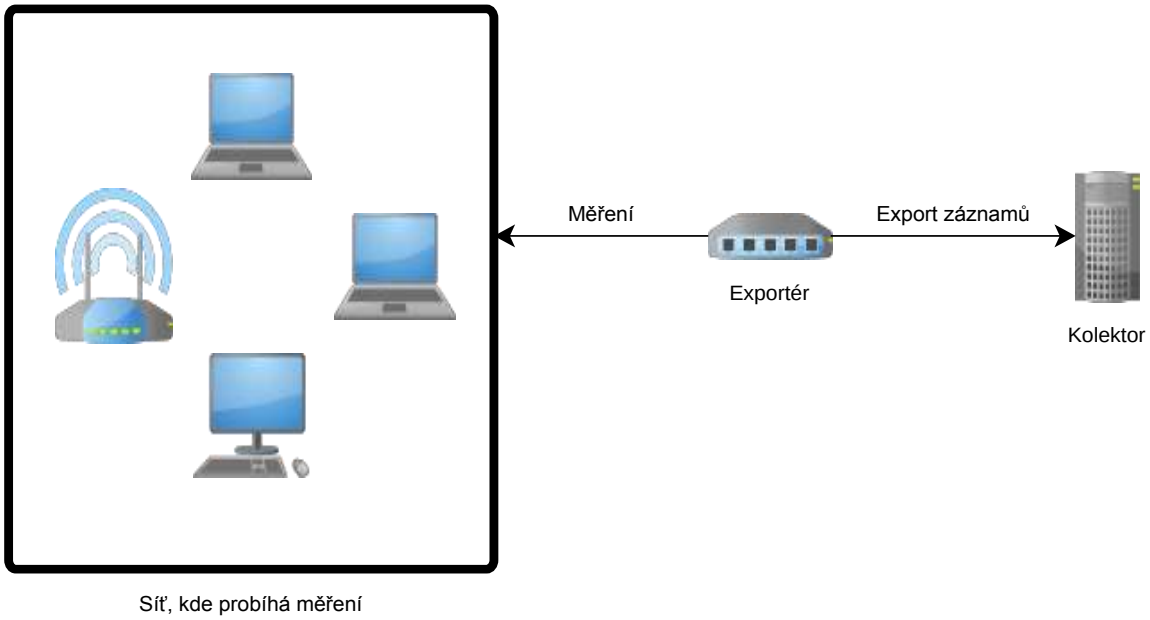
2.1 Přenos informací o tocích

IPFIX je protokol, který slouží k přenosu informací o IP tocích v síti. Byl vytvořen kvůli potřebě obecného, rozsáhlejšího a univerzálního standardu pro exportování záznamů o IP tocích z routerů, sond a dalších zařízení, které se používají pro různé zprostředkovací systémy, systémy pro správu sítě a to s cílem usnadnění služby, jako je měření toků, fakturace služeb, atd [1]. Vychází z protokolu NetFlow v9, ze kterého přebírá použití šablon a k původní definici přidává další hodnoty, které lze měřit a zpracovávat. IPFIX standard definuje, jak mají být informace o IP tocích formátovány a následně převedeny (poslány) z exportéru na kolektor, tj. zařízení sbírající záznamy o tocích v síti.

2.2 Architektura IPFIX

Klíčové prvky architektury IPFIX tvoří (obrázek 2.1) [2]:

- Exportér: je zařízení, které monitoruje procházející provoz. Vytváří a odesílá IPFIX záznamy o provozu na jeden či více kolektorů.
- Kolektor: je zařízení, které přijímá IPFIX záznamy z jednoho či více exportérů. Tyto záznamy dále zpracovává a ukládá. Je možná jejich grafická vizualizace nebo je možné vytvářet nejrůznější statistiky.
- IPFIX záznam: IPFIX umožňuje přenášet tři druhy záznamů a to šablonu, rozšířenou šablonu (rozšíření o další informace) a datový formát.
- Měřící bod: je místo v síti, kde dochází k pozorování IP paketů. Může to být například místo připojení sondy, port routeru, atd.
- Tok: IPFIX definuje tok jako množinu IP paketů procházejících měřícím bodem v síti během určitého časového intervalu. Pakety, které přísluší k určitému toku, sdílí řadu společných vlastností, jako je např. stejná cílová IP adresa, stejný cílový port a protokol.
- IPFIX protokol: jde o protokol definující formát zprávy a způsob komunikace mezi exportérem a kolektorem.



Obrázek 2.1: Architektura IPFIX.

2.3 Popis protokolu IPFIX

IPFIX protokol zajišťuje komunikaci mezi zařízeními a definuje formát IPFIX zprávy. V IPFIX protokolu jsou n-tice (typ, délka, hodnota) popsány v šabloně, která obsahuje dvojici (typ, délka) a tato dvojice dále specifikuje, která pole (hodnota) jsou v datových záznamech. Tato vlastnost poskytuje velikou flexibilitu.

V porovnání s daty jsou šablony zasílány velmi zřídka, což vede ke snížení zatížení sítě. Odlišné datové formáty můžeme také jednoduše posílat. Stačí definovat novou šablonu s upřesněním dvojice (typ, délka) pro nový datový formát [1].

IPFIX definuje velké množství standardních informačních prvků, které jsou nezbytné (typ) pro šablony. Použití těchto standardních prvků umožňuje kompatibilitu mezi implementacemi různých dodavatelů. Navíc lze definovat specifické prvky pouze pro privátní použití. IPFIX zpráva může být zasílána skrze protokol SCTP, ale také skrze TCP i UDP [1]. Příklad IPFIX zprávy je v tabulce 2.1.

IPFIX message	
Bits 0 .. 15	Bits 16 .. 31
Version = 0x000a	Message Length = 64 B
Export Timestamp	
Sequence number = 0	
Observation Domain ID = 123456789	
Set ID = 2 (Template)	Set Length = 20 B
Template ID = 256	Number of Fields = 3
Typ = sourceIPv4Address	Field Length = 4 B
Typ = destinationIPv4Address	Field Length = 4 B
Typ = packetDeltaCount	Field Length = 4 B
Set ID = 256 (Data Set using Template 256)	Set Length = 28 B
Record 1, Field 1 = 192.168.0.201	
Record 1, Field 2 = 192.168.0.1	
Record 1, Field 3 = 235 Packets	
Record 2, Field 1 = 192.168.0.202	
Record 2, Field 2 = 192.168.0.1	
Record 3, Filed 3 = 42 Packets	

Tabulka 2.1: Příklad IPFIX zprávy. Převzato[1].

2.4 Využití IPFIX

IPFIX má stejně jako NetFlow řadu využití [2] [9]:

- Monitorování sítě: můžeme monitorovat využití celé sítě, ale také jednotlivých částí sítě nebo konkrétních zařízení. Díky tomu lze odhalit různé problémy, které mohou v síti nastat, jako je např. omezená propustnost částí sítě, zařízení, atd. Monitorovat můžeme také jednotlivé aplikace a uživatele v sítí, tzn. můžeme zjistit, jaké aplikace jsou nejvíce používané, jaké zařízení uživatel nejvíce využívá, jak dlouho je uživatel připojený v síti, atd.

- Účtování: IPFIX záznamy poskytují širokou škálu informací, tudíž lze služby účtovat podle různých kritérií jako je např.: podle denní doby, podle aplikací, na základě využití šířky pásma, na základě skutečně přenesených dat, atd.

- Detekce útoků: na základě anomálií v síťovém provozu lze detekovat různé útoky, jako např. DoS, DDoS útoky, viry nebo skenování.

- Rozvoj sítě: díky statistikám získaným na základě monitorování sítě můžeme naplánovat rozvoj sítě, tj. nákup nových síťových zařízení, zvýšení rychlosti internetu, atd.

[1] Dostupné z: http://en.wikipedia.org/wiki/IP_Flow_Information_Export

3 Detekce operačních systémů

Detekce operačních systémů je nejdůležitější částí této práce. Abychom mohli navrhnout funkční plugin, je nutné znát principy, možnosti a metody, které lze využít pro detekci OS v síťovém provozu. Mezi nejzajímavější nástroje patří aplikace P0f a Nmap a je jim věnovaná velká část následující kapitoly. Kromě těchto nástrojů kapitola pojednává o dalších aplikacích, které můžeme pro klasifikaci OS využít. Detekce operačních systémů v síti probíhá na základě tzv. OS fingerprintingu, což je proces, který pomocí kombinace parametrů a informací zjištěných v síťovém provozu dokáže určit příslušný OS. Téměř všechny techniky fingerprintingu jsou založeny na detekci odlišností v paketech zasílaných různými OS. Tyto techniky běžně analyzují IP datagram (konkrétně hodnoty TTL a ID), TCP protokol (Window size a příznaky SYN a SYN+ACK), DHCP request a ICMP request. Některé techniky pak využívají běžicí služby nebo otevřené porty. Rozlišujeme aktivní a pasivní fingerprinting [3] [4].

Aktivní fingerprinting je založen na zasílání paketů cílovému hostiteli a analyzování následné odpovědi. Typickým zástupcem aktivního fingerprintingu je nástroj Nmap.

Pasivní fingerprinting, jehož zástupce je například nástroj p0f, monitoruje síťový provoz bez vytváření a zasílání speciálních paketů a analyzuje pouze zachycené pakety cílového hostitele. Jak hlavička IP datagramu, tak TCP protokolu nemá konkrétní defaultní hodnoty, tyto hodnoty jsou většinou doporučené. Ovšem, jak ukazuje tabulka 3.1, různé OS se projevují odlišnostmi v určitých hodnotách IP a TCP hlavičky.

Protokol	Pole	Hodnota	OS
IP	Initial TTL	64	Nmap, BSD, Mac OS, Linux
		128	Novell, Windows
		255	Cisco IOS, Palm OS, Solaris
IP	Don’t fragment	Set	BSD, Mac OS, Linux, Novell, Windows, Palm OS, Solaris
		Not set	Nmap, Cisco IOS
TCP	Max segment size	0	Nmap
		1440	Windows, Novell
		1460	BSD, Mac OS,
TCP	Window size	1024-4096	Nmap
		65535	BSD, Mac OS
		2920-5840	Linux
		16384	Novell
		4128	Cisco IOS
		24820	Solaris
		Variable	Windows
TCP	SackOK	Set	Linux, Windows, OpenBSD
		Not set	Nmap, FreeBSD, Mac OS, Novell, Cisco IOS, Solaris

Tabulka 3.1: Běžné hodnoty OS při pasivním fingerprintingu. Převzato[2].

[2] Sanders, Ch.: Practical Packet Analysis using Wireshark to solve real-word network problems. No Starch Press, druhé vydání, červenec 2011, s. 194-197. ISBN 978-1-59327-266-1.

Objevuje se zde i nástroj Nmap, který provádí klasifikaci OS na základě aktivního fingerprintingu. Nmap totiž sám posílá do sítě speciálně upravené pakety a tím šíří svůj specifický podpis.

3.1 Aplikace p0f

P0f[3] je nástroj, který využívá řadu sofistikovaných a čistě pasivních mechanismů fingerprintingu k identifikaci hostitele z jakékoli TCP/IP komunikace. Běžné použití zahrnuje průzkum během penetračních testů, rutinní monitorování sítě, zjištění neoprávněného síťového propojení v podnikové síti, poskytování nástrojů prevence proti zneužití a řadu různých forenzik (forenzních věd).

P0f pasivně monitoruje síťový provoz bez vytváření extra paketů, tzn., že sám nevysílá žádné pakety. Operační systém rozpozná na základě analýzy zachycených paketů cílového hostitele, konkrétně pracuje s hodnotami IP datagramu TTL, DF a ToS a s Window size z TCP protokolu. Zjištěné hodnoty následně porovná s podpisy v podpisovém souboru a vyhodnotí nejlepší shodu. Používá tři podpisové soubory (implicitně p0f.fp, dále méně používaný p0fra.fp a nejméně používaný p0frb.fp), na základě zadaného parametru lze zvolit, s kterým podpisovým souborem má pracovat.

Většina metrik, které p0f používá, byla navržena přímo pro tento nástroj. Tento nástroj využívá data získaná z IPv4 (případně IPv6) a TCP hlavičky a další údaje, především z aplikační vrstvy, jako například HTTP User-agent nebo MTU. P0f umí pracovat ve třech režimech:

- offline, kdy vstup probíhá prostřednictvím pcap souboru
- online, kdy odposlouchává síťový provoz na zvoleném rozhraní
- online, kdy k odposlouchávání provozu využívá API socket.

Po odchycení paketu nejprve dojde k jeho analýze. Nejdůležitější je jeho hlavička, ze které získá p0f potřebné informace k další práci. Všechny zjištěné informace si ukládá do připravených struktur (IPv4, IPv6, TCP, atp. mají vlastní strukturu), tak aby s nimi bylo možné poté dál pracovat. Následně se snaží p0f najít shodu pomocí získaných informací a vlastní databáze operačních systémů pof.fp. Tato databáze je obyčejný textový soubor a je rozdělena na několik částí, podle typu provozu:

- TCP signatures: komunikace na základě TCP protokolu.
- HTTP signatures: využívá údaje z HTTP hlavičky, nepoužívá se primárně pro detekci OS, ale především pro detekci NAT zařízení.
- MTU signatures: na základě MTU určí typ spojení.

V případě TCP a HTTP signatures vždy ještě rozlišuje klient a server, tzn. u TCP SYN a SYN+ACK, u HTTP request a response.

V případě, že nalezne shodu v této databázi, vrátí název příslušného operačního systému. V případě, že nenalezne shodu, vypíše údaje, podle kterých shodu zjišťoval a k nim hodnotu Unknown, která značí, že příslušný OS v databázi nenalezl.

[3] p0f v3. [online], [cit. 2013-10-22]. Dostupné z: http://lcamtuf.coredump.cx/p0f3/

3.1.1 Popis databáze aplikace p0f

P0f implicitně používá pro detekci OS soubor *p0f.fp*. Soubor je rozdělen na 3 části – MTU signatures, TCP signatures a HTTP signatures. Pro nás je nejzajímavější část TCP signatures. Záznam z databáze můžeme vidět na obrázku 3.1.

```
label = s:win:Windows:XP
sig   = *:128:0:*:16384,0:mss,nop,nop,sok:df,id+:0
sig   = *:128:0:*:65535,0:mss,nop,nop,sok:df,id+:0
sig   = *:128:0:*:65535,0:mss,nop,ws,nop,nop,sok:df,id+:0
sig   = *:128:0:*:65535,1:mss,nop,ws,nop,nop,sok:df,id+:0
sig   = *:128:0:*:65535,2:mss,nop,ws,nop,nop,sok:df,id+:0
```

Obrázek 3.1: podpis OS z databáze nástroje p0f.

Popis záznamu

Každému podpisu předchází řádek `label`. P0f používá `label` pro seskupování podpisů, které představují stejný operační systém nebo aplikaci.

```
label=type:class:name:flavor
```

- **type**: ‚s' představuje podpis, který již nelze nějakým rozumným způsobem zpřesnit, ‚g' naopak představuje obecný podpis.
- **class**: rodina OS, většina TCP podpisů mají rodinu OS specifikovanou („win", „unix", atd.).
- **name:** jméno OS čitelné pro uživatele.
- **flavor:** další informace o OS nebo aplikaci, např. verze OS, atd.

TCP signatures

Řádek, začínající `sig` představuje jeden podpis specifický pro OS. Formát podpisu je následující a obsahuje tyto položky:

```
sig = ver:ittl:olen:mss:wsize,scale:olayout:quirks:pclass
```

- **ver**: verze IP protokolu. IPv4 (4), IPv6 (6), případně oba (*).
- **ittl:** initial TTL, většina OS používá hodnoty 64, 128 a 255. Starší systémy pak mohou používat 32, některé systémy také používají hodnotu iTTL 60.
- **olen:** délka IPv4 options nebo IPv6 extension headers. Pro normální IPv4 obvykle 0, pro IPv6 0 vždy.
- **mss**: maximum segment size, pokud je uvedeno v TCP options. Může být *, pokud je mss závislé na parametrech síťové připojení odesílatel.
- **wsize:** window size. Může být vyjádřena jako konkrétní hodnota, ale také jako násobek MSS nebo MTU, nebo násobek nějakého čísla. Tyto případy p0f automaticky detekuje. Potom je možný zápis MSS*4, MTU*4, nebo %8192.
- **scale**: window scale specifikovaný v TCP options. Může být buď konkrétní hodnota, nebo *.
- **olayout:** čárkou oddělené položky TCP options, pokud paket obsahuje options. Olayout zachovává pořadí hodnot v TCP options. Sledované položky jsou uvedeny v tabulce 3.2.

Hodnota	Popis
EOL	End of Options list
NOP	No operation
MSS	Maximum segment size
WS	Window scale
SOK	Selective ACK permitted
SACK	Selective ACK
TS	Timestamp
?n	Neznáme option ID n

Tabulka 3.2: Položky olayout.

- **quirks:** čárkou oddělené další vlastnosti zjištěné z IP a TCP hlavičky (tabulka 3.3).

Hodnota	Popis
df	Don't fragment, nepoužívá se u IPv6
id+	DF nastaven, ale IP ID je nenulové, nepoužívá se u IPv6
id-	DF nenastaven, ale IP ID je 0, nepoužívá se u IPv6
ecn	Podpora ECN
0+	„must be a zero" pole je nenulové, nepoužívá se u IPv6
flow	Nenulové flow ID u IPv6, nepoužívá se u IPv4
seq-	Sequence number je 0
ack+	ACK number je nenulové, ale příznak ACK není nastaven
ack-	ACK number je 0, ale příznak ACK je nastaven
uptr+	URG pointer je nenulový, ale příznak URG není nastaven
urgf+	URG příznak je nastaven
pushf+	PSH příznak je nastaven
ts1-	Timestamp je 0
ts2+	Nenulové timestamp při TCP SYN
opt+	Nenulová data v options segmentu
exws	Excessive window scaling factor je větší jak 14
bad	Špatné TCP options

Tabulka 3.3: Položky quirks.

- **pclass:** představuje velikost užitečného zatížení. V případě, že paket obsahuje data, tak se tato položka vypočítá jako rozdíl velikosti paketu a datového offsetu. Pokud je výsledek nula, hodnota je 0, pokud je rozdíl nenulový, hodnota je +, jinak je hodnota *. Většinou je tato hodnota nulová.

Kromě výše uvedené databáze dále program p0f používá databázi *p0f.fp*, která je pouze na unixových systémech a je umístěná ve složce */etc/p0f/*. Tato databáze obsahuje otisky OS z TCP SYN provozu a je největší a nejpoužívanější databází pro určování OS v aplikaci p0f na unixových systémech. Vzor podpisu obsahuje obrázek 3.2.

```
S20:64:1:60:M*,S,T,N,W0:.:Linux:2.2 (1)
S22:64:1:60:M*,S,T,N,W0:.:Linux:2.2 (2)
S11:64:1:60:M*,S,T,N,W0:.:Linux:2.2 (3)
```

Obrázek 3.2: Vzor podpisu z unixové databáze P0f.

Popis záznamu

Podpis z unixové databáze nástroje p0f má odlišný formát od předchozí databáze, ale sleduje téměř stejné hodnoty.

```
wwww:ttt:D:ss:OOO...:QQ:OS:Details
```

- **www:** window size. U některých OS je nastavena konkrétní hodnota, někde je naopak násobkem MSS nebo MTU a někde může být hodnota window size proměnná.

- **ttt**: initial Time to live. Při získávání dat pro ověření podpisu nemůže být aktuální hodnota TTL větší než initial TTL, ani nemůže být výrazně nižší. U p0f je nastaven limit hops na 40.

- **OOO:** tabulka 3.4 obsahuje sledované hodnoty TCP options. Hodnoty jsou odděleny čárkou.

Hodnota	Popis
N	No operation
E	End of options list
Wnnn	Window scale, hodnota nnn nebo * nebo %nnn
Mnnn	Maximum segment size, hodnota nnn nebo * nebo %nnn
S	Selective ACK permitted
T	Timestamp
T0	Timestamp, pokud se rovná 0
?n	Neznámé TCP option ID n

Tabulka 3.4: Položky OOO.

- **QQ** – další vlastnosti z IP a TCP hlaviček (tabulka 3.5). Obvykle je hodnota QQ prázdná.

Hodnota	Popis
P	Options za EOL
Z	Nulové IP ID
I	IP options specified
U	Nenulový URG pointer
X	2x je nevyužité pole nenulové
A	ACK number je nenulové
T	Nenulové druhé timestamp
F	Neobvyklé příznaky (PSH, URG, atd.)
D	Data payload
!	Chybné TCP options

Tabulka 3.5: Položky QQ.

- **D** – don't fragment, 0 = nenastaveno, 1= nastaveno.
- **ss** – celková velikost SYN paketu
- **OS** – rodina OS, např. Windows, Linux, FreeBSD, atd.
- **Details** – popis OS nebo aplikace, např. verze.

Pole Window size je rozděleno na 5 druhů:

1) WIN_TYPE_NORMAL – konkrétní hodnota 1 – 65535
2) WIN_TYPE_MOD – modulo win size, 2 – 65535
3) WIN_TYPE_MSS – násobek MSS, 1 – 1000
4) WIN_TYPE_MTU – násobek MTU, 1 – 1000
5) WIN_TYPE_ANY – ostatní

Algoritmus detekce

P0f data získaná z pcap souboru nebo ze síťového provozu postupně porovnává s podpisy v databázi, položku po položce. Pokud program zjistí, že se data neshodují v nějaké položce s podpisem, přeskočí další porovnávání a začne data porovnávat s dalším podpisem. Příklad klasifikace OS pomocí nástroje p0f obsahuje obrázek 3.3. V případě drobných odchylek u hodnot quirks nebo ttl aplikace označí shodu za nepřesnou (fmatch) a pokračuje v hledání lepší shody. Pokud lepší shodu nenajde je tento OS klasifikován podle nepřesné shody (fmatch) a uživatel je informován o neúplné shodě dat s podpisem. Program p0f je schopen na základě dat získaných z TCP paketů klasifikovat přes třicet různých operačních systémů a zařízení (firewall, router, atd.).

```
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on 'ostatni/http_gzip.cap', 262 sigs (14 generic
192.168.69.2:34059 - Linux 2.6 (newer, 3) (up: 5587 hrs)
  -> 192.168.69.1:80 (distance 0, link: ethernet/modem)
[+] End of input file.
```

Obrázek 3.3: Příklad klasifikaci pomocí nástroje p0f.

3.2 Detekce NAT zařízení – natdet

Natdet[4] vznikl jako výsledek výzkumu Masarykovy univerzity a je pluginem pro NfSen. Detekuje podezřelá zařízení za pomoci rozšířených NetFlow záznamů. Díky tomuto je možné detekovat i různé OS a to na základě odlišného chování v určitých aspektech (hodnot IP datagramu TTL a ID, délce TCP paketů s příznakem SYN). Další, nová metoda, která umí detekovat různé OS, sleduje sekvence přidělených zdrojových TCP a UDP portů a opět využívá odlišných vlastností v chování.

[4] MUNI: Detekce NAT zařízení. [online], [cit. 2013-10-22].
Dostupné z: http://www.muni.cz/research/projects/4622/web/natdet?lang=cs

3.3 Nástroj Yaf – yet another flowmeter

Yaf[5] je soubor nástrojů sloužících k měření toků v síti. Používá se jako senzor pro zachycení informací v síti a exportu získaných informací ve formátu IPFIX. Mezi nástroje yaf patří dále yafscii, yaf application labeling, yaf deep packet inspection a yaf DHCP fingerprinting.

Z pohledu detekce operačních systémů je důležitý nástroj yaf DHCP fingerprinting. DHCP fingerprinting sleduju pořadí DHCP option v DHCP request, konkrétně option 55, kde je seznam dalších parametrů, které DHCP klient požaduje (jako je například DNS server, proxy server, výchozí brána, atd.). Na základě pořadí, ve kterém klient žádá další informace, je možné určit OS klienta. Yaf dále exportuje i specifické informace o HW, pokud jsou k dispozici. Yaf pro určení OS obsahuje a používá databázi podpisů Fingerbank.org *dhcp_fingerprints.conf*. Příklad záznamu z této databáze můžeme vidět na obrázku 3.4.

```
[os 109]
description=Microsoft Windows 8
fingerprints=<<EOT
1,15,3,6,44,46,47,31,33,121,249,252,43
EOT
```

Obrázek 3.4: Ukázka databáze *dhcp_fingerprints.conf*.

3.4 Program Nmap

Nmap[6] je nástroj, který se používá především pro zjišťování a mapování sítí a pro bezpečnostní audit, ale také pro audit služeb a systémů v počítačových sítích, pro správu sítě a síťových zdrojů atd. Tento nástroj je multiplatformní a původně byl vyvinut pro skenování rozsáhlých sítí, ovšem pracuje velice dobře i pro malé sítě a samotné cílové hostitele. Nmap využívá aktivní fingerprinting. Posílá speciálně upravené pakety, díky nimž detekuje otevřené porty a služby běžící na těchto portech, typ OS, jaký typ firewallu hostitel používá atd. Po poslání těchto paketů Nmap čeká na odpověď. Následně je mnoho atributů z těchto odpovědí analyzováno a jejich kombinací získáme podpis operačního systému. Tento nástroj využívá vlastní databázi OS *nmap-os-db*.

Nmap pošle až šestnáct TCP, UDP a ICMP sond (paketů) na otevřené porty cílového hostitele a čeká na jejich odpovědi. Na základě odpovědí na zaslané sondy provede Nmap více jak dvacet testů. A právě výsledky těchto testů představují otisk zařízení, který se uloží do paměti. Tento otisk je dále dekódován. Algoritmus vlastní detekce OS spočívá v porovnání získaného otisku se všemi referenčními podpisy z databáze *nmap-os-db*, přičemž se porovnává každá hodnota. V případě, že nějaká hodnota chybí, je tento test vynechán. Pokud algoritmus nenalezne perfektní shodu OS, vrací tu nejpravděpodobnější variantu.

3.4.1 Popis databáze programu Nmap

Databáze OS pro Nmap je uložena v souboru nmap-os-db. Tato databáze je velice rozsáhlá a kromě detekce OS umožňuje také určit různá zařízení, např. firewall, router, switch, atd. Příklad záznamu z této databáze můžeme vidět na obrázku 3.5.

[5] Yaf. [online], [cit. 2013-10-22]. Dostupné z: http://tools.netsa.cert.org/yaf/docs.html

[6] Nmap. [online], [cit. 2013-10-22]. Dostupné z: http://nmap.org/

Popis záznamu z databáze

- Pole **Fingerprint** obsahuje informace o operačním systému a jeho popis.
- Pole **Class** obsahuje čtyři položky oddělené znakem ‚|'. Každý otisk má jednu nebo více tříd Class. Struktura pole Class: vendor | OS family | OS generation | device type.
- Pole **CPE**[7] má formu URL. Common platform enumeration (CPE) je standardizovaný způsob, jak pojmenovávat software, OS a hardware.

```
# Microsoft Windows Server Version 6.0 (Build 6002: Service Pack 2)
Fingerprint Microsoft Windows Server 2008 SP2
Class Microsoft | Windows | 2008 | general purpose
CPE cpe:/o:microsoft:windows_server_2008::sp2
SEQ(CI=I%II=I%TS=7)
OPS(O1=M2300ST11%O2=M2300ST11%O3=M2300NNT11%O4=M2300ST11%O5=M2300ST11%O6=M2300ST11)
WIN(W1=2000%W2=2000%W3=2000%W4=2000%W5=2000%W6=2000)
ECN(R=Y%DF=Y%T=7B-85%TG=80%W=2000%O=M2300NNS%CC=N%Q=)
T1(R=Y%DF=Y%T=7B-85%TG=80%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
T3(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=O%F=AR%O=%RD=0%Q=)
T4(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T5(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(DF=N%T=7B-85%TG=80%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(DFI=N%T=7B-85%TG=80%CD=Z)
```

Obrázek 3.5: Záznam z databáze programu Nmap.

- Ostatní pole mají následující formát:
  `Název_pole(Test1=Výsledek%Test2=Výsledek1|Výsledek2)`
  Každý řádek začíná názvem pole. Název pole označuje sondu, které se řádek týká. V závorkách poté následují jednotlivé testy a jejich výsledky. Jednotlivé testy jsou odděleny znakem ‚%'. V případě, že má test více jak jeden výsledek, tak jsou výsledek takového testu odděleny znakem ‚|'. Jednotlivé sondy, testy a jejich možné výsledky jsou popsány níže.

Sondy zasílané programem Nmap

Nmap zasílá až šestnáct sond (paketů) a na základě analýzy odpovědí na tyto pakety se snaží klasifikovat OS. Co znamenají jednotlivé testy a jaké jsou jejich výsledky, je popsáno v sekci Prováděné testy (Response tests) [5].

- **Generovaná sekvence SEQ, OPS, WIN, T1 (Sequence generation)**
  Prvních šest sond (paketů) zasílaných programem Nmap. Všechny tyto pakety jsou TCP pakety s příznakem SYN a jsou zasílané na otevřené porty cílového hostitele. Pakety jsou odesílány co 100 ms. Tabulka 3.6 obsahuje podrobný popis jednotlivých paketů.

[7] CPE. [online], [cit. 2014-05-12]. Dostupné z: http://cpe.mitre.org/

Čas [ms]	Paket č.	Položky
0	1	WinScale(10), Nop, MSS(1460), TS, SACK, WS 1
100	2	MSS(1400), WinScale(0), SACK, TS, EOL, WS 63
200	3	TS, Nop, Nop, WinScale(5), Nop, MSS(640), WS 4
300	4	SACK, TS, WinScale(10), EOL, WS 4
400	5	MSS(536), SACK, TS, Win Scale(10), EOL, WS 16
500	6	MSS(265), SACK, TS, WS 512

Tabulka 3.6: Popis jednotlivých sond.

- TS = Timestam(0xFFFFFFFF, 0)
- WS = Window size field
- SACK = Selective Acknowlidge permitted
- Nop = No operation
- EOL = End of option list

Po odeslání jednotlivých sond Nmap analyzuje odpovědi na tyto sondy a provádí následující testy (viz Prováděné testy):

- Pro pole SEQ (sekvenční analýza paketů) provádí testy GCD, SP, ISR, TI, II, TS a SS.
- Pro pole OPS (hodnoty TCP options) provádí testy O1 – O6.
- Pro pole WIN (hodnoty Window size) provádí testy W1 – W6.
- T1 (1. paket generované sekvence) provádí testy R, DF, T, TG, W, S, A, F, O, RD a Q.

- **ICMP echo IE**
  Tato sonda zasílá dva ICMP echo request pakety s těmito hodnotami:

  1) První paket obsahuje tyto hodnoty - IP DF = 0x02, ToS = 0, code = 9, sequence number = 256, random IP ID, random ICMP request ID, data payload = 120B s následujícím obsahem 0x00.

  2) Druhý paket je podobný, změna je u hodnot ToS = 4, code = 0, IP ID a ICMP ID jsou inkrementovány, data payload má velikost 150B a obsahuje hodnotu 0x00.

  Testy, které jsou prováděny pro pole IE, jsou R, DFI, T, TG a CD (viz Prováděné testy).

- **TCP explicit congestion notification ECN**
  ECN[8] je metoda pro zlepšení výkonnosti internetu. Směrovač signalizuje problémy s přetížením, ještě než musí zahazovat pakety. Sonda posílá TCP paket s příznakem SYN a s nastavenými příznaky CWR a ECE. Tyto dva příznaky signalizují přetížení.
  Testy prováděné pro pole ECN jsou R, DF, T, TG, W, O, CC a Q (viz Prováděné testy).

[8] RFC 3168. Dostupné z: http://tools.ietf.org/html/rfc3168

- **TCP T2 – T7**
  Tato sonda zasílá celkem šest TCP paketů (sond). Hodnoty položek jednotlivých paketů obsahuje tabulka 3.7. Pro všechny pakety jsou pak společné tyto hodnoty:

  - Data,
  - WinScale = 10,
  - 1x Nop,
  - Maximum segment size = 256,
  - Timestamp(0xFFFFFFFF, 0),
  - selective ACKnowlidge permitted (u sondy T7 winScale = 15).

Paket	Flags	IP DF	Window size field	Port
T2	No flags	0x02	128	Open
T3	SYN, FIN, URG, PSH	0x00	256	Open
T4	ACK	0x02	1024	Open
T5	SYN	0x00	31 337	Closed
T6	ACK	0x02	32 768	Closed
T7	FIN, PSH, URG	0x00	65 535	Closed

Tabulka 3.7: Popis zasílaných paketů.

Prováděné testy pro pole T2 – T7 jsou testy R, DF, T, TG, W, S, A, F, O, RD a Q (viz Prováděné testy).

- **UDP U1**
  Sonda U1 posílá UDP paket na uzavřený port cílového hostitele. Položka data obsahuje třista krát znak ‚C‘ (0x43). Pokud je port opravdu uzavřený a není zde firewall, tak Nmap očekává ICMP zprávu o nedostupném portu.
  Testy prováděné pro pole U1 jsou R, DF, T, TG, IPL, UN, RIPL, RID, RIPCK, RUCK a RUD (viz Prováděné testy).

Prováděné testy (Response tests)

Program Nmap klasifikuje OS právě podle prováděných testů. Pro každou sondu, přesněji pro každou odpověď na paket zaslaný sondou, je prováděno několik testů. Výsledky těchto testů pak představují podpis operačního systému [5].

- **TCP ISN greatest commom diviser (GCD)**
  Test GCD zkoumá odpovědi s příznakem SYN/ACK, konkrétně položku ISN (32-bit, Initial sequence number) a snaží se určit nejmenší číslo, o které ISN hostitel navyšuje. Vypočítá postupně navýšení ISN z odpovědí pro dvě po sobě jdoucí sondy. Toto navýšení uloží do pole a následně určí nejmenší číslo. Výsledkem tohoto testu je tedy nejmenší číslo, o které byla položka ISN navýšena.

- **TCP ISN counter rate (ISR)**
  Test ISR se snaží určit průměrnou hodnotu, o kterou se navyšuje položka ISN. Opět je využito pole jako v předchozím testu. Rozdíl mezi dvěma po sobě jdoucími sondami je

vydělen časem, který uplynul mezi posláním těchto sond. Pokud je hodnota menší než 1, výsledek je 0. Jinak se výsledek vypočte jako 8 * binární logaritmus z průměrné hodnoty.

- **TCP ISN sequence predictability index (SP)**
  Tento test provádí určení standardní odchylky ISN. Snažíme se určit, jak obtížné by bylo předpovědět další ISN od známé sekvence šesti reakcí na sondy. Využíváme k tomu výsledek testu GCD. Pokud je hodnota menší nebo rovna 1, výsledek je 0. Jinak se výsledek vypočte jako 8 * binární logaritmus výsledku testu GCD.

- **IP ID sequence generation algorithm (TI, CI, II)**
  Zde se provádí celkem tři testy:

  1) Test TI – test odpovědí na SEQ sondy, musíme mít alespoň tři odpovědi.
  2) Test CI – test odpovědí na sondy T5 – T7, musíme mít alespoň dvě odpovědi.
  3) Test II – test odpovědí na IE sondy, musí přijít obě odpovědi.

  Jakým způsobem tyto testy probíhají:

  1) Pokud jsou všechny ID = 0, tak výsledek testu je Z.
  2) Pokud se sekvence IP ID zvyšuje vždy alespoň o 20 000, výsledek je testu RD. Kontrola IP ID nelze provést u testu II.
  3) Pokud jsou všechny IP ID stejné, nastavíme hodnotu IP ID jako výsledek.
  4) Pokud je rozdíl mezi dvěma po sobě jdoucími IP ID větší jak 1000 a není dělitelný 256, výsledek je RI. Pokud je dělitelný 256, tak musí být hodnota minimálně 256 000, aby byl výsledek opět RI.
  5) Pokud jsou všechny rozdíly dělitelné 256 a nejsou větší jak 5120, tak je výsledek BI.
  6) Pokud jsou všechny rozdíly menší než 10, výsledek je I.
  7) Pokud testy 1 – 6 nebyly úspěšné, tak jsou hodnoty TI, CI a II vynechány z otisku.

- **Shared IP ID sequence boolean (SS)**
  Test SS zkoumá, zda TCP a ICMP pakety sdílí IP ID sekvenci. Provádí se pouze, pokud je hodnota testu II rovno RI, BI nebo I (stejně to platí pro test TI). Pokud pakety sdílí sekvenci IP ID, tak je výsledek tohoto testu S. Jinak je výsledek O. Jestli pakety sdílí sekvenci IP ID, můžeme zjistit pomocí tohoto výpočtu:

  - Nejdříve vypočítáme hodnotu *avg = (poslední TCP IP ID – první TCP IP ID) / (číslo poslední sondy – číslo první sondy)*.
  - Pokud je první ICMP echo request IP ID menší, než poslední *TCP IP ID + 3*avg*, tak je výsledek S, jinak O.

- **TCP timestamp option algorithm (TS)**
  Test TS sleduje položku TCP timestamp u SEQ sondy, konkrétně hodnotu TSval. TSval představuje první 4 B z položky TCP timestamp. Rozdíl mezi dvěma po sobě jdoucími TSval dělí množstvím času mezi zasláním dvou sond. Počítá míru přírůstku položky TCP timestamp za sekundu. Následně vypočítá průměrný přírůstek:

  1) Pokud je nějaký paket bez položky TCP timestamp, výsledek je U.
  2) Pokud má kterákoliv položka TCP timestamp hodnotu 0, výsledek je 0.

3) Pokud průměr přírůstku spadá do intervalu, výsledek se rovná:
    a. 0 – 5.66 = 1
    b. 70 – 150 = 7
    c. 150 – 350 = 8
4) Jinak se výsledek vypočítá jako binární logaritmus průměrného přírůstku.

- **TCP options (O, O1 – O6)**
  Testy O, O1 – O6 zaznamenávají TCP options jako string. Zachovávají pořádí v TCP options. V tabulce 3.8 jsou popsány hodnoty, které testy zaznamenávají a zda mají tyto hodnoty nějaký argument.

Hodnota	Popis	Argument
L	EOL – end of option list	Nemá
N	Nop – No operation	Nemá
S	Selective Acknowlidge permitted	Nemá
M	Maximum segment size	Hodnota MSS
W	Window scale	Hodnota winScale
T	Timestamp	2 binární hodnoty (0 bez hodnoty)

Tabulka 3.8: Zaznamenávané položky z TCP option.

- **TCP initial window size (W, W1 – W6)**
  Tyto testy zaznamenávají z odpovědí hodnoty položky window size TCP paketu. Výsledkem je tedy hodnota window size.

- **Responsiveness (R)**
  Test R zkoumá, zda přišla odpověď na zaslanou sondu. V případě, že odpověď nepřišla, výsledek je N. Jinak je výsledek testu Y. Tento test se neprovádí pro sondy IE a U1.

- **IP don’t fragment bit (DF)**
  DF testuje, jestli je u odpovědi na sondu nastaven bit Don’t fragment. Pokud ano, výsledek je Y. Pokud není nastaven, výsledek tohoto testu je N.

- **Don’t fragment ICMP (DFI)**
  Test DFI je modifikací předchozího testu speciálně upraveného pro IE sondu. Sleduje hodnotu bitu Don’t fragment u odpovědi na ICMP sondy. Možné výsledky testů jsou uvedeny v tabulce 3.9.

Hodnota	Popis
N	Žádná odpověď nemá nastaven DF
Y	Obě odpovědi mají nastaven DF (0x02)
S	Odpovědi mají DF nastaven stejně jako sondy
O	Odpovědi mají DF nastaven různě

Tabulka 3.9: Výsledky testů DFI.

- **IP initial time-to-live (T)**
  Test T se snaží určit hodnotu initial TTL. Využívá k tomu sondu U1. V případě, že je port nedostupný, v odpovědi přijde i originální paket (sonda U1) s dekrementovanou hodnotou TTL. Díky tomu můžeme zjistit hodnotu hops (odečteme TTL, které obsahuje originální paket v odpovědi od počáteční hodnoty TTL sondy U1) a následně ji přičíst k TTL z odpovědi a tím získáme počáteční hodnotu TTL.

- **IP initial time-to-live guess (TG)**
  U testu TG se snažíme určit hodnotu intial TTL. Pokud nelze zjistit initial TTL jinak, využijeme znalosti, že běžné OS mají nastaveny hodnoty TTL na 32, 60, 64, 128 a 255. Hodnota TTL z odpovědi je pak zaokrouhlena k nebližší další hodnotě běžných initial TTL. Je vynechána pouze hodnota 60. Hodnota 60 je vynechána, protože rozdíl mezi hodnotou 60 a 64 je minimální. Tím pádem by k zaokrouhlení na hodnotu 64 docházelo velmi zřídka.

- **Explicit congestion notification (CC)**
  Test CC kontroluje odpověď na ECN sondu. Kontroluje nastavení příznaků ECE a CWR. Možné výsledky tohoto testu jsou uvedeny v tabulce 3.10.

Hodnota	Popis
Y	Je nastaven pouze ECE bit, podporuje ECN
N	Není nastaven žádný bit, nepodporuje ECN
S	Nastaveny oba bity, nepodporuje však ECN
O	Jinak

Tabulka 3.10: Výsledky testu CC.

- **TCP miscellaneous quirks (Q)**
  Test Q provádí dvě porovnání nad odpovědí na sondu ECN. Výsledek tohoto testu je string. První zkoumá hodnotu Reserved bit u TCP paketu v odpovědi na sondu ECN. Pokud je rovna 1, k výsledku je přidáno R. Následně, pokud není nastaven příznak URG v odpovědi na sondu ECN a pole urgent field je nenulové, tak je k výsledku přidáno U. Jinak je Q prázdné.

- **TCP sequence number (S)**
  Test S porovnává sequence number z odpovědi a acknowlidgment number ze sond T1 – T7. Výsledky porovnání obsahuje tabulky 3.11.

Hodnota	Popis
Z	sequence number je 0
A	sequence number = Ack number
A+	sequence number = Ack number + 1
O	Jinak

Tabulka 3.11: Výsledky testu S.

- **TCP RST data checksum (RD)**
  Test RD se provádí pro sondy T1 – T7. Pokud má odpověď na sondu příznak RST a zároveň obsahuje data, je proveden kontrolní součet algoritmem CRC 31. Tento kontrolní součet představuje výsledek tohoto testu. Pokud v tomto paketu nejsou žádná data, výsledek je 0.

- **IP total length (IPL)**
  Test IPL zaznamená hodnotu IP total length z hlavičky odpovědi na sondu U1.

- **TCP acknowlidgment number (A)**
  Test A porovnává acknowlidgment number z odpovědi a sequence number ze sond T1 – T7. Možné výsledky porovnání jsou uvedeny v tabulce 3.12.

Hodnota	Popis
Z	Ack number je 0
S	Ack number = sequence number
S+	Ack number = sequence number + 1
O	Jinak

Tabulka 3.12: Výsledky testu A.

- **Unused port unreachable field non-zero (UN)**
  Test UN zaznamenává poslední 4B hlavičky ICMP odpovědi na sondu U1 o nedostupném hostiteli. Podle RFC 792 mají být poslední 4B této hlavičky nulové. Některé OS ale nastavují i poslední 4B hlavičky ICMP v případě nedostupného hostitele.

- **TCP flags (F)**
  Test F zaznamenává příznaky odpovědí na sondy T1 – T7. Výsledkem tohoto testu je string složený z hodnot, které test přiřazuje jednotlivým příznakům (tabulka 3.13).

Hodnota	TCP flag
E	ECN echo (ECE)
U	URG
A	ACK
P	PSH
R	RST
S	SYN
F	FIN

Tabulka 3.13: TCP příznaky.

- **Returned probe IP total length value (RIPL)**
  Test RIPL zkoumá hodnotu IP total length v odpovědi na sondu U1 o nedostupném hostiteli. Pokud je hodnota IP total length 0x148, tak výsledek tohoto testu je G. Jinak je výsledek hodnota IP total length.

- **Returned probe IP ID value (RID)**
  Test RID zkoumá hodnotu IP ID v odpovědi na sondu U1. Pokud je hodnota IP ID v odpovědi rovna 0x1042, výsledek testu je G. Jinak je výsledek testu vrácená hodnota IP ID.

- **Integrity of returned probe IP checksum value (RIPCK)**
  RIPCK je test, která zkoumá hodnotu checksum v hlavičce ICMP odpovědi. Pokud je hodnota checksum v hlavičce odpovědi na sondu U1 stejná, jako na této sondě, výsledek je G. Pokud je vrácená hodnota 0, výsledek tohoto testu je Z. Jinak je výsledek I.

- **Integrity of returned probe UDP checksum (RUCK)**
  Test RUCK zkoumá hodnotu checksum v hlavičce odpovědi UDP. Pokud je položka checksum v hlavičce UDP odpovědi stejná, jako v sondě U1, výsledek je G. Jinak je výsledkem hodnota položky checksum.

- **Integrity of returned UDP data (RUD)**
  Test RUD zkoumá hodnotu dat vrácených v odpovědi na sondu U1. V případě, že mají data v odpovědi hodnotu ‚C' (0x43) nebo 0, výsledek testu je G. Jinak je výsledek tohoto testu I.

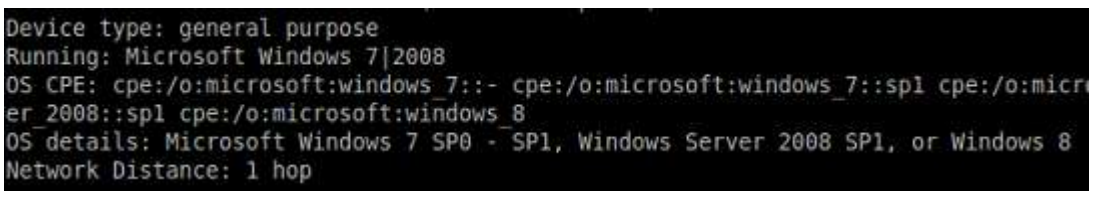
- **ICMP response code (CD)**
  Test CD je zaměřen na kódy vrácené jako odpověď na IE sondy. Na základě kombinace kódů v odpovědích na IE sondy určí výsledek tohoto testu (tabulka 3.14).

Výsledek testu CD	Kombinace
Z	Oba kódy = 0
S	Oba kódy jsou shodné, jako kódy v sondě
<NM>	Oba kódy mají stejnou nenulovou hodnotu <NM>
O	Jinak

Tabulka 3.14: Výsledky testu CD.

Algoritmus detekce

Získaná data testuje s každým referenčním otiskem v databázi *nmap-os-db*. Jestliže najde test pro danou položku, tak inkrementuje čítač possiblePoints. Následně porovná položky, a pokud se shodují, tak inkrementuje čítač NumMatchPoints. Vydělením NumMatchPoints a possiblePoints získáme pravděpodobnost shody. Pro ohodnocení testů Nmap využívá speciální strukturu MatchPoints. Příklad klasifikace OS programem Nmap můžeme vidět na obrázku 3.6.



Obrázek 3.6: Příklad detekce OS programem Nmap.

3.5 Fingerbank – DHCP fingerprinting

DHCP fingerprinting[9] je pasivní technologie identifikace specifického operačního systému nebo typu zařízení a díky svému plošnému charakteru není nákladná ani složitá. Technologie je užitečná pro analýzu síťového provozu a NAC (Network access control – přístup k bezpečnosti sítě).

Princip zjištění OS probíhá na základě pořadí v jakém si DHCP klient žádá o další nastavení a informace (DHCP option – DNS server, výchozí brána, atd.). Toto pořadí se porovná s databází OS *dhcp-fingerprints.conf* (obrázek 3.7) a díky tomu, že je poměrně unikátní, tak umožňuje určit příslušný OS.

```
[os 201]
description=Mac OS 9
fingerprints=<<EOT
1,3,6,15,33,42,44,45,46,47,69,70,71,74,78,79
EOT
```

Obrázek 3.7: Záznam z databáze *dhcp-fingerprints.conf*.

3.6 HTTP User agent

Informaci o příslušném operačním systému lze také zjistit z HTTP protokolu. Tato informace se nachází konkrétně v položce User agent, což je textový řetězec, který má předepsaný formát [6]. Tuto položku lze najít pouze u metody GET nebo POST. Obrázek 3.8 obsahuje hlavičku protokolu HTTP s položkou User agent.

```
Hypertext Transfer Protocol
⊞ GET /test/ethereal.html HTTP/1.1\r\n
  Host: cerberus\r\n
  User-Agent: Mozilla/5.0 (X11; U; Linux ppc; rv:1.7.3) Gecko/20041004 Firefox/0.10.1\r\n
  Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,
  Accept-Language: en-us,en;q=0.5\r\n
  Accept-Encoding: gzip,deflate\r\n
  Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
  Keep-Alive: 300\r\n
  Connection: keep-alive\r\n
  Cookie: FGNCLIID=05c04axp1yaqynldtcdiwis0ag1\r\n
  \r\n
  [Full request URI: http://cerberus/test/ethereal.html]
  [HTTP request 1/1]
  [Response in frame: 6]
```

Obrázek 3.8: Příklad HTTP hlavičky obsahující položku User agent.

3.7 Vyhodnocení metod

Většina zkoumaných metod fingerprintingu v této práci je pasivních, tedy neposílají žádné speciálně vytvořené pakety. Jediným zástupcem aktivního fingerprintingu je nástroj Nmap, který se využívá zejména pro mapování sítí. Jeho výhodou je určitě to, že nemusí čekat na informace, aby mohl detekovat operační systém. O tyto informace si sám řekne, ovšem celá detekce a především

[9] Fingerbank. [online], [cit. 2013-10-22]. Dostupné z: http://www.fingerbank.org/

algoritmus je v porovnání s pasivními metodami složitější. Všechny ostatní nástroje využívají pasivní fingerprinting. Jejich princip je velice podobný:

- Zachycení paketu
- Analýza hlavičky paketu
- Získání potřebných informací
- Porovnání získaných informací s vnitřní databází
- Určení operačního systému

Nástroj natdet je plugin NfSenu, tudíž pro další práci není vhodný. Další nástroje, Yaf a DHCP fingerprinting, využívají komunikace mezi DHCP klientem a DHCP serverem. Tato komunikace však nastává velice zřídka, záleží na době výpůjčky IP adresy DHCP serverem. Jako nevhodnější kandidátem pro vlastní návrh se jeví nástroj p0f, který pro detekci OS využívá IP a TCP pakety. Výhodou pasivních metod je, že jsou prakticky nezjistitelné, nejeví žádnou aktivitu, pouze analyzují hlavičky průchozích paketů. Jejich nevýhoda spočívá v tom, že nemusí mít všechny potřebné informace pro určení operačního systému, z tohoto důvodu je nutné zde zavést určité heuristiky.

	Metoda	Protokol	Platforma	Licence	Poznámky
P0f	pasivní	TCP, HTTP	Linux, Windows	GNU LGPL	
Natdet	pasivní	TCP/UDP		MUNI	Plugin NfSenu
Yaf	pasivní	DHCP	Linux, Mac OS, Posix, Unix	GNU GPL	
Fingerbank	pasivní	DHCP		ODBL, DBCL	Databáze podpisů
Nmap	aktivní	TCP/UDP	multi	GNU GPL	

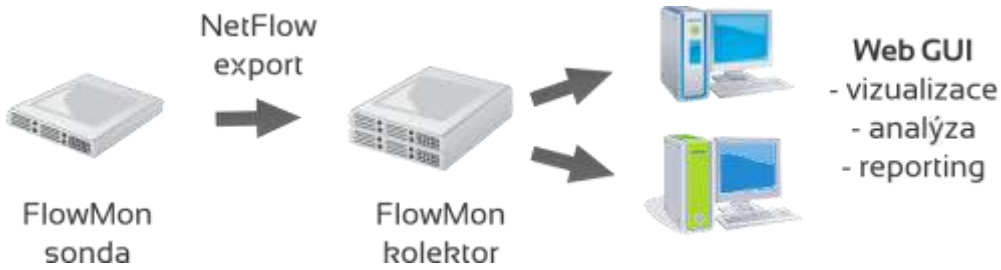
Tabulka 3.15: Porovnání nástrojů pro detekci OS.

4 Návrh pluginu pro sondu FlowMon

Před vlastním návrhem pluginu bylo nutné se dobře seznámit s technologií a architekturou sondy FlowMon od společnosti INVEA-TECH a také analyzovat a prozkoumat možné metody detekce operačních systémů v síti. Způsoby a nástroje, jak detekovat operační systémy, jsme viděli v předchozí kapitole. Následující kapitola se zabývá především architekturou sondy FlowMon a návrhem pluginu pro detekci OS.

4.1 FlowMon sonda

FlowMon sonda[10] patří do portfolia produktů FlowMon společnosti INVEA-TECH, které tvoří kompletní řešení pro monitorování sítí na bázi toků. Portfolio tvoří FlowMon sondy, FlowMon kolektory a FlowMon pluginy. Sonda analyzuje každý procházející paket a na základě zjištěných informací vytváří NetFlow/IPFIX statistiky [7]. Její velkou výhodou je schopnost pracovat na linkách s propustností až 10 GB/s, možnost umístění do libovolného bodu sítě a také transparentnost na L2 i L3 vrstvě. Kolektor je určen pro sběr NetFlow/IPFIX statistik z FlowMon sond a pro jejich další zpracování, jako je např. vizualizace [8].



Obrázek 4.1: Zjednodušený princip technologie FlowMon [8].

Vlastnosti sondy FlowMon[8]:

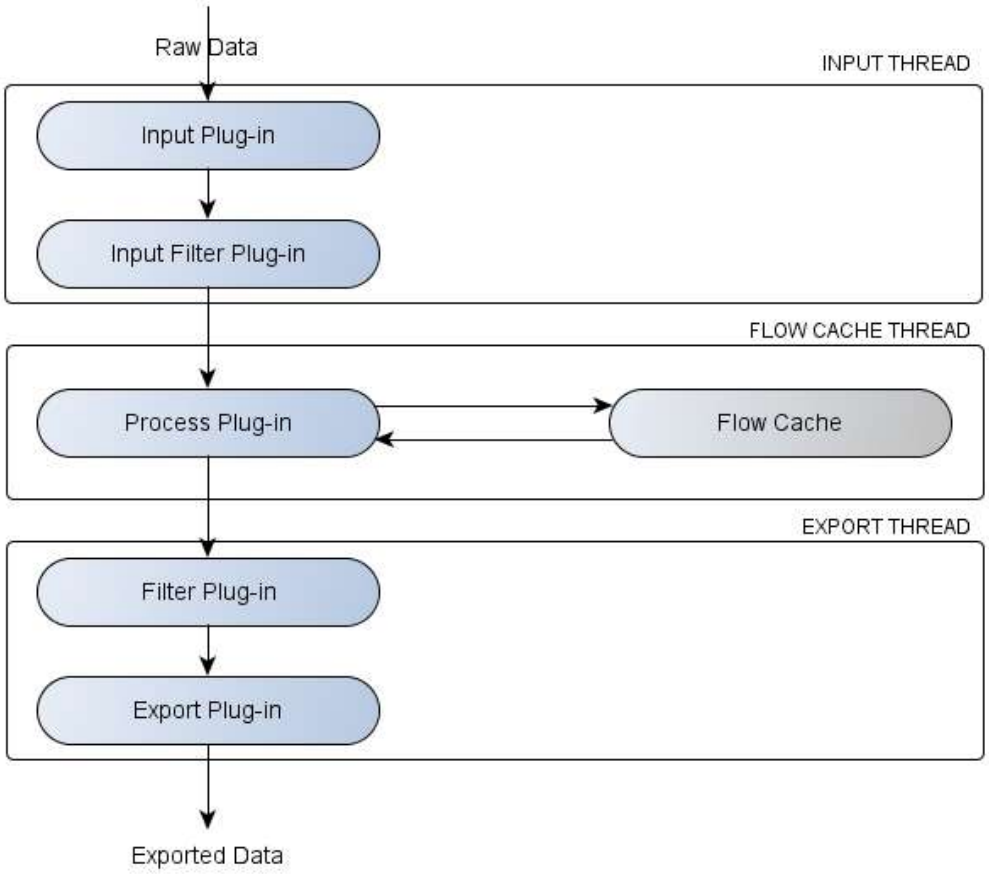
- Vysoce výkonná autonomní NetFlow sonda.
- Standardní a hardwarově akcelerované modely.
- Zpracování na rychlosti linky bez ztráty paketu.
- Podpora pro 10/100/1000 a 10 Gb Ethernet.
- Neinvazivní instalace, jednoduchá konfigurace pomocí webového rozhraní.
- 1x 10 Gb/s nebo až 4x 10/100/1000 monitorovací rozhraní
- Plně kompatibilní s nejrozšířenějšími NetFlow kolektory ostatních výrobců.
- Schopna zpracovat až 512 000 souběžných toků a 6 milionů paketů za sekunud.
- Podpora pro IPv4, IPv6, VLAN a MPLS.
- Transparentní na L2 i L3 vrstvě.
- Podporuje formáty NetFlow v5/v9 a IPFIX.

[10] INVEA-TECH: FlowMon sonda. Dostupné z: https://www.invea.com/cs/produkty-sluzby/flowmon/flowmon-sondy

FlowMon sondu lze využít pro monitorování sítě v reálném čase, analýzu síťového provozu, sledování uživatelů a služeb, pro účtování a fakturaci, kontrolu FUP (kontrola čerpání datových limitů), dohled nad přístupem k internetu, plánování kapacity sítě a datových linek, kontrolu peeringu a SLA (Service Level Agreement).

4.1.1 Tvorba pluginů pro sondu FlowMon

Pluginy umožňují rozšířit funkcionalitu sondy i kolektoru a poskytují pokročilé analýzy NetFlow/IPFIX statistik. Monitorují dostupnost a výkonnosti počítačů a služeb v síti [7]. Společnost INVEA-TECH také podporuje tvorbu pluginů svým komunitním programem. Vývoj pluginů pak probíhá na virtuální sondě FlowMon, která nemá takový výkon, ovšem pro tvorbu pluginů je dostatečná.



Obrázek 4.2: Architektura tvorby pluginů [7].

Nejdůležitější pluginy jsou plugin vstupní (Input Plug-in) a plugin procesní (Process Plug-in).

- Vstupní plugin slouží především pro analýzu paketů a získávání potřebných dat. Při práci můžeme využít i vstup např. z PCAP souboru. Plugin podporuje několik metod pro vstup, konkrétně to jsou metody GET_PACKET, GET_FLOW, GET_FINAL_FLOW. Vstupní plugin také umožňuje rozšířit flow record, který obsahuje základní data o toku, o vlastní položky potřebné pro další práci.

- Procesní plugin pak slouží pro zpracování dat a údajů získaných ve vstupním pluginu a ukládá informace o tocích do paměti flow cache.

- Filtrovací a exportní plugin slouží pro definici filtrů pro expiraci záznamů a pro vytváření a odeslání záznamů na kolektor.

4.2 Návrh pluginu

Při návrhu pluginu bylo nutné dodržet technologii a architekturu FlowMon sondy. Po analýze a prozkoumání všech metod detekce OS v kapitole 3., se jevila jako nejlepší varianta pasivní fingerprinting. Aktivní fingerprinting není možné využít, jelikož posílá do sítě speciálně upravené pakety, a tímto by porušil transparentnost samotné sondy FlowMon. Z dostupných a prozkoumaných nástrojů bylo nutné dále zvolit ten, který bude jako vzor pro plugin nejvhodnější. Nástroje Fingerbank a Yaf určují operační systém na základě komunikace mezi dhcp klientem a dhcp serverem. Tato komunikace je ovšem závislá na době zápůjčky IP adresy dhcp serverem, což může být i několik dní. Detekce OS tudíž probíhá po vzoru nástroje p0f (a tabulky 3.1), který využívá hodnot z IP a TCP protokolu. TCP protokol se objevuje v síťovém provozu poměrně často, ovšem je nutné, aby tyto pakety měly příznaky SYN nebo SYN+ACK. Pakety s těmito příznaky totiž obsahují data, která potřebujeme pro klasifikaci OS. V případě HTTP paketů, které obsahují metodu GET nebo POST, lze informace o příslušném OS zjistit přímo z položky User-agent.

Kombinací příslušných hodnot z IP a TCP protokolu v tabulce 4.1 můžeme určit, o jaký se jedná operační systém. Většinu těchto hodnot používá pro klasifikace nástroj P0f (kapitola 3.1) nebo jsou uvedeny v literatuře [3]. Můžeme také využít skutečnosti, že ve velkém množství případů obsahuje TCP paket s příznaky SYN nebo SYN+ACK všechny uvedené hodnoty.

Protokol	Položka
IP	Initial Time To Live
IP	Don’t fragment
IP	SYN packet size
TCP	Max segment size
TCP	SackOK
TCP	Window size
TCP	No Operation
TCP	Window scale

Tabulka 4.1: Položky potřebné pro určení OS.

U hodnoty Initial TTL je nutné brát v potaz, že ji nastavuje odesílatel paketu. Tato hodnota se může lišit s tou, která přijde v paketu na sondu. To je způsobeno tím, že každý router, kterým projde takovýto paket, dekrementuje hodnotu TTL [10]. Z toho důvodu nemůžeme porovnávat referenční hodnotu TTL z podpisu přímo se zjištěnou hodnotou. Snažíme se určit, do jakého intervalu spadá zjištěná hodnota TTL a podle toho odhadneme její počáteční hodnotu.

Vojtěch Krmíček ve své disertační práci provádí porovnání TCP hlavičky a TCP/IP fingerprintingu u dnešních operačních systémů. Dnešní operační systémy se mimo jiné liší také ve velikosti SYN paketu a v poli Options, kde rozlišujeme počet NOP bytů (No Operation) [12].

Operační systém	SYN packet size	NOP
Windows Vista	52	3 x
Windows XP	48	2 x
Linux	60	1 x
FreeBSD	60	1 x
NetBSD	64	5 x
Mac OS	64	3 x (+ 2x eol)

Tabulka 4.2: Hodnoty SYN packet size a NOP [12].

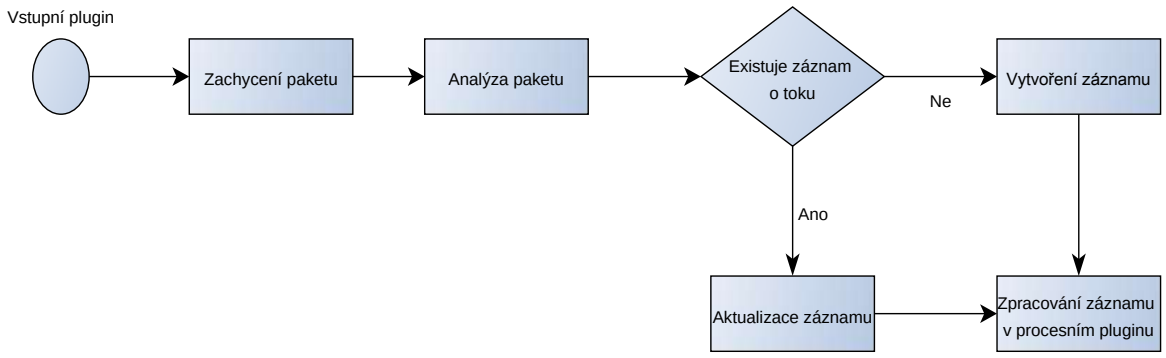
Při dalším zkoumání TCP paketů a hodnot v poli Options si můžeme všimnout, že operační systémy se liší také v hodnotě Window Scaling (tabulka 4.3).

Operační systém	Hodnota Win Scale
Windows XP Prof	None
Windows 7 Home	2
Windows 8.1 Prof	8
FreeBSD	3
RedHat	7
Ubuntu	7
Mac OS	4

Tabulka 4.3: Hodnoty Win Scale pro různé OS.

4.2.1 Rozdělení návrhu pluginu

Návrh samotného pluginu je rozdělen na dvě části, na plugin vstupní a plugin procesní. Na úrovní vstupního pluginu proběhne analýza paketu a jeho zpracovaní, tedy získání potřebných informací pro určení OS. Detekce OS probíhá pro každý tok, který zachytí sonda FlowMon. Proto je nutné na úrovni vstupního pluginu rozšířit flow record o veškeré hodnoty potřebné k určení operačního systému a dále také o informaci, o jaký OS se jedná, a o informaci, která nám říká, kolik položek bylo shodných s referenčním podpisem OS, tj. na kolik procent může uživatel důvěřovat informaci o OS. Analyzovat budeme pouze TCP pakety s příznaky SYN nebo SYN+ACK, které obsahují potřebné informace pro určení OS.



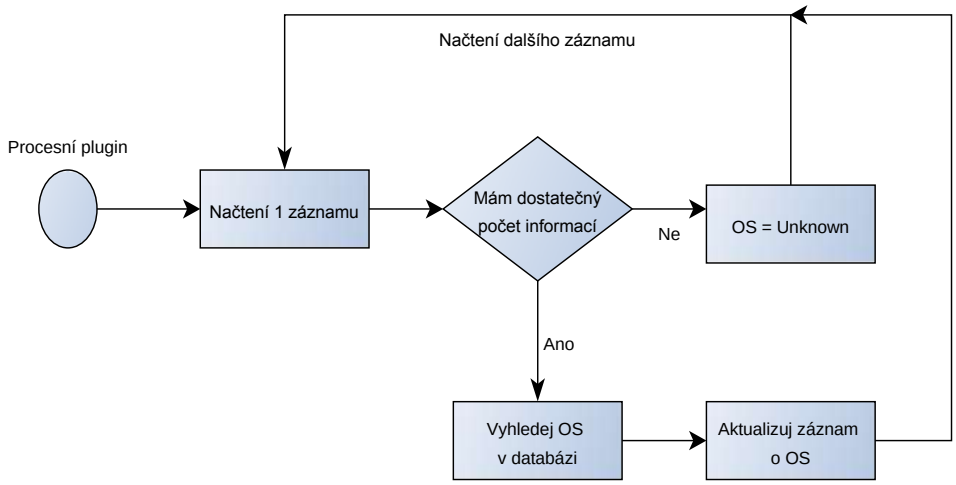
Obrázek 4.3: Vývojový diagram vstupního pluginu.

Nejdůležitější část detekce probíhá na úrovni procesního pluginu, kde dochází ke zpracování dat a vlastnímu určení operačního systému. Proto je nutné zde implementovat algoritmus pro detekci OS a také interní databázi operačních systémů, která bude obsahovat příslušné hodnoty z hlaviček IP a TCP protokolu (mluvíme zde o podpisu operačního systému, se kterým budou porovnávána data získaná ze síťového provozu) a samozřejmě bude obsahovat i konkrétní operační systém, který se vztahuje k těmto hodnotám a také údaj, který představuje počet shodných položek s podpisem v databázi.

Porovnávání naměřených hodnot s databází OS probíhá na základě vyhledávacího klíče, který má tento formát:

```
IP_TTL; IP_DF; IP_SYNlength; TCP_WinSize; TCP_MaxSegSize;
TCP_SackOK; TCP_Nop; TCP_WinScale;
```

V případě, že některá z hodnot není zjištěná z hlavičky paketu, bude na její pozici v klíči prázdná hodnota (např. `NULL, 0`). K určení OS je nutné mít k dispozici alespoň pět hodnot. Samozřejmě čím více hodnot budeme mít k dispozici, tím přesnější bude výsledek. Podle počtu shodných hodnot klíče a položek z databáze vrátí algoritmus zjištěný OS. Může ovšem nastat situace, kdy bude informace o zjištěném OS nejasná, tzn. že klíč odpovídá více než jednomu typu OS v databázi. V takovém případě rozhoduje položka z TCP paketu Window Scale, jelikož její hodnoty se mezi sebou vzájemně nejvíce odlišují. Pokud se nám ovšem nepodaří klasifikovat OS z IP/TCP hlavičky, můžeme se pokusit určit OS na základě údaje z HTTP, konkrétně z položky User agent.



Obrázek 4.4: Vývojový diagram procesního pluginu.

Jak můžeme vidět v tabulce 4.4, nelze přesně říci, jaké implicitní hodnoty má určitá položka v IP a TCP protokolu pro konkrétní operační systém. Tyto hodnoty nastavuje odesílatel, tedy sám operační systém, a mohou se lišit na základě verze OS nebo konkrétní distribuce. To znamená, že jeden konkrétní OS může mít více podpisů. Tabulka 4.4 obsahuje skutečně naměřené hodnoty pro jednotlivé operační systémy. Tučně označené hodnoty jsou ty, které jsou odlišné od hodnot udávaných v literatuře a zdrojích. Například Windows XP může mít dva podpisy. Jeden podpis bude mít hodnotu Maximum segment size 1440 (podle literatury) a druhý hodnotu 1460 (na základě skutečného měření).

	Windows XP	Windows 7 Home	Windows 8 Prof	Ubuntu, Redhat	FreeBSD	MAC OS
Time to live	128	**64**	128	64	64	64
DF (0x02)	Set	Set	Set	Set	Set	Set
SYN packet size	48	52	52	60	60	64
Maximum segment size	**1460**	**1452, 1460**	**1460**	1460	1460	1460
Window size	65535	8192	8192	**14600**	65535	65535
Selective acknowledge (kind 0x04)	Set	Set	Set	Set	**Set**	Not set
No operation (0x01)	2x	3x	3x	1x	1x	3x
Window scale	None (0)	2	8	3	7	4

Tabulka 4.4: Zjištěné hodnoty z IP a TCP hlaviček pro různé OS.

4.2.2 Rozšíření IPFIX záznamu

V tomto bodě plugin umí zjistit potřebné informace pro klasifikaci OS, dokáže OS klasifikovat a odeslat záznamy na kolektor pomocí protokolu IPFIX. IPFIX ovšem neví, jak zpracovat námi přidané informace, proto je nutné rozšířit IPFIX záznam právě o tyto položky. Tyto položky přidáme do šablony pro exportní plugin *ipfix-template-file.txt*. Po exportování záznamu na kolektor musíme dále upravit šablonu pro ipfixcol *ipfix-elements.xml*. Ipfixcol sice přijme rozšířený záznam, ovšem neví, jak má s těmito rozšířenými hodnotami pracovat. Na kolektor zasíláme veškeré údaje, tj. i data, podle kterých jsme klasifikovali operační systém, dále informaci o OS a také počet shodných položek. Samotná úprava definovaných šablon se provede pouze jednou.

Plugin jako celek je navržen na základě principu pasivního fingerprintigu tak, aby neodporoval technologii a architektuře sondy FlowMon. Využívá data získáná z IP a TCP hlavičky, případně položku User agent z HTTP hlavičky, pro klasifikaci OS. Zde můžeme využít znalosti, kdy víme, že TCP pakety s příznaky SYN nebo SYN+ACK s sebou nenesou HTTP protokol. Tudíž můžeme říci, že pokud nemáme TCP paket s příznaky SYN nebo SYN+ACK, můžeme se ihned pokusit klasifikovat OS na úrovni vstupního pluginu na základě protokolu HTTP, pokud bude k dispozici. V tomto případě není nutné využít algoritmus detekce OS z procesního pluginu.

5 Implementace pluginu pro sondu FlowMon

Sonda FlowMon pasivně monitoruje provoz v síti a je většinou zapojena na vstupním, výstupním bodu sítě nebo také na kritických místech, kde chceme monitorovat provoz a zde zpracovává procházející pakety. Plugin pro detekci OS neslouží primárně pro ochranu sítě, má spíše informativní charakter. Ovšem při dlouhodobé analýze operačních systémů můžeme detekovat NAT zařízení nebo neautorizovaná zařízení v naší síti.

Po získání všech potřebných informací a znalostí můžeme přejít k implementaci. Implementací, principy fungování pluginu a jeho výstupy se zabývá tato kapitola. Dále jsou zde také uvedena možná rozšíření a vylepšení pluginu. Vstupní i procesní plugin jsou implementovány v jazyce C.

5.1 Popis implementace pluginu

Implementace pluginu je rozdělena na dvě části a to na vstupní plugin a procesní plugin. Na úrovni vstupního pluginu dochází k analýze procházejících paketů. Snažíme se zde získat potřebné údaje pro následnou klasifikaci operačních systémů. Při spuštění pluginu nejprve dojde k inicializaci všech potřebných struktur a je zde provedena kontrola vstupů. Nastavení počátečních hodnot provádíme pro privátní strukturu **`osDet_input_private_t`**, která obsahuje data, se kterými budeme dále pracovat. Tato data jsou pak přístupná na úrovni celého pluginu. Na vstupu pluginu pak může být pcap soubor nebo rozhraní, na kterém chceme monitorovat provoz. O inicializaci se stará funkce **`plugin_input_init`**.

Po inicializaci a kontrole vstupů můžeme přejít ke zpracování dat. O zpracování dat se starají funkce **`plugin_input_get_flow`** a **`parse_data`**. Funkce **`plugin_input_get_flow`** načte paket, který předá dále ke zpracování funkci **`parse_data`**. Především vrací hash hodnotu, kterou dále využijeme pro vytvoření nebo aktualizaci záznamu v paměti sondy FlowMon.

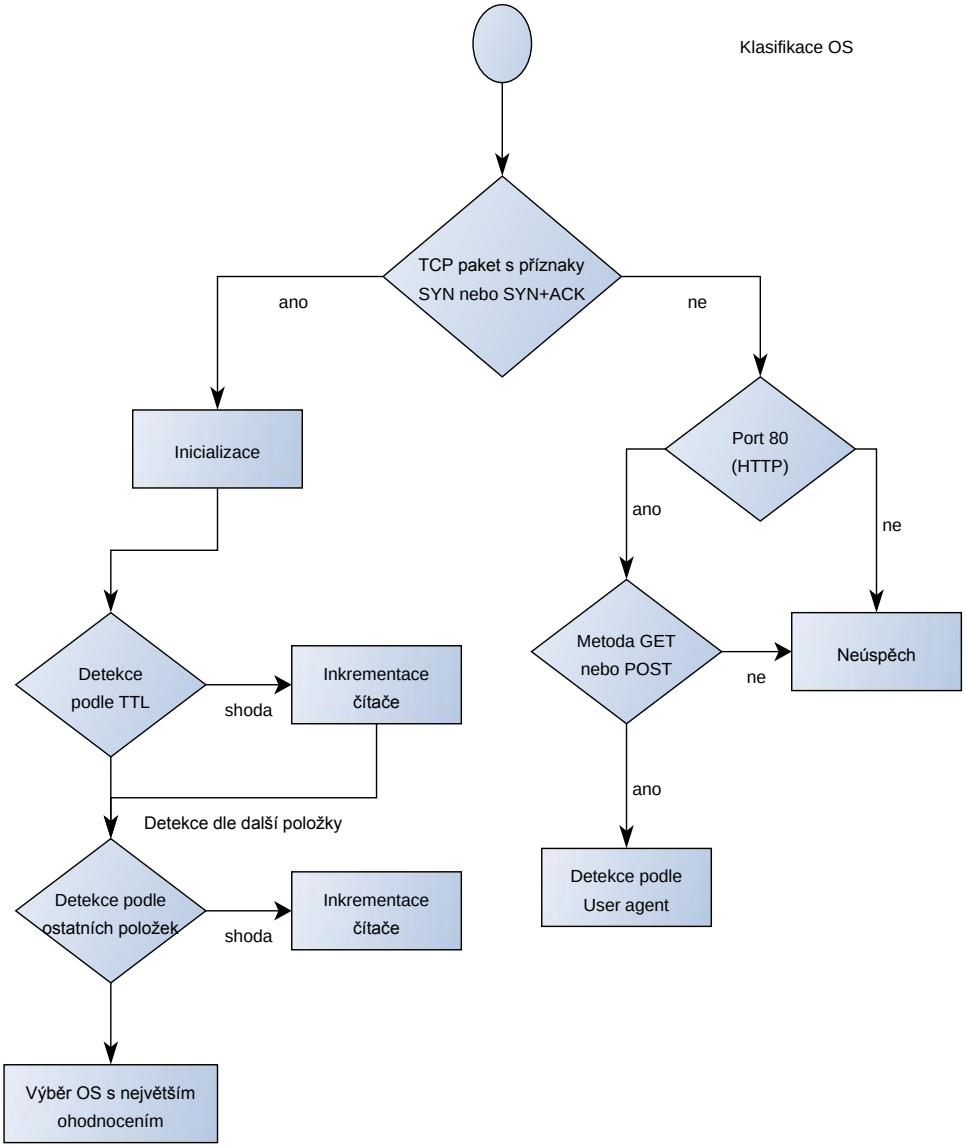
Funkce **`parse_data`** pak provádí vlastní zpracování paketu. Nejdříve provede kontrolu, zda jde o IP verze 4 a zda se jedná o TCP paket. Pokud ne, nemá smysl dále pokračovat, protože nemůžeme získat požadovaná data pro určení OS. Paket je dále rozdělen do dvou předpřipravených struktur: jedna slouží pro data z IP paketu, druhá pro data z TCP paketu. Tyto struktury jsou v hlavičkovém souboru, který přísluší ke vstupnímu pluginu. Zajímají nás pouze TCP pakety, které mají příznak SYN nebo SYN+ACK. V tomto bodu se plugin dělí na dvě části:

1) TCP paket s příznakem SYN nebo SYN+ACK – v tomto případě pokračujeme dál ve zpracování a snažíme se získat potřebné informace z TCP option a TCP window size.
2) TCP paket bez příznaku SYN nebo SYN+ACK – pokud je v tomto případě cílový port 80, tedy HTTP, snažíme se klasifikovat OS pomocí položky User agent z hlavičky protokolu HTTP. Pokud se nám podaří zde klasifikovat OS, v procesním pluginu už nedochází k jeho opětovné klasifikaci.

Veškeré informace, které jsme získali, uložíme do struktury **`osDet_record_t`**, která slouží pro rozšíření záznamu v paměti FlowMon cache.

Vlastní klasifikace operačních systémů probíhá podle schématu na obrázku 5.1. V případě, že nemáme k dispozici TCP paket s příznaky SYN nebo SYN+ACK, probíhá klasifikace OS na úrovni vstupního pluginu a využíváme k ní položku User agent protokolu HTTP. Zde je nutné, aby metoda

HTTP byla GET nebo POST. Pokud ale máme TCP paket s příznaky SYN nebo SYN+ACK, potom probíhá klasifikace OS na úrovni procesního pluginu. Samotná klasifikace probíhá na základě ohodnocení, kdy má každý OS v databázi na začátku každé klasifikace (jedna klasifikace vždy náleží k jednomu toku) toto ohodnocení nulové. Nejprve dojde k inicializaci všech potřebných pomocných proměnných a všechny OS v databázi mají ohodnocení nulové. Následně dochází k porovnání všech získaných dat s referenčními podpisy OS v databázi. Postupně se porovnávají jednotlivé sledované položky (TTL, DF, velikost SYN paketu, atd.) a v případě, že se shoduje údaj získaný z toku s údajem v databázi, dojde k inkrementaci ohodnocení konkrétního OS, ke kterému přísluší údaj z databáze.



Obrázek 5.1: Zjednodušené schéma klasifikace OS.

Potom stačí jen vybrat ten operační systém, který má toto ohodnocení nejvyšší. Pokud mají dva OS stejné ohodnocení, tak rozhoduje položka Windw scale, z TCP option. V položce Window scale se referenční podpisy sledovaných OS liší nejvíce. Výběr operačního systému s nejvyšším ohodnocením však neprobíhá pro všechny ohodnocené OS. Aby byl OS zařazen do výběru, musí dosáhnout jeho ohodnocení minimálně hodnoty šest, tzn., že musíme nalézt minimálně šest shodných položek údajů zjištěných ze vstupního pluginu s databázi OS v procesním pluginu. V opačném případě je OS

označen jako neznámý, tedy unknown. Stačilo by nám i pět položek, jak je uvedeno v knize Practical Packet Analysis using Wireshark to solve real-word network problems [3], ovšem čím více shodných položek máme, tím je klasifikace přesnější (více v kapitole 6.1).

Procesní plugin obsahuju jednoduchou statickou databázi OS, která představuje referenční podpisy jednotlivých OS v databázi a podle níž probíhá samotná klasifikace. V současné době není tato databáze určena k rozšíření bez zásahu programátora. V případě, kdy budeme chtít přidat do databáze nový podpis operačního systému, musíme to udělat ručně. Opět zde jako první probíhá funkce, která se stará o inicializaci a kontrolu vstupů, funkce **`plugin_process_init`**. V tomto případě toho ovšem nemá moc na práci, pouze alokuje paměť pro strukturu **`osDet_private_t`**. Vstup u procesního pluginu není žádný, veškerá data a informace získáme ze vstupního pluginu.

První paket patřící k určitému toku zpracovává funkce **`plugin_process_create`**, která dále volá funkci **`detect_os`**, která klasifikuje operační systém podle dat získaných ve vstupním pluginu. V případě, že se jedná o druhý a další paket určitého toku, tak zde tento paket zpracovává funkce **`plugin_process_update`**, kde dochází opět k volání funkce **`detect_os`**.

K záznamu ve FlowMon cache o operačním systému je také přidána položka, která nám říká, kolik bylo shodných položek při klasifikaci OS.

5.2 Výstup pluginu

FlowMon sonda představuje exportér, který má za úkol monitorovat síťový provoz a získaná data následně odesílá na kolektor. Na kolektoru následně probíhá sběr dat, analýza a vizualizace dat prostřednictvím webového rozhraní. V případě pluginu pro detekci operačních systémů máme dvě možnosti vizualizace výsledků:

1) Textové rozhraní – za pomoci nástroje fbitdump[11]: Zde máme opět dvě možnosti, jak budeme výsledky vizualizovat, respektive, můžeme si zvolit, jaké informace chceme na výstupu.

   a) Zkrácený výstup obsahuje IP adresu, OS a počet shodných položek.

Src IPv4	Operation system	Hits
192.168.10.101	MAC OS	7
192.168.10.101	MAC OS	7
31.13.81.128	Windows 7,8	6

Obrázek 5.2: Zkrácený výstup pomocí nástroje fbitdump.

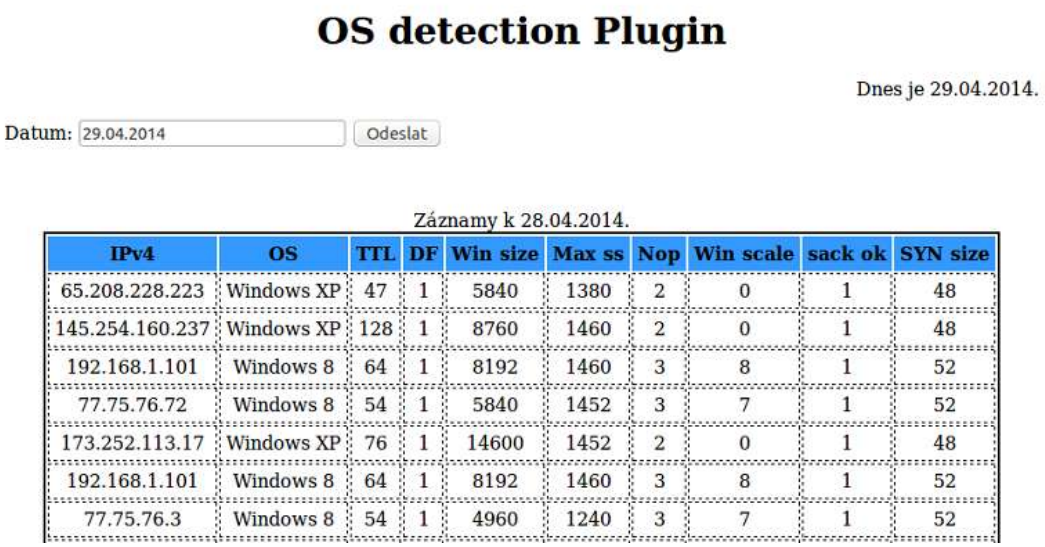
   b) Kompletní výstup obsahuje kromě IP adresy a OS i veškeré údaje, podle kterých probíhala klasifikace.

Src IPv4	Operation system	TTL	DF	Win size	Max ss	Nop	Win Scale	Sack OK	SYN packet size
192.168.1.101	Windows 8	64	1	8192	1460	3	8	1	52
77.75.76.72	Windows 8	54	1	5840	1452	3	7	1	52
173.252.113.17	Windows XP	76	1	14600	1452	2	0	1	48
192.168.1.101	Windows 8	64	1	8192	1460	3	8	1	52
77.75.76.3	Windows 8	54	1	4960	1240	3	7	1	52
77.75.76.3	Windows 8	55	1	12400	1240	3	7	1	52

Obrázek 5.3: Kompletní výstup pomocí nástroje fbitdump.

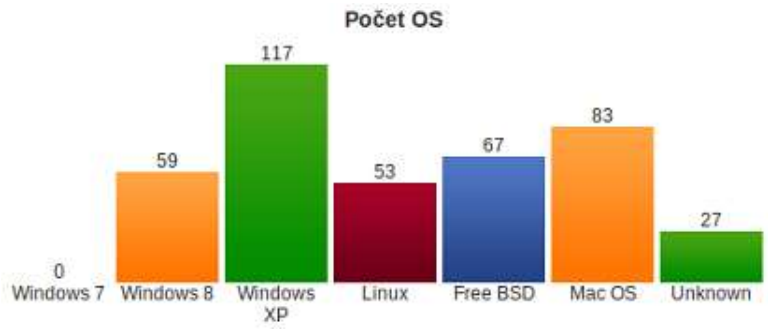
[11] Dostupné z: https://homeproj.cesnet.cz/rpm/liberouter/devel/x86_64/

2) Grafické rozhraní – za pomoci webového rozhraní: Jednoduchá webová stránka, implementovaná v php a html, která zobrazí tabulku s veškerými údaji, podle kterých jsme klasifikovali OS, a graf, který udává počet OS. Jak tabulka, tak i graf se vztahují k určitému dni, který zadá uživatel. Samozřejmě, pokud k zadanému dni nejsou žádné údaje, tak webová stránka nemůže zobrazit žádné výsledky a pouze informuju uživatele, že nenašla žádné záznamy k určitému dni.



IPv4	OS	TTL	DF	Win size	Max ss	Nop	Win scale	sack ok	SYN size
65.208.228.223	Windows XP	47	1	5840	1380	2	0	1	48
145.254.160.237	Windows XP	128	1	8760	1460	2	0	1	48
192.168.1.101	Windows 8	64	1	8192	1460	3	8	1	52
77.75.76.72	Windows 8	54	1	5840	1452	3	7	1	52
173.252.113.17	Windows XP	76	1	14600	1452	2	0	1	48
192.168.1.101	Windows 8	64	1	8192	1460	3	8	1	52
77.75.76.3	Windows 8	54	1	4960	1240	3	7	1	52

Obrázek 5.4: Grafický výstup - tabulka.



Obrázek 5.5: Grafický výstup - graf.

5.3 Možná rozšíření

Největší nevýhodou výsledného pluginu i všech ostatních nástrojů umožňujících detekci OS, je fakt, že jejich databáze OS je statická a v případě, že chceme přidat OS, jsme nuceni ručně upravit databázi OS, případně i upravit algoritmus detekce. Proto by bylo dobré, kdyby tato databáze byla dynamická. Tedy aby se sama učila nové podpisy neznámých operačních systémů z údajů získaných ze síťového provozu. V tomto případě je nutné získat podpis takového OS, ale především je nutné získat informaci o tomto OS, tedy zjistit, o jaký se jedná operační systém. Tady nám může pomoct například právě protokol HTTP a jeho položka User agent.

Další možné rozšíření je klasifikace OS nad protokolem IP verze 6. Tato úprava by vyžadovala samostatný výzkum. Jednak by bylo nutné získat referenční podpisy OS nad protokolem IP verze 6, dále by bylo nutné upravit samotnou databázi a také algoritmus detekce OS. Plugin provádí detekci operačních systémů pouze pro IP verze 4.

5.3.1 Metoda shlukové analýzy K-means

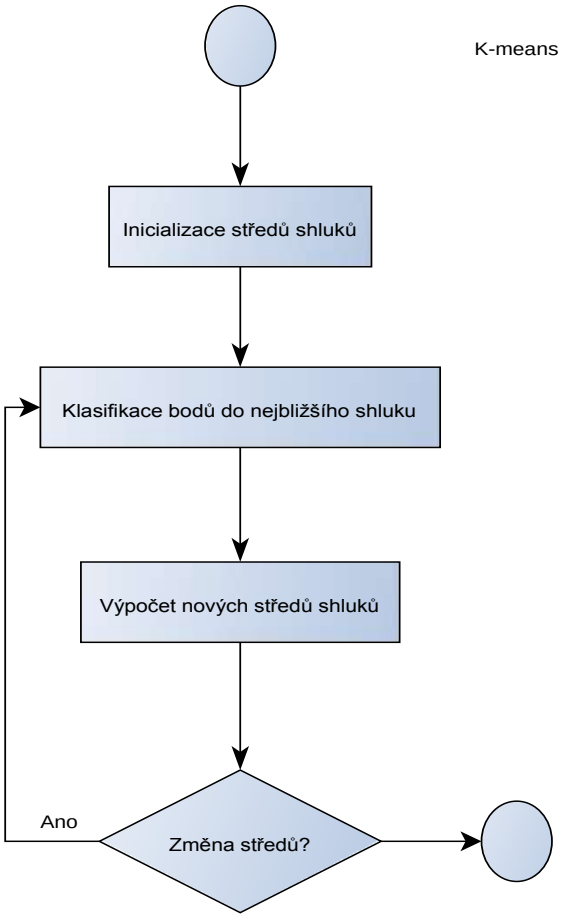
Při tvorbě dynamické databáze a algoritmu detekce, který by umožnil s touto databází pracovat, nám může pomoci například metoda shlukové analýzy K-means. Shluková analýza je statistická metoda a jejím cílem je rozdělení množiny objektů do *K* shluků na základě podobnosti vlastností těchto objektů. Objekty, u kterých je podobnost vlastností vysoká, budou ve stejné skupině (shluku). Naopak podobnost mezi objekty z různých skupin bude nízká. Shluková analýza se využívá především pro detekci anomálií a identifikaci charakteristických rysů n-dimenzionálních dat a jako metoda pro organizaci dat[13].

Metoda K-means je nehierarchický algoritmus, který třídí n-dimenzionální data do shluků na základě jejich vlastností. Na začátku je důležité znát tři parametry, které jsou [13]:

1) Počet shluků.
2) Inicializace středů shluků – musíme být schopni určit počáteční stav shluků.
3) Metrika vzdálenosti – jakým způsobem bude probíhat určení, ke kterému shluku náleží bod, např. Euklidovská vzdálenost.

K-means pak každý bod přiřadí do shluku, ke kterému má bod nejblíže. V případě, že došlo k přesunu některého z bodů do jiného shluku, algoritmus vypočítá nové středy shluků, například jako aritmetický průměr všech bodů ve shluku. Cílem tohoto algoritmu je pak dosáhnout co nejmenších rozdílů uvnitř shluku [13].

Pro použití algoritmu K-means pro detekci operačních systému by bylo ovšem nutné provést několik změn oproti základnímu algoritmu (obrázek 5.6). Inicializace středů shluků by neznamenalo nic jiného, než definování referenčních podpisů OS (jednotlivé středy shluků představují podpisy OS). Při klasifikaci bodů pak dochází k určení příslušnosti OS k nebližšímu shluku. Zde je nejdůležitější metrika, podle které tato klasifikace probíhá. V podstatě je pouze na nás, jakou metriku, jaký způsob, zvolíme. Musíme mít na paměti, že údaje, které jsme získali ze síťového provozu, jsou velmi odlišné. Např. Window size u operačních systémů rodiny Windows mohou teoreticky nabývat hodnot 1 – 65535. Nebo hodnoty Window size v porovnání s ostatními, např. s DF, kdy tato položka nabývá pouze dvou hodnot, jsou rozdíly obrovské a referenční podpisy by pak byly od sebe příliš vzdáleny a klasifikace by nebyla přesná. Toto vše musíme při návrhu metriky vzít v úvahu. Výpočet nových středů pak není nic jiného, než přidání nových podpisů. Největším přínosem metody K-means může být dynamická databáze, která bude schopná učit se nové podpisy operačních systémů sama, bez zásahu programátora. Další výhodou může být i to, že při použití tohoto rozšíření bude vždy OS přiřazen ke shluku, ke kterému bude mít nejblíže. Tedy vždy bude přiřazen k nějakému operačnímu systému.



Obrázek 5.6: Princip algoritmu K-means.

Implementace pluginu pro sondu FlowMon je rozdělena na vstupní plugin a procesní plugin. Na úrovni vstupního pluginu dochází k analýze a zpracování procházejících paketů. V případě, že nemůže získat potřebné informace (TCP paket nemá příznaky SYN nebo SYN+ACK), snažíme se klasifikovat operační systém na základě položky User agent protokolu HTTP. Klasifikace OS v pluginu probíhá pro:

- Každý zachycený tok,
- IP verze 4,
- TCP pakety s příznaky SYN nebo SYN+ACK,
- případně probíhá klasifikace z položky User agent protokolu HTTP.

Na úrovni procesního pluginu se nachází statická databáze OS a probíhá zde klasifikace pomocí dat získaných ve vstupním pluginu. Největší nevýhodou pluginu je právě jeho statická databáze. Tato nevýhoda by však mohla jít odstranit pomocí navrženého rozšíření.

6 Testování

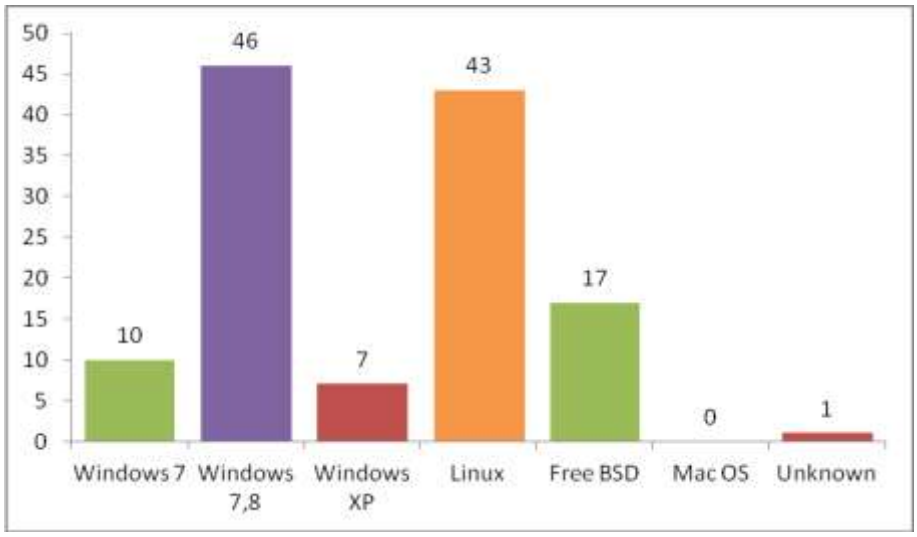
Nedílnou součástí každé aplikace je její testování, které si klade za cíl otestovat aplikaci před jejím nasazením. Testování závisí na druhu aplikace, pro jaký účel byla vytvořena, pro koho byla vytvořena a kdo s ní bude pracovat. V našem případě aplikace slouží pro detekci OS, analýzu síťového provozu. Pracovat by s ní měl administrátor nebo správce sítě. Testování probíhalo především na datech z pcap souborů. Nebylo nutné plugin rozsáhle testovat na reálném provozu, jelikož data, která potřebuje pro svoji práci, nejsou závislá na čase nebo jiných vlastnostech specifických pro reálný provoz. Plugin tvoří z větší části analyzátor dat (vstupní plugin) a algoritmus detekce (procesní plugin), který provádí pouze porovnávání. Režie spojená s během pluginu tedy není velká.

Data pro testování jsem získal především monitorováním různých sítí (laboratoře FIT VUT, koleje, domácí síť) a také monitorováním provozu na počítačích s různými operačními systémy, nejčastěji za pomoci nástroje Wireshark a tcpdump. Při testování jsem využil také databázi P0f pro kontrolu dosažených výsledků.

Testování probíhalo po celou dobu práce. Jednak bylo nutné testovat správnost získaných dat z pcap souborů a ověřit správnost referenčních podpisů. Dále bylo nutné otestovat běh samotného pluginu, zda se chová podle návrhu a pracuje tak, jak má.

6.1 Přesnost algoritmu detekce OS

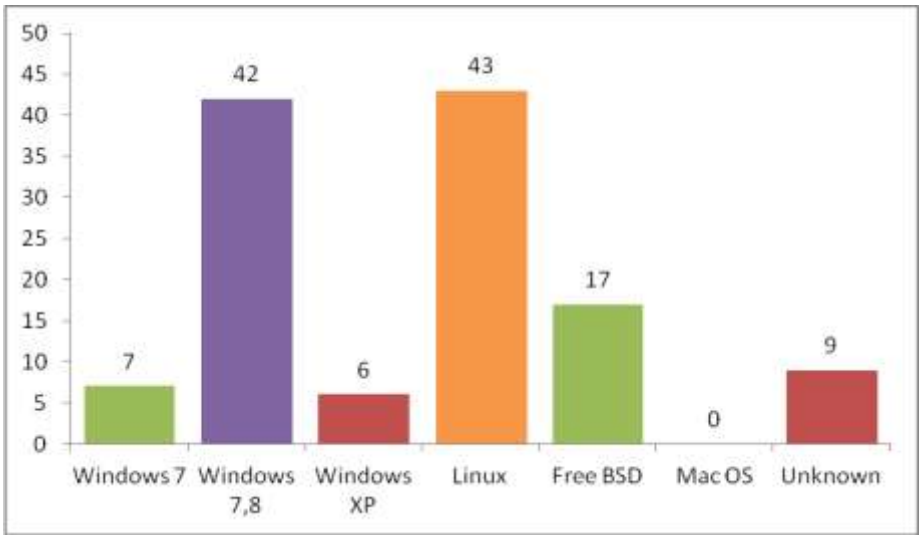
Jak jsem uvedl v kapitole 5.1, pro určení OS nám stačí znát pět z osmi sledovaných hodnot. Plugin je nastaven na šest hodnot z důvodu větší přesnosti klasifikace. Hodnota, která určuje, v kolika položkách musí nastat shoda s referenčním podpisem OS v databázi, je velmi důležitá a jak můžeme vidět níže (obrázek 6.1, 6.2 a 6.3), tak samozřejmě ovlivní i přesnost detekce OS.



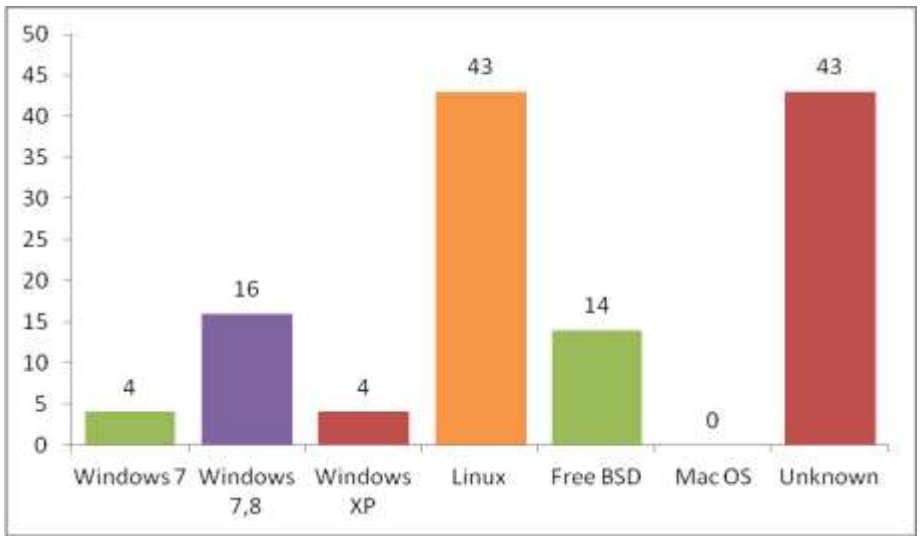
Obrázek 6.1: Přesnost algoritmu - 5 položek.

Jak můžeme vidět na obrázcích 6.1, 6.2 a 6.3, tak největší rozdíl v klasifikaci při různé přesnosti je především u operačních systémů Windows. To je způsobeno jednak tím, že OS rodiny Windows mají největší odchylky od referenčních podpisů v databázi OS, ale také, že nemají pevně danou hodnotu Window size. Ta může nabývat hodnot teoreticky od 1 do 65535. Dále také může způsobit problém

to, že podpisy některých verzí OS jsou stejné, například tomu tak je u některých verzí Windows 7 a Windows 8.



Obrázek 6.2: Přesnost algoritmu - 6 položek.



Obrázek 6.3: Přesnost algoritmu - 7 položek.

Přesnost	Windows 7	Windows 7/8	Windows XP	Linux	FreeBSD	Mac OS	Unknown
5 položek	10	46	7	43	17	0	1
6 položek	7	42	6	43	17	0	9
7 položek	4	16	4	43	14	0	43
Skutečný počet OS[12]	30	21	9	43	17	0	4

Tabulka 6.1: Porovnání přesnosti algoritmu při různém počtu shodných položek.

[12] Sloupec Windows 7 představuje výskyt všech OS Windows 7 v testovacím souboru. Sloupec Windows 7/8 představuje výskyt všech OS Windows 8 v testovacím souboru.

Tabulka 6.1 porovnává přesnost algoritmu při různém počtu shodných položek společně se skutečným výskytem OS v testovacím souboru. Při porovnání výsledků testů různé přesnosti shody se skutečným výskytem operačních systémů vidíme, že největší rozdíl je u klasifikace OS rodiny Windows, především od verze 7 dále.

6.2 Podpisy OS Windows 7 a Windows 8

Při testování pluginu jsem narazil na fakt, kdy operační systémy Windows 7 a Windows 8 měly stejné podpisy. Tato shoda nastává u systémů Windows 7 Home (32 bit) a Windows 8 u více verzí. Ostatní systémy Windows 7 pak mají svůj jedinečný podpis. Bylo tedy nutné upravit databázi. Pokud nejsme schopni rozlišit, zda se jedná o Windows 7 nebo Windows 8, tak je v tomto případě OS klasifikován jako Windows 7/8. Ostatní verze Windows 7 mají jedinečný podpis, tudíž zde není s klasifikací problém. Stejné řešení je použito i u databáze nástroje P0f, přičemž databáze tohoto nástroje sleduje podstatně více informací, ovšem stejně nedokáže rozhodnout, o jaký OS se jedná.

Rozdíl mezi podpisy u Windows 8 a ostatními verzemi Windows 7 je také minimální. Ovšem liší se nám vždy alespoň v jedné hodnotě a to navíc v hodnotě Window scale. Tato hodnota je u každého podpisu v databázi téměř jedinečná. Tabulka 6.2 obsahuje vzorové podpisy operačních systémů Windows 7 a 8 a můžeme vidět, že podpis Windows 7 Home je identický s podpisem Windows 8. Ostatní operační systémy Windows 7 se od Windows 7 Home a Windows 8 liší v hodnotě Window scale.

Operační systém	Podpis (TTL; DF; SYN size; Win size; Max ss; Nop; Win scale; sACK)
Windows 7 Home	128; SET; 52; 8192; (1440, 1460); 3x; 8; SET;
Windows 8	128; SET; 52; 8192; (1440, 1460); 3x; 8; SET;
Windows 7 ostatní	(64, 128); SET; 52; 8192; (1440, 1460); 3x; 2; SET;

Tabulka 6.2: Porovnání podpisů OS Windows 7 a Windows 8.

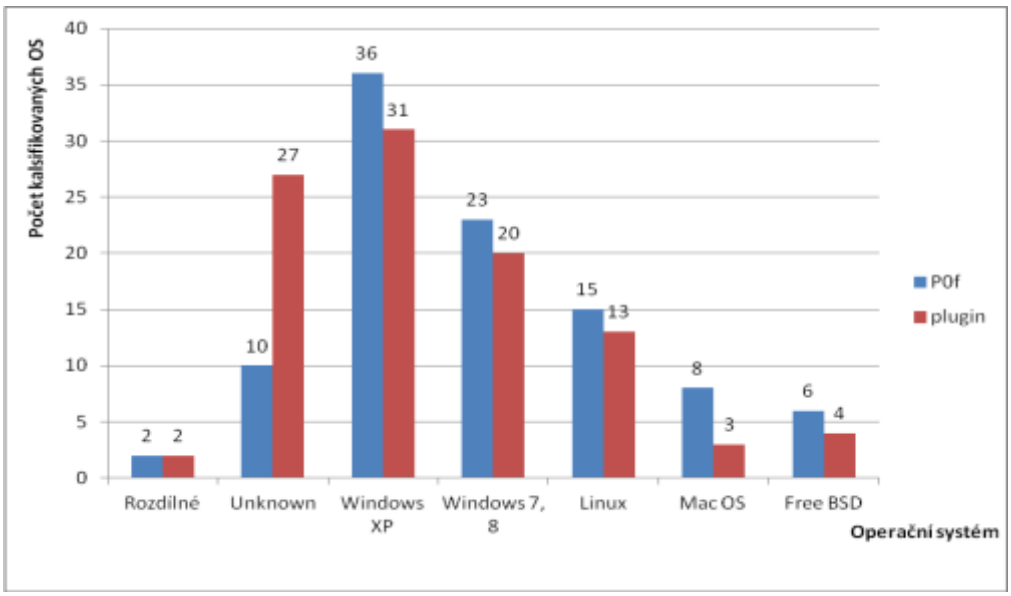
6.3 Porovnání detekce OS u p0f a pluginu

Plugin implementovaný pro sondu FlowMon je navržen na základě pasivního fingerprintingu a nástroje p0f, který využívá právě tuto metodu k detekci operačních systémů. Proto se nabízela možnost porovnat klasifikaci OS právě mezi nástrojem p0f a samotným pluginem. Nástroj p0f má v porovnání s pluginem rozsáhlou databázi OS, která obsahuje velké množství podpisů různých OS, a sleduje mnohem více položek. Proto toto testování probíhalo pouze na datech, kde by se měly objevit OS, které obsahuje databáze pluginu pro sondu FlowMon.

Jako testovací soubor posloužil pcap soubor, který obsahoval pakety od různých operačních systémů. Jak aplikace p0f, tak i plugin pro sondu FlowMon zpracovávají vstupy sekvenčně, tedy paket po paketu, tudíž jsem se mohl zaměřit například na prvních sto výsledků. Při analýze prvních sto výsledků se plugin pro sondu FlowMon a nástroj p0f lišily pouze ve dvou případech (graf 6.1):

1) Plugin určil OS jako Windows 7, p0f jako neznámý.
2) Plugin určil OS jako Free BSD, p0f jako Linux 2.2.

Samozřejmě nástroj p0f při klasifikaci OS poskytuje podrobnější informace a je také přesnější. To je dáno především tím, že sleduje na třicet položek a k jednomu konkrétnímu operačnímu systému má v některých případech k dispozici několik podpisů a jeho databáze je velice rozsáhlá.



Graf 6.1: Porovnání klasifikace OS u nástroje p0f a pluginu pro sondu FlowMon.

6.4 Testovací program

Pro zjednodušení práce a lepší zpracování zdrojových souborů vznikl program, který zpracovává pcap soubory a je schopný klasifikovat OS. Prochází hlavičky paket po paketu a získává potřebné informace a následně určí, o jaký se jedná OS. Operační systém vždy přiřadí k IP adrese. Pokud nemá dostatek informací, nebo se zjištěná data neshodují s žádným operačním systémem v databázi, program určí jako OS Unknown, tedy neznámý OS.

Veškerý zdrojový kód byl nejdříve implementován a otestován před jeho nasazením na sondu FlowMon v rámci tohoto testovacího programu. Stejně tak byly testovány i referenční podpisy. Jediný rozdíl mezi pluginem a tímto programem je, že plugin přidá informaci o OS k toku, kdežto testovací program ke zdrojové IP adrese.

```
IP: 173.194.113.68      OS: Unknown     ttl: 50  df: 2  synsize 0       win size 0
IP: 173.194.70.125      OS: Unknown     ttl: 44  df: 2  synsize 0       win size 0
IP: 192.168.20.1        OS: MAC OS      ttl: 64  df: 1  synsize 64      win size 65535
IP: 31.13.81.128        OS: Windows 8   ttl: 82  df: 1  synsize 60      win size 14480
IP: 81.91.84.180        OS: Free BSD    ttl: 53  df: 1  synsize 60      win size 5792
IP: 173.194.113.78      OS: Unknown     ttl: 50  df: 2  synsize 60      win size 42540
IP: 68.232.35.139       OS: Free BSD    ttl: 50  df: 1  synsize 60      win size 14480
IP: 185.31.17.185       OS: Free BSD    ttl: 47  df: 1  synsize 60      win size 14280
IP: 173.194.70.84       OS: Unknown     ttl: 43  df: 2  synsize 60      win size 42540
IP: 8.37.70.21  OS: Free BSD    ttl: 45  df: 1  synsize 60      win size 14480  max ss
IP: 23.209.127.139      OS: Free BSD    ttl: 52  df: 1  synsize 60      win size 14480
IP: 23.62.237.104       OS: Free BSD    ttl: 54  df: 1  synsize 60      win size 14480
IP: 23.209.114.110      OS: Free BSD    ttl: 52  df: 1  synsize 60      win size 14480
```

Obrázek 6.4: Výstup testovacího programu.

Testování probíhalo po celou dobu práce na této bakalářské práci. Nejdříve bylo nutné otestovat, zda jsou data získaná na úrovni vstupního pluginu správná a validní, tedy zda získáváme požadované informace pro klasifikaci operačních systémů. Také bylo nutné otestovat, zda je vstupní plugin schopen zpracovat oba vstupy, tedy vstup z pcap souboru a vstup z určeného rozhraní. Na úrovni procesního pluginu pak bylo nezbytné zjistit, zda algoritmus detekce OS funguje a je schopný klasifikovat OS na základě údajů, které mu poskytuje vstupní plugin. Dále bylo nezbytné otestovat, zda referenční podpisy na úrovni procesního pluginu jsou opravdu podpisy těch operačních systémů, ke kterým jsou přiřazeny v rámci databáze. Veškeré testování probíhalo tak, že jsem porovnával výstup dílčí části pluginu s předem známým vzorkem dat. Tedy dopředu jsem věděl, jak má vypadat výstup jednotlivých částí. Plugin všemi testy prošel. Při testování správnosti databáze mi pomohl především nástroj p0f a jeho databáze, s kterou jsem mohl porovnat dosažené výsledky klasifikace z pluginu pro sondu FlowMon.

7 Závěr

Cílem této práce bylo vytvořit plugin pro sondu FlowMon, který bude schopen klasifikovat OS na základě síťového provozu. V teoretické části práce bylo nutné seznámit se s protokolem IPFIX, protože jedním z úkolů bylo rozšířit IPFIX záznam o potřebné informace, které mají být přenášeny po síti. Dále bylo nezbytné seznámit se s možnostmi detekce operačních systémů v síti. Zde jsem se setkal s pojmem fingerprinting, který se dále dělí na aktivní a pasivní. Výhodou aktivního fingerprintingu je, že si můžeme v podstatě říci o data a informace, které potřebujeme pro určení operačního systému. To přináší zároveň nevýhodu. Musíme posílat speciálně upravené pakety a následně zpracovávat jejich odpovědi, což zbytečně zatěžuje síť. Také je tento způsob detekce v rozporu s technologií a architekturou sondy FlowMon, která je transparentní. Tudíž tato metoda nemohla být použita. Naopak pasivní fingerprinting je vhodný pro detekci OS na sondě FlowMon. Toto řešení s sebou nese mírnou nevýhodu v tom, že musíme čekat do té doby, než přijdou správné pakety s potřebnými informacemi. Z nich lze následně určit, o jaký OS se jedná. V případě TCP paketů je tento čas zanedbatelný, jelikož velká část síťového provozu je tvořena právě těmito pakety. U DHCP by to ale mohl být problém, jelikož doba zápůjčky IP adresy je většinou různá a může překračovat několik dní. V neposlední řadě bylo nutné porozumět a pochopit technologii a architekturu sondy FlowMon, pro kterou je výsledný plugin určen.

V praktické části práce jsem na základě poznatků a informací zjištěných v teoretické části mohl navrhnout a implementovat plugin pro sondu FlowMon. Tento plugin je rozdělen na dvě části: na část vstupní a procesní. Na úrovni vstupního pluginu získáváme potřebné informace pro klasifikaci OS a na úrovni procesní pluginu pak probíhá vlastní detekce OS. Bylo nutné také výsledný plugin otestovat. Testování probíhalo především na pcap souborech. Největší nevýhodou výsledného pluginu je jeho databáze. Tato databáze je statická, tudíž v případě, kdy ji budeme chtít rozšířit, to musíme provést ručně, tedy někdo musí nový podpis přidat. Tuto nevýhodu by mohlo vyřešit možné rozšíření, tedy použít pro klasifikaci upravený algoritmus K-means.

Během práce jsem se detailně seznámil s možnosti detekce operačních systému v síťovém provozu, dále jsem se seznámil s protokolem IPFIX a sondou FlowMon, která poskytuje zajímavé řešení monitorování sítě.

Na tématu detekce operačních systémů v síťovém provozu chci dále pokračovat. Především bych rád dotáhl do konce myšlenku dynamické databáze, tedy takové databáze, která je schopna sama zjistit a správě přiřadit operační systém k neznámému podpisu. Dynamickou databázi zatím neposkytuje žádný nástroj věnující se klasifikace OS v síti.

S touto bakalářskou prací jsem se zúčastnil studentské soutěže EEICT pořádanou fakultami FIT a FEKT VUT v Brně a obsadil jsem 2. místo v kategorii Komunikace a počítačové sítě.

Literatura

[1] Claise, B., Trammell, B., Aitken, P. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011. Září 2013.

[2] Sadasivan, G., Brownlee, N., Claise, B., Quittek, J. Architecture for IP Flow Information Export. RFC 5470. Březen 2009.

[3] Sanders, Ch. Practical Packet Analysis using Wireshark to solve real-word network problems. No Starch Press, druhé vydání, červenec 2011, s. 194-197. ISBN 978-1-59327-266-1.

[4] Forensic Wiki: OS fingerpinting. [online], [cit. 2013-10-22].
Dostupné z: http://www.forensicswiki.org/wiki/OS_fingerprinting

[5] Nmap: Remote OS detection. [online], [cit. 2014-04-14].
Dostupné z: http://nmap.org/book/osdetect.html

[6] Fielding, R., Gettys, J., Mogul, J., et al. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616. Červen 1999.

[7] INVEA-TECH: FlowMon. [online], [cit. 2013-10-21].
Dostupné z: http://www.invea.cz/products/flowmon

[8] INVEA-TECH: FlowMon sonda. [online], [cit. 2013-10-21].
Dostupné z: https://www.invea.com/cs/produkty-sluzby/flowmon/flowmon-sondy

[9] CISCO: NetFlow Overview. [online], únor 2004, [cit. 2013-12-16]
Dostupné z: http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_presentation0900aecd80311f57.pdf

[10] Postel, J. Internet Protocol Specification. RFC 791. 1981.

[11] Nmap: Reasons for OS detection. [online], [cit. 2013-12-10].
Dostupné z: http://nmap.org/book/osdetect.html#osdetect-reasons

[12] Krmíček, V. Hardware-Acceleratd Anomaly Detection in High-Speed Networks. [online], 2011, [cit. 2013-12-17], disertační práce.

[13] Jain, K., A. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters.* 2010, roč. 31, č. 8, s. 653-655. ISSN 0167-8655.

Seznam příloh

A Obsah DVD

- Readme.txt – textový soubor obsahující manuál.
- BP_Martin_Vymlatil.pdf – text bakalářské práce ve formátu PDF.
- BP_Martin_Vymlatil_text.doc – text bakalářské práce ve formátu doc.
- \osDet – adresář obsahující zdrojové soubory pluginu
- \IPFIX – adresář obsahující šablony a soubory nutné pro rozšíření IPFIX záznamu
- \web – webová stránka pro grafickou vizualizaci výsledků

B Manuál

Pro spuštění pluginu musíme mít sondu FlowMon verze 3.05 nebo vyšší. Jako první musíme přeložit zdrojové soubory pluginu. To provedeme příkazem `make` na úrovni složky `osDet`. Plugin pak můžeme spustit příkazem:

```
sudo flowmonexp -X /home/flowmon/ipfix_export/flowmon-export-ipfix.so -X ./process/process-osDet.so -I input-osDet:pcap_file=srcFile -P process-osDet -E ipfixx:host=localhost,port=4739
```

Při volbě vstupu pluginu máme dvě možnosti:

- Vstup ze soubor - pcap_file=file
- nebo vstup z rozhraní - pcap_if=interface

V tomto stavu nám plugin poskytuje na výstup pouze informace o tom, jaký OS právě klasifikoval. Abychom dostávali lepší a přesnější informace, musíme využít buďto textový výstup pomocí nástroje fbitdump nebo grafický výstup pomocí webové stránky.
V tomto případě je nutné zkopírovat soubory ze složky IPFIX na sondu FlowMon:

1) Soubor `ipfix-template-file.txt` do `/etc/flowmon/`,
2) Soubor `ipfix-elements.xml` do `/etc/ipfixcol/`,
3) Soubor `fbitdump.xml` do `/usr/share/fbitdump/`.

Dále je nutné zapnout před spuštěním pluginu ipfixcol příkazem `Ipfixcol –v2`.
Nyní můžeme využít textový výstup pomocí nástroje fbitdump. Použitím přepínače –o osdet dostaneme na výstupu kompletní informace včetně dat podle kterých probíhala klasifikace OS. Použitím přepínače –o osdetIP dostane na výstupu pouze IP adraseu, OS a počet shodných položek.

Pro grafický výstup je nutné provést kroky 1) – 3) uvedené v předchozím odstavci. Navíc musíme zkopírovat obsah adresáře `web` do `/var/www/shtml/community/`.
Nyní máme k dispozici webovou stránku dostupnou přes webové rozhraní sondy FlowMon.

Další možnosti:

- Pokud chceme vypnout klasifikaci OS pomocí protokolu HTTP a jeho položky User agent, tak musíme v `input-osDet.c` změnit hodnotu makra `HTTP` z `true` na **`false`**.
- Databáze OS dostupná v procesním pluginu obsahuje podpisy jedenácti operačních systémů. Bezpečně otestované OS jsou Windows XP, 7 a 8, Linux, FreeBSD, Mac OS a Android 4.x. Z toho důvodu je databáze implicitně nastavena tak, aby klasifikace probíhala pouze s podpisy těchto OS. Pokud chceme do klasifikace OS zařadit i podpisy ostatních OS (Symbian, Palm OS, NetBSD, OpenBSD), tak stačí v `process-osDet.c` změnit makro `DB` z hodnoty `7` na hodnotu **`11`**.

C Databáze OS

OS	Podpis operačního systému							
	TTL	DF	SYN size	Win size	Win scale	Max ss	sACK	Nop
Windows XP	128	SET	48	variable	0	1440 1460	SET	2x
Windows 7	64 128	SET	52	variable	2	1440 1460	SET	3x
Windows 7,8	128	SET	52	variable	8	1440 1460	SET	3x
Linux	64	SET	60	2920-5840 14600	3	1460	SET	1x
FreeBSD	64	SET	60	65535	7	1460	SET	1x
Mac OS	64	SET	64	65535	4	1460	Not	3x
Android 4.x	64	SET	52 | 60	65535	6	1460	SET	3x
Symbian *	255	Not	44	8192	0	1460	Not	0
Palm OS *	255	Not	44	16384	0	1350	Not	0
NetBSD *	64	SET	64	32768	0	1416	Not	5x
OpenBSD *	64	SET	64	32768	0	1440	SET	5x

Tabulka C.1: Databáze OS.

Položky označené symbolem * nejsou otestované na dostatečném množství dat. V implicitním stavu databáze bere v úvahu prvních sedm položek z tabulky C.1. V případě, kdy budeme chtít, aby byla k dispozici celá databáze, tak stačí změnit makro `DB` ve zdrojovém souboru `process-osDet.c` z hodnoty 7 na hodnotu 11.