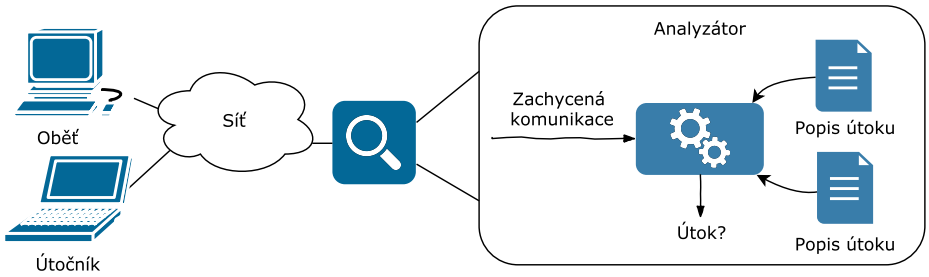
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2018
http://excel.fit.vutbr.cz

Detekce síťových útoků pomocí jejich deklarativního popisu

Jindřich Dudek*



Abstrakt
Tento článek se zabývá návrhem nástroje pro detekci síťových útoků ze zachycené síťové komunikace, který ke své činnosti využívá paketový analyzátor tshark. Význam paketového analyzátoru spočívá v převedení vstupního souboru se zachycenou komunikací do textového formátu PDML, přičemž účelem této konverze je flexibilnější zpracování vstupních dat. Při návrhu nástroje je kladen důraz na rozšiřitelnost o detekci nových síťových útoků a jejich snadnou integraci. Z tohoto důvodu je součástí článku také navržení obecného deklarativního zápisu síťových útoků v serializačním formátu YAML. Ten umožňuje specifikovat klíčové vlastnosti síťových útoků a podmínky pro jejich detekci. Výsledný nástroj tedy funguje jako interpret deklarativních zápisů, čímž je umožněna jeho snadná rozšiřitelnost o nové typy útoků.

Klíčová slova: detektor síťových útoků — deklarativní zápis síťových útoků — tshark — bezpečnost sítí — síťové útoky

Přiložené materiály: *N/A*

*xdudek04@stud.fit.vutbr.cz, *Fakulta informačních technologií, Vysoké učení technické v Brně*

1. Úvod

Počítačové sítě patří v dnešní době k neodmyslitelné součásti lidské populace. Kromě nespočtu benefitů, které nám nepochybně poskytují, je ovšem nezbytné řešit otázku zabezpečení sítí vůči potenciálním síťovým útokům [2], či jiným pokusům o narušení jejich bezproblémového chodu. Mnoho síťových útoků bylo již objeveno a podrobně popsáno, nicméně stále se objevují další zranitelnosti, které jsou příčinou vzniku nových útoků, proti kterým je nutné hledat detekční a obranné mechanismy vedoucí k jejich eliminaci. Společně s rozvojem Internetu věcí (IoT) také stále vzrůstá počet nových zařízení se síťovou konektivitou, u kterých mohou být opomenuty bezpečnostní aspekty, díky čemuž mohou být zneužity již známé zranitelnosti, případně jejich obdoby a tato zařízení se tak stávají novými potenciálními cíli síťových útoků. Z tohoto důvodu je nezbytné se neustále přizpůsobovat nově vznikajícím hrozbám a hledat nové a efektivní metody, které by umožňovaly jednoduchý způsob jejich detekce.

Tato práce si klade za cíl popsat návrh nástroje pro detekci různých síťových útoků ze zachycené síťové komunikace. Hlavním požadavkem, na který je při návrhu kladen důraz, je možnost co nejsnazší rozšiřitelnosti tohoto nástroje o nové síťové útoky, které je schopen nástroj detekovat. Je tedy nezbytné navrhnout takové rozhraní nástroje, které by tento požadavek

dokázalo naplnit bez nutnosti výrazných změn v jeho implementaci.

2. Existující řešení

Obdobnou funkcionalitu jako navrhovaný nástroj rovněž poskytují tzv. systémy pro detekci průniku (IDS) [1], které pro detekci síťových útoků používají tzv. signatury, což je strukturovaný soubor pravidel, pomocí nichž je možné útok identifikovat. Pokud dojde k objevení nového typu útoku, dodavatel systému musí vytvořit novou signaturu a provést aktualizaci IDS, který je pak schopen nový útok detekovat. Na základě složitosti vytvořené signatury je pak typicky v případě komplexnějších signatur účtován poplatek, zatímco ty jednodušší jsou obvykle distribuovány zdarma.

Kromě signatur jsou pro detekci síťových útoků využívány i pokročilejší metody, které jsou založeny na vytváření dlouhodobých statistik o síťovém provozu, či využití umělé inteligence pro získání modelu obvyklého síťového provozu. V případě odchylky od získaných statistik či naučeného modelu je pak zahlášena detekce útoku. Tento princip je označován jako *anomaly based detection*.

V rámci porovnání IDS a nástroje popisovaného v tomto článku je nezbytné zmínit, že nástroj nemá ambice stát se plnohodnotnou náhradou IDS systémů, ale pouze doplňující alternativou vhodnou například v případech, kdy finanční prostředky vynaložené na koupi signatury jsou nerentabilní a pro detekci síťového útoku postačují prostředky poskytované nástrojem.

Existující řešení dostupná na trhu jsou často zaměřena na detekci pouze určitých typů útoků (např. útoky typu DDoS, detekce malware, webové útoky, atp.) a pro detekci útoků často kromě síťového provozu vyžadují i logy různých služeb, změny v souborovém systému, či registry operačního systému. Detekci síťových útoků typicky provádějí v reálném čase, což ovšem vyžaduje větší množství času a prostředků pro úspěšnou detekci útoků a tento fakt se obvykle negativně podepisuje na propustnosti v případě vysokorychlostních sítí. Výhodou tohoto přístupu je ovšem velmi rychlá detekce útoku.

Navrhovaný nástroj detekuje útoky pouze na základě informací získaných ze zachycené síťové komunikace, díky čemuž je možné provádět forenzní analýzu komunikace na libovolném zařízení, nikoliv pouze na stroji, jehož komunikace je analyzována. Další výhodou může být fakt, že analýza komunikace není prováděná v reálném čase, což vede k nižšímu využití prostředků a omezení vlivu na síťovou propustnost. Využití tohoto přístupu ovšem vede k pozdější detekci útoku od okamžiku jeho provedení.

Konkrétními příklady produktů dostupných na trhu mohou být nástroje *Snort*[1], či *Suricata*[2]. *Snort* [6] detekuje síťové útoky pomocí souboru pravidel, která jsou vyhodnocena pro každý přijatý či odeslaný paket a v případě jejich splnění je zahlášena detekce útoku. Pravidla obsahují informaci, zda mají být aplikována na přijaté, či odeslané pakety, dále informace o zdrojovém a cílovém portu a následně specifikaci, jaká data má daný paket obsahovat, aby byl označen za pozitivní detekci. Nástroj je tedy schopný detekovat pouze útoky, které jsou detekovatelné z jednoho paketu.

Oproti těmto produktům se popisovaný nástroj snaží navrhnout framework, který by umožňoval uživatelsky přívětivější specifikaci síťových útoků a zároveň poskytl prostředky pro detekci různých typů útoků, k jejichž detekci mohou být zapotřebí složitější postupy, než pouze kontrola obsahu jednotlivých paketů.

3. Návrh aplikace

Při návrhu nástroje je zapotřebí zohlednit fakt, že nástroj musí poskytovat co nejsnazší rozšiřitelnost o nové typy útoků bez nutnosti provádět výrazné změny v jeho implementaci. Z tohoto důvodu je nevhodné, aby detekce každého útoku byla napevno implementována v kódu aplikace, což by vedlo k nutnosti neustále rozšiřovat kód programu při potřebě detekovat nové síťové útoky. Naopak je zapotřebí proces detekce útoků co nejvíce zobecnit.

Tento požadavek je řešen pomocí textových deklarativních zápisů, které strukturovaně popisují způsob detekce jednotlivých útoků. Je tedy nezbytné navrhnout obecný způsob zápisu, který by poskytoval prostředky pro popis detekce širší škály síťových útoků. Deklarativní popis rozděluje síťové útoky na tři typy. Jednotlivé typy se od sebe navzájem odlišují postupem, který je aplikován při samotném vyhledávání útoku ve vstupním souboru se zachycenou komunikací. Jejich bližšímu popisu se věnuje sekce 4.

Nástroj pak funguje jako interpret těchto popisů a na jejich základě jsou v předloženém souboru se zachycenou síťovou komunikací vyhledávány konkrétní síťové útoky. V případě výskytu nového typu útoku si pak může uživatel nástroje vytvořit nový popis, který je nástroj schopen automatizovaně zpracovat, aniž by bylo zapotřebí provádět jeho implementační úpravy.

V deklarativních zápisech je často zapotřebí nastavit specifické parametry, které mohou mít pro každou síť odlišné hodnoty. Z tohoto důvodu je vhodné, aby každý síťový administrátor měl vlastní databázi deklarativních popisů, ve kterých jsou tyto parametry nas-

[1] Viz. https://www.snort.org/
[2] Viz. https://suricata-ids.org/

taveny tak, aby v dané síti docházelo k minimalizaci nekorektních detekcí. Tato databáze může být založena na již existujících deklarativních zápisech (s případně přizpůsobenými parametry), nebo může obsahovat nové popisy vytvořené síťovým administrátorem.

Při samotném procesu detekce je nejdříve vstupní soubor se zachycenou síťovou komunikací převeden do textového formátu *Packet Description Markup Language* (PDML) pomocí paketového analyzátoru `tshark` [3]. Formát PDML je založen na značkovacím jazyku XML a díky této konverzi je tedy možné vstupní soubor zpracovat pomocí libovolné knihovny pro zpracování XML dokumentů a není tak nezbytné implementovat disketory jednotlivých síťových protokolů. Nástroj `tshark` zároveň jednoznačně specifikuje jména polí v paketu, díky čemuž je možné se na ně odkazovat v deklarativních popisech útoků. Další výhodou, kterou tato konverze poskytuje, je možnost přesně specifikovat, která pole paketu mají být ve výstupu uvedena a stejně tak je možné filtrovat i pakety, které jsou pro detekci klíčové. Tím je možné výrazně zmenšit velikost výstupního souboru a zvýšit rychlost samotné detekce útoků.

Konvertovaný soubor je poté procházen pomocí knihovny pro zpracování XML souborů a na základě deklarativních popisů jsou v něm vyhledávány specifikované síťové útoky. Výše uvedený postup je graficky ilustrován na obrázku 1.

Při vysokém počtu paketů ve vstupním souboru dosahují konvertované výstupy ve formátu PDML velikostí v řádu gigabytů. Při výběru knihovny pro zpracování XML souborů je tedy nezbytné brát tento fakt v potaz a vybírat takové knihovny, které poskytují vysokou rychlost zpracování a umožňují iterativní načítání XML dokumentů. Pro procházení jednotlivých paketů v PDML souboru je pak možné využít například jazyk XPath.

4. Formát pro deklarativní popis útoků

Kvůli snadné rozšiřitelnosti nástroje o nové typy síťových útoků je nezbytné navrhnout obecný deklarativní formát, který poskytuje prostředky pro popis široké škály síťových útoků. Deklarativní zápisy útoků jsou realizovány pomocí serializačního formátu YAML, mezi jehož vlastnosti patří především dobrá čitelnost jak člověkem, tak i strojem, která je v tomto případě zásadní.

Deklarativní popis rozděluje síťové útoky celkem do tří typů, které se od sebe navzájem liší postupem, jenž je aplikován při vyhledávání útoku ve vstupním souboru se zachycenou síťovou komunikací. Názvy těchto typů útoků jsou následující:

- *atomic*;
- *stream*;
- *group*.

Následující sekce jsou věnovány podrobnému popisu prostředků, které jednotlivé typy útoků poskytují pro tvorbu deklarativních zápisů síťových útoků, včetně postupů, které jsou aplikovány při jejich zpracování.

4.1 Útok typu atomic

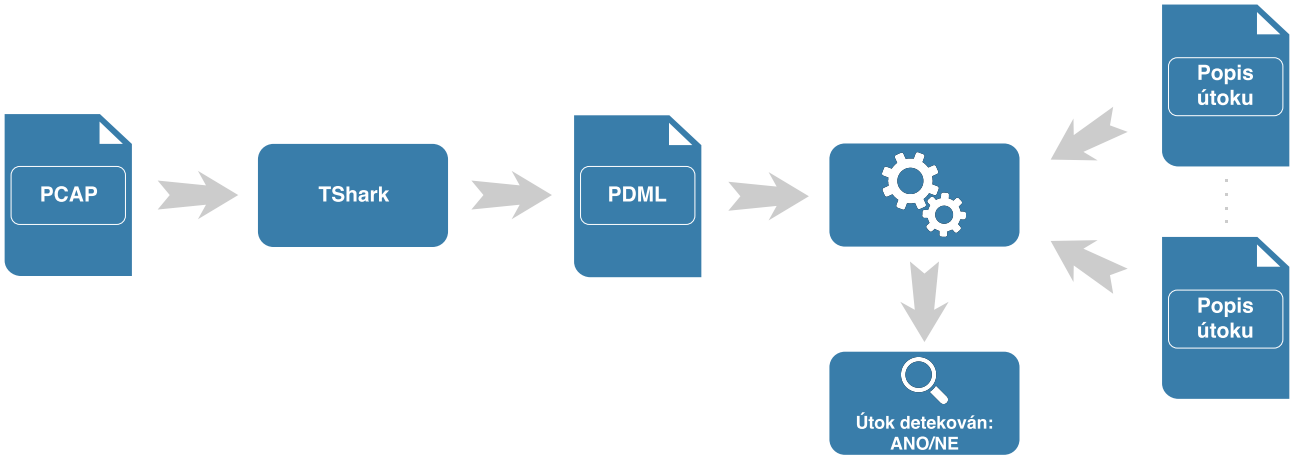
Typ útoku s označením *atomic* se vyznačuje tím, že pro jeho detekci postačuje procházet jednotlivé pakety v zachycené síťové komunikaci a u každého z nich jsou ověřovány uživatelem specifikované podmínky, jež daný útok identifikují. Pro tento typ útoku je podstatné, že pro jeho odhalení není důležité, v jakém síťovém toku se zkoumané pakety vyskytují. Z toho vyplývá, že pro korektní detekci útoku není zapotřebí zjišťovat, jaké další pakety jsou obsaženy ve stejném síťovém toku, v němž se vyskytuje aktuálně zkoumaný paket. Obecně platí, že pro úspěšnou detekci útoku může být zapotřebí více paketů, nicméně tyto pakety mohou být součástí různých síťových toků. Pro identifikaci síťového útoku je v tomto typu útoků možné využít čtyři následující podmínky:

- *field-name*;
- *field-value*;
- *field-count*;
- *packet-ratio*;
- *expression*.

Pomocí podmínky *field-name* lze specifikovat výskyt konkrétního pole v paketu (v tomto případě se kontroluje pouze výskyt, nikoliv jeho hodnota). Při zadání *field-value* se kromě výskytu pole detekuje i jeho hodnota. Pokud je tedy dané pole v paketu nalezeno (a případně odpovídá jeho specifikovaná hodnota), je paket označen za pozitivní detekci. Podmínka taktéž umožňuje zjistit celkový počet různých paketů, které mají stejnou (nebo naopak rozdílnou) hodnotu specifikovaného pole a pokud tento počet překročí uživatelem určenou hodnotu, je zahlášena pozitivní detekce síťového útoku.

Další podmínka typu *field-count* se opět týká konkrétního pole v paketu, v tomto případě se ale neřeší jeho samotný výskyt nebo hodnota, ale počet výskytů pole v paketu. Pokud tedy počet výskytů specifikovaného pole v paketu odpovídá uživatelem určené hodnotě, je paket označen za pozitivní detekci.

Pomocí podmínky typu *packet-ratio* lze specifikovat poměr dvou různých typů paketů. Uživatel určí jejich vlastnosti, na základě kterých jsou tyto pakety vyfiltrovány. Následně je zjištěn počet obou typů paketů,



Obrázek 1. Grafické znázornění procesu detekce síťových útoků navrhovaným nástrojem. Vstupní soubor se zachycenou síťovou komunikací je převeden pomocí nástroje `tshark` do formátu PDML, ve kterém jsou pak vyhledávány síťové útoky na základě textových deklarativních popisů těchto útoků.

ze kterého je vypočítán jejich poměr. Pokud poměr překročí uživatelem stanovenou hodnotu, je zahlášena pozitivní detekce.

Poslední typ podmínky s názvem *expression* umožňuje vyhodnotit uživatelem zadaný booleovský výraz ve formě textového řetězce. Ve výrazu je možné používat vlastní proměnné, u kterých je ovšem zapotřebí uvést, jaká hodnota se má za proměnnou dosadit při vyhodnocování výrazu. Většinou se jedná o konkrétní hodnotu nějakého z polí v paketu, pro který je aktuálně výraz vyhodnocován, nicméně je možné použít i hodnoty, které se týkají více paketů (např. celkový počet paketů v souboru, počet různých hodnot určitého pole napříč všemi pakety apod.). Jednotlivé pakety jsou tedy postupně procházeny a specifikovaná podmínka je pro každý z nich vyhodnocena. Pokud je podmínka pro daný paket pravdivá, je možné paket označit za pozitivní detekci.

Všechny výše uvedené podmínky je možné libovolně kombinovat za účelem definování co nejpřesnějších detekčních podmínek. Pokud je součástí deklarativního zápisů více různých podmínek, lze navíc specifikovat, zda musí platit všechny současně, nebo postačuje platnost pouze jedné z nich. Uživatel rovněž musí specifikovat i počet pozitivních detekcí, které musí být ve vstupním souboru nalezeny, aby bylo vypsáno příslušné hlášení o nalezení útoku.

Síťovým útokem, který se dá tímto typem popsat je např. Local Area Network Denial (LAND). Deklarativní zápis útoku je uveden v příkladu 1. Princip tohoto útoku spočívá v zaslání TCP paketu s nastaveným příznakem SYN a shodnou cílovou i zdrojovou IP adresou, která je nastavena na IP adresu oběti. Při přijetí tohoto paketu na stanici oběti se oběť pokouší ustavit TCP spojení sama se sebou (kvůli nastavenému SYN příznaku), což v některých případech může vést k zamrznutí, čí pádu systému [7].

```
1  name: LAND
2  scope: atomic
3  properties:
4    - element-name: tcp.flags
5      value: '0x00000002' #SYN paket
6  detection-conditions:
7    type: and
8    conditions:
9      - condition-type: expression
10       expression: SrcIp == DstIp
11       variables:
12         - name: SrcIp
13           type: field-value
14           element-name: ip.src
15         - name: DstIp
16           type: field-value
17           element-name: ip.dst
18 threshold-error: 1
```

Příklad 1. Deklarativní zápis síťového útoku Local Area Network Denial (LAND).

Útok je možné detekovat procházením jednotlivých paketů s nastaveným příznakem SYN a kontrolovat, zda zdrojová a cílová IP adresa jsou nastaveny na stejnou hodnotu. V případě deklarativního zápisu jsou nejdříve pomocí pravidla uvedeného v sekci *properties* (řádek č. 3) vyfiltrovány všechny pakety s nastaveným příznakem SYN. Následně je pro každý takový paket vyhodnocen výraz uvedený pomocí atributu *expression* (řádek č. 10). Kolekce *variables* pak pro každou proměnnou ve výrazu specifikuje, jaká hodnota se má za proměnnou dosadit. V tomto případě se jedná o konkrétní hodnotu polí s názvy *ip.src* a *ip.dst*.

Počet pozitivních detekcí je pak porovnán s hodnotou atributu *threshold-error* (řádek č. 20). Pokud je tento počet větší, nebo roven hodnotě tohoto atributu, je vypsáno chybové hlášení o pozitivní detekci útoku.

4.2 Útok typu stream

Útok typu *stream* je specifický tím, že síťové útoky jsou v tomto typu útoku vyhledávány v obousměrné komunikaci mezi dvěma síťovými entitami, identifikovanými prostřednictvím IP adres, na konkrétních portech, pomocí transportního protokolu TCP, či UDP. Pakety jsou tedy před samotným vyhledáváním síťových útoků rozčleněny do odpovídajících TCP, či UDP streamů na základě indexu streamu, který do paketů automaticky doplňuje nástroj `tshark`. Tyto streamy jsou následně skenovány za účelem nalezení síťového útoku. Detekce útoku probíhá na základě výskytu uživatelem specifikovaných paketů v jednotlivých streamech.

Síťovým útokem tohoto typu je např. skenování portů pomocí TCP paketů s nastaveným příznakem SYN. Deklarativní zápis tohoto útoku je uveden v příkladu 2:

```
1  name: Port Scanning - SYN/TCP
2  scope: stream
3  properties:
4    - element-name: tcp
5  packets-specification:
6    follow: true
7    specified-only: false
8    specification:
9      - count: 1
10       packet-properties:
11         - element-name: tcp.flags
12           value: '0x00000002'#SYN
13     - count: 1
14       packet-properties:
15         - element-name: tcp.flags
16           #RST+ACK:
17           value: '0x00000014'
18 threshold-warning: 200
19 threshold-error: 800
```

Příklad 2. Deklarativní zápis síťového útoku skenování portů metodou SYN/TCP.

Princip tohoto skenování spočívá v zaslání TCP paketu s nastaveným příznakem SYN na konkrétní port stanice oběti. Pokud je v reakci na tuto zprávu přijat paket s nastavenými příznaky SYN a ACK, jedná se o otevřený port, v případě přijetí paketu s nastavenými příznaky RST a ACK pak o port uzavřený [5].

Útok lze detekovat prohledáváním jednotlivých TCP streamů a hledáním dvojice paketů s nastavenými příznaky SYN a RST+ACK (tzn. streamy, kde je cílový port uzavřen). V případě zápisu uvedeného v příkladu 2 jsou tedy nejdříve pomocí pravidla uvedeného v kolekci *properties* (řádek č. 3) vyfiltrovány všechny pakety obsahující pole s názvem *tcp* (tzn. všechny TCP pakety). Díky tomuto kroku jsou odfiltrovány pakety, které jsou pro detekci útoku nepodstatné. V této fázi zpracování tedy nedochází k samotnému rozčleňování jednotlivých paketů do streamů, ale pouze se vyfiltrují pakety obsahující TCP hlavičku, aby při následném vytváření TCP streamů nebyly zbytečně procházeny nepodstatné pakety.

Následně jsou vyfiltrované pakety rozděleny do odpovídajících TCP streamů, které jsou postupně procházeny a je v nich hledána dvojice paketů s nastavenými příznaky SYN a RST+ACK. Specifikaci těchto paketů je možné najít v kolekci *specification* (řádek č. 8), kde je pro každý typ paketu uvedeno, pomocí atributu *count*, kolikrát se má ve streamu vyskytovat. Následuje kolekce *packet-properties*, ve které už jsou pro každý typ paketu uvedeny jeho vlastnosti, které jej jednoznačně identifikují. V tomto případě je paket s příznakem SYN identifikován prostřednictvím hodnoty pole *tcp.flags* nastaveným na `0x00000002`. Obdobně je identifikován i paket s nastavenými příznaky RST+ACK.

Atribut *follow* nastavený na hodnotu `true` pak indikuje, že všechny specifikované pakety musí v daném TCP streamu následovat bezprostředně za sebou. Pomocí atributu *specified-only* nastaveného na hodnotu `false` je určeno, že kromě specifikovaných paketů se mohou v TCP streamu vyskytovat i jiné pakety.

Pokud jsou v některém TCP streamu nalezeny specifikované pakety a splňují všechny požadované podmínky, je takový stream označen za pozitivní detekci. Celkový počet pozitivních detekcí je pak porovnán s hodnotami atributů *threshold-warning* a *threshold-error* (řádky č. 17 a 18). Pokud počet pozitivních detekcí překročí hodnotu prvního z uvedených atributů, je vypsáno varovné hlášení. Pokud dokonce přesáhne hodnotu druhého uvedeného atributu, vypíše se chybové hlášení, které má vyšší váhu. Pokud je počet pozitivních detekcí menší než hodnota obou uvedených atributů, není zobrazeno žádné hlášení.

4.3 Útok typu group

Útok typu *group* je zobecněním útoku typu *stream*. Obdobně jako v tomto typu útoku jsou útoky vyhledávány v určité skupině paketů, nicméně nedochází

zde ke třídění paketů dle jejich příslušnosti do TCP, či UDP streamu. Uživatel v tomto případě musí specifikovat seznam polí, na základě jejichž společné hodnoty jsou pakety umístěny do identické skupiny. Specifikovaný seznam polí je tedy postupně procházen a pokud je v paketu dané pole nalezeno, je paket umístěn do skupiny s identifikátorem odpovídajícímu hodnotě nalezeného pole. Pokud pole nalezeno není, je pokračováno dalším polem v seznamu. V případě, že se v paketu nevyskytuje žádné ze specifikovaných polí, je takový paket ignorován.

Identicky jako v předchozím typu útoku uživatel specifikuje jednotlivé pakety, jejichž výskyt indikuje přítomnost síťového útoku. Tyto pakety jsou pak ve skupinách vyhledávány a v případě jejich nalezení a splnění všech specifikovaných podmínek je skupina paketů označena za pozitivní detekci. Celkový počet detekcí je opět porovnán s uvedenými prahy a případně je ohlášeno nalezení útoku.

5. Optimalizace paměťových nároků

Při zpracování souborů formátu PDML je nutné řešit problémy spojené s jejich značnou velikostí, která v mnohých případech dosahuje hodnoty několika gigabytů. Obrovská velikost konvertovaného PDML souboru oproti vstupnímu souboru je způsobena především tím, že PDML obsahuje již interpretovaná data. Kromě samotných hodnot polí paketu tedy obsahuje i názvy těchto polí a další informace interpretované paketovým analyzátorem. Hodnoty jednotlivých polí jsou navíc často uvedeny v několika různých formátech současně (např. ve formě raw bytů, textových řetězů, hexadecimálního zápisu bytů, apod.). Určitý nárůst velikosti je rovněž spojen s vlastnostmi jazyka XML, na kterém je formát PDML založen. Mezi ně patří například nutnost ke každé párové značce specifikovat její ukončující značku apod. V paketech jsou také často přenášena obsáhlá aplikační data, která jsou pro detekci síťových útoků nepodstatná, ale mají výrazný vliv na velikost zpracovávaného souboru.

Při zpracování těchto souborů pomocí knihoven pro parsování XML dokumentů navíc jejich velikost v paměti několikanásobně vzroste, kvůli nutnosti uchování informací o návaznosti jednotlivých XML značek. Z těchto důvodů je prakticky nereálné zpracovávat soubory o vyšší velikosti s využitím současných výpočetních prostředků čistě v operační paměti bez využití diskového uložiště, což činnost nástroje značně zpomaluje. Z tohoto důvodu je nezbytné velikost souborů snížit pomocí několika optimalizací.

První optimalizací je redukce počtu polí v paketu pouze na ta pole, která jsou nezbytná pro úspěšnou detekci útoku. Před samotnou konverzí jsou tedy postupně procházeny všechny deklarativní zápisy útoků a je vytvořen seznam nezbytných polí. Nástroj `tshark` následně umožňuje specifikovat, která pole mají být zahrnuta v konvertovaném výstupu, díky čemuž dojde nejen k redukci výsledné velikosti souboru, ale i k výraznému zrychlení samotné konverze.

Velikost vstupu [MB]	Počet paketů [tis.]	Velikost PDML [MB]	Velikost opt. PDML [MB]	Zmenšení velikosti [%]
3,4	29	320	28	91,3
56	94	2200	104	95,3
98	142	3500	156	95,5
153	193	4100	192	95,3
202	228	5000	256	94,9
275	282	6400	315	95,1
352	402	8300	450	94,6
453	443	9800	498	94,9

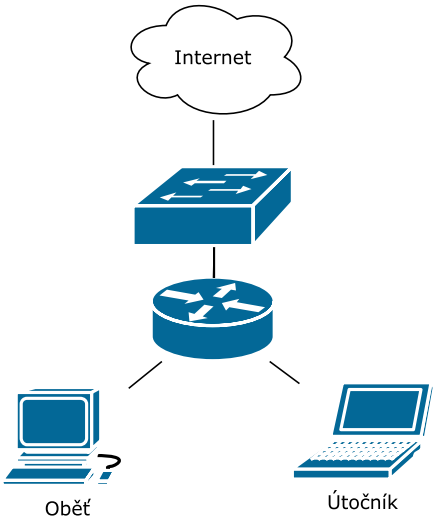
Tabulka 1. Porovnání velikostí výstupů konverze do formátu PDML před a po aplikování optimalizací.

Dále je možné výstupní velikost konvertovaného souboru optimalizovat pomocí filtru, jenž umožňuje specifikovat, které pakety mají být zahrnuty ve výstupu konverze. Tento filtr je předán jako parametr nástroji `tshark` před samotnou konverzí a jeho syntaxe je identická jako u display filterů[3] využívaných v nástroji `Wireshark`. Filtry jsou vytvořeny na základě informací uvedených v kolekci *properties* v deklarativních zápisech útoků (viz. příklad 2, řádek č. 3). Díky tomuto kroku je zajištěno, že ve výstupním konvertovaném souboru nebudou přítomny pakety, které jsou pro detekci útoků zbytečné.

Tyto optimalizace zajišťují i určitou variabilitu v použití nástroje, pokud zařízení, na kterém je nástroj spouštěn, disponuje nižší kapacitou operační paměti. Pokud je zapotřebí snížit paměťovou náročnost nástroje, je možné ho spouštět vícekrát po menších počtech deklarativních zápisů. Díky výše popsaným optimalizacím se tak redukuje celkový počet polí zpracovávaných při jednom běhu nástroje, stejně tak jako celkový počet paketů, které jsou pro detekci síťových útoků zapotřebí. To vede ke snížení paměťových nároků nástroje, což je často důležité při zpracování vstupních souborů s větším počtem zachycených paketů.

V tabulce 1 je uvedeno porovnání velikostí výstupů konverze do formátu PDML před a po aplikování výše uvedených optimalizací. První dva sloupce určují velikost vstupního souboru a počet paketů v něm obsažených. Třetí a čtvrtý sloupec udává velikost PDML souboru bez optimalizací a s optimalizacemi. Poslední

[3] Viz. https://wiki.wireshark.org/DisplayFilters



Obrázek 2. Topologie sítě při realizaci síťových útoků za účelem zachycení síťové komunikace.

sloupec pak udává procentuální rozdíl ve velikostech. Z tabulky je možné určit, že velikost výstupního souboru byla redukována přibližně o 95 %.

6. Testování

Testování nástroje probíhalo celkem ve třech fázích, přičemž první dvě fáze byly věnovány testování funkčnosti nástroje a třetí testům výkonnosti. V první fázi došlo k testování, zda útoky popsané v deklarativních zápisech jsou korektně detekovány. K tomu bylo zapotřebí získat soubory se zachycenou síťovou komunikací, v niž jsou útoky přítomny. Soubory s některými síťovými útoky bylo možné získat z různých veřejných databází na internetu, nicméně většinu útoků bylo zapotřebí reálně provést a zachytit pomocí některého z paketových analyzátorů. K tomu bylo využito virtuální testovací prostředí vytvořené pomocí nástroje `VirtualBox` a reálné domácí sítě. Topologie domácí sítě je vizualizována na obrázku 2. Pro samotné generování útoků bylo využito množství volně dostupných nástrojů, či paketový manipulátor Scapy[4] pro jazyk Python. Samotné testování probíhalo tak, že nástroji byl jako vstup předložen soubor obsahující daný síťový útok (potvrzeno pomocí paketového analyzátoru *Wireshark*) a deklarativní zápis, který tento útok popisuje. Následně bylo ověřeno, zda skutečně došlo ke korektní detekci útoku.

V druhé fázi bylo zapotřebí otestovat tzv. false positives [4], což jsou případy, ve kterých je zahlášena pozitivní detekce útoku i při normální síťové aktivitě. Pro tento účel bylo z veřejných internetových databází staženo množství souborů obsahujících zachycený reálný síťový provoz, ve kterých se rozsah paketů pohyboval v rozmezí od pěti tisíc do pěti set tisíc. Na základě provedených testů pak docházelo k úpravám či rozšířením formátu pro deklarativní zápis útoků tak, aby k případným false positives nedocházelo.

[4]Viz. http://www.secdev.org/projects/scapy/

Velikost vstupu [MB]	Počet paketů [tis.]	Doba zpracování [s]	Max. využití oper. paměti [MB]
3,4	29	19,5	358
56	94	88	1350
98	142	136	1900
153	192	191	2600
202	228	372	3150
275	282	441	3800
352	402	507	5500
453	443	691	6100

Tabulka 2. Porovnání doby běhu a paměťové náročnosti nástroje pro různé velikosti vstupních souborů se zachycenou síťovou komunikací.

Třetí fáze byla věnována testování výkonnosti implementovaného nástroje. V tabulce 2 je uvedeno porovnání doby běhu a paměťových nároků nástroje pro různou velikost vstupních souborů se zachycenou síťovou komunikací. V rámci jednoho běhu nástroje musí dojít ke konverzi vstupního souboru do formátu PDML. Následně musí být pomocí knihovny *cElementTree* iterativně zpracovány jednotlivé pakety z výstupu konverze. V poslední části je pak vyhodnoceno dvacet tři deklarativních popisů útoků. Hodnoty v tabulce 2 byly naměřeny na zařízení s procesorem Intel Core i5-3230M (2.6/3.2 GHz) a operační pamětí 8GB.

Z tabulek 1 a 2 je patrné, že na současných průměrných počítačích je možné analyzovat vstupní soubory až o velikostech přibližně půl gigabytu, přičemž vyhodnocení dvaceti tří různých síťových útoků na takto velkém vstupu zabere přibližně dvanáct minut. Nástroj tedy dosahuje propustnosti přibližně 5,2 Mb/s. Běh nástroje je limitován především kapacitou operační paměti, protože při zpracování souborů o velikosti půl gigabytu již dochází k zaplnění přibližně 8 GB paměti RAM. Pokud dojde k vyčerpání dostupné kapacity operační paměti, unixové operační systémy začínají využívat diskového uložiště pro ukládání nadbytečných dat, což vede ke značnému zpomalení při detekci. Pokud je tedy zapotřebí zpracovávat soubory o větších velikostech, je nezbytné rozšířit kapacitu operační paměti nebo využít disků typu SSD, na kterých zpomalení již není tolik výrazné.

7. Zhodnocení

Výsledný nástroj je implementován v jazyce Python ve verzi 2.7. Jazyk Python byl vybrán především díky jeho flexibilitě a množství dostupných knihoven.

Pro zpracování paketů byla použita knihovna *cElementTree*, která je implementována v čistém jazyce C a ve srovnání[5] s ostatními knihovnami dosahuje nejlepších výsledků jak v době zpracování, tak v oblasti paměťových nároků. Aktuální verze nástroje poskytuje dvacet tři deklarativních popisů a je schopná detekovat následující útoky [7]:

- *ARP Spoofing*;
- *Duplicate address detection attack*;
- *DHCP Spoofing*;
- *ICMP Redirect*;
- *MAC Flooding*;
- *Skenování sítě IPv4 a IPv6*;
- *Ping of Death*;
- *Skenování portů (různé metody)*;
- *Router Advertisement Flooding*;
- *SYN Flooding*;
- *Local Area Network Denial (LAND)*;
- *Teardrop*;
- *VLAN hopping: Double tagging*;
- *VLAN hopping: Switch Spoofing*.

8. Závěr

Tato práce pojednávala o návrhu nástroje pro automatizovanou detekci síťových útoků ze zachycené síťové komunikace s využitím nástroje tshark. Při návrhu byl kladen důraz na snadnou rozšiřitelnost nástroje o nové síťové útoky bez nutnosti implementačních změn nástroje. To je řešeno pomocí textových deklarativních zápisů jednotlivých útoků, které pro každý útok popisuje způsob jeho detekce. Součástí práce je tedy i návrh obecného formátu pro deklarativní zápis síťových útoků. Výsledný nástroj funguje jako interpret těchto zápisů a při výskytu nového útoku postačuje vytvořit nový deklarativní zápis, namísto reimplementace nástroje.

Funkčnost navrženého nástroje byla otestována na zachycené síťové komunikaci obsahující všechny síťové útoky, jejichž detekci výsledný nástroj podporuje. Testování bylo zaměřeno i na falešné detekce (tzv. false positives), na jejichž základě pak docházelo k upravě či rozšiřování navrženého řešení. Zároveň bylo navrženo několik optimalizací při konverzi vstupní zachycené komunikace do PDML, díky čemuž došlo k razantnímu snížení velikosti výstupu konverze a tudíž i snížení paměťové náročnosti nástroje.

Budoucí vývoj nástroje bude věnován především rozšiřování prostředků pro deklarativní zápisy útoků v případě, že bude vyžadována podpora nových síťových útoků, pro jejichž zápis jsou současné prostředky nedostatečné. Zároveň bude zapotřebí reagovat na případné další falešné detekce a upravovat navržený deklarativní zápis tak, aby došlo k jejich maximální možné eliminaci.

[5] Viz. http://effbot.org/zone/celementtree.htm#benchmarks

Poděkování

Velice rád bych tímto poděkoval svému vedoucímu Ing. Martinu Holkovičovi za jeho ochotu, časovou flexibilitu a odborný dozor při celém procesu tvorby nástroje, kterému se tento článek věnuje.

Literatura

[1] BACE, R.: *Intrusion Detection*. Indianapolis: Macmillan Technical Publishing, 2000, ISBN 1578701856, 339 s.

[2] CANVAN, J. E.: *The Fundamentals of Network Security*. Artech House, 2001, ISBN 1580531768, 218 s.

[3] COMBS, G.; RAMIREZ, G.; HARRIS, G.: GitHub repozitář nástroje TShark. červen 2014, [Online].
URL https://github.com/wireshark/wireshark

[4] GLEN, S.: False Positive and False Negative: Definition and Examples. Duben 2015, [Online], [cit. 2018-03-25].
URL https://goo.gl/fJHJL8

[5] McCLURE, S.; SCAMBRAY, J.; KURTZ, G.: *Hacking Exposed: Network Security Secrets and Solutions*. McGraw-Hill Education, sedmé vydání, 2012, ISBN 0071780289, 768 s.

[6] NAYYAR, A.: The Best Open Source Network Intrusion Detection Tools. Duben 2017, [Online], [cit. 2018-04-19].
URL https://goo.gl/4yYYHz

[7] Tulloch, M.: *Microsoft® Encyclopedia of Security*. Microsoft Press, 2003, ISBN 0735618771, 480 s.