Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Vizualizace síťového provozu s ohledem na bezpečnostní události

Bakalářská práce

Jan Soubusta

Vedoucí práce: Ing. Petr Břehovský

České Budějovice 2020

Bibliografické údaje

Soubusta J., 2020: Vizualizace síťového provozu s ohledem na bezpečnostní události. [Network traffic visualisation with respect to security events, Bc. Thesis, in Czech] – 67 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Abstrakt:

Práce se zabývá metodami vizualizace síťového provozu a identifikací nekalých či útočných aktivit. Teoretická část je ověřena pomocí experimentů s virtuálními počítači, kdy je simulován útočník generující škodlivý provoz. Cílem práce je popsat a otestovat způsoby zobrazení dat v síti tak, aby byla lidsky čitelná a byla možná identifikace potenciálně nebezpečných aktivit útočníka.

Klíčová slova:

sítový útok; vizualizace datového toku; virtualizace

Abstract:

The thesis deals with methods of visualization of network traffic and identification of unfair or offensive activities. The theoretical part is verified by experiments with virtual machines, where an attacker, generating malicious traffic, is simulated. The target of the thesis is to describe and test the ways of displaying data in the network so that it is humanly readable and possible to identify potentially dangerous activities of the attacker.

Keywords:

network attack; network flow visualisation; virtualisation

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenu a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 22. 5. 2020 _____

Jan Soubusta

Poděkování

Děkuji panu Ing. Petru Břehovskému za odborné vedení, trpělivost, cenné rady a ochotu při řešení problémů. V neposlední řadě bych rád poděkoval celé své rodině, která mě po celou dobu studia podporovala.

Obsah

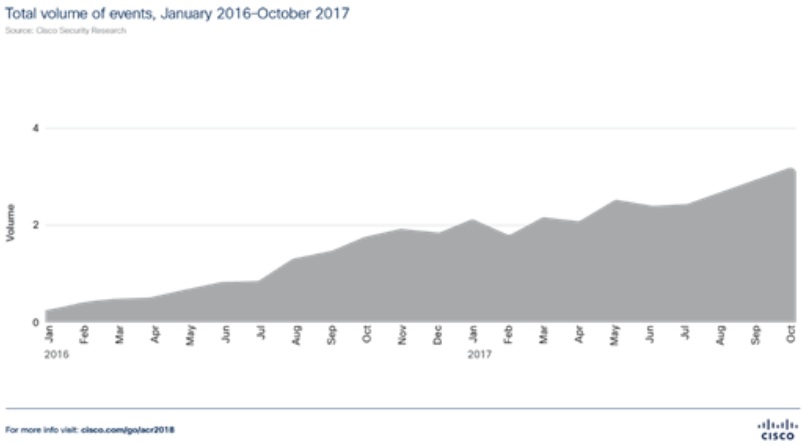
1 Úvod

Detekce (zachycení) a prevence síťových útoků dnes bezesporu patří mezi nejdůležitější bezpečnostní opatření v rámci jak otevřeného internetu, webových služeb, tak i interních sítí. Chráníme data, která mají pro různé cílové skupiny jiný význam a důležitost, ale všechny spojuje jedna obava, a to z jejich odcizení (útočník data zkopíruje na své systémy a nenávratně je vymaže z našich – krádež kriticky důležitých dat), pozměnění (změna významu dat, a tím ohrožení dalších systémů, či společností) nebo vyzrazení (útočník pouze data odposlechne – například odposlech v síti).

Detekce útoku sama o sobě nemůže poskytovat dostatečnou službu [1] (monitorovací systémy, které vyhodnocují tok dat, na základě známých vzorů či neobvyklých znaků, jako podezřelý), ale důležitou roli hrají i systémy, které jsou na detekci útoků napojeny. Jde například o systémy, jež na podezřelé aktivity samočinně reagují, a systémy nebo služby, které upozorňují správce či uživatele na nějakou podezřelou aktivitu. Tyto služby mohou poskytovat například systémy pro detekci vniknutí (IDS), které pomocí databáze známých útoků a behaviorálních analýz vyhodnocují tok dat a podezřelé či nebezpečné aktivity.

Poté v závislosti na jejich nastavení aktivně blokují nebo se dotáží uživatele na další akci.

V souvislosti s nárůstem síťového provozu se zvyšuje i tlak na analýzu a detekci potencionálních hrozeb.



Obrázek 1 Počet útoků v období leden 2016 – říjen 2017, zdroj: [2]

Je důležité vhodně přizpůsobit uživateli/správci data tak, aby byl schopen účinně, rychle a spontánně reagovat v případech, kdy automatizované systémy selžou a z nějakého důvodu útok nedetekují [3]. Proto vznikla poptávka po řešeních, která umožňují čitelně a názorně vizualizovat dění ve spleti síťového provozu – a to je též hlavním obsahem této práce.

2 Cíle

Hlavním cílem bakalářské práce je popsat a otestovat metody vizualizace síťových aktivit, které jsou identifikované jako nekalé a útočné. Tohoto cíle je dosaženo pomocí definic rámcových podcílů. Pro dosažení cílů práce bude nutné se zaměřit na provedení následujících dílčích kroků:

a) Komentovaná rešerše k tématům síťových útoků (jaké jsou, jak fungují, ukázky), logování a uchovávání záznamů o událostech (co je log, jak se uchovávají, jak se dají převést do lidsky pochopitelného formátu/vizualizace);
b) Zachycení útoku – detekce podezřelé aktivity, metody vyhodnocování síťové aktivity;
c) Metody zpracování a vizualizace síťové aktivity;
d) Vytvoření podpůrného virtuálního počítače pro ověření teoretických znalostí;
e) Simulace vybraných síťových útoků na virtuálním počítači;
f) Data a výstupy z experimentů popsat a okomentovat.

3 Vizualizace síťového provozu

Hlavním smyslem této práce je ověřit, zda lze vizualizaci metrik, poskytovanou systémem, využít v reálných situacích, např. v datových centrech. Tento způsob v rámci vizualizovaných změn monitorovaných parametrů může sloužit k detekci útoků, kde je v roli detektorů správce sítě, který dokáže určit, zda se jedná o útok, či nikoliv, a proto jsem se rozhodl využít tento způsob ve své bakalářské práci.

Vizualizací rozumíme jasné a efektivní prezentování dat pomocí grafických prvků neboli převedení strukturovaných dat do lidsky uchopitelného formátu. Nejprve nás zajímá srozumitelnost prezentovaných dat, až poté se zabýváme grafickou stránkou vizualizace, která dokáže zaujmout pozornost a data zpřehlednit. Data se dají formou vizualizace prezentovat nebo také zkoumat. V první řadě je důležité zjednodušit vstupní soubor dat a zobrazit údaje tak, abychom mohli vyvodit nějakou souvislost [4].

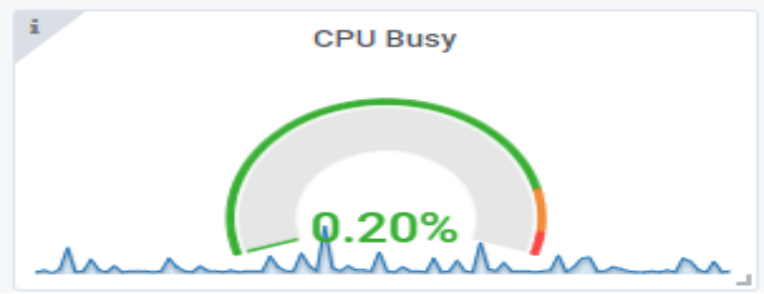
3.1 Způsob vizualizace síťového provozu

Data, která byla vygenerována systémem a uložena do databází, je možné dále zpracovávat a transformovat:

a) filtrovat pouze vybrané (numerické, textové) hodnoty;
b) provádět statistické a matematické úpravy (průměr, medián, denní součet atd.);
c) vizualizovat.

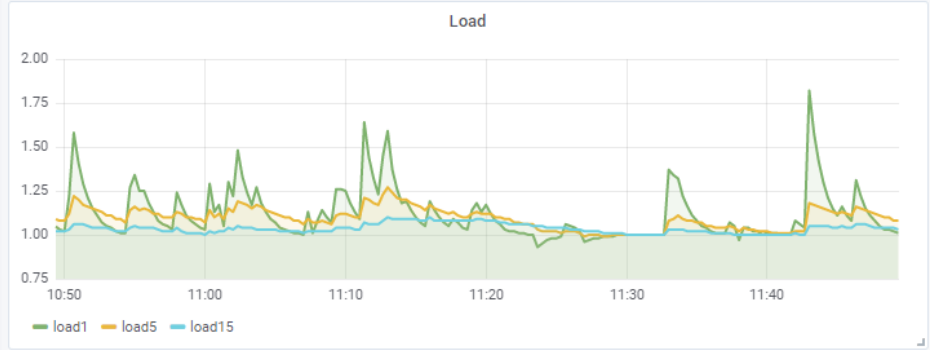
Člověk snadněji zpracovává vizuální vjemy než textové a strukturované informace [5]. Vizualizace je v podstatě prezentací dat [6] grafickými prvky neboli převedení strukturovaných dat do lidsky uchopitelného formátu. Důležité je zjednodušit vstupní soubor dat a zobrazit údaje tak, abychom mohli vyvodit nějakou souvislost. Složitost prezentace můžeme rozdělit na:

a) jednoduché grafy, které na osy vynášejí četnost výskytu hodnot;



Obrázek 2 Jednoduchý graf, zdroj: vlastní výzkum

b) grafické prezentace spojující různé souvislosti, které vznikají statistickými metodami a dolováním dat.



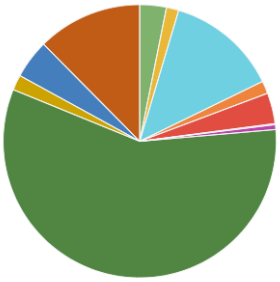
Obrázek 3 Další grafy, zdroj: vlastní výzkum

Pro zobrazení jednodušších grafů můžeme využít 2D i 3D reprezentace sloupcových a koláčových grafů, histogram a kartografické zobrazení místa (mapy). Složitější zobrazení závislostí několika veličin je možné znázornit pomocí grafu rozptylu nebo krabicového grafu. Grafy z kategorie a) se využívají pro zobrazení v reálném čase, jelikož přehledně zobrazují veličiny v čase (např. zatížení procesoru). Tabulka ukazuje příklady využití grafu v prostředí síťového provozu.

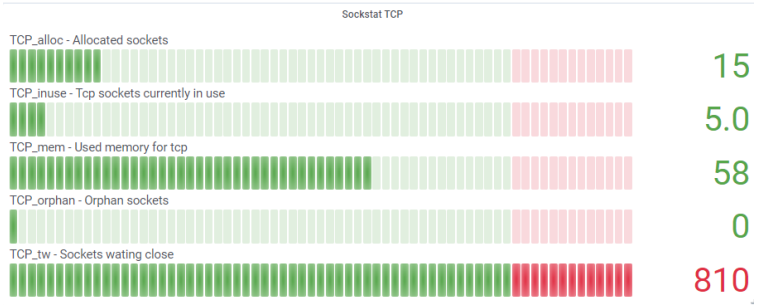
Tabulka 1 Typy grafů, zdroj: vlastní výzkum

Typ grafu	**Příklad využití**
Sloupcový	Časová řada – porovnání hodnot z různých systémů
Spojnicový	Časová řada – využití určitého prostředku v čase
Koláčový	Poměr vytížení prostředků vůči celému systému
Histogram	Četnost výskytu jednotlivých priorit logů
Body XY	Využití zdroje v síti jednotlivými počítači

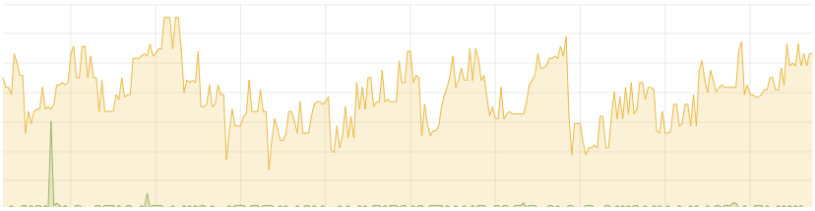
Další ukázky použitých grafů:



Obrázek 4 Výsečový graf, zdroj: vlastní výzkum



Obrázek 5 Bar graf, zdroj: vlastní výzkum



Obrázek 6 Čárový graf, zdroj: vlastní výzkum

3.2 Monitorování síťového provozu

Abychom mohli zpracovávat události a stavy sítě, je potřeba do systému zavést nástroje, které umožní sběr dat. **Události v systému se monitorují** [7]:

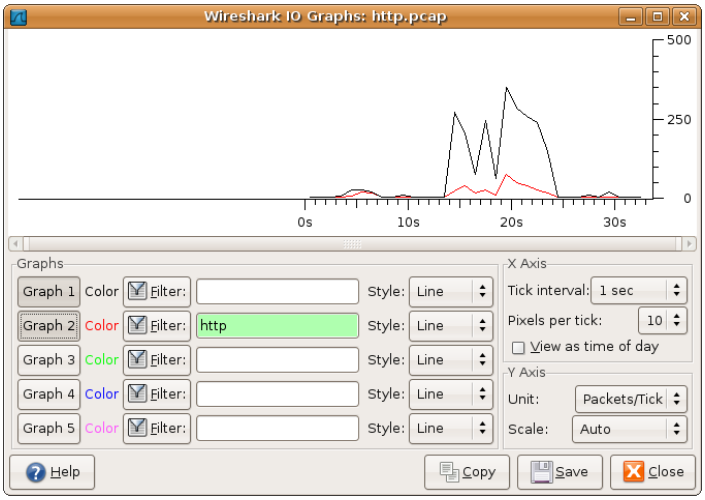
1. zachytáváním paketů:
   a) zdrojová a cílová IP adresa,
   b) služba (port),
   c) obsah paketu.

2. sledováním využití jednotlivých zdrojů:
    a) počet TCP/IP spojení,
    b) počet paketů na síťovém rozhraní,
    c) využití systémových prostředků (procesor, operační paměť).

Uvedené body lze zachytávat ve dvou úrovních:

a) **lokálně**, na úrovni jednotlivých zařízení;
b) **globálně**, v rámci celého systému.

Lokálně lze zachytávat pakety pomocí programů nebo modulů známých jako „packet sniffer“ [8]. Tyto programy či moduly jsou softwarovými nástroji, které zachytí příchozí (mohou i odchozí) paket, dokážou ho analyzovat a volitelně o něm provést záznam (log). Nástroje mohou analyzovat hlavičky jednotlivých paketů a dekódovat jejich data. Mnoho nástrojů může také sloužit jako IDS. Příkladem nástrojů, které mohou analyzovat pakety, jsou: Wireshark [9], tcpdump [10], nebo PRTG [11]. Příkladem modulu může být součást firewallu klientského zařízení.



Obrázek 7 Lokální vizualizační nástroj, zdroj: [12]

Globální zachytávání vyžaduje komplexnější řešení. V síti jsou rozmístěna zařízení, jež můžeme chápat jako „probes“ (sondy) [13]. Tyto sondy mohou být buď na softwarové, nebo hardwarové bázi. Umisťují se na místa, kde prochází síťový provoz, nebo se připojují k zrcadleným portům switche. Jedná se o firewally, IDS/IPS systémy a routery. Vizualizace v globálním měřítku je možná díky topologickým mapám, kde jsou vyobrazeny jednotlivé uzly a poté lze data z každého jednotlivého systému kombinovat do grafů.

Síťová topologie je zobrazením logické struktury sítě – jeho aktivních (routery, switche, servery, klientské stanice) a pasivních prvků (huby, volitelně drátové a bezdrátové spoje). V takovéto topologii můžeme zobrazovat i tok dat, vzdálenost mezi jednotlivými prvky, fyzická spojení, rychlosti spojení atd.

3.3 Převod strukturovaných dat v grafickou reprezentaci

Pro generování grafů a vizualizace existuje mnoho nástrojů [14], které programátorům usnadňují zpracování a publikaci dat. V tabulce je sestaven přehled nástrojů, díky kterým lze vizualizovat data síťového provozu. Záměrem tabulky je porovnat jednotlivé nástroje z hlediska aplikace, zda jde o jednoúčelové nástroje čistě pro grafické zpracování vložených dat, nebo o víceúčelové nástroje, jež umožňují i sběr a zpracování (transformace). Dalšími parametry je způsob vykreslování, který udává, zda nástroj již ve výchozím nastavení umožňuje vizualizovat data v reálném čase a složitost implementace. Způsob vykreslování definuje, zda nástroj umožňuje generovat grafy v reálném čase, tedy bez zásahu uživatele anebo je potřeba každou aktualizaci grafu ručně vyvolat. Složitost implementace jsem posoudil podle množství nabízené funkcionality, použitého jazyka a požadovaných dodatečných úprav (jako je instalace dalších knihoven).

Složitost je rozdělena na:

a) lehkou – lze jednoduše zprovoznit a okamžitě využívat;
b) střední – lze jednoduše nasadit, ale vyžaduje dodatečné úpravy na serveru;
c) těžkou – je potřeba zprovoznit dodatečné knihovny a ručně vytvářet grafy.

Tabulka 2 Porovnání grafických a vizualizačních nástrojů, zdroj: vlastní výzkum

Nástroj	**Aplikace**	**Vykreslování**	**Složitost**	**Licence**
jqPlot [15]	Grafická knihovna	V reálném čase	Střední	Open Source
Flot [16]	Grafická knihovna	V reálném čase	Střední	Open Source
Highcharts [17]	Grafická knihovna	V reálném čase	Střední	Komerční
Wireshark [18]	Aplikace na ploše	V reálném čase	Lehká	Open Source
RRDGraph [19]	Terminálová aplikace	Jednorázově	Střední	Open Source
VnStat [20]	Terminálová aplikace	Jednorázově	Těžká	Open Source
Darkstat [21]	Webová aplikace	V reálném čase	Těžká	Open Source
Cacti [22]	Webová aplikace	Jednorázově	Těžká	Open Source
GlassWire [23]	Aplikace na ploše	V reálném čase	Lehká	Freeware
ZABBIX [24]	Webová aplikace	V reálném čase	Těžká	Open Source
Nagios [25]	Webová aplikace	Jednorázově	Těžká	Open Source
Grafana [26]	Webová aplikace	V reálném čase	Lehká	Open Source

Na základě provedené analýzy jsem vybral nástroj Grafana, který budu dále konkretizovat v kapitole 5.3.1 Grafana.

4 Síťové útoky

Jedná se o systematickou činnost zaměřenou na snížení nebo narušení bezpečnosti, jež je vykonávaná účelově s cílem něčeho dosáhnout (např. osobních cílů, získání dat, poškození cílového subjektu atd.). Základem této činnosti je využít slabých míst operačního systému nebo softwaru na něm umístěném. Jedná se o jakékoliv protiprávní jednání, které směřuje proti zájmům jiné osoby [27]. Obecně je cílem útoku:

a) získání nebo zničení informací;
b) odstavení nebo narušení služby;
c) zneužití služby.

Útoky, které jsou vedené pomocí softwarových prostředků, v rámci světové (internet), ale i lokální sítě, spadají do oblasti kybernetické kriminality. Definuje ji terminologie, jež je využita i v této práci. Osoba, jež se pohybuje v kybernetickém prostoru a která realizuje samotný útok, se nazývá útočníkem. Způsob, jakým dochází ke zneužití zranitelnosti systému, se nazývá vektor útoku. Zranitelnost je zpravidla softwarová chyba, ale může se jednat i o chybu způsobenou lidským faktorem (např. nedostatečné proškolení či povědomí o bezpečném užívání systému). Samotná kybernetická kriminalita je dle Matějky [28] definována jako trestná činnost, v níž je využit počítač nebo některé z jeho komponentů, případně větší množství počítačů samostatných, nebo propojených jako předmět zájmu trestné činnosti, anebo jako nástroj trestné činnosti.

Ohrožení síťové bezpečnosti (integrity) je charakteristické anomáliemi, které během provádění jednotlivých útoků vznikají, a které je možné detekovat s různou mírou úspěšnosti, anebo jim předcházet. Na takovéto anomálie se zaměřují různé prostředky pasivní a aktivní prevence, jako jsou firewally, systémy IDS/IPS [29] a administrátoři jednotlivých systémů.

4.1 Rozdělení podle způsobu

Jako se liší způsoby fyzického narušení bezpečnosti (napadení banky, poškození nebo zcizení majetku movitého i nemovitého atd.), tak i v síťové komunikaci existuje vícero způsobů, jak provést útok prostřednictvím síťových prostředků a protokolů. Způsoby, jakými lze narušit bezpečnost cílového systému, jsou [30]:

1) **Aktivní útoky**, které jsou zaměřené na systémové prostředky a jejich uživatele. Účelem může být narušení integrity cílového systému (služby).

Jedná se o:

a) sociální inženýrství,
b) skenování sítě, či systému,
c) odstavení služby,
d) pozměňování informací,
e) zneužití slabin systému.

2) **Pasivní útoky** cílí na získání a zneužití důvěrnosti informací, jež jsou v systému generovány tak, aby nedošlo k prozrazení této nekalé činnosti (čím déle může útočník monitorovat provoz, tím více podstatných informací může získat). Pasivní jednání může předcházet aktivnímu útočení. Jedná se o:
   a) odposlech,
   b) analýzu provozu.

4.2 Druhy útoků

V této kapitole se zabývám obecným popisem jednotlivých útoků, které následně provádím a testuji v experimentální části.

4.2.1 Útok hrubou silou

Útoky Brute Force se nejčastěji využívají na prolomení heslových frází, kdy systém útočníka zkouší „uhádnout“ aktuální přístupové údaje do systému. Útočníci předpokládají, že něž oběť stačí zaznamenat nějaké napadení integrity a bezpečnosti, tak stihnou heslo uhádnout. Pro útoky vedené na přístupové údaje existuje mnoho nástrojů a skriptů, které lze jednoduše převzít a spouštět. Některé fungují na principu náhodného či postupného generování řetězců, ostatní využívají předgenerované tabulky heslových frází. Hrubou sílu využívají útočníci také na prolomení jiných tajemství, například klíčů SSH a certifikátů. Tímto stylem útoku můžeme odstavit službu také tak, že ji přehltíme požadavky (Denial of Service).

Pro tento typ útoku jsem očekával změny v následujících metrikách, které budu dále uvádět v praktické části.

1. Počet TCP spojení,
2. síťový provoz,
3. zatížení procesoru,
4. celkové zatížení systému,
5. využití paměti ram,
6. počet neúspěšných přihlášení v ssh serveru.

4.2.2 ARP spoofing

Útočník využívá zranitelnosti protokolu ARP, jenž mapuje IP adresy na MAC adresy. V podstatě to funguje tak, že ARP posila dotazy, které se ptají na MAC adresu dané IP adresy, kde ARP pakety jsou vyslány všem zařízením v broadcast doméně.

Každé zařízení tak analyzuje příchozí ARP dotazy a zasílá odpověď v případě shody IP adres. Pro redukci počtu ARP paketu se příchozí odpovědi průběžně ukládají do cache. Pokaždé, když zařízení obdrží odpověď, uloží si do cache aktuální kombinaci IP adresy a MAC adresy.

Většina operačních systémů mění cache tabulku nehledě na to, zda se o to dotazovaly či nikoli. ARP spoofing je tedy o tom – poslat zařízení A sestavený arp paket tak, aby si k IP adrese zařízení B přiřadil MAC adresu zařízení X. Zařízení A tak pochopitelně bude považovat zařízení X za zařízení B. Bude tedy komunikovat s X bez sebemenšího podezření, že nejde o zařízení B [31].

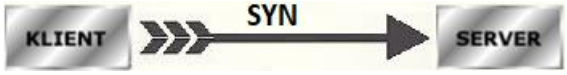
Pro tento typ útoku jsem očekával změny v následující metrice, kterou budu dále rozvádět v praktické části:

- počet záznamů v ARP tabulce.

4.2.3 SYN Flood

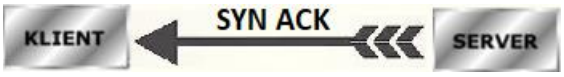
Jde o útok typu DoS (odepření služby), jehož cílem není to, aby se útočník dostal do systému, a nemá ani za cíl získat, smazat nebo jiným způsobem poškodit nějaká data umístěná na serveru. DoS útok zapříčiní většinou dočasnou nedostupnost určité služby, například webu nebo e-mailu. Podstatou SYN Flood útoku je zneužít jedné z vlastností TCP protokolu, který se nazývá three-way handshake, jinými slovy třísměrné potřesení rukou, které ověřuje, zda obě strany o komunikaci stojí.

Můžeme si představit situaci, že klient zahájí spojení se serverem. Klient tedy zašle první paket s nastaveným SYN bitem.



Obrázek 8 Inicializace navázání TCP spojení, zdroj: vlastní výzkum

SERVER odpovídá paketem, který má nastavený SYN a ACK bit a ukládá si informace o nadcházejícím spojení do interní databázové struktury. Tento stav se nazývá polootevřené spojení (half-open connection).



Obrázek 9 Polootevřené spojení, zdroj: vlastní výzkum

Klient nyní za běžné situace dokončí následující krok, který odešle paket s nastaveným ACK bitem. V tuto chvíli je spojení navázáno a mezi klientem a SERVEREM může začít výměna dat.

Obrázek 10 Úspěšné navázání spojení, zdroj: vlastní výzkum

SYN Flood nedělá vlastně nic jiného, než že započne odesílat pakety se SYN bitem. Vypadá to, jako by chtěl začít komunikaci, ale neprovede již poslední fázi handshaku, tudíž na stroji, na který je tento útok zacílen, dojde postupně k zaplavení bufferu pro polootevřená spojení. Tímto jsme dosáhli cíle, server není nyní schopen přijímat další pokusy o navázání spojení, a tak se stává nedostupným. Pokud není nastaven maximální počet spojení, může tento útok také zapříčinit pád serveru, což způsobí úplné vyčerpání volné paměti.

Pro tento typ útoku jsem očekával změny v následujících metrikách, které jsem proto výrazněji sledoval:

1. počet pokusů o navázání TCP spojení,
2. počet neúspěšných TCP spojení,
3. nárůst využití RAM paměti,
4. nárůst swap mezi diskem a RAM pamětí,
5. síťový provoz.

4.2.4 Útok na skenování portů

Skenování sítě je velice zajímavým způsobem získání informací [32], jelikož umožňuje útočníkovi získat přehled o cílovém systému (softwarovým, hardwarovým) a prostředcích, které se v něm nacházejí. Tato technika však už může generovat nežádoucí pozornost automatizovaných systémů, jelikož každou provedenou akci je možné logovat. Způsoby, jakými lze získat informace jsou:

a) **Analýza síťového rozsahu** nebo také **horizontální skenování**, kdy je zkoumán konkrétní segment sítě a jsou hledány připojené systémy (počítače, servery, služby). Každý systém může být klasifikován podle MAC adresy (výrobce a typ přístroje) a mohou u něj být analyzovány služby, které na svém vnější rozhraní poskytuje.

b) **Analýza síťového rozhraní konkrétního stroje** nebo také **vertikální skenování**. V této fázi může útočník testovat celý rozsah služeb a u každé služby poté analyzovat možná bezpečnostní rizika, jako jsou například zranitelnosti verzí jednotlivých služeb, otevřené síťové porty stroje nebo možnost získat administrátorská práva. Tento sken probíhá většinou na transportní vrstvě ISO/OSI modelu.

Pro tento typ útoku jsem očekával změny v následujících metrikách, které jsem proto výrazněji sledoval:

1. počet TCP spojení,
2. síťový provoz,
3. využití CPU.

4.2.5 Přetížení síťového rozhraní provozem

V momentě, kdy útočník nepotřebuje, nechce nebo nemá možnost získat informace cílového systému, může provést útok typu odepření služby neboli také anglicky známé Denial of Service (DoS). Jedná se narušení bezpečnosti odstavením systému (nezřídka kritického), jeho zahlcením, a tím znepřístupněním pro uživatele. Smyslem útoku je omezení funkčnosti poskytované služby. Konkrétně se pak jedná o vyčerpání prostředků (kapacity) systému jako je procesorový čas, paměť či dostupné síťové porty.

Útoky DoS můžeme rozdělit na tři typy[33]:

a) zaměřené na objem,
b) zneužívající TCP/IP protokol,
c) útoky na aplikační vrstvu.

Cílem útoku zaměřeného na objem je vytížit volné pásmo (využití síťového rozhraní) cílového webu. Z pohledu přípravy před samotným útokem a úrovní znalostní, které jsou potřeba pro provedení útoku, se z těchto tří typů jedná o nejjednodušší verzi. Útočník si vystačí s pouhým počtem infikovaných počítačů, které naráz zasílají požadavky na napadený síťový prvek, jehož přidělené zdroje nezvládají takový objem zpracovat. Tento typ útoku jsem si vybral ke zkoumání v praktické části.

Útoky na síťové vrstvě ISO/OSI zneužívají zdroje serveru na úrovni, kde dochází k prvotnímu zpracování paketů (firewally, load balancery). Tyto útoky podvrhují pakety či hlavičky IP rámců tak, aby zahltily síťovou kartu a procesor cílového počítače, u kterého dojde k postupnému „odstavení“ z normálního provozu.

Pro tento typ útoku jsem očekával změny v následujících metrikách, které jsem proto výrazněji sledoval:

1. počet přenesených bajtů,
2. počet TCP Spojení,
3. využití CPU,
4. využití RAM.

5 Experimentální část

V praktické části této práce popisuji provedený způsob vizualizace síťového provozu a zachycení podezřelé aktivity. Dále rozebírám metodiku výběru softwarových prostředků, pomocí kterých jsou provedeny navržené experimenty a jejich aplikace.

V této kapitole je uveden metodický přehled, pomocí něhož postupuji při experimentování s virtuálním prostředím, dále při generování podezřelých aktivit a monitorování síťového provozu. V kapitole popisuji postup od vytvoření virtuálního prostředí, až po provedení jednotlivých experimentů. Výsledkem je přehledný souhrn provedených experimentů a komentované závěry.

5.1 Metodika praktické části

Tato podkapitola uvádí obecný postup pro experimentální část, kterým je:

1. **Příprava hardwaru a nasazení virtuálního testovacího prostředí** – tento bod je detailně rozveden a zdůvodněn, jelikož na tomto prostředí je celý experiment proveden, a jedná se tedy o nosný pilíř celého testovacího prostředí.
2. **Instalace softwaru, který je potřebný pro zpracování síťového provozu** – software, jenž umožňuje generování a monitoring síťového provozu.
3. **Instalace vizualizačního nástroje Grafana a všech prerekvizit** – příprava a nastavení webového rozhraní pro vizualizaci, jež umožňuje snadnou správu a sledování metrik.
4. **Generování síťového provozu** – způsoby vytváření umělého síťového provozu, které byly použity pro ztížení experimentů a možnost srovnání stavů před a po útoku.
5. **Analýza síťového provozu a vizualizace** – analýza metrik, jež jsou použité pro vizualizaci síťového provozu a lepší pochopení chování na síti.
6. **Nastavení limitů, které určují běžný síťový provoz** – v rámci dlouhodobého dohledu nad sítí byly stanoveny limity, které definují stav sítě, jenž považujeme za běžný. V případě překročení této hranice dochází k notifikaci pověřené osoby.
7. **Provedení útoků pomocí OS[1] Kali Linux –** výběr správných utilit a jejich obsluha při generování nekalého provozu.
8. **Popis přechodu metrik –** popis přechodu z monitorování všech metrik na monitorování konkrétních metrik, které jsou k provedeným útokům relevantní.

[1] Operační systém.

9. **Monitorování a určení anomálií** – na základě sledování stavů sítě je možné určit, za jakých okolností dochází v systému k anomálii.
10. **Určení výstražných upozornění** a definice jejich nastavení.
11. **Testování funkčnosti upozornění** – provedení experimentálních útoků s cílem otestovat nastavené limitní hodnoty a upozorňovací funkce.
12. **Souhrn dosažených výsledků** – soupis a komentář ohledně dosažených výsledků experimentů.

5.2 Vytvoření experimentálního prostředí

Pro experiment jsem vytvořil kontrolované prostředí, které umožnilo simulaci útoků, snadnou správu a obnovení prostředí do původního stavu. Z těchto tří důvodů jsem zvolil možnost virtualizace.

Pro virtuální prostředí jsem potřeboval zvolit hypervizor[2], který by splnil všechny tři výše dané předpoklady. Zvažovány byly softwary:

1. KVM/QEMU [34]
2. PROXMOX [35]
3. VIRTUALBOX [36]
4. VMWARE [37]

Pro srovnání byla sestavena vícekriteriální tabulka:

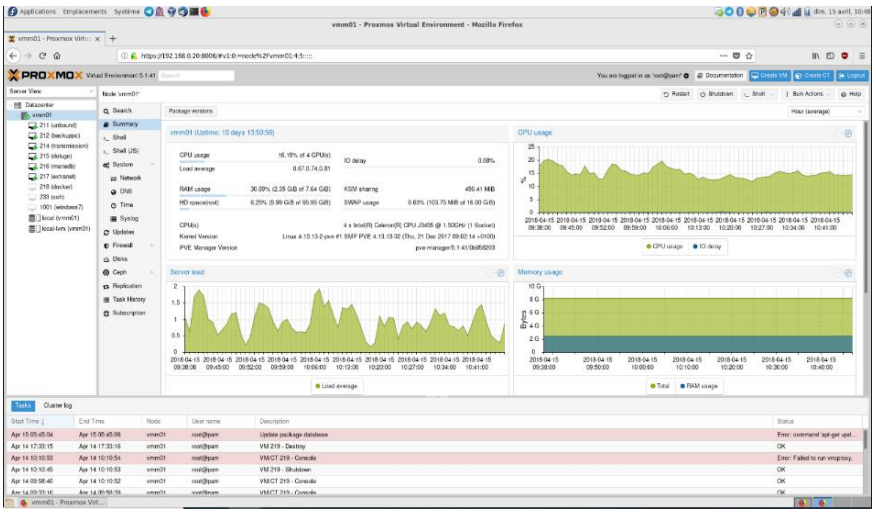
Tabulka 3 Vícekriteriální porovnání virtualizačních platforem, zdroj: vlastní výzkum

Volba	**OSS[3]**	**Křivka učení[4]**	**Hardwarové nároky**	**Obsluha**	**Proces zprovoznění**
1	Ano	Strmá	Nízké	Příkaz. řádka	Náročný
2	Ano	Mírná	Nízké	Webové rozhraní	Střední
3	Ano	Mírná	Střední	Aplikace na ploše	Jednoduchý
4	Ne	Strmá	Střední	Webové rozhraní	Náročný

[2] Arbitr, který spravuje hardwarové prostředky hostitele a umožňuje přístup virtualizovaným systémům známým jako hosté.
[3] Open Source Software.
[4] Popisuje míru porozumění předmětné oblasti v závislosti na délce učení.

Po otestování jednotlivých možností jsem vybral druhý jmenovaný software, jelikož podle mého subjektivního vnímání nabízí ze všech uvedených možností:

1. nejjednodušší správu virtuální sítě,
2. jednouché přiřazení hardwarových prostředků,
3. snadno spravovatelný interface,
4. dostupnost přes prohlížeč (servery tedy mohou běžet bez monitoru),
5. a jedná se o software zdarma, OSS[5].



Obrázek 11 Administrační rozhraní nástroje PROXMOX, zdroj: vlastní výzkum

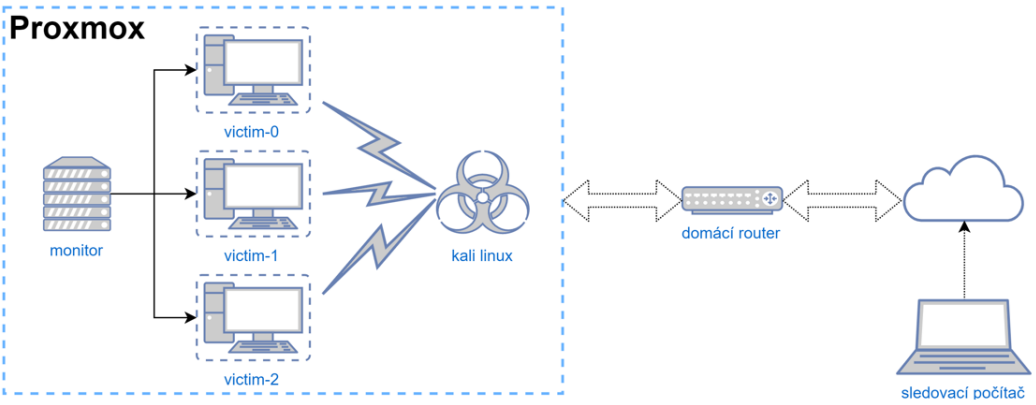
Hypervizor PROXMOX jsem nainstaloval na samostatně stojící počítač, který má čtyř jádrový procesor a 16 GiB paměti RAM. Při instalaci a nastavení jsem postupoval dle návodu na webu výrobce [38], jenž obsahuje tyto kroky:

1. stažení instalačního softwaru a následné nahrání na USB disk;
2. připojení USB disku k počítači a jeho zapnutí;
3. spuštění instalačního procesu;
4. rozdělní disků a nastavení hardwarového RAIDu[6];
5. instalace nutných součástí a nastavení hesla správce;
6. restart systému;
7. připojení k webovému administrativnímu rozhraní a přihlášení;
8. příprava virtuálních počítačů;
9. instalace OS ve virtuálních počítačích.

[5] Open Source Software.

[6] Redundant Array of Independent Disks – diskové pole s redundancí dat.

Virtuální model jsem navrhl jako prostředí třech uživatelských počítačů, jež jsou konstantě monitorovány čtvrtým dohledovým počítačem, a do sítě je připojen pátý počítač, jenž simuluje útočníka, který pronikl do tohoto prostředí.



Obrázek 12 Model virtuálního prostředí, Zdroj: vlastní výzkum

Tabulka 4 Přehledová tabulka virtuálního prostředí, Zdroj: vlastní výzkum

Označení	**Úloha**	**Prostředky**	**Operační systém**
Victim-0	Hlavní oběť útoků	2 vCPU / 4GiB RAM	Ubuntu 18.04
Victim-1	Kontrolní skupina	2 vCPU / 4GiB RAM	Ubuntu 18.04
Victim-2	Kontrolní skupina	2 vCPU / 4GiB RAM	Ubuntu 18.04
Monito-0	Monitorovací prvek	4 vCPU / 8GiB RAM	Ubuntu 18.04
Kali Linux	Útočník	4 vCPU / 4GiB RAM	Kali Linux 2019.03

Tři virtuální počítače „victim“ jsem vytvořil za účelem dosažení vysoké dostupnosti a navození zajímavého prostředí pro útočníka. Jelikož jsou v jednu chvíli získávány údaje ze tří samostatných instancí, máme k dispozici přímé srovnání metrik napadeného počítače a dvou zdravých. Také v případech, kdy útoky docílí odstavení počítače, máme k dispozici další dva “k dobru”. Na počítače victim jsem nainstaloval operační systém Ubuntu 18.04, jelikož právě s tímto OS mám největší zkušenosti, co se týče administrace systému.

Kali Linux je systém útočníka, jehož prostřednictvím provádím útoky na cílové oběti. Tento operační systém jsem vybral, jelikož obsahuje širokou škálu penetračních testů, obsahuje nástroje pro snadné provádění útoků a mám s ním dobré zkušenosti.

Všechny virtuální počítače jsem připojil do společné virtuální sítě a jsou dosažitelné z běžné domácí sítě skrze své IP adresy, které jim přiřadil domácí router.

5.3 Mé monitorovací řešení

V této kapitole popisuji důvod výběru jednotlivých softwarů, jejich instalaci a dále pojednávám o tom, jak jsem je použil v rámci své bakalářské práce.

5.3.1 Grafana

Nástroj **Grafana** jsem vybral jako vizualizační nástroj, jelikož umožňuje:

a) Jednoduchou správu datových zdrojů – což znamená, že vstupy a data se vkládají jednoduše a jednoduše se s nimi pracuje. Pro tyto účely nabízí Grafana interaktivního průvodce pro vytváření a správu datových zdrojů.

b) Přístup vícero uživatelům, přičemž svou výslednou nástěnku je možné sdílet s dalšími uživateli jako odkaz, na který se pouze klikne a zobrazí se okamžité výstupy. Uživatelé, kteří mají možnost užívat tuto nástěnku, mohou mít omezená práva a nemusí mít možnost změnit složení nástěnky [39].

c) Interaktivní práci s vizualizacemi, jež vyjadřuje vzájemné působení dvou i více činitelů a slouží k okamžité viditelnosti výsledku, popřípadě úpravě své práce.

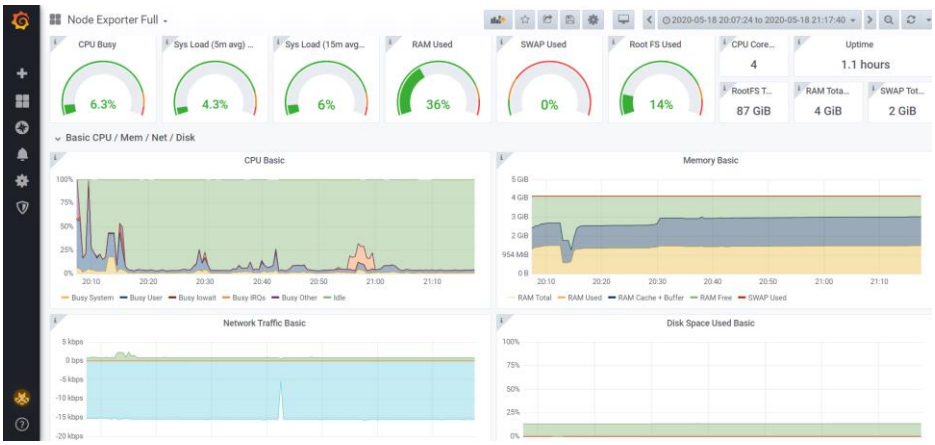
d) Nastavení upozornění, kdy pomocí databázového dotazu zvolíme metriku, kterou potřebujeme kontrolovat a nastavíme horní hranici (threshold), při jejímž překročení má dojít k upozornění.

e) Efektní grafickou reprezentaci dat, přičemž data jsou koncipovaná přehledně tak, aby umožňovala rychlé zorientování pro uživatele. To hraje velkou roli v reakci na anomálie.

f) Pro Grafanu bylo vytvořeno velké množství doplňků [40], jež lze do ní přímo integrovat a stáhnout je z oficiálního portálu výrobců. Uživatel si poté může rozšířit funkcionalitu nástěnek a/nebo pracovat s atypickými datovými zdroji.

d) Je zdarma pro komerční i nekomerční užívání.

Grafana se používá jako nástroj k monitorování a analýze dat. Slouží k vytváření grafů, upozornění a ukazatelů, které je potřeba pozorovat. Po připojení databáze můžeme tvořit grafy, alerty apod. z dat, jež se v ní nachází.



Obrázek 13 Náhled na vizualizační nástroj Grafana, zdroj: vlastní výzkum

5.3.2 Popis metrik, které lze Grafanou sledovat

Metriky monitorujeme a sbíráme proto, abychom získali náhled na chování systému, historické srovnání stavů a abychom mohli vytvářet predikce o stavu budoucím, případně detekovat nenadálé změny systému. Toto zpracování dat nám také umožňuje hledat příčiny vzniklých problémů, abychom mohli vytvářet opatření. Souhrnně je tedy pro nás důležité znát chování systému, kterého docílíme sběrem údajů ve formě metrik.

Metrika je veličina, jež se uvádí v měrných jednotkách a obsahuje nějakou hodnotu (např. 20 % CPU). Andrew Jaquith ve své publikaci Security Metrics [41] uvádí pět základních požadavků, kterých by mělo být dosaženo. Dle těchto pravidel by měla být dobrá metrika taková, aby:

a) měření bylo objektivní,
b) získání vstupních dat by nemělo být nákladné,
c) měření mohlo být prováděno opakovaně,
d) výsledek měření mohl být vyjádřen jako číslo či procento,
e) výsledek měření byl vztažen ke konkrétní veličině.

Po osvojení si vytváření jednotlivých vizualizací (grafu) jsem si vytvořil grafickou nástěnku všech dostupných metrik.

Existuje celá řada již vytvořených metrik [42], které je možné použít pro hodnocení systému. Pro můj experiment je k dispozici cca 45 těchto metrik. Seznam metrik je popsaný v repozitáři projektu. Jedná se o výrobcem pevně daný seznam všech metrik, které lze pomocí nástroje sbírat. Pro účely experimentů jsem vybral následující, jež jsem klasifikoval jako stěžejní:

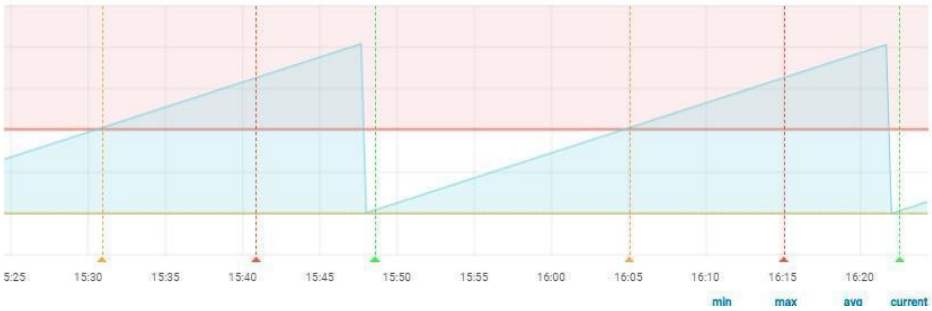
Tabulka 5 Popis metrik, zdroj: vlastní výzkum

CPU Busy	Reprezentuje zatížení procesoru v jednotkách času – kolik procesorového času z celkového tráví zpracováním úloh – je možné ho vyjádřit procentuálně jako poměr. Poté můžeme zatížení procesoru vyjadřovat v procentech. Procesor počítače zpracovává požadavky, které mu přichází (ne)jen přes síťové rozhraní. Útoky, se kterými jsou prováděny experimenty, mají přímý vliv na vytížení procesoru, jelikož ten musí každý požadavek, např. na přístup k webovému serveru nebo ke zpracování události přihlášení, zpracovat.
Used RAM	Tato metrika sleduje zaplnění paměti RAM v počtech MB. V paměti RAM si operační systém uchovává běžící úlohy. Paměť RAM zaplňuje webový server, který si zde uchovává dočasné kopie svých souborů. Pokud útočník přistupuje často k jednomu síťovému zdroji nebo provádí útok s cílem přetečení paměti, může tato metrika tyto aktivity nepřímo prozradit.
System load	Reprezentuje metriku, kterou si můžeme zobrazit linuxovým příkazem „uptime“ – tedy systémovým zatížením za 1,5 a 15 minut. Tato metrika je přímo ovlivněna počtem úloh, se kterými si operační systém musí poradit a zobrazuje tedy celkovou zátěž systému. Jedná se spíše o údaj, který se opírá o znalost administrátora serveru, který ví, jaké systémové zatížení je ještě normální nebo v pořádku.
Network traffic	Údaj měřený v jednotkách bajtů za sekundu. Tato metrika reprezentuje síťový provoz, tedy počet přijatých a odeslaných bajtů na síťovém rozhraní. Pokud bude útočník po síti přenášet velká množství dat, může tato metrika na tuto skutečnost poukázat prudkým zvýšením.
Context switches	Útočník může například pomocí DoS útoku mít za cíl odstavit operační systém velkým množstvím požadavků. Operační systém poté musí často přeskakovat mezi jednotlivými úlohami, a tím se zvyšuje množství přepnutých kontextů.
TCP stats	Počet TCP spojení za jednotku času. Jedná se o metriku, která sbírá aktivní spojení na síťovém rozhraní.
UDP stats	Tato metrika zobrazuje počet přenesených UDP rámců – útočník nemusí navazovat TCP spojení, aby mohl přenést velká množství dat na naše síťové rozhraní, proto je dobré sledovat i tuto metriku.

5.3.3 Způsob upozornění uživatele (alerty)

Alert (upozornění) – tento způsob notifikace jsem použil pro jednoduší a okamžité nalezení anomálie, přičemž díky této funkci nemusí správce sítě pracně prohledávat celou nástěnku. V některých případech by mohlo zabrat i několik hodin, než by správce nalezl probíhající anomálii.

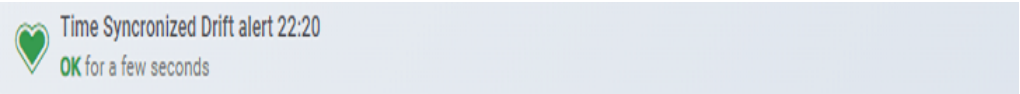
Mezi základní upozorňovací funkce Grafany patří Alert. Jedná se o programovou funkci, kterou si vlastnoručně nadefinuji pro každou hodnotu grafu nebo pro celý graf. Skládá se z definice vybrané hodnoty, z limitní hodnoty, dále z toho, jak často dochází ke kontrole stavu, po jaké době detekování nadlimitní hodnoty dojde ke spuštění akce a z definice akce, která se provede, pokud je alert spuštěn. Spuštěním alertu je myšlen stav, kdy se překročí nastavená prahová hodnota a provede se akce, jež je definovaná v nastavení alertu. Slouží také k automatické notifikaci správce a pověřených osob. Vizualizace spuštění alertu je na obrázku č. 14, kdy prahová hodnota je maximální přijatelnou hodnotou metriky. To znamená, že Alert sleduje hodnoty metriky, ke které je přiřazen.



Obrázek 14 Vývoj upozornění zdroj: vlastní výzkum

Alert má 3 stavy:

1. **OK** – na obrázku č. 15 je vidět jako zelená svislá čára. Tento stav nám říká, že nedochází k žádné anomálii vzhledem k normálnímu provozu.

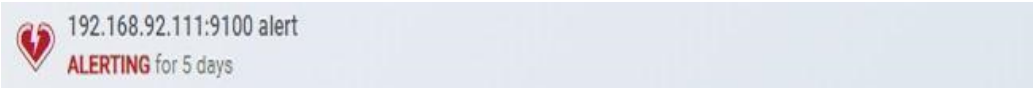


Obrázek 15 Stav vizualizace OK, zdroj: vlastní výzkum

2. **Pending** – na obrázku č. 16 je vidět jako oranžová svislá čára. Tento stav signalizuje překročení threshold. To nutně neznamená nějaký útok, pouze dochází k vyhodnocování dalšího SQL dotazu. Pokud se tento stav vrátí zpět do stavu OK, pravděpodobně šlo pouze o chvilkové zvýšení této metriky a dalo by se říct, že nešlo o žádnou hrozbu.

Obrázek 16 Stav vizualizace pending, zdroj: vlastní výzkum

3. **Alerting** – na obrázku č. 17 je vidět jako červená svislá čára. V případě, kdy SQL dotaz po určitý čas překročil například maximální hodnotu thresholdu, se pending mění na Alerting a dochází k již zmiňovanému alertu, který může dále upozorňovat techniky síťového dohledu.



Obrázek 17 Stav vizualizace upozornění, zdroj: vlastní výzkum

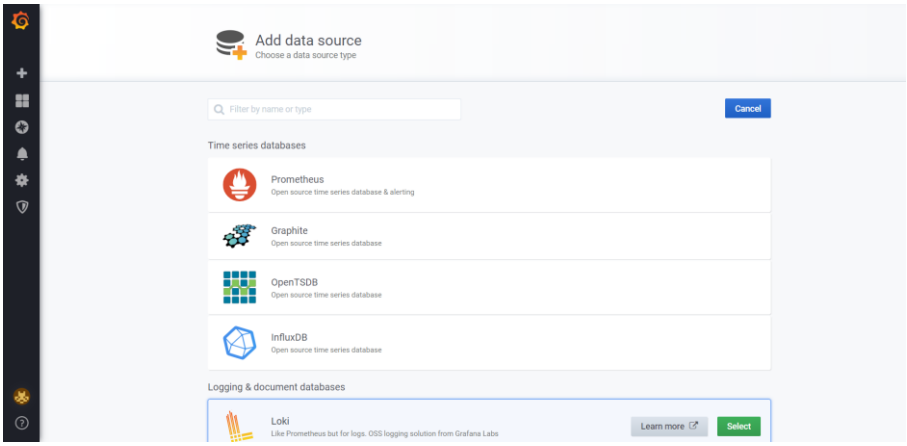
5.3.4 Nastavení monitorovacího řešení

V této kapitole jsem uvedl metodický přehled o tom, jak nastavit nástroj Grafana od začátku, až po vlastní upozornění.

Grafanu jsem nainstaloval společně s datovým zdrojem Prometheus (který je rozveden v kapitole 5.4) na virtuální počítač monitor-0. Je ale možné tuto službu provozovat i samostatně na lokálním počítači. Instalaci na server jsem provedl z důvodu snadného přístupu přes IP adresu z jakéhokoliv počítače v domácí síti.

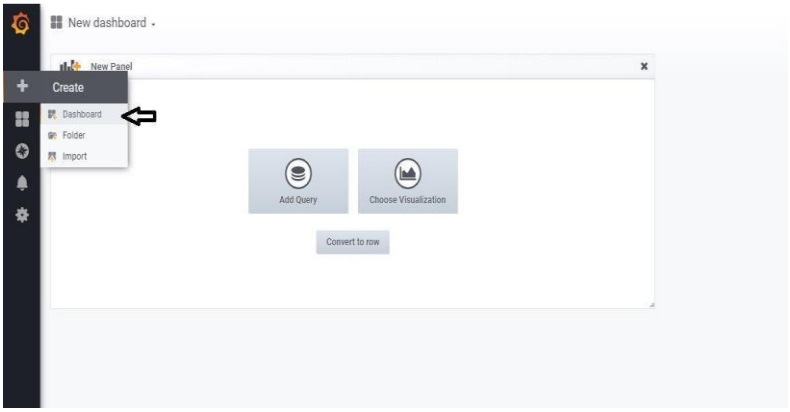
V základním nastavení Grafana naslouchá na: „http://ip-adresa-počítače:3000“.

Na začátku je potřeba připojit datový zdroj. Pro účel této bakalářské práce jsem vybral zdroj Prometheus. Kliknutím na logo Grafana se zobrazí nabídka panelu, kde si můžeme přidat datový zdroj.



Obrázek 18 Cesta k založení nového Dashboardu, zdroj: vlastní výzkum

Po připojení datového zdroje jsem si vytvořil novou nástěnku. Po stisknutí křížku vlevo nahoře se mi otevřela nabídka Create – Dashboard (nástěnka). Pomocí této funkce si vytvořím nástěnku, kde už snadno mohu začít s vizualizací dat. Stačí pouze kliknout na „Add Query“, čímž vyvoláme nabídku, kde nalezneme základní parametry pro nastavení vizualizace.



Obrázek 19 Cesta k založení nového Dashboardu, zdroj: vlastní výzkum

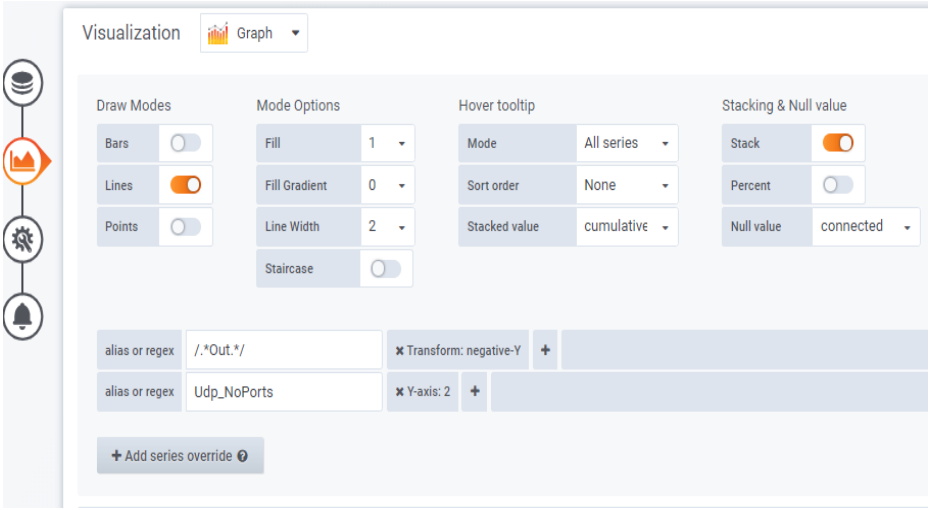
Za pomoci pole „Metrik“ jsem si vyhledal metriky pomocí automatického doplňování a pomocí databázového dotazu (v jazyce PromQL [43] jsem si zvolil, které metrické údaje mi bude graf vracet / vizualizovat).

irate(node_netstat_Udp_InDatagrams{instance=~”$node”}[5m])

Tento dotaz pracuje s metrikou vstupních UDP datagramů (node_netstat_Udp_InDatagrams), která se počítá z instance victim (zastoupená proměnnou node). Nad touto metrikou se počítá rozdíl hodnot v 5minutových intervalech (irate).

Obrázek 20 Implementace grafu, zdroj: vlastní výzkum

Další z hlavních nabídek nastavení grafu se nazývá vizualizace. V této části jsem si vybral vhodný typ grafu, který byl relevantní k typu dat. Zbylé nastavení se týká především vizuální prezentace grafu (jako je tloušťka čáry, podbarvení grafu, atd.).



Obrázek 21 Styl zobrazení vizuálních vlastností grafu, zdroj: vlastní výzkum

Po osvojení si vytváření jednotlivých vizualizací (grafů) jsem si vytvořil grafickou nástěnku všech dostupných metrik.

Na obrázku č. 22 je mnou vytvořená nástěnka podle návodu, který jsem popisoval ve své práci výše.

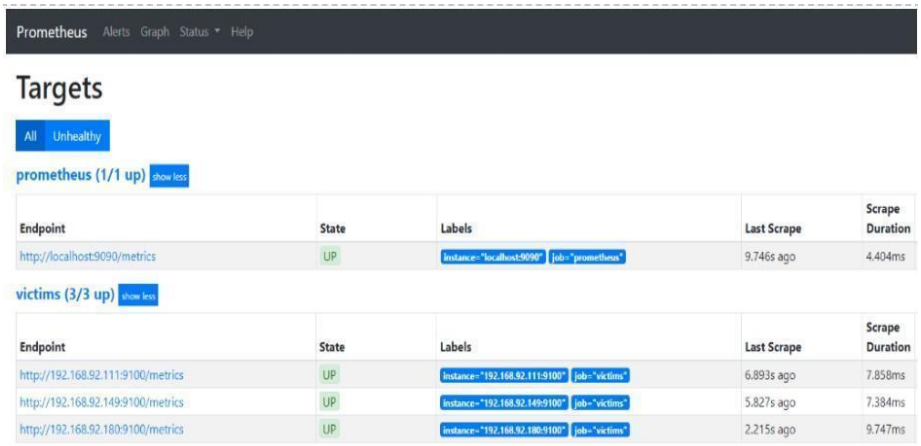
Obrázek 22 Vytvořená vlastní nástěnka, zdroj: vlastní výzkum

5.4 Prometheus

Vybral jsem agregační nástroj Prometheus, jelikož umožňuje automatizovaný sběr a zpracování časových řad. Údaje získává z endpointu node exporteru (o kterém budu hovořit v kapitole Node Exporter 5.5). Nabízí také podporu pro Grafanu a je v ní jako výchozí datový zdroj.

Tato služba sama o sobě umí vizualizovat zpracovaná data, ale nenabízí takovou škálu možností jako Grafana. Proto v naší architektuře vizualizačního řešení slouží jako prostředník, který sbírá data z Node Exporteru a transformuje je do souvislé časové řady, kterou poté načítáme do Grafany.

Prometheus sbírá data tak, že v každém N časovém úseku endpoint Node Exporteru zavolá a provede načtení a zpracování (parsing) dat, která jsou zde uvedena. Jde o bezplatnou softwarovou aplikaci používanou pro monitorování událostí a varování.



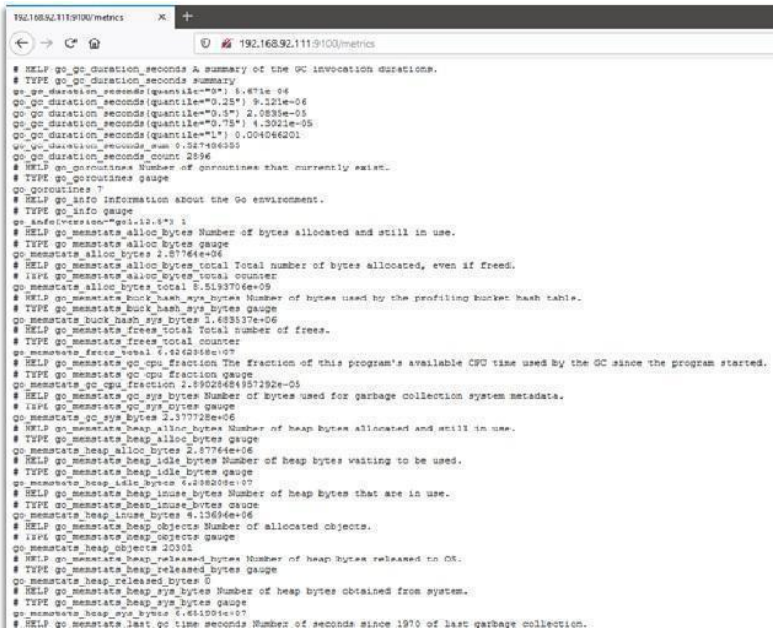
Obrázek 23 Přehled nástroje Prometheus, Zdroj: vlastní výzkum

Nástroj Prometheus jsem nainstaloval na virtuálním počítači monitor-0 z důvodu, aby při útočení nedocházelo k výpadkům dat a vizualizace. Pravidelně sbírá data z jednotlivých virtuálních počítačů victim a ukládá je do své lokální databáze, na kterou se poté napojuje vizualizační nástroj Grafana.

5.5 Node Exporter

Důvodem pro výběr Node Exporteru byla jednoduchost instalace a konfigurace společně s doporučením tohoto projektu v souvislosti s Prometheem. Pro Prometheus nabízí přímou integraci a jde o produkt od stejných tvůrců. Node Exporter také nabízí širokou škálu metrických údajů, které získává přímo ze statistik operačního systému. Je také doporučován v nesčetných návodech pro konfiguraci dohledových systémů, jakožto léty ověřený OSS produkt.

Node exporter jsem použil pro tzv. „vystavení“ metrik systému pod endpointem[7], který je ve výchozí instalaci pod URL http://ip-adresa-počítače:9100/metrics. Přístupem („zavoláním“) k tomuto endpointu dojde k vygenerování sestavy (přehledu) všech podporovaných metrik v systému v rámci jednoho souvislého textu.



Obrázek 24 Surový výpis metrik z virtuálních strojů victim, zdroj: vlastní výzkum

Node exporter jsem nainstaloval na jednotlivé virtuální počítače victim, ze kterých sbírám metrické údaje.

[7] Většinou URL adresa, která je součástí aplikačního rozhraní (interface)

5.6 Provedení experimentů

Pro vizualizaci jednotlivých útoků bylo zapotřebí provést následující kroky:

1. **Výběr útoku** – nejdříve jsem prováděl útoky, které jsem nejvíce teoreticky znal a se kterými jsem měl i praktické zkušenosti. Dále jsem prováděl útoky, kde jsem si musel načíst teorii o tom, jak se daný útok provádí a kde se v systémů projevuje.

2. **Provedení útoku** – druhou fází experimentu je provedení útoku, který jsem vedl z distribuce Kali Linux. Pro jeho provedení jsem potřeboval pro tyto účely nejvhodnější nástroj a potřeboval jsem vědět, jakým způsobem tento nástroj použít. To jsem si dohledal v odborných zdrojích.

3. **Monitorování stavu bez jakéhokoliv jiného provozu –** jedním z důvodů, proč monitorujeme síťový provoz, je detekce anomálií, jak je uvedeno v práci detekce anomálií v lokální síti [44]. Anomálie je událost, která se vymyká stavu, který je definovaný jako normální a běžný. V případě mého výzkumu se za anomálii považuje překročení limitní hodnoty, kterou jsem si nastavil tak, že při překročení této hranice se jedná o stav neobvyklý a dá se považovat za potencionální hrozbu. Monitorování stavu jsem prováděl:

    a) **Při běžném provozu (bez útoků a jiného umělé vytvořeného provozu),** přičemž na začátku experimentu jsem zvolil sledování síťového provozu bez jakéhokoliv uměle vytvořeného provozu, čímž bylo dosaženo maximální citlivosti jednotlivých metrik, a díky tomu jsem mohl zachytit jakoukoliv změnu v metrikách. Cílem bylo sledovat virtuální počítače (oběti) po delší časový úsek, aby se dalo určit, co je běžným stavem sítě v tomto režimu.

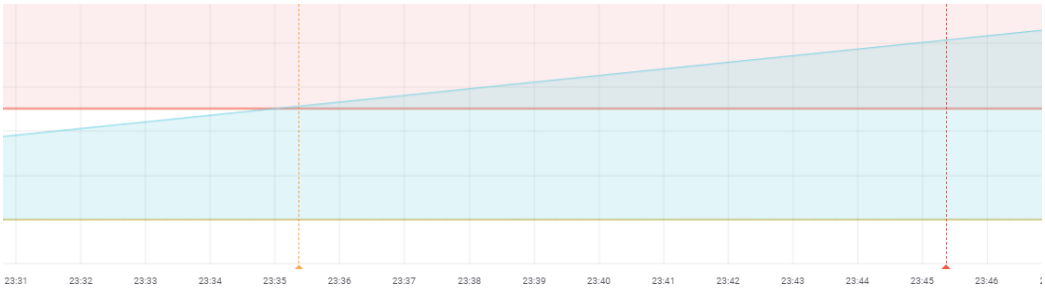
    b) **Při provedení útoku (bez přidání simulace provozu)** jsem, po zjištění, jak se chová běžný stav sítě, začal útočit na prvek v síti. Cílem bylo zjistit, které metriky se projevily jako pozitivní a zajímavé vůči útoku, jenž jsem na systém vedl. Tyto metriky jsem zařadil do užšího výběru ke zkoumání, jelikož by mohly mít vypovídající hodnotu k odhalení útoku.

4. **Nalezení význačných bodů (kde dochází k výkyvům)** - všechny metriky, popsané v kapitole 5.3.2, jsem naimportoval do Grafany, kde poté proběhla jejich vizualizace. Metriky jsem vybral podle následujících kritérií:

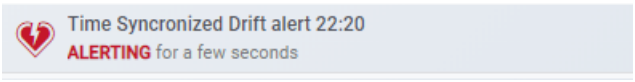
    1. nemají nulová data,
    2. zaznamenaly pohyb (navýšení hodnot) během provádění útoků,
    3. jsou pro celý výzkum jedinečné (některé metriky zobrazují jeden údaj, ale z různých úhlů),
    4. vykazovaly opakované znaky anomálií při provádění útoku.

Za velmi užitečnou funkci Grafany jsem shledal možnost si nastavit upozornění pro každou sledovanou metriku. To mi velmi ulehčovalo práci při hledání anomálií a mohl jsem si dovolit odejít od obrazovky, jelikož mezi funkcemi, které se daly nastavit, byla i automatická notifikace, kde jsem si mohl nadefinovat, komu a jak má přijít upozornění a mohl jsem přidat i text k doručenému upozornění.

Limitní hodnota je znázorněna vodorovnou červenou čarou uprostřed grafu.



Obrázek 25 Překročení prahové hodnoty, zdroj: vlastní výzkum



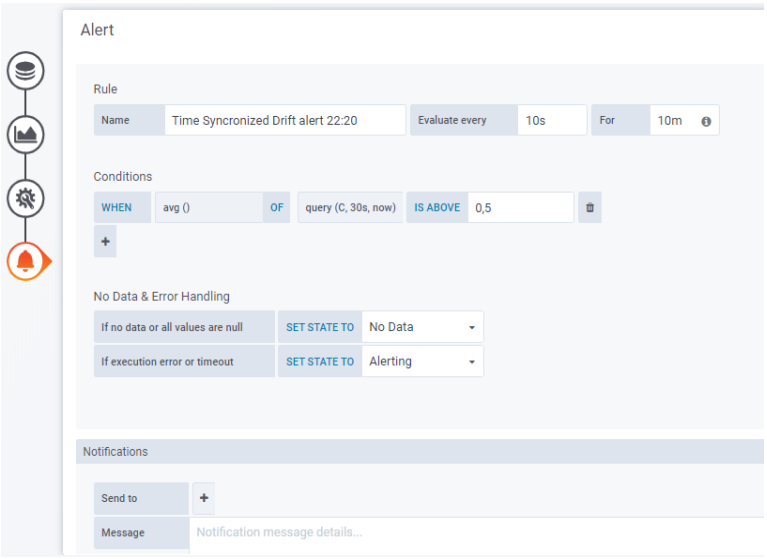
Obrázek 26 Probíhající alert, zdroj: vlastní výzkum

Tato metoda se ukázala jako velice účinná pro **filtraci** nepotřebných metrik, které se v rámci útoku nijak neprojevily a ty, co se projevily, jsem mohl zařadit do složky útoků k dalšímu prozkoumání.

Nastavení upozornění na podezřelé události

Při vytváření vizualizace metrik jsem si také mohl zvolit vytvoření alertu pro danou metriku. Na obrázku č. 27 si můžeme všimnout oranžově zvýrazněného zvonečku, který zastupuje funkci pro vytvoření alertu.

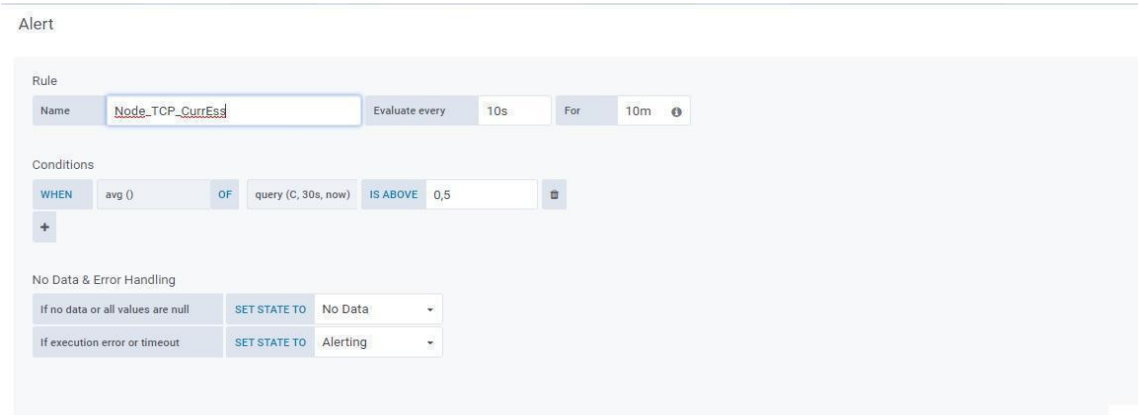
Nejprve jsem nakonfiguroval pravidla pro zavedení alertu. Možnosti této funkce popisuji níže.



Obrázek 27 Pravidla alertu (notifikace), zdroj: vlastní výzkum

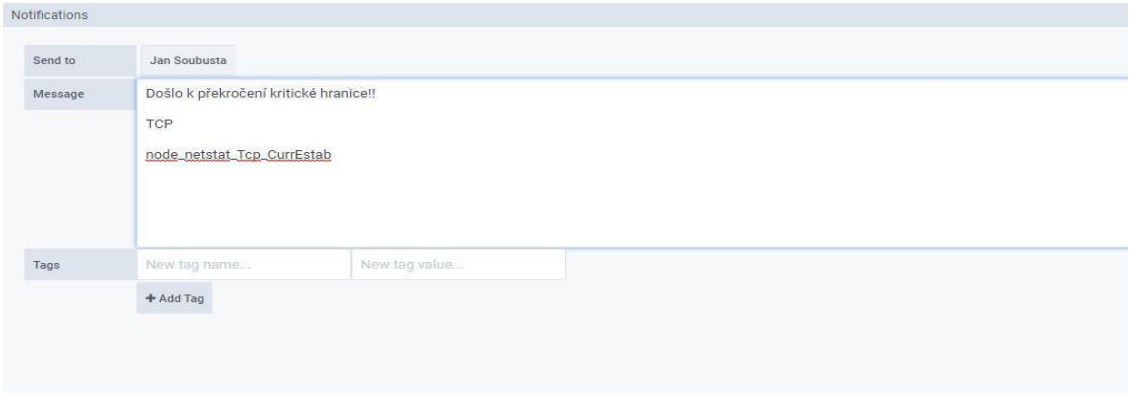
Nabídka nastavení alertu:

- **Name** – jméno alertu;
- **Evaluate every** – spuštění testu (každých 10 s);
- **For** – doba, po kterou, pokud je splněna podmínka (Conditions), dochází k alertu a následně k notifikaci (10 min);
- **Conditions** – logická podmínka – pokud průměr po spuštění testu přesahuje mnou nadefinovanou hodnotu po dobu 10 min, dojde k alertu a pověřená osoba dostane notifikační správu (pokud je nastavena). V této části je také nastavení prahové hodnoty, kterou si můžeme nastavit posuvníkem na grafu nebo uvedením hodnoty do políčka;
- **No data / Error handing** – toto slouží pro případ, když ve vizualizaci nejsou detekovaná data nebo je hodnota nula. V tomto případě máme rozbalovací seznam, kde si můžeme zvolit, co se má stát (bez dat, upozornění, ponechat poslední stav, OK);



Obrázek 28 Ukázka nastavení alertu, zdroj: vlastní výzkum

- Send to – komu se má notifikace zaslat;
- Message – text zprávy, kterou příjemce obdrží.



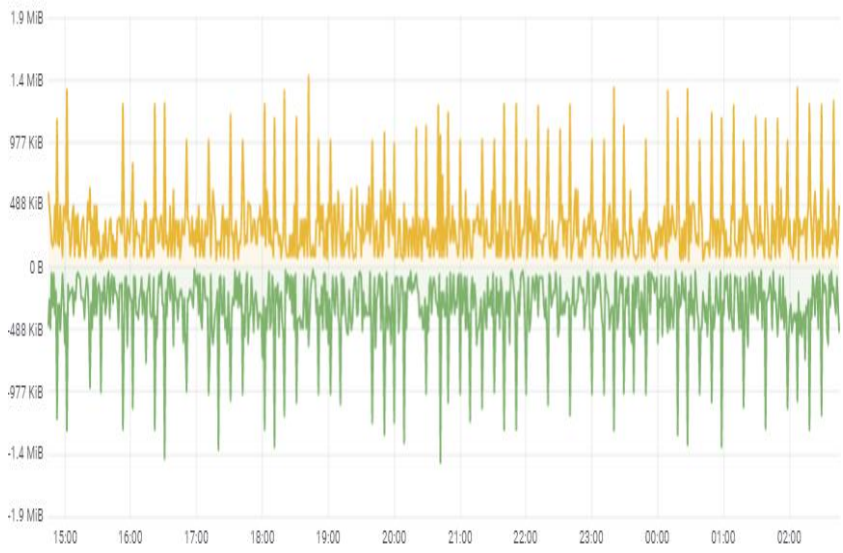
Obrázek 29 Ukázka nastavení alertu (správy pro pověřenou osobu), zdroj: vlastní výzkum

5. **Provedení útoku s generováním obvyklého provozu.** V této části popisuji potřebu generování vedlejšího provozu. Je to proto, že doposud naměřené hodnoty by mohly být zavádějící a mohly by mít špatnou vypovídající hodnotu, jelikož v běžném provozu, který je standartní v síti malých nebo středních firem, by se mohly doposud naměřené hodnoty ztratit a mohly by být od legitimního provozu naprosto nerozpoznatelné. Tímto způsobem jsem docílil toho, že v další selekci metrik jsem mohl související metriky s prováděným útokem filtrovat již tak, aby mohly sloužit k detekci útoku. Pokud bych sledoval pouze nulový provoz a poté výrazný nárůst, byly by výsledky experimentů příliš přímočaré, proto jsem připravil sadu generátorů, pomocí kterých simuluji reálný provoz na síti.

Pomocí vlastního shell skriptu

```
#/bin/bash

# tuto proměnnou získáme z parametrů volání skriptu
cil_provozu=$1

# deklarace proměnných
maximalni_mezera_mezi_iteracemi_s=5
maximalni_prutok_bits=2000
maximalni_doba_prutoku_s=30

# nekonečný cyklus
while :
do
# rychlost dat, které se bude dosahovat
  prutok_bits=`shuf -i 1-$maximalni_prutok_bits -n 1`
  # po jakou dobu bude probíhat odesílání dat
  doba_prutoku=`shuf -i 1-$maximalni_doba_prutoku_s -n 1`
  # jak dlouho bude skript čekat mezi další iterací
  mezera_mezi_iteracemi=`shuf -i 1-
$maximalni_mezera_mezi_iteracemi_s -n 1`
  # spuštění provozu
     iperf3 --client $cil_provozu --time $doba_prutoku --
bandwidth $prutok_bits
  # simulace procházení webové stránky (Apache Bench)
ab -n 100 -c 3 http://$cil_provozu/
  # skript počká po náhodnou dobu před provedením další
iterace
  sleep $mezera_mezi_iteracemi
Done
```

Pomocí tohoto skriptu generuji v náhodných intervalech náhodný počet paketů v síti. Skript je spuštěn ve dvou variantách: a) s TCP; b) s UDP provozem. Jelikož se jedná o nepředvídatelnou náhodu, vytváří tento skript v monitorovacím systému dostatečně zajímavý tok dat – šum.



Obrázek 30 Provoz generovaný nástroji, zdroj: vlastní výzkum

Jelikož tento skript jednoduchý provoz pouze generuje, je dále rozšířen o nástroj Apache Bench.

Apache Bench [45]

Pomocí tohoto programu testuji připojení přímo k webovému serveru, přičemž se přistupuje na webové stránky, které stahuje nebo prochází. Jeho hlavním cílem je testovat propustnost webových stránek. Pokud však náhodně regulujeme dobu běhu a počet spojení, dosáhneme simulace síťového provozu, přičemž se jednotliví klienti připojují k webovému serveru.

IPERF3

Nástroj iperf3 [46] jsem vybral, jelikož s ním mám největší zkušenosti a dá se snadno nainstalovat z repozitářů Ubuntu [47].

Jedná se o nástroj používaný k testování propustnosti sítě, jenž testuje maximální propustnost. Pokud jej pustíme na jeden cíl paralelně vícekrát, efektivně přetěžujeme cílový počítač (podobně jako útok DoS).

Pro účely experimentů konfiguruji nástroj pomocí následujících přepínačů:

1. S – režim serveru je puštěn na straně oběti a čeká na připojení;
2. C – režim klienta, s tímto přepínačem dochází k vysílání paketů;
3. TCP – režim TCP, pokud potřebuji v síti generovat potvrzovaná spojení;
4. UDP – režim UDP, pokud chci jen maximálně vytížit síť;
5. B – počet bajtů za sekundu (bandwidth = šířka pásma), pokud potřebuji regulovat propustnost;
6. T – délka jednoho spuštění využívaná při generování síťového provozu.

6. Monitorování stavu vybraných metrik při generování vedlejšího provozu:

a) Při běžném provozu (bez útoků).

V tomto případě monitorování sítě probíhá na oběti s uměle generovaným provozem, který by se dal přirovnat provozu, jenž se vyskytuje na síti malých a středních firem. To způsobilo, že metriky, jež se projevily v dřívějším experimentu, se s takto produkovaným provozem už nijak významně neprojevily, a tím pro mě přestaly být zajímavé a z užšího výběru metrik jsem je tak mohl vyřadit.

Cílem tedy bylo opět stanovit vývoj normálního stavu sítě se spuštěným skriptem, který simuluje běžný provoz. Normální stav sítě jsem mohl konstatovat opět po delším časovým úseku, kdy jsem sledoval průběh chování sítě.

b) Při provedení útoku.

Cílem bylo zjistit metriky, které se opět výrazně projevily ve vizualizačním nástroji Grafana. Tyto metriky už jsem založil do složky s daným útokem a několikrát jsem testoval, zda mají tyto metriky opravdu vliv na probíhající útok. Po prokázání opakovaného chování na daný útok jsem konstatoval, že s dalšími metrikami je zřejmé, že tyto metriky reagují na provedený útok a lze některé z prováděných útoků detekovat.

6 Vyhodnocení provedených experimentů

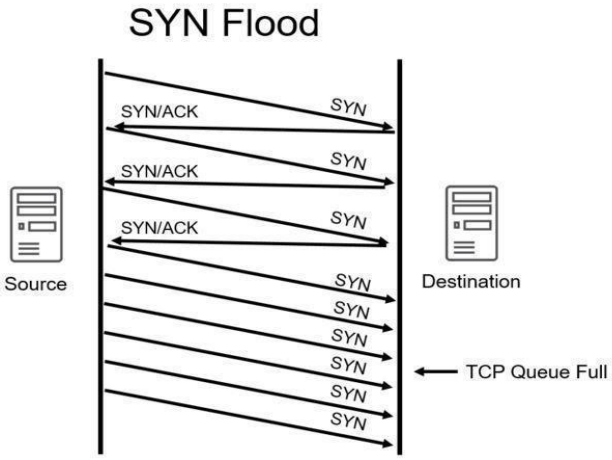
Tato kapitola se věnuje popisu oběti, která byla použita pro experimenty. Tento popis je stejný pro všechny experimenty.

Jedná se o virtuální počítač, jenž má přiřazený výkon 2 jader CPU a 1 GB paměti RAM. SWAP byl zajištěn automaticky pomocí systému Ubuntu 19.04, který nepotřebuje vyčlenit SWAP oddíl s omezenou velikostí, ale podobně jako systém Windows odkládá RAM do souboru, který může růst do teoretické velikosti volného místa na disku. Síťová konektivita je zajištěna pomocí virtuálního switche v Proxmoxu, s teoretickou propustností odpovídající rychlosti síťového rozhraní fyzického serveru, tedy 1 Gbps. Na virtuálním počítači je spuštěna základní sada systémových programů a webových serverů (apache2, mysql, ftp, ssh). Správa počítače byla prováděna pomocí náhledového okna v administračním rozhraní Proxmoxu, kde probíhala i instalace metrického serveru (node exporter) a výpis systémových hodnot, viz obrázky v této kapitole.

6.1 SYN food

Smyslem experimentu bylo detekovat a vizualizovat útok SYN Flood. Ten jsem prováděl pomocí nástroje hping3 a simuloval jsem útočníka, jehož cílem bylo odstavit cílový počítač victim od síťového provozu pomocí pokusů o navázání TCP spojení. Útok byl od začátku tak efektivní, že během provádění experimentu došlo jeho vlivem ke kompletnímu zahlcení domácí sítě.

Jak funguje útok z pohledu útočníka



Obrázek 31 Způsob provádění útoku, zdroj: [48]

Tento útok zneužívá tří-cestného potvrzování při navazování TCP/IP spojení. Útočník zasílá na cílový počítač SYN pakety v co nejrychlejším sledu tak, že zahltí tabulku TCP spojení. Tento útok můžeme provádět v několika variantách:

a) **DDoS** (z více počítačů) - Útočník využívá celou řadu infikovaných počítačů (síť botnet), které najednou zahlcují TCP/IP tabulku oběti. Tento útok můžeme provést tak, že na jednotlivé počítače distribuujeme spustitelný kód (skript, program), který bude útok provádět naráz podle příkazu útočníka. Demonstrovat tento způsob lze tak, že na více virtuálních počítačích naplánujeme provedení příkazu **hping3** v této formě:

```
hping3 -c 20000 -d 120 -S -p 22 --flood ip-adresa-oběti
-a neexistující-adresa
```

b) **DoS** (z jednoho počítače) - Tuto formu útoku spouštíme z jednoho počítače, nejlépe z počítače, jenž je ve stejné síti, kde jsme mohli převzít kontrolu jinou formou útoku. Tento příklad demonstrujeme tak, že v jednom virtuálním počítači spustíme příkaz **hping3** s přidáním dalšího přepínače.

```
hping3 -c 20000 -d 120 -S -p 22 --flood --rand-source
ip-adresa-oběti
```

Jaký program jsem použil a proč

Pro experimenty jsem použil nástroj hping3 z následujících důvodů:

a) je předinstalovaný v OS Kali Linux [49],
b) je široce podporován,
c) má jasně zdokumentované chování a modifikátory (přepínače).

Ve výchozím nastavení tento příkaz funguje na TCP/IP protokolu, přičemž lze toto chování upravit pomocí přepínačů na ICMP nebo UDP. Já používám tento příkaz s následujícím nastavením [50]:

```
hping3 -c 20000 -d 120 -S -p 22 --flood --rand-source ip-adresa-oběti
```

Tabulka 6, X – popis přepínačů pro příkaz hping3, zdroj: vlastní výzkum

Přepínač	Hodnota	Vysvětlení
`-c`	`20000`	Počet paketů, které se pošlou **a zároveň** přijmou. Program poběží do doby, než se potvrdí 20 000 paketů.
`-d`	`120`	Velikost obsahu těla každého odeslaného paketu v bytech. Celková velikost paketu je velká jako velikost hlavičky paketu + velikost obsahu těla paketu.
`-S`		U TCP paketů bude nastaven příznak SYN.
`-p`	`22`	Určuje cílový port, na který se paket zašle. Pro určení čísla portů je vhodné provést nejprve skenování portů na zařízení oběti, abych mohl cílit na konkrétní port. Ve výchozím nastavení je hodnota 0.
`--flood`		Tento přepínač způsobí, že program zasílá pakety co nejrychleji může a nečeká na potvrzení. Tento přepínač je důležitý pro provedení SYN flood útoku.
`--rand-source`		Program pro každý odeslaný paket generuje náhodnou zdrojovou adresu v hlavičce TCP/IP.
`-a`	`222.222.222.222`	Nastavení podržené IP adresy, která se zapíše do hlavičky TCP paketu jako zdrojová. Server oběti se poté bude snažit odpovědět na tuto v síti neexistující adresu.

Tento program byl součástí repositářů většiny populárních linuxových distribucí, např. v Ubuntu:

```
sudo apt-get install hping3 -y
```

Distribuce, kterou používám (Kali Linux), je už součástí výchozí instalace.

Jak se tento útok projevuje na cílovém zařízení oběti

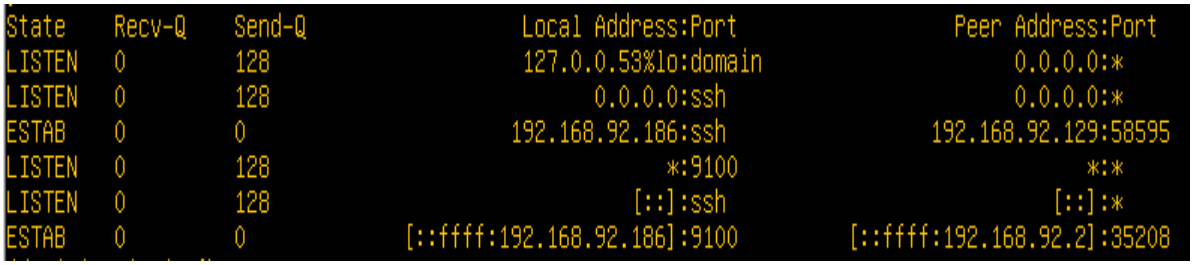
Napadený počítač pracuje se spojením, jako kdyby se jednalo o normální běžný provoz. Podle specifikace TCP/IP protokolu zasílá odpovědi na zdrojovou adresu a čeká potvrzení, které nikdy nepřijde. Tím si zahlcuje tabulku TCP/IP spojení tak dlouho, dokud není schopen přijímat další spojení.

Tabulku spojení na počítači oběti můžeme vizualizovat pomocí Grafany a také vypsat následujícím příkazem **ss** (socket summary), který je nainstalovaný v běžně dostupných distribucích. Příkaz **ss** spouštím s přepínači:

a) -t zobrazit pouze TCP sokety;
b) -a zobrazit sokety se všemi stavy (ve výchozím nastavení jsou pouze navázané).

Následující obrázky ukazují výstup tohoto programu v klidovém stavu.

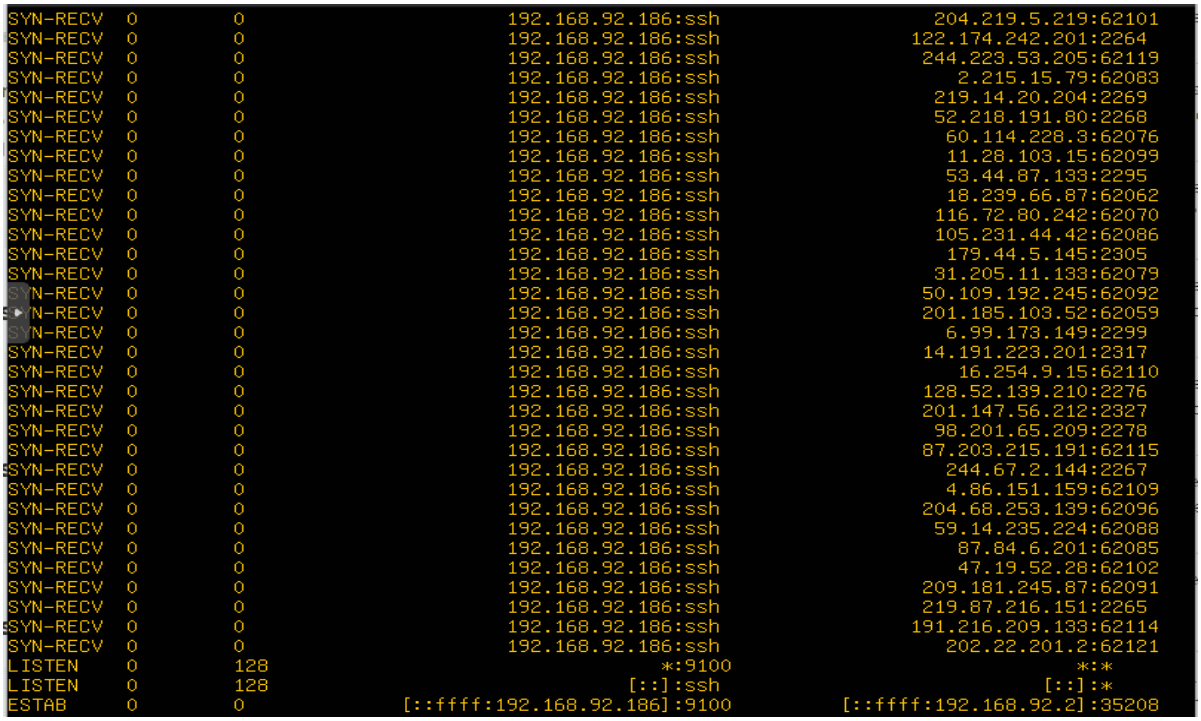
```
ss -t -a # výpis všech TCP soketů
```



State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
LISTEN	0	128	127.0.0.53%lo:domain	0.0.0.0:*
LISTEN	0	128	0.0.0.0:ssh	0.0.0.0:*
ESTAB	0	0	192.168.92.186:ssh	192.168.92.129:58595
LISTEN	0	128	*:9100	*:*
LISTEN	0	128	[::]:ssh	[::]:*
ESTAB	0	0	[::ffff:192.168.92.186]:9100	[::ffff:192.168.92.2]:35208

Obrázek 32 Ukázka v klidovém stavu, zdroj: vlastní výzkum

Při provádění útoku s přepínačem --rand-source poté výpis příkazem **ss -t -a** vypadá následovně:



SYN-RECV	0	0	192.168.92.186:ssh	204.219.5.219:62101
SYN-RECV	0	0	192.168.92.186:ssh	122.174.242.201:2264
SYN-RECV	0	0	192.168.92.186:ssh	244.223.53.205:62119
SYN-RECV	0	0	192.168.92.186:ssh	2.215.15.79:62083
SYN-RECV	0	0	192.168.92.186:ssh	219.14.20.204:2269
SYN-RECV	0	0	192.168.92.186:ssh	52.218.191.80:2268
SYN-RECV	0	0	192.168.92.186:ssh	60.114.228.3:62076
SYN-RECV	0	0	192.168.92.186:ssh	11.28.103.15:62099
SYN-RECV	0	0	192.168.92.186:ssh	53.44.87.133:2295
SYN-RECV	0	0	192.168.92.186:ssh	18.239.66.87:62062
SYN-RECV	0	0	192.168.92.186:ssh	116.72.80.242:62070
SYN-RECV	0	0	192.168.92.186:ssh	105.231.44.42:62086
SYN-RECV	0	0	192.168.92.186:ssh	179.44.5.145:2305
SYN-RECV	0	0	192.168.92.186:ssh	31.205.11.133:62079
SYN-RECV	0	0	192.168.92.186:ssh	50.109.192.245:62092
SYN-RECV	0	0	192.168.92.186:ssh	201.185.103.52:62059
SYN-RECV	0	0	192.168.92.186:ssh	6.99.173.149:2299
SYN-RECV	0	0	192.168.92.186:ssh	14.191.223.201:2317
SYN-RECV	0	0	192.168.92.186:ssh	16.254.9.15:62110
SYN-RECV	0	0	192.168.92.186:ssh	128.52.139.210:2276
SYN-RECV	0	0	192.168.92.186:ssh	201.147.56.212:2327
SYN-RECV	0	0	192.168.92.186:ssh	98.201.65.209:2278
SYN-RECV	0	0	192.168.92.186:ssh	87.203.215.191:62115
SYN-RECV	0	0	192.168.92.186:ssh	244.67.2.144:2267
SYN-RECV	0	0	192.168.92.186:ssh	4.86.151.159:62109
SYN-RECV	0	0	192.168.92.186:ssh	204.68.253.139:62096
SYN-RECV	0	0	192.168.92.186:ssh	59.14.235.224:62088
SYN-RECV	0	0	192.168.92.186:ssh	87.84.6.201:62085
SYN-RECV	0	0	192.168.92.186:ssh	47.19.52.28:62102
SYN-RECV	0	0	192.168.92.186:ssh	209.181.245.87:62091
SYN-RECV	0	0	192.168.92.186:ssh	219.87.216.151:2265
SYN-RECV	0	0	192.168.92.186:ssh	191.216.209.133:62114
SYN-RECV	0	0	192.168.92.186:ssh	202.22.201.2:62121
LISTEN	0	128	*:9100	*:*
LISTEN	0	128	[::]:ssh	[::]:*
ESTAB	0	0	[::ffff:192.168.92.186]:9100	[::ffff:192.168.92.2]:35208

Obrázek 33 Stav, kdy útočím na SSH, zdroj: vlastní výzkum

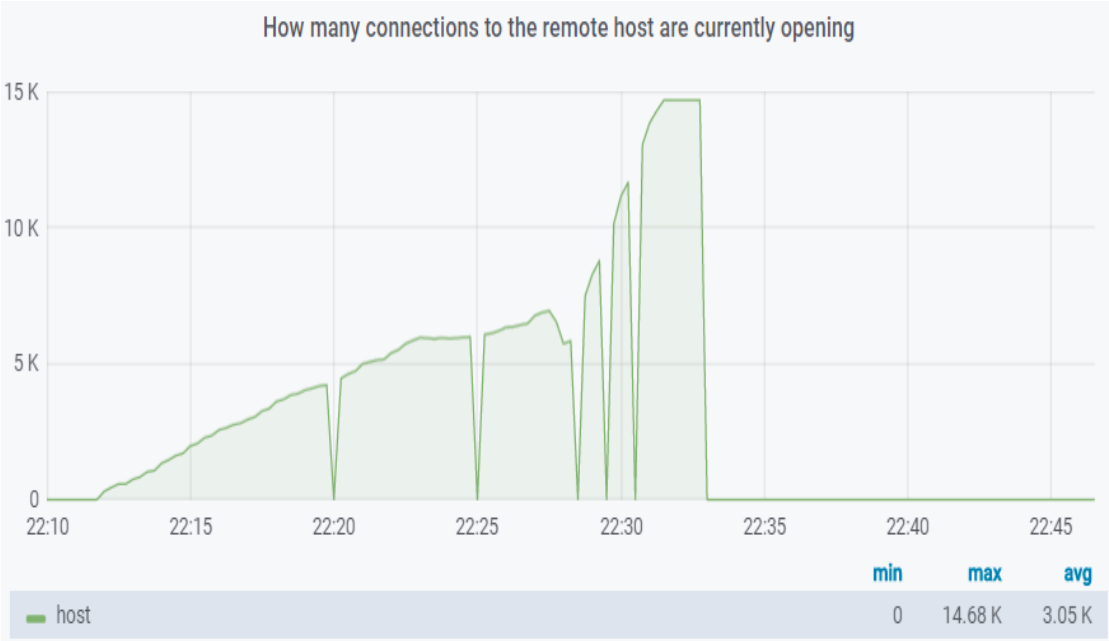
Při obou formách útoku můžeme v Grafaně pozorovat změny stavů těchto metrik:

a) Počet TCP spojení a jejich stavy

Pomocí pluginu conntrack_exporter exportujeme stavy, v jakých se nachází TCP spojení: OPEN, OPENING, CLOSING. Tato metrika je stěžejní pro zachycení tohoto útoku, jelikož se zde útok projevuje tím, že má velký počet SYN-RECV segmentů.

Na následujících dvou grafech jsou statistiky TCP spojení. U všech grafů je vidět výpadek, kdy došlo k odstavení victim, a tím pádem se statistiky přestaly sbírat.

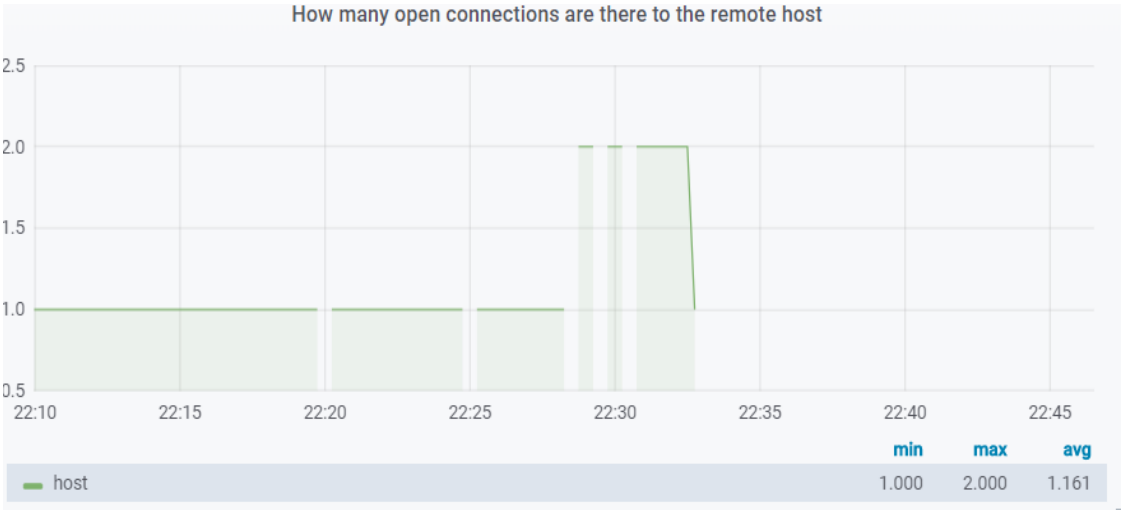
První graf nám zobrazuje metrický údaj o navázání spojení, na ose X čas a na ose Y počet pokusů. Pokud tato metrika výrazně roste v čase, pravděpodobně jde o útok SYN-FLOOD. Jedná se o jednu z indikujících metrik.



Obrázek 34 Na tomto grafu je přehled spojení, která jsou aktuálně ve stavu OPENING, zdroj: vlastní výzkum

Druhý graf zobrazuje počet spojení aktuálně otevřených na serveru. To znamená spojení, která jsou úspěšně navázaná. Na grafu jsou vidět výpadky, které vznikly během útoku. Výpadky jsou způsobeny tím, že virtuální stroj už nestíhal odpovídat na dotazy na endpoint s metrikami.

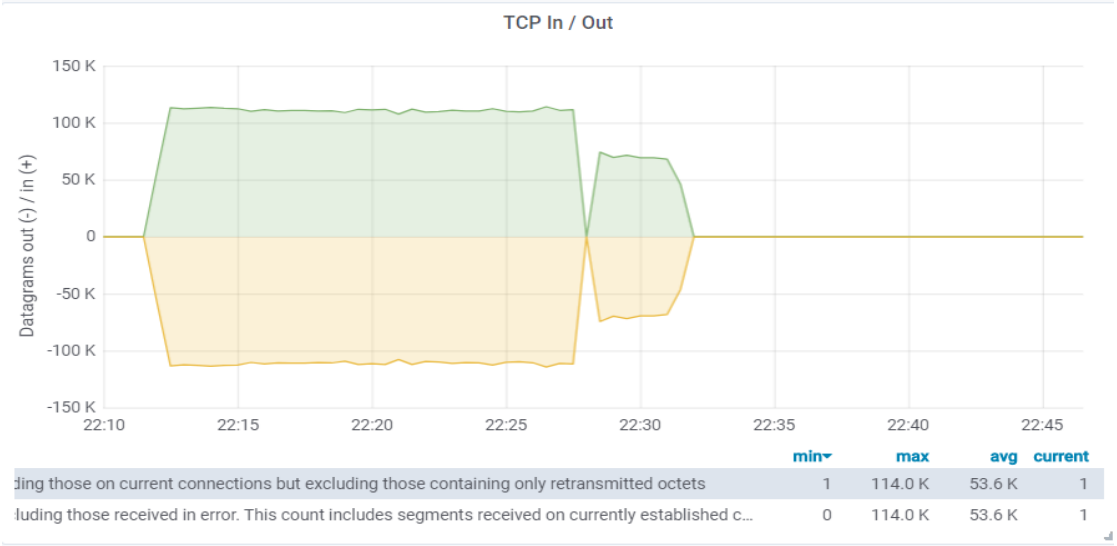
Můžeme si všimnout, že je zde výrazný rozdíl mezi pokusem o navázání spojení a počtem navázaných spojení. Z toho lze usoudit, že se jedná o SYN flood útok. Osa X uvádí čas, osa Y uvádí počet navázaných spojení.



Obrázek 35 Počet navázaných TCP spojení, zdroj: vlastní výzkum

b) Počet příchozích a odchozích TCP/IP segmentů

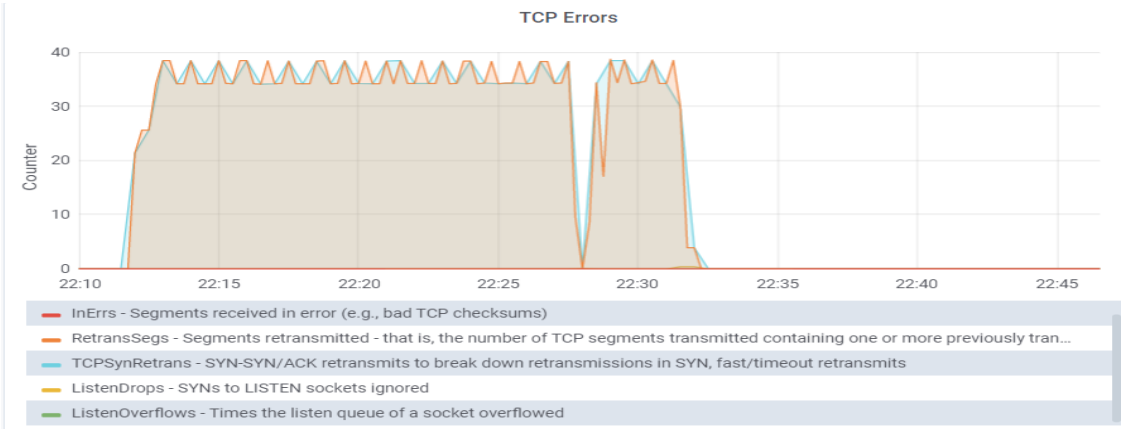
Tato metrika je první na ráně, jelikož zobrazuje hrubá data o tom, kolik TCP segmentů dorazilo na síťové rozhraní počítače a kolik se odeslalo. Oproti běžnému provozu se liší rychlostí, jakou se změní její stav a počtem spojení, který je výrazně jiný. Důležité je sledovat souvislost s ostatními metrikami, rychlost náběhu a rozdíl oproti běžnému provozu. Osa X uvádí čas, osa Y uvádí počet přijatých a odeslaných Datagramů.



Obrázek 36 Metrika, která vizualizuje množství příchozích a odchozích TCP/IP segmentů, zdroj: vlastní výzkum

c) Počet opakovaných přenosů TCP segmentů

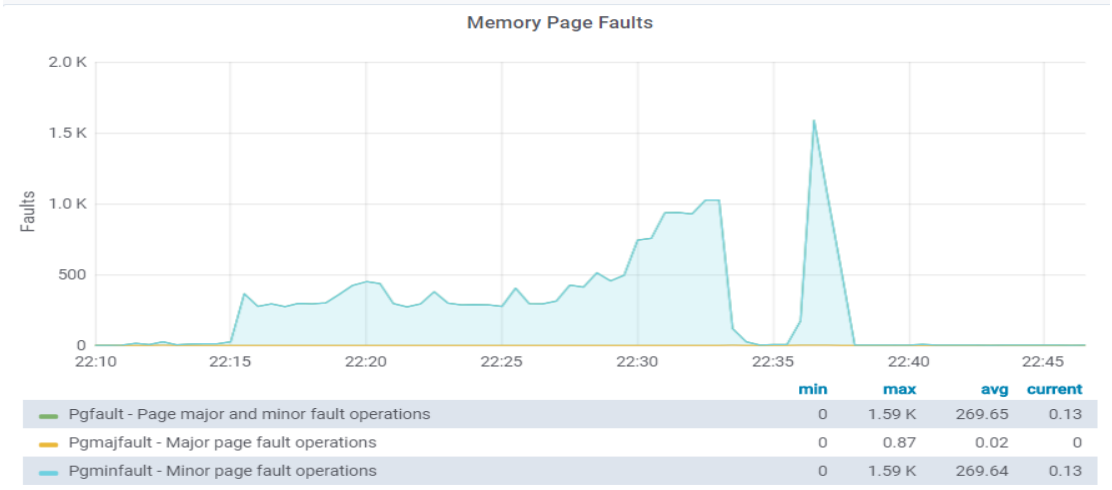
Z podstaty útoku na potvrzované TCP spojení se počítač oběti nedočká potvrzení navázání spojení, a proto posílá SYN-ACK pakety znovu. Tento jev zpravidla vzniká při přetížení sítě nebo výpadku konektivity, kdy se počítač snaží navázat spojení. Ale jelikož data od útočníka k oběti dorazí, tak víme, že tato metrika reaguje na přetížení sítě a je pro nás dobrým ukazatelem tohoto útoku. Osa X uvádí čas, osa Y uvádí počet opakovaných přenosů TCP segmentů.



Obrázek 37 Počet opakovaných přenosů TCP segmentů, zdroj: vlastní výzkum

d) Chyby v paměťových stránkách

Během experimentů došlo k zahlcení paměti RAM a došlo k výraznému nárůstu odkládání do SWAPu. Operační systém nestačil odkládat a načítat data ze SWAP oddílu a procesor, který si žádal konkrétní bloky paměti, je od manažera paměti nedostal a logoval chyby. Na tuto metriku může mít vliv více oddílů, proto zdůrazňuji, že je třeba na ni nahlížet v souvislosti s ostatními metrikami v této kapitole.

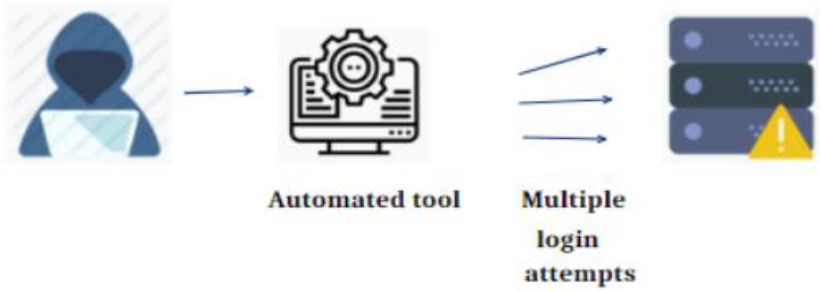


Obrázek 38 Chyby stránkování paměti, zdroj vlastní výzkum

6.2 Útok na prolomení hesla SSH služby

Smyslem experimentu bylo detekovat a vizualizovat útok na SSH autentizaci jménem a heslem. Pomocí nástroje Hydra jsem simuloval útočníka, jehož cílem je uhádnout heslo SSH služby s cílem ovládnout cílový počítač victim.

Jak funguje útok z pohledu útočníka



Obrázek 39 Způsob provedení útoku, zdroj: [51]

Běžně je nejprve nutné zjistit, na jakém portu naslouchá **SSH** služba. Můžeme vyzkoušet standardní port **22,** nebo pokud není dostupný, tak najít alternativní port utilitou **nmap**. Po zjištění portu můžeme zkusit slovníkový útok nebo útok hrubou silou (zkoušení všech možných kombinací znaků), abychom uhádli heslo uživatele.

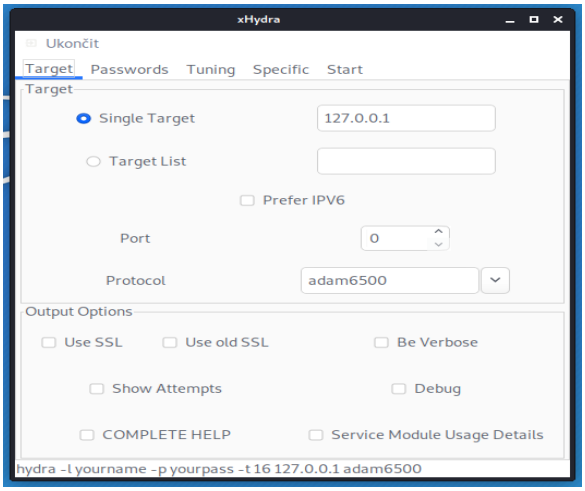
Jaký program jsem použil a proč

Pro experimenty jsem použil nástroj **xHydra** [52] z následujících důvodů:

a) je předinstalovaný v OS Kali Linux,

b) mám s ním dobré zkušenosti,

c) jedná se o desktopovou aplikaci s dobře zdokumentovaným nastavením.

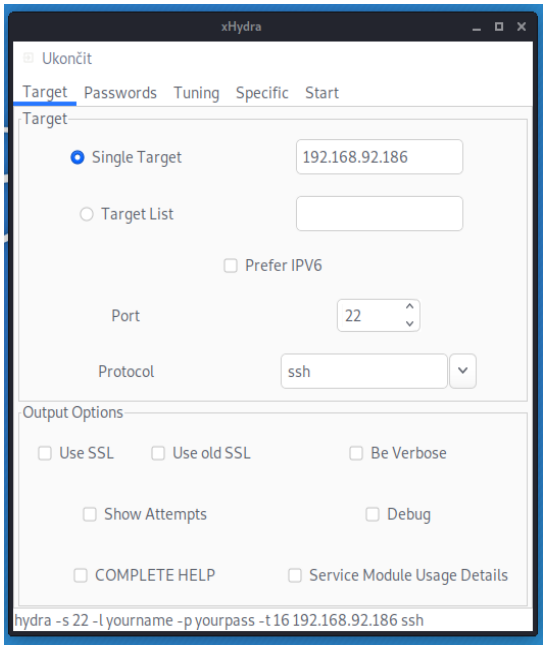


Obrázek 40 Výchozí nastavení nástroje xHydra, zdroj: vlastní výzkum

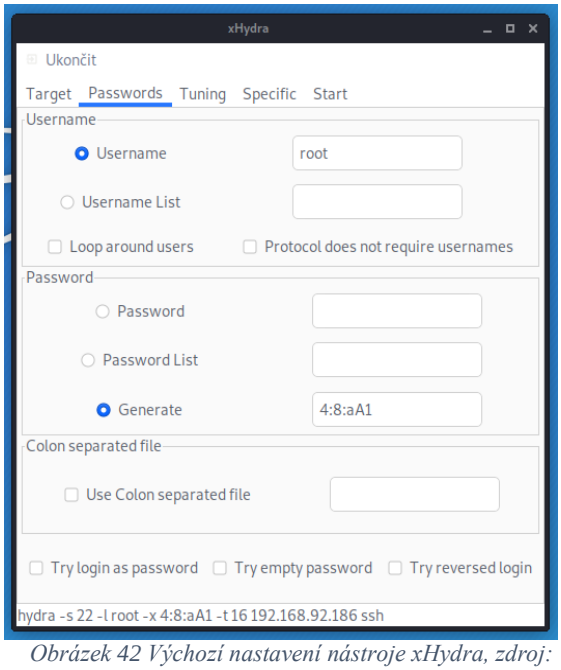
Tento nástroj nabízí možnost provést útok na jeden cíl (Single Target) nebo seznam cílů (Target List). Umožňuje také nastavit port (v našem případě 22), na který chceme cílit a můžeme si zvolit protokol, který chceme použít (v našem případě ssh).

Uživatele a hesla můžeme definovat v záložce Passwords, kde můžeme nechat program generovat náhodné řetězce nebo použít seznam předdefinovaných uživatelských jmen a hesel.

Pro potřeby mé práce a názornosti ukázky používám program s následujícím nastavením:



Obrázek 41 Výchozí nastavení nástroje xHydra, zdroj: vlastní výzkum



Obrázek 42 Výchozí nastavení nástroje xHydra, zdroj: vlastní výzkum

Nastavena je IP adresa počítače victim (192.168.92.186), známý port SSH serveru (22), uživatelské jméno (root) a zadali jsme kód pro generování hesel (4:8:aA1). Kód pro generování je nastaven tak, aby náhodně generoval hesla, která jsou minimálně 4 znaky krátká a 8 znaků dlouhá a obsahují malé znaky (a), velké znaky (A) a číslice (1). Příkaz, který se poté spustí, vypadá takto:

```
hydra -s 22 -l root -x 4:8:aA1 -t 4 192.168.92.186 ssh
```

V tomto příkazu si můžeme povšimnout přepínače -t 4, kterým specifikujeme počet paralelních spojení na ssh server.

Během provádění útoku je možné sledovat výpisy nástroje:

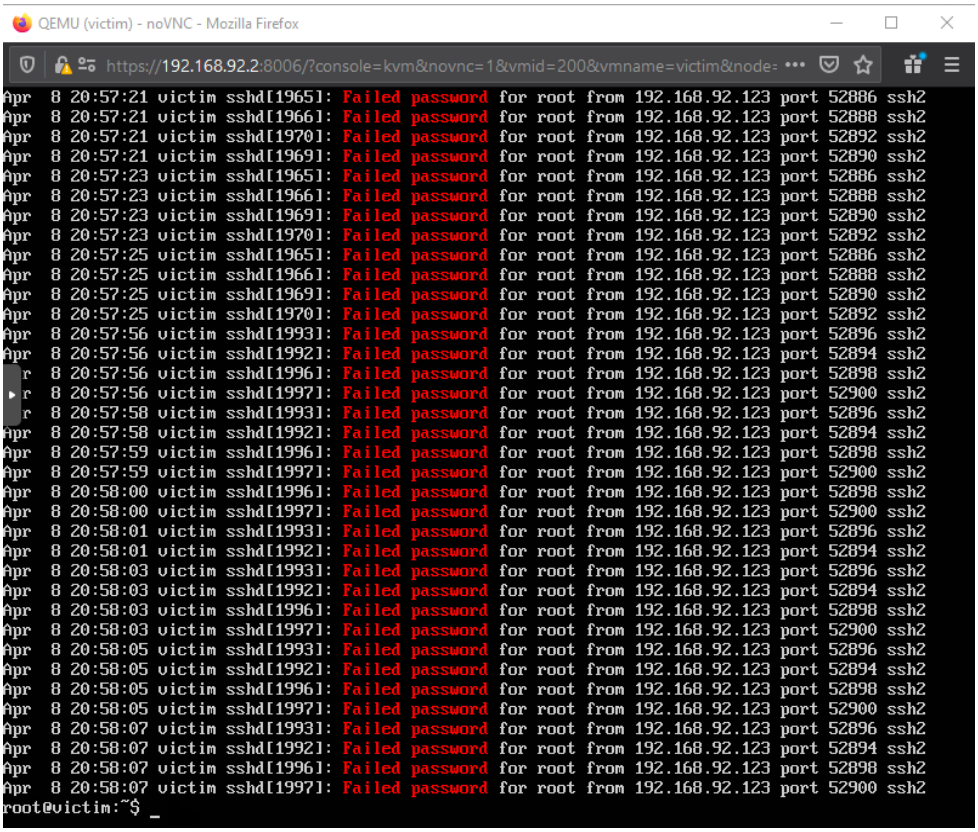
```
Output
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-04-08 20:55:39
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting
[DATA] max 4 tasks per 1 server, overall 4 tasks, 100000000 login tries (l:1/p:100000000), ~25000000 tries per task
[DATA] attacking ssh://192.168.92.186:22/
[STATUS] 44.00 tries/min, 44 tries in 00:01h, 99999956 to do in 37878:47h, 4 active
[STATUS] 34.67 tries/min, 104 tries in 00:03h, 99999896 to do in 48076:53h, 4 active
[ERROR] Can not create restore file (./hydra.restore) - Permission denied
[STATUS] 29.14 tries/min, 204 tries in 00:07h, 99999796 to do in 57189:26h, 4 active
```

Obrázek 43 Výstup nástroje Hydra, zdroj: vlastní výzkum

Jak se projevuje tento útok na cílovém zařízení oběti

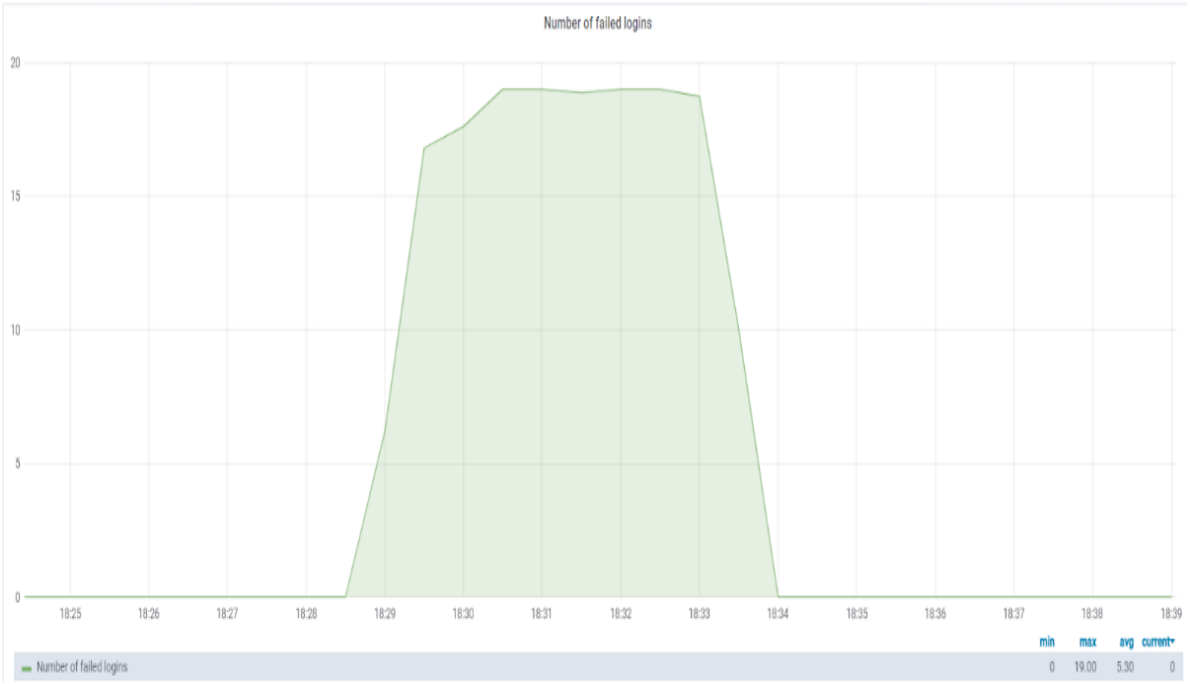
SSH server si loguje každé neúspěšné přihlášení, proto je možné výsledek útoku pozorovat i ve výpisu logu pomocí příkazu:

```
grep "Failed password" /var/log/auth.log
```



Obrázek 44 Průběh útoku Brute force, zdroj: vlastní výzkum

Tyto logy si také můžeme načítat a vizualizovat počet neúspěšných přihlášení. Osa X uvádí čas, osa Y uvádí počet neúspěšných přihlášení.



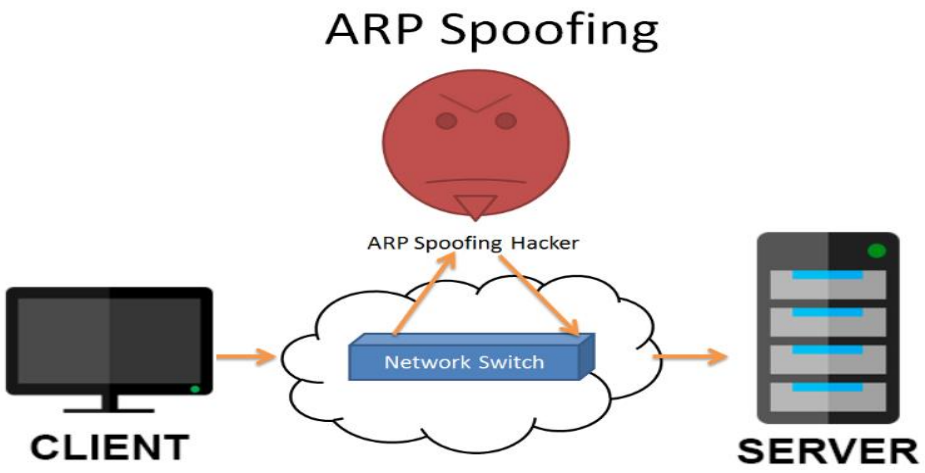
Obrázek 45 Počet neúspěšných přihlášený pro službu SSH, zdroj: vlastní výzkum

V této formě útoku tedy došlo ke změně stavu metriky **Počet neúspěšných přihlášení.** Tato metrika je již všeříkající. Pozoroval jsem i metriku **Počet TCP Spojení**, kde nedošlo k žádnému průkaznému jevu. U ostatních obecných metrik, jako je např. **Využití CPU** nebo **Využití paměti RAM**, nedošlo k žádné výrazné změně. Zvažoval jsem i metriku, která by sledovala počet přenesených dat na jednotlivých portech, a tím bych byl schopen i určit, že na portu, kde běží SSH, se přeneslo velké množství dat. Avšak pomocí nástrojů, které používám, nejsem schopný rozlišit příchozí pakety podle portu (služby), na který přišly.

6.3 Útok ARP spoofing

Smyslem experimentu bylo detekovat provádění útoku main-in-the-middle na síti pomocí ARP spoofingu. Tento útok je nebezpečný, jelikož útočník se může v síti vydávat za důvěryhodné zařízení, a tím pádem zachytávat komunikaci. Proto je významnost zachycení tohoto útoku velká.

Jak funguje útok z pohledu útočníka



Obrázek 46 Ukázka ARP spoofing, zroj: [53]

Útočník nejprve musí zjistit IP adresu oběti, za kterou se chce vydávat vůči jinému počítači v síti (např. webovému serveru). Tomuto útoku může předcházet skenování sítě, kdy zjistíme, které IP adresy jsou aktivní, případně si vytipujeme, jakou IP adresu chceme spoofowat. Principiálně se jedná o to, že útočník (za použití programu Ettercap), který chce zachytávat pakety oběti, neustále zasílá odpověď na ARP se svou MAC adresou a zařízení oběti si uloží podvrženou MAC adresu a všechny pakety směřuje na ni. Útočník poté již může fungovat jako proxy, navázat spojení se serverem a přeposílat je oběti, zatímco je upravuje nebo analyzuje.

Jaký program jsem použil a proč

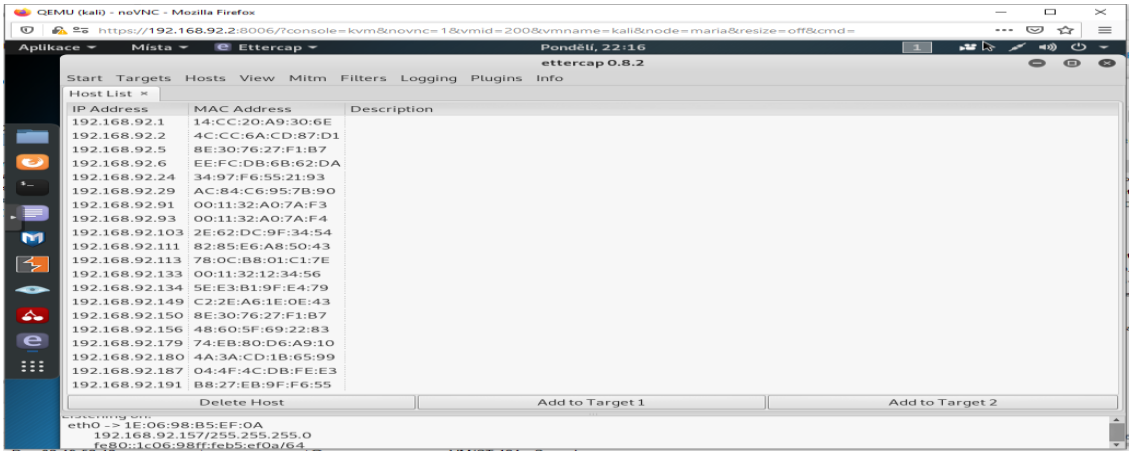
Pro experimenty jsem použil nástroj Ettercap [54] z následujících důvodů:

a) je předinstalovaný v OS Kali Linux,

b) jedná se o desktopovou aplikaci s dobře zdokumentovaným nastavením.

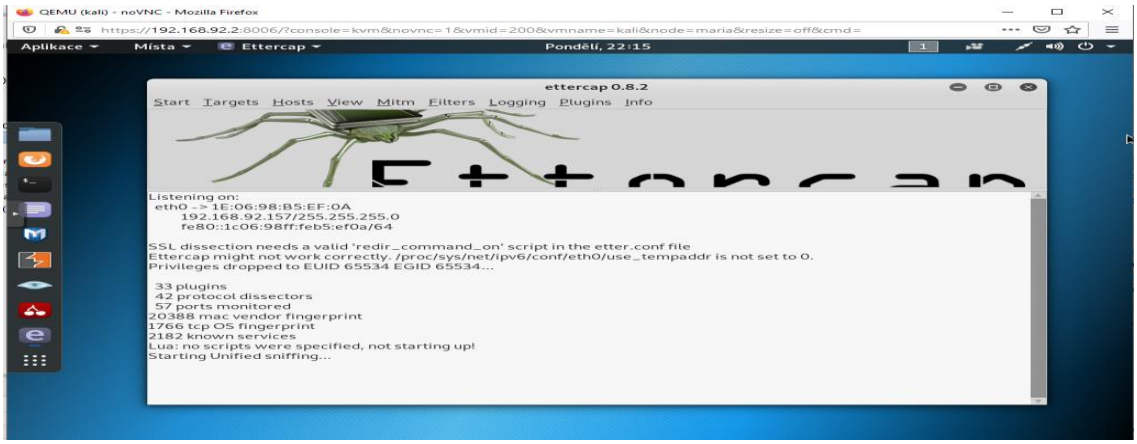
Obrázek 47 Spouštění nástroje Ettercap, zdroj: vlastní výzkum

Tento nástroj jsem spustil tak, že jsem nejprve skenoval dostupné IP adresy:



Obrázek 48 Skenování dostupných IP adres, zdroj: vlastní výzkum

Poté jsem přidal IP adresu počítače victim (192.168.92.xxx) jako target a spustil útok MITM (Man In The Middle) Attack: ARP Poisoning.

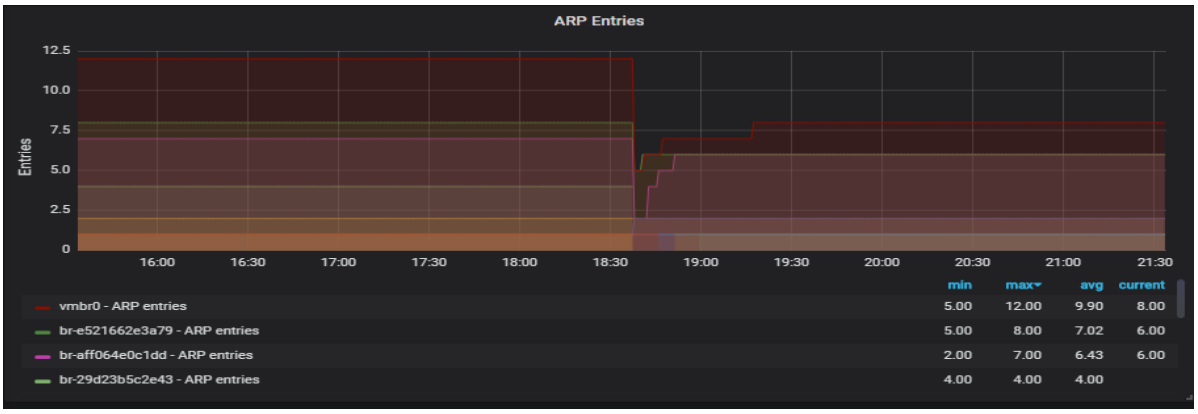


Obrázek 49 Průběh útoku ARP spoofing, zdroj: vlastní výzkum

V prvních 3 minutách docházelo k výkyvům v metrice ARP entries, která zobrazuje ARP záznamy na počítači.

Jak se projevuje ARP útok na cílovém zařízení oběti

V běžném provozu dochází v pravidelných intervalech k následujícím výkyvům v ARP tabulce počítače victim:
Osa X uvádí čas, osa Y uvádí počet záznamů v ARP tabulce.



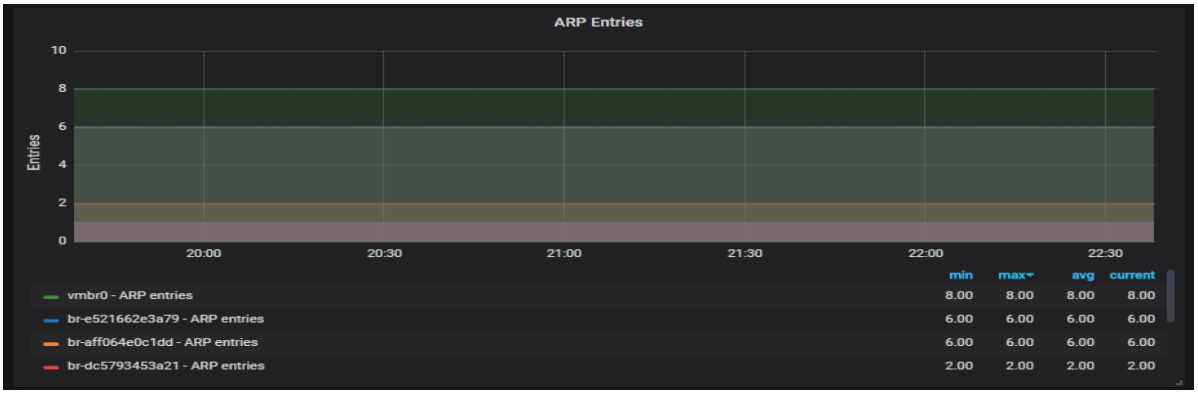
Obrázek 50 Počet záznamů v ARP tabulce, zdroj vlastní výzkum

Tento graf zobrazuje metriku **Počet ARP záznamů na zařízení victim**. Počet ARP záznamů je za každý jeden bod na ose X.

Tyto záznamy jsou způsobeny tím, že ARP tabulka se v Linuxu pravidelně pročišťuje a pokud nějaké zařízení dlouho nekomunikuje s počítačem victim, tak se jeho záznam vymaže. Následujícím příkazem můžeme ověřit, jak dlouho se záznamy cachují:

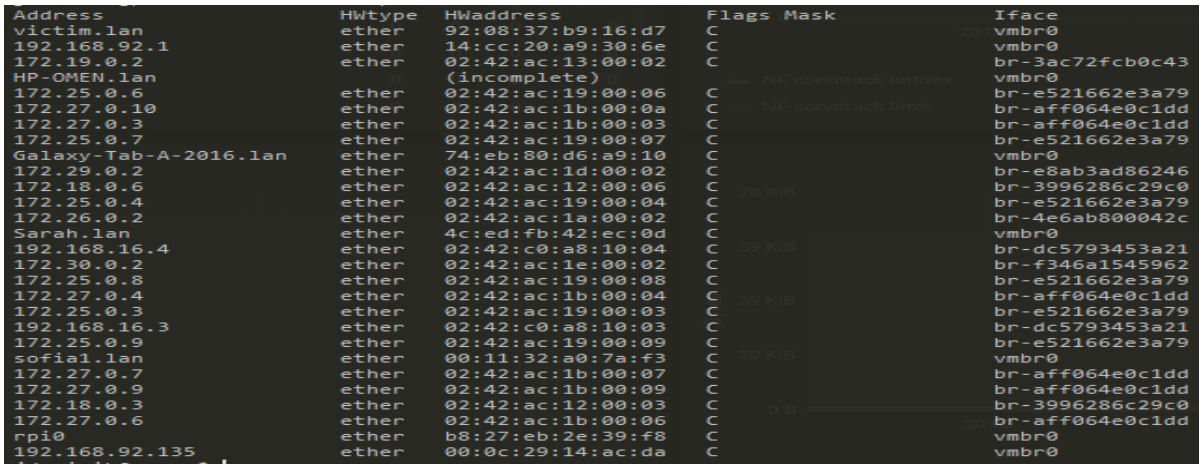
```
cat /proc/sys/net/ipv4/neigh/default/gc_stale_time
```

Příkaz vypíše hodnotu 60, což znamená, že na zařízení victim se záznamy cachují po dobu 60 sekund a poté se vymažou. Během útoku však nedošlo k žádné změně v počtu ARP záznamů.



Obrázek 51 Metrika počet ARP záznamů, zdroj: vlastní výzkum

S touto metrikou také souvisí výpis ARP tabulky v konzoli:



Address	HWtype	HWaddress	Flags	Mask	Iface
victim.lan	ether	92:08:37:b9:16:d7	C		vmbr0
192.168.92.1	ether	14:cc:20:a9:30:6e	C		vmbr0
172.19.0.2	ether	02:42:ac:13:00:02	C		br-3ac72fcb0c43
HP-OMEN.lan		(incomplete)			vmbr0
172.25.0.6	ether	02:42:ac:19:00:06	C		br-e521662e3a79
172.27.0.10	ether	02:42:ac:1b:00:0a	C		br-aff064e0c1dd
172.27.0.3	ether	02:42:ac:1b:00:03	C		br-aff064e0c1dd
172.25.0.7	ether	02:42:ac:19:00:07	C		br-e521662e3a79
Galaxy-Tab-A-2016.lan	ether	74:eb:80:d6:a9:10	C		vmbr0
172.29.0.2	ether	02:42:ac:1d:00:02	C		br-e8ab3ad86246
172.18.0.6	ether	02:42:ac:12:00:06	C		br-3996286c29c0
172.25.0.4	ether	02:42:ac:19:00:04	C		br-e521662e3a79
172.26.0.2	ether	02:42:ac:1a:00:02	C		br-4e6ab800042c
Sarah.lan	ether	4c:ed:fb:42:ec:0d	C		vmbr0
192.168.16.4	ether	02:42:c0:a8:10:04	C		br-dc5793453a21
172.30.0.2	ether	02:42:ac:1e:00:02	C		br-f346a1545962
172.25.0.8	ether	02:42:ac:19:00:08	C		br-e521662e3a79
172.27.0.4	ether	02:42:ac:1b:00:04	C		br-aff064e0c1dd
172.25.0.3	ether	02:42:ac:19:00:03	C		br-e521662e3a79
192.168.16.3	ether	02:42:c0:a8:10:03	C		br-dc5793453a21
172.25.0.9	ether	02:42:ac:19:00:09	C		br-e521662e3a79
sofia1.lan	ether	00:11:32:a0:7a:f3	C		vmbr0
172.27.0.7	ether	02:42:ac:1b:00:07	C		br-aff064e0c1dd
172.27.0.9	ether	02:42:ac:1b:00:09	C		br-aff064e0c1dd
172.18.0.3	ether	02:42:ac:12:00:03	C		br-3996286c29c0
172.27.0.6	ether	02:42:ac:1b:00:06	C		br-aff064e0c1dd
rpi0	ether	b8:27:eb:2e:39:f8	C		vmbr0
192.168.92.135	ether	00:0c:29:14:ac:da	C		vmbr0

Obrázek 52 Výpis ARP tabulky, zdroj: vlastní výzkum

Empirickým zkoumáním jsem došel k závěru, že mnou zvolenou metodikou a vizualizací není možné detekovat útok ARP spoofing, jelikož nedochází ke změně počtu ARP záznamů. A i v případě, kdy měření metriky počtu ARP záznamů změním na měření změny za časový úsek, tak ta změna je tak nepatrná, že ji nelze objektivně a spolehlivě detekovat. Jednou z možností by bylo sledovat změny MAC záznamů u jednotlivých IP adres v ARP tabulce, které však nástroj node exporter neumí sledovat a detekovat – jediná poskytovaná metrika je počet záznamů v celé ARP tabulce.

6.4 Útok skenování portů (SYN sken)

Smyslem experimentu bylo provést a detekovat útok, při kterém útočník skenuje síťové rozhraní cílového zařízení s cílem zjistit otevřené TCP/UDP porty, ze kterých by poté mohl získat další informace. Provedl jsem SYN sken variantu nastavení nástroje nmap, která je opakem TCP Connect Scan. Při ní se neotevírají spojení s cílovým zařízením.

Tento útok slouží jako základ pro další útoky, jelikož otevírá dveře útočníkovi k proniknutí do systému. Souvisí tedy s ostatními útoky, které jsem v rámci bakalářské práce provedl.

Jak funguje útok z pohledu útočníka

Útočník skenuje porty proto, aby zjistil, které služby na jakých portech naslouchají, aby mohl zjistit verzi služeb a s pomocí databáze zranitelností efektivně útočit. Tento útok je možné provádět v několika variantách:

a) základní sken:

`nmap ip-adresa-oběti`

Tento příkaz spustí sken známých (well-know) portů na ip adrese oběti a slouží jako jednoduchý průzkum.

b) sken specifických portů:

`nmap -p 1-65535 ip-adresa-oběti`

Tímto příkazem definujeme rozsah portů, které chceme skenovat.

c) sken celých síťových rozsahů:

`nmap -p ip-adresa-oběti/24`

Tímto příkazem můžeme skenovat více počítačů najednou.

d) sken nejpoužívanějších portů:

`nmap --top-ports 20 ip-adresa-oběti`

Pokud provádíme sken větších sítí, můžeme zrychlit skenování vybráním nejpoužívanějších portů.

Jelikož skenování větších rozsahů portů a IP adres trvá dlouho, můžeme skenování provádět i z vícero počítačů například tak, že si rozsah portů rozdělíme. Například:

- Útočník 1: **`nmap -p 1-20000 ip-adresa-oběti`**
- Útočník 2: **`nmap -p 20001-40000 ip-adresa-oběti`**
- Útočník 3: **`nmap -p 40001-65535 ip-adresa-oběti`**

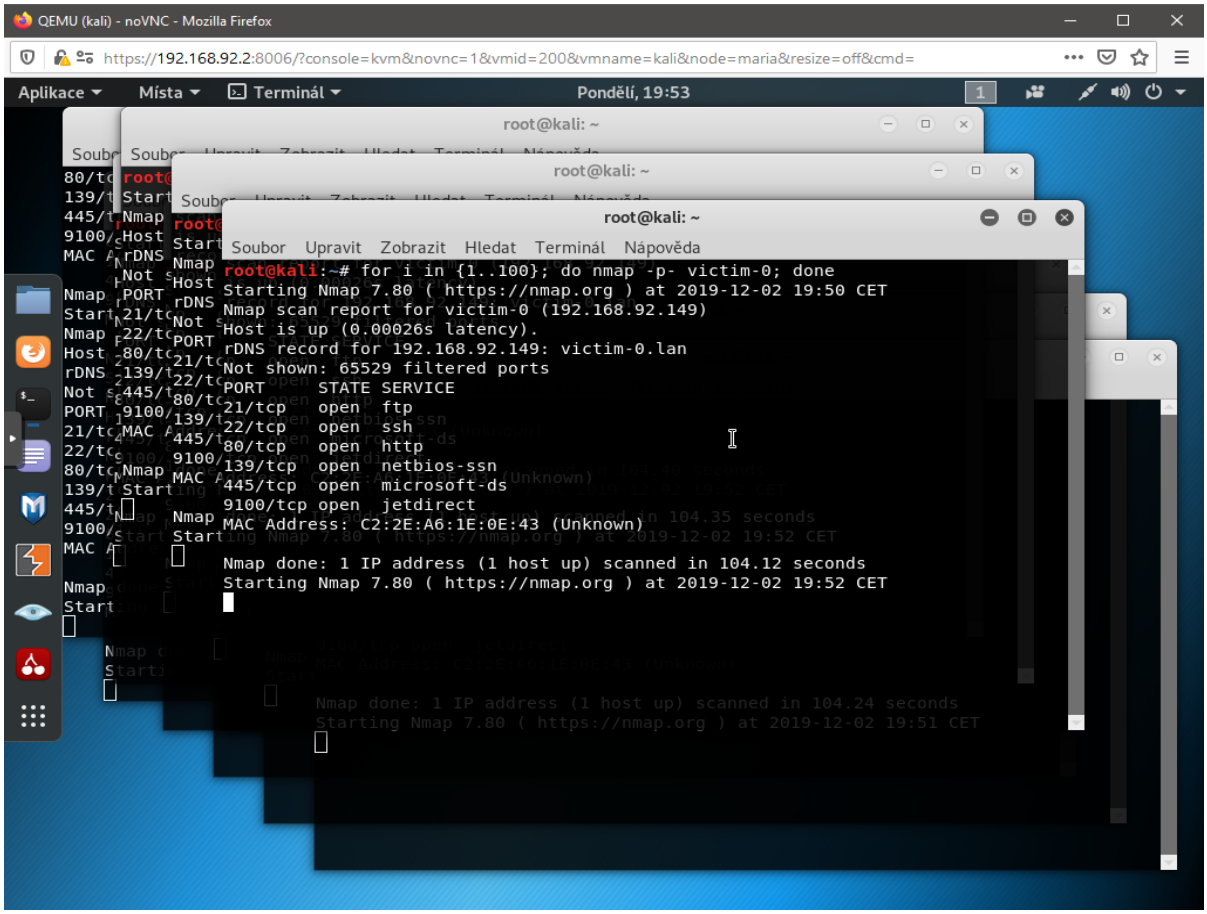
Tento útok lze tedy škálovat pomocí množství útočících zařízení.

Jaký program jsem použil a proč

K útokům jsem využil program nmap [55], který lze doinstalovat do většiny linuxových distribucí a také existuje implementace pro windows [56]. Tento program jsem použil proto, že s ním mám dobré zkušenosti a jedná se také o de-facto nejvíce doporučovaný [57] program pro provádění skenů.

Pro účely experimentu jsem použil nmap v následující konfiguraci, kterou jsem pustil několikrát, abych dosáhl měřitelných výsledků:

```
for i in {1..100}; do nmap -p- victim-0; done
```



Obrázek 53 Průběh útoku nmap, zdroj: vlastní výzkum

Tento zápis říká, aby se nmap pustil 100x za sebou v řadě a po každé provedl kompletní sken všech TCP portů na cílovém zařízení victim-0.

Tabulka 7 popis přepínače pro příkaz nmap, zdroj: vlastní výzkum

Přepínač	Hodnota	Vysvětlení
-p-	žádná	Provede sken všech TCP portů na cílovém zařízení.

Jak se projevuje tento útok na cílovém zařízení oběti

I dle tvůrců aplikace je sledování (potažmo vizualizace) systémových logů bráno jako nedostatečné [58] a změny se mohou projevovat v počtech pokusů o navázání TCP spojení.

Bohužel i s naškálovaným nmapem pomocí for-each cyklu a puštěním na více počítačích jsem nedosáhl žádné viditelné změny na grafech. Sledoval jsem metriky Počtu TCP navázaných spojení a počtu paketů, které dorazily na síťové rozhraní.

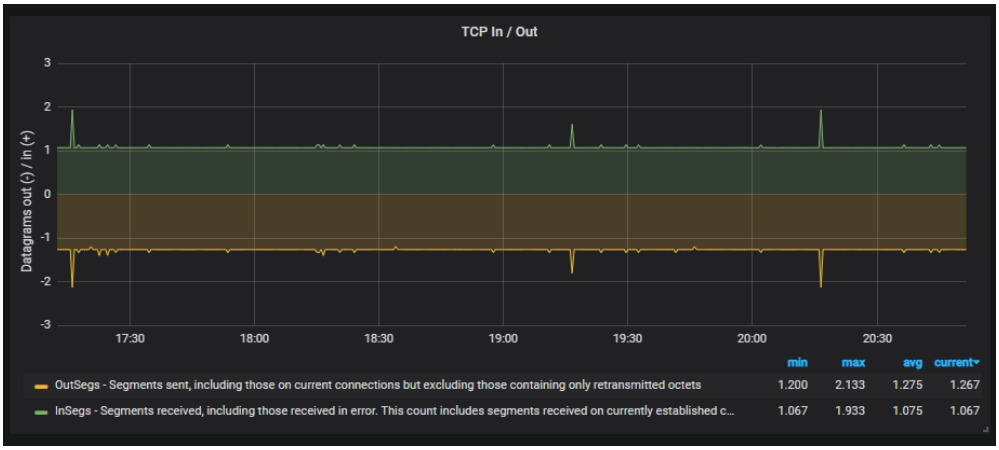
Ve výsledku jsem v grafech nezaznamenal žádný viditelný pohyb a změny se mohou snadno splést s běžným provozem.

Pro detekci skenování portů jsou obecně doporučovány [59] dva nástroje:

1. ScanLooD [60]
2. Sentrytools [61],

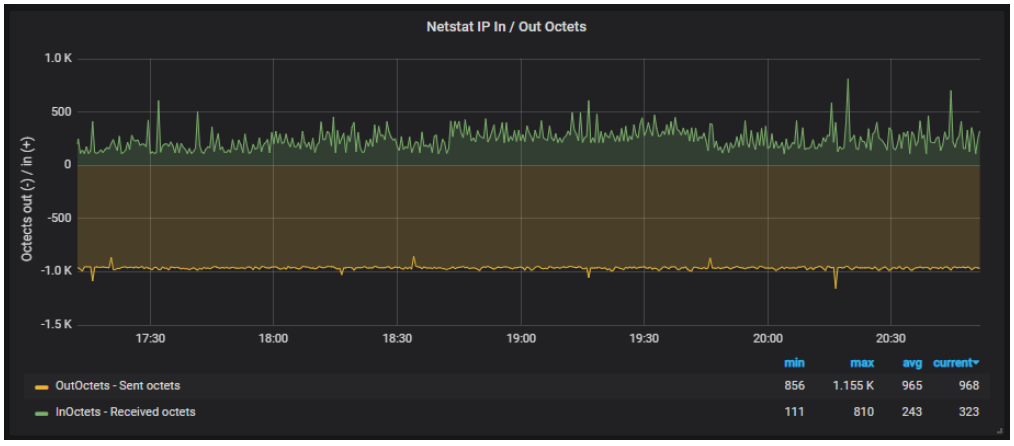
pro které jsem nenalezl žádnou konfiguraci NodeExporteru či jiného sběru metrik, kterou bych mohl integrovat s Grafanou a vizualizovat tak síťové skenování.

Na ose X je uveden čas, Osa Y uvádí počet spojení.



Obrázek 54 Na grafu nejsou vidět žádné změny v počtu TCP spojení zdroj: vlastní výzkum

Na dalším grafu na Osa X uvádí čas, Osa Y uvádí počet Příchozích TCP oktetů.



Obrázek 55 Na grafu nejsou vidět žádné změny v počtu Příchozích TCP oktetů zdroj: vlastní výzkum

6.5 Útok DoS přetížení síťového rozhraní provozem

Smyslem tohoto experimentu bylo simulovat velký síťový provoz (klient – server) s cílem odstavit cílový server victim.

Jak funguje útok z pohledu útočníka

Tento útok využívá nástrojů na generování síťového provozu, které naváží spojení s cílovým serverem a **odesílají na něj** velké množství dat. Tento útok je možné provádět dvěma způsoby:

a) **Z jednoho zařízení**. Tento útok má velké nároky na síťové připojení útočníka, proto je výhodnější jej provozovat, pokud útočník napadl počítač ze stejné sítě jako je cílové zařízení (stejná síť lan), jelikož může dojít i k přetížení aktivních prvků v síti (switche, routery).

b) **Z více zařízení** (botnet, zombie počítače). Tento útok využívá faktu, že útočník delší dobu buduje síť napadených počítačů, ve kterých čeká připravený škodlivý kód, kdy po odeslání povelu k útoku začnou všechny počítače v botnet síti posílat požadavky a velké množství dat na cílové zařízení oběti.

Pro provedení tohoto útoku je možné použít mnoho nástrojů, pro účely tohoto experimentu jsem vybral nástroj iperf3.

Jaký program jsem použil a proč

Vybraný nástroj iperf3 jsem vybral, protože nabízí jednoduché rozhraní pro nastavení chování programu a lze jej jednoduše doinstalovat do mnou zvolené distribuce. Tento program potřebuje stejnou verzi programu na koncovém zařízení, která se pustí v režimu serveru a my poté pouštíme náš program v režimu clienta, který na server zasílá požadavky. Pro nás je výhodou, že jsme schopni kontrolovat, kdy k útoku dochází a kdy jej ukončit. Také můžeme sledovat chování programu přes administrační konzoli proxmoxu a ladit parametry serveru, jako je například port.

Program jsem použil v následující konfiguraci:

```
iperf3.exe -c victim-0–P 20 -t 3600
```

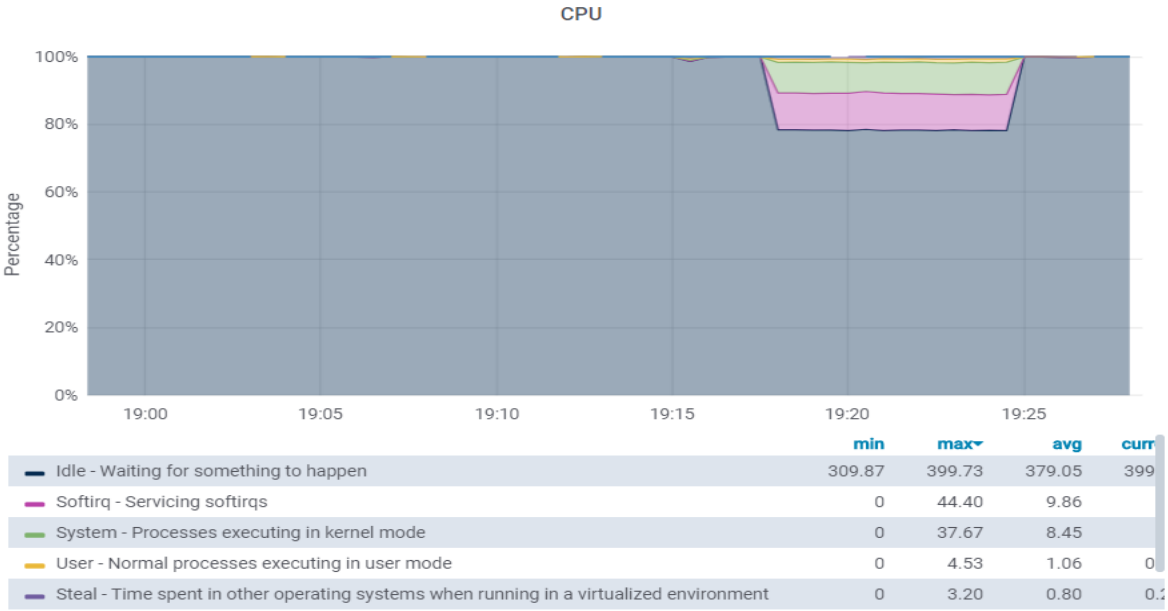
Tabulka 8 Nastavení programu iperf3., zdroj: vlastní výzkum

Přepínač	Hodnota	Vysvětlení
-c	ip adresa nebo hostname	Tímto přepínačem specifikujeme cílovou adresu serveru, na který chceme útočit.
-P	Počet paralelních spojení	Tímto přepínačem volíme, kolik se naváže TCP spojení najednou.
-t	Jak dlouho pustit útok	Tímto přepínačem říkáme, že útok chceme provádět 3600 sekund, tedy 1 hodinu.

Tento útok jsem dále škáloval tak, že jsem příkaz pustil z více počítačů najednou.

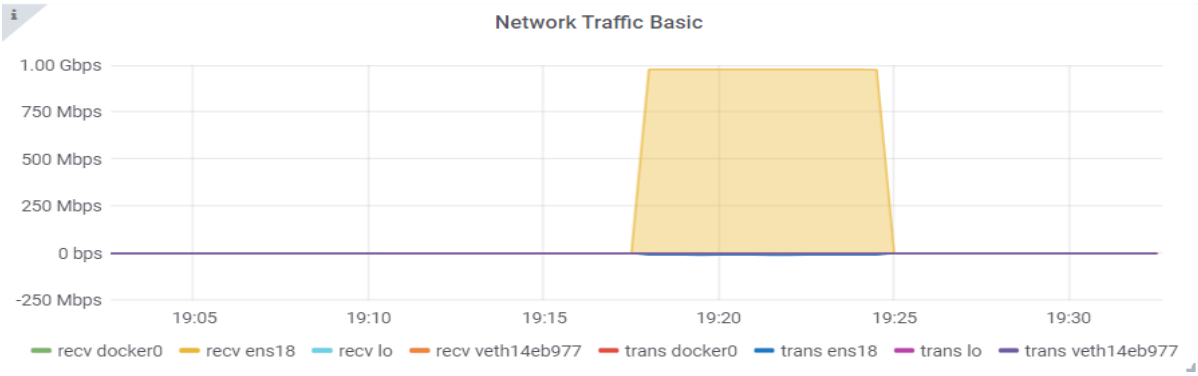
Jak se tento útok projevuje na cílovém zařízení oběti

Na zařízení oběti došlo k okamžitému nárůstu většiny systémových metrik. V následujícím grafu je vidět metrika vytížení CPU. Tento graf je skládaný, kdy celkový součet dává 100 %. Největší plochu zabírá IDLE (v normálním režimu), jelikož procesor čeká na příkazy, které má zpracovávat. Můžeme si všimnout, že při provádění útoku došlo k výraznému (>20 %) náběhu v zátěži CPU. Toto sice není na odstavení cílového stroje, ale pokud bychom na něj útočili z více počítačů, tak už by zatížení mohlo být vyšší. Na ose Y je využití procesoru, které je rozděleno na jednotlivé kategorie viz legenda.



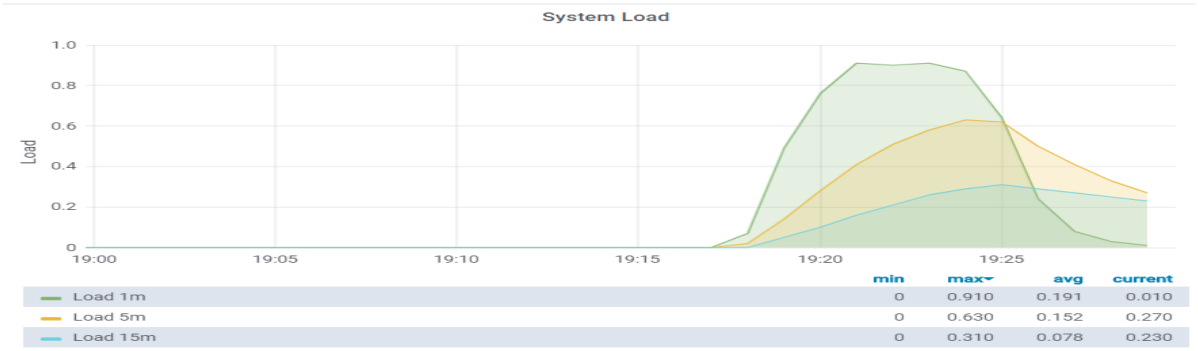
Obrázek 56 Vytížení procesoru v procentech, zdroj: vlastní výzkum

Metrika Network Traffic Basic (základní síťový provoz) vzrostla naprosto okamžitě a v podstatě narazila na strop (100 % vytížení) linky síťového rozhraní. Na ose Y je vytížení síťového rozhraní (odeslané/přijaté) v bitech za sekundu. Limitem pro tuto osu je 1 Gigabit za sekundu, což je teoretická maximální rychlost síťového rozhraní. Pro docílení tohoto vytížení stačil pouze jeden útočící počítač.



Obrázek 57 Vytížení síťového rozhraní v Megabajtech za sekundu, zdroj: vlastní výzkum

S vytížením procesoru souvisí i metrika celkového zatížení systému, kdy z pohledu administrátora vidíme, že nejrychlejší nárůst je v metrice zatížení za 1 minutu. Na ose Y je zatížení od 0 do 1, které odpovídá 0–100 % zátěže systému a Load 1m představuje průměrnou zátěž za jednu minutu – což nám značí, že zátěž byla náhlá.



Obrázek 58 Vytížení systému v minutových, 5 minutových a 15 minutových maximech, zdroj: vlastní výzkum

Tento útok je jednoduchý na indikaci, jelikož si můžeme nastavit jednoduchý alert s limitní hodnotou na počet přenesených bajtů. Pokud tento přenos trvá delší dobu, můžeme detekovat útok a upozornit správce.

Tento útok je možné zaměnit například s tím, že uživatelé stahují velké množství dat, proto bychom měli měření v Grafaně kombinovat například se sledováním odezvy serveru z externích zdrojů a před nasazením alertů delší dobu monitorovat chování uživatelů na síti a dobu největšího zatížení.

7 Závěr

Hlavním cílem bakalářské práce je popsat a otestovat metody vizualizace síťových aktivit identifikovaných jako nekalé a útočné. Tohoto cíle jsem dosáhl splněním rámcových podcílů, které vedly ke splnění hlavního cíle.

Komentovaná rešerše k tématům síťových útoků logování a uchovávání záznamů o událostech. V tomto prvním cíli, který byl základem pro celou tuto práci, bylo zjištěno, jakým způsobem dochází k logování a uchovávání záznamů o událostech. Zároveň byla napsaná komentovaná rešerše k tématům síťových útoků.

Zachycení útoku – detekce podezřelé aktivity, metody vyhodnocování síťové aktivity. Po implementaci a zprovoznění vizualizačního softwaru jsem byl schopen vizualizovat většinu prováděných útoků. Objevily se však i takové útoky, jež se zachytit nepodařilo, jelikož nedošlo k žádným anomáliím, kterých by si v našem případě člověk (detektor) byl schopen všimnout i s nastavenou podpůrnou funkcí upozornění. Buď se útoky v běžném provozu ztratily a nebylo možné je od běžného provozu rozpoznat, nebo mnou navržená nástěnka pro kontrolu síťového provozu nebyla schopná tuto anomálii zachytit.

Metody zpracování a vizualizace síťové aktivity. Tento bod byl zpracován ve třetí kapitole, kde je popsána metoda zachytávání a interpretace síťových aktivit pomocí grafické reprezentace datových bodů. V této kapitole jsem zpracoval dostupné nástroje pro vizualizace a porovnal jsem vícekriteriální tabulku.

Vytvoření podpůrného virtuálního počítače pro ověření teoretických znalostí. Tohoto cíle bylo dosaženo výběrem vhodného virtualizačního řešení – PROXMOX, které posloužilo jako supervizor pro správu virtuálního prostředí, kde byly vytvořeny virtuální stroje k potřebám testování.

Simulace vybraných síťových útoků na virtuálním počítači. Po prostudování materiálů na téma síťových útoků bylo tohoto cíle dosaženo, jelikož, na základě těchto dovedností, jsem byl schopen provést vybrané síťové útoky na mnou vytvořeným virtuálních počítačích pomocí nástroje Kali Linux známého jako distributora pro penetrační testy.

Data a výstupy z experimentů popsat a okomentovat. Na základě provedených experimentů jsem provedl zhodnocení dosažených výsledků a došel jsem k závěru, že pomocí zvolených nástrojů je možné vizualizovat následující útoky SYN food, útok na prolomení hesla SSH služby, ARP spoofing, skenování portů, DoS přetížení síťového rozhraní provozem.

8 Seznam použité literatury

[1] KOLOUCH, J., BAŠTA, P. et al. 2019. *Cybersecurity*. Praha: CZ.NIC, z. s. p. o. 556 s. ISBN 978-80-88168-34-8.

[2] Inno3. *Cisco Cybersecurity Report: i criminali informatici non lasciano tregua* [Online]. 2018 [Citace: 19. 5. 2020]. Dostupné z: https://inno3.it/2018/02/27/cisco-cybersecurity-report-i-criminali-informatici-non-lasciano-tregua/?fbclid=IwAR0r8kMXHCvw7Xy2HcGWtiSY3OFiILJYu4iEda9TicrrM0wf8xjn1FkaYXQ.

[3] BARTEL, P. Diplomová práce. *Detekce anomálií a vizualizace síťového provozu*. [Online]. 2011 [Citace: 10. 12. 2019]. Dostupné z: https://is.muni.cz/th/wuoos/dp.pdf?fbclid=IwAR0sck5h4xWPMMnrujiDJZtKFO7.MGbouf9yY51yNe5I9IreyJw-Fki6LgGE.

[4] BARTEL, P. Diplomová práce. *Detekce anomálií a vizualizace síťového provozu*. [Online]. 2011 [Citace: 10. 12. 2019]. Dostupné z: https://is.muni.cz/th/wuoos/dp.pdf?fbclid=IwAR0sck5h4xWPMMnrujiDJZtKFO7.MGbouf9yY51yNe5I9IreyJw-Fki6LgGE.

[5] BARTEL, P. Diplomová práce. *Detekce anomálií a vizualizace síťového provozu*. [Online]. 2011 [Citace: 10. 12. 2019]. Dostupné z: https://is.muni.cz/th/wuoos/dp.pdf?fbclid=IwAR0sck5h4xWPMMnrujiDJZtKFO7.MGbouf9yY51yNe5I9IreyJw-Fki6LgGE.

[6] DULÍK, P. 2007. Diplomová práce. *Informace, vizuální vnímání, vizualizace* [Online]. Brno: Masarykova univerzita [Citace: 02. 23. 2020]. Dostupné z: https://is.muni.cz/th/64795/ff_m/DP_Dulik_Pavel.pdf?fbclid=IwAR2WyGmYZjILnCX4ptdSFmKJs7h57S4qlWKy9b8IMzhJuDNRSgrL4Q7hQv8.

[7] KŮŽEL, P. Diplomová práce. *Analýza síťového provozu.* [Online]. 2006 [Citace: 11. 12. 2019]. Dostupné z: https://is.muni.cz/th/ys7rx/ diplomka.pdf ?fbclid=IwAR3JtusMj6mt6Vpe3Bi3m7sktM4TSuwKFtrJdooeMGTQQ4pdOYp0XaZ9qfs.

[8] Wireshark. *About Wireshark.* [Online]. 2019 [Citace: 10. 12. 2019]. Dostupné z: https://www.wireshark.org/?fbclid=IwAR1Arw7KCXNBc0YYawQ8lmLewMG-BfUCb7CZVmpu7NEDEwa6V4MKo64uhQc.

[9] Wireshark. *About Wireshark* [Online]. 2020 [Citace: 19. 5. 2019]. Dostupné z: https://www.wireshark.org/?fbclid=IwAR0015CcLizLSuTv4jTr62VKOFC96VuxdxQDZeufxnfxu0409XDMDkEejvk.

[10] Martin. *TCPDUMP základy*. [Online]. 2014 [Citace: 26. 2. 2020]. Dostupné z: https://m-linux.cz/2014/10/tcpdump-zaklady/.

[11] Paessler. *PRTG Network Monitor.* [Online].[Online]. ©2020 [Citace: 27. 2. 2020]. Dostupné z: https://www.paessler.com/prtg.

[12] Wireshark. *The "I/O Graphs" Window* [Online]. 2011 [Citace: 19. 5. 2020]. Dostupné z:https://www.wireshark.org/docs/wsug_html_chunked/ChStatIOGraphs.html?fbclid=IwAR26EOjmvVLwKAWg671FscR6wj1CWT3ZF9gQwZPVS57SQ57BggpzDtY3i1Q.

[13] DNSstuff. *Top Free Network Monitoring Tools.* [Online].[Online]. 2019 [Citace: 27. 2. 2020]. Dostupné z: https://www.dnsstuff.com/free-network-monitoring-software.

[14] NADEEM, S. F., HUANG, CH., 2018. *Data Visualization in Cybersecurity* [Online]. USA: Kean University [Citace: 02. 23. 2020]. Dostupné z: https://www.researchgate.net/publication/333191303_Data_Visualization_in_Cybersecurity?fbclid=IwAR1tRFUh8yQxDwV_kaVHBhXiprfh7PaibD7OXErvkg5Berrsv07W_ri95mA.

[15] JqPlot. *A Versatile and Expandable jQuery Plotting Plugin!* [Online]. [Citace: 10. 12. 2019]. Dostupné z: http://www.jqplot.com/index.php.

[16] Flot. *Attractive JavaScript plotting for jQuery.* [Online]. © 2007 - 2014 [Citace: 10. 12. 2019]. Dostupné z: https://www.flotcharts.org/?fbclid=IwAR2fqYolsAEkAxbf2hDUqDsFzB_X_z49bON6L5iq8A3a-YQzxjupi4Roy9Y.

[17] Highcharts. *Make your data come alive.* [Online]. © 2019 [Citace: 10. 12. 2019]. Dostupné z: https://www.highcharts.com/?fbclid=IwAR0LJqzSpcEvk3xjCOZs_jCjYAcvnRlf- oYjjuqJ2uUoYGgnOHkyFIgl0oM.

[18] Wireshark. *About Wireshark.* [Online]. 2019 [Citace: 10. 12. 2019]. Dostupné z: https://www.wireshark.org/?fbclid=IwAR1Arw7KCXNBc0YYawQ8lmLewMG-BfUCb7CZVmpu7NEDEwa6V4MKo64uhQc.

[19] Alex van den Bogaerdt. *Rrdgraph.* [Online]. 2019 [Citace: 10. 12. 2019]. Dostupné z: https://oss.oetiker.ch/rrdtool/doc/rrdgraph.en.html?fbclid=IwAR0DFtulFexwGlNXMAZvNha1J0i6n0DX0Vhu_Hdu2ISSyZ58jb5DRB5WNUE.

[20] Humdi. *Vnstatl* [Online]. © 2002-2019 [Citace: 10. 12. 2019]. Dostupné z: https://humdi.net/vnstat/?fbclid=IwAR1mTFl2sfhHKNIpfLw6MZ3UUXvXAJAUtPcKrDu2lelD7_UrHAbuAzbtuXo.

[21] Unix4lyfe.org. *Darkstat.* [Online]. © 2001-2016 [Citace: 11. 12. 2019]. Dostupné z: https://unix4lyfe.org/darkstat/?fbclid=IwAR1LXX7JZh0xkEG55cwVLzUSqlVkQcQr8PRxntjpCfTqWrKtNfARNf1wKYk.

[22] The Cacti Group. *Cacti: The complete rrdtool-based graphing solution* [Online]. © 2004-2019 [Citace: 11. 12. 2019]. Dostupné z: https://www.cacti.net/?fbclid=IwAR3cC0a1ZBaaTxCbsP7yv1zxMxY3wyGZamJrHT9Ace1hbbioBxLju-JZT-M.

[23] GlassWire. *Privacy & Security Features.* [Online]. © 2019 [Citace: 11. 12. 2019]. Dostupné z: https://www.glasswire.com/?fbclid=IwAR2oTTjNWH2pOJ8cgBrapq3gBQLh8QJ35J-DQBCaa__yCV91jlunzEKr0mQ.

[24] Zabbix. *Monitor anything.* [Online]. © 2001-2019 [Citace: 11. 12. 2019]. Dostupné z: https://www.zabbix.com/?fbclid=IwAR2O7PsQ58x16W2GLcCA9H6RJHftxNYk2mgNVCxwIkELeGqW56uSMoK6Ggs.

[25] Nagios. *What can Nagios help you do?* [Online]. © 2009 - 2020 [Citace: 26. 2. 2020]. Dostupné z: https://www.nagios.org/.

[26] Grafana Labs. *Download Grafana*. [Online]. 2019 [Citace: 7. 12. 2019]. Dostupné z: https://grafana.com/grafana/download?fbclid=IwAR1TXFQH2Sm3yJOtOqDIUzplKzsiktouCyAtZt5-XNM8SqTh4yXSPBowqTc.

[27] MATĚJKA, M. *Počítačová kriminalita* [Online]. 2002 [Citace: 27. 2. 2020]. Dostupné z: https://www.martinus.cz/?uItem=10934.

[28] MATĚJKA, *Michal. Počítačová kriminalita* [Online]. 2002 [Citace: 27. 2. 2020]. Dostupné z: https://www.martinus.cz/?uItem=10934.

[29] KŮŽEL, P. Diplomová práce. *Analýza síťového provozu.* [Online]. 2006 [Citace: 11. 12. 2019]. Dostupné z: https://is.muni.cz/th/ys7rx/diplomka.pdf?fbclid=IwAR3JtusMj6mt6Vpe3Bi3m7sktM4TSuwKFtrJdooeMGTQQ4pdOYp0XaZ9qfs.

[30] MATĚJKA, M. *Počítačová kriminalita* [Online]. 2002 [Citace: 27. 2. 2020]. Dostupné z: https://www.martinus.cz/?uItem=10934.

[31] 0xb1t. *ARP Spoofing* [Online]. 2006 [Citace: 14. 5. 2020]. Dostupné z: https://www.security-portal.cz/clanky/arp-spoofing?fbclid=IwAR1x213HjMlUKH-OaSi2m4MODlDjJo8lgTZOPSKcL7KNlxd1PPcqNKIMnAQ.

[32] BARTEL, P. Diplomová práce. *Detekce anomálií a vizualizace síťového provozu.* [Online]. 2011 [Citace: 10. 12. 2019]. Dostupné z: https://is.muni.cz/th/wuoos/dp.pdf?fbclid=IwAR0sck5h4xWPMMnrujiDJZtKFO7.MGbouf9yY51yNe5I9IreyJw-Fki6LgGE.

[33] KOLOUCH, J. *Cybercrime* [Online]. 2016 [Citace: 14. 5. 2020]. Dostupné z: https://knihy.nic.cz/files/edice/cybercrime.pdf?fbclid=IwAR14Im2d80tayJ5DPBw_WK-ht1MFmDl7N7eHJ8DdptBkGw_9KRei-UtlMSQ.

[34] DONATO, R. *What Is the Difference between QEMU and KVM?* [Online]. 2020 [Citace: 27. 2. 2020]. Dostupné z: https://www.packetflow.co.uk/what-is-the-difference-between-qemu-and-kvm/.

[35] Proxmox. *Proxmox Virtual Environment* [Online]. © 2004 - 2020 [Citace: 27. 2. 2020]. Dostupné z: https://www.proxmox.com/en/.

[36] VirtualBox. *Welcome to VirtualBox.org!* [Online]. 2020 [Citace: 27. 2. 2020]. Dostupné z: https://www.virtualbox.org/.

[38] Proxmox. *Proxmox ve administration guide* [Online]. 2020 [Citace: 27. 2. 2020]. Dostupné z: https://pve.proxmox.com/pve-docs/pve-admin-guide.pdf.

[39] GrafanaCONline. *Grafana 7.0 is here* [Online]. 2020 [Citace: 19. 5. 2020]. Dostupné z:https://grafana.com/?fbclid=IwAR39_r1HrmRstql_u6FuOf_RAopmyk9NC4vo2h_zHhZqKPF4UMmZ7DYrVD8.

[40] GrafanaLabs. *Plugins: Official & community built plugins* [Online]. ©2020 [Citace: 19. 5. 2019]. Dostupné z: https://grafana.com/grafana/plugins?fbclid=IwAR2U0XjGR5hnfx9OFnr4rBDcLnUY4DtpSE5mDGCshlZVstiJmY-1QsibqQU&orderBy=weight&direction=asc.

[41] JAQUITH, A. *2007. Security Metrics: Replacing Fear, Uncertainty, and Doubt*. English: AddisonWesley Professional. 336 s. ISBN: 978-03-21349-98-9.

[42] GitHub. *Node exporter* [Online]. © 2020 [Citace: 27. 2. 2020]. Dostupné z: https://github.com/prometheus/node_exporter.

[43] Prometheus. *Querying prometheus* [Online]. © 2014-2020 [Citace: 19. 5. 2020]. Dostupné z: https://prometheus.io/docs/prometheus/latest/querying/basics/?fbclid=IwAR29FUxFbqP9zU2NHtnb9vVGbJEoZdXXfL7Tc4EFSuTB5xmiQNdLxo1YpEc.

[44] FAJKUS, K. Diplomová práce. *Detekce anomálií v lokální síti* [Online]. 2018 [Citace: 19. 5. 2019]. Dostupné z: https://is.muni.cz/th/moca6/DP-full.pdf?fbclid=IwAR1pFsvO9xWNiOqRWFCvPMZM9FX8_GYtXGfVVnYUzgD7cmtF3oRaGCgTuKk.

[45] Apache. *ab - Apache HTTP server benchmarking tool* [Online]. 2020 [Citace: 19. 5. 2019]. Dostupné z: https://httpd.apache.org/docs/2.4/programs/ab.html?fbclid=IwAR2KWc0p523WeV9Xpdef3jvl_sLKmDNCaqFwJ-q9KTRUaZAUROBX4sjwXpg.

[46] DUGAN, J., et al. *What is iPerf / iPerf3?* [Online]. ©2020 [Citace: 19. 5. 2019]. Dostupné z: https://iperf.fr/?fbclid=IwAR3uJDhYTlrCbucwPVUpbCfk6NPrOBwIrqaaDEZZgK-Jc6ITuVXbE0iAdaU.

[47] ESnet. *Throughput Tool Comparision* [Online]. 2019-2020 [Citace: 19. 5. 2020]. Dostupné z: https://fasterdata.es.net/performance-testing/network-troubleshooting-tools/throughput-tool-comparision/?fbclid=IwAR0_nug4nSQ3OfxKO04tN6UvL09xL9NUyndOoZod8IjpEv8y3oLC946gQxE.

[48] Realsecurity. *DDoS: It's Not a Matter of If, But When* [Online]. 2019 [Citace: 19. 5. 2020]. Dostupné z: https://www.real-sec.com/2019/05/ddos-its-not-a-matter-of-if-but-when/?fbclid=IwAR3frD8JuLV6Mh6cgP4qHWNcEjmtN0kcaqmBhY7HeAvINuRcFbGvnE_rtB0.

[49] Kali Linux. *Latest Kali Linux News and Tutorials* [Online]. ©2020 [Citace: 19. 5. 2019]. Dostupné z: https://www.kali.org/?fbclid=IwAR1K4JVkDc0y3cIjuJXt9uldkjuUjdZiraNY_rWNqH3QP-kI2klVivzY1VY.

[50] SANFILIPPO, S. *Hping3(8) - Linux man page* [Online]. ©2020 [Citace: 19. 5. 2019]. Dostupné z: https://linux.die.net/man/8/hping3.

[51] ManageEngine. *What is a brute force attack?* [Online]. © 2020 [Citace: 19. 5. 2020]. Dostupné z: https://www.manageengine.com/log-management/cyber-security-attacks/what-is-brute-force-attack.html?fbclid=IwAR2FFlXptMy9KtKdt6o54q0vpi3T6nY0epgAPgH1ByjrY_zZSIAyWfBHw-Q.

[52] Kali. *Hydra Package Description* [Online]. ©2020 [Citace: 19. 5. 2019]. Dostupné z: https://tools.kali.org/password-attacks/hydra?fbclid=IwAR1dHwnEKPKQ9Ey-XOv1qDFf5pf2YS8Nis32QodNrkRkooQyelZrjRvSGr8.

[53] MICHAEL, B. *Arp spoofing* [Online]. 2019 [Citace: 19. 5. 2019]. Dostupné z: https://blog.octanetworks.com/arp-spoofing/?fbclid=IwAR31SsRYXUPuunvslRUW7cdpy2t4fOHc75oIyX0yHpCIhJTT7WYjPQSeQ6w.

[54] Ettercap. *Welcome to the ettercap project* [Online]. ©2020 [Citace: 19. 5. 2019]. Dostupné z: https://www.ettercap-project.org/?fbclid=IwAR0TdUw1 NJm49ScCnXTwO5F_2F7eAd01geu4oiWiKAaP14BB-SviEhBU5q8.

[55] Nmap.org. *Free Security Scanner* [Online]. 2017 [Citace: 19. 5. 2019]. Dostupné z: https://nmap.org/?fbclid=IwAR1K4JVkDc0y3cIjuJXt9uldkjuUjdZiraNY_rWNqH3QP-kI2klVivzY1VY.

[56] Nmap.org. *Windows* [Online]. 2013 [Citace: 19. 5. 2019]. Dostupné z: https://nmap.org/book/inst-windows.html?fbclid=IwAR3K_ZnhlcfNKOwXfMGgBKqD_V-Q9OUY7p8YLveij-k9lXr7aJtytjm9tIs.

[57] LYON, G. *Nmap Extensible: open-source network mapper with OS detection to scan networks for hosts and services* [Online]. 2020 [Citace: 19. 5. 2019]. Dostupné z: https://alternativeto.net/software/nmap/?fbclid=iwar3xhq53jqjrbduixyfaagjdmrwa27na_bmqjhlzcf0umrd3thp_lqxmwjo.

[58] Nmap.org. *Detect Nmap Scans* [Online]. ©2020 [Citace: 19. 5. 2019]. Dostupné z: https://nmap.org/book/nmap-defenses-detection.html?fbclid=IwAR3F0Hd4Kqab75foNnaSSfbMgi0yciSLTHEMnmB_JIcovCvIYkPrL5Wjw9U.

[59] Nmap.org. *Detect Nmap Scans* [Online]. ©2020 [Citace: 19. 5. 2019]. Dostupné z: https://nmap.org/book/nmap-defenses-detection.html?fbclid=IwAR3F0Hd4Kqab75foNnaSSfbMgi0yciSLTHEMnmB_JIcovCvIYkPrL5Wjw9U.

[60] Openwall. *Scanlogd - a port scan detection tool* [Online]. ©2020 [Citace: 19. 5. 2019]. Dostupné z: https://www.openwall.com/scanlogd/?fbclid=IwAR00L3Rt3auAOwKsg7soARcqrn-kUpvBQiThF6yIfoqmCIHj4p0ZMJAotGQ.

[61] Sourceforge. *Sentry Tools download* [Online]. ©2020 [Citace: 19. 5. 2019]. Dostupné z: https://sourceforge.net/projects/sentrytools/?fbclid=IwAR2WkKaZCvezS6Vk7DMpZwM5M1LgPPJOHXVsDK3RO4UtdyWBZHAMf2TI_Mc.

9 Seznam obrázků

10 Seznam tabulek

11 Seznam zkratek

MAC	Media Access Control
ARP	Adress Resolution Protocol
DOS	Denial of Service
DDOS	Distributed Denial of Service
SSH	Secure Shell
MITM	Man in the Middle