VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. JAN KLEČKA

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

MONITOROVACÍ SONDA SÍŤOVÉ KOMUNIKACE
NETWORK COMMUNICATION MONITORING PROBE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jan Klečka**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Blažek**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Jan Klečka ***ID:*** 195811
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Monitorovací sonda síťové komunikace

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce je realizace síťové sondy postavené na operačním systému Linux a jednodeskovém mini počítači. Proveďte analýzu a výkonové testování (procesor, RAM, síťové rozhraní) jednodeskových počítačů (např. Raspberry Pi, Banana Pi, ODROID a další), ze kterých vyberte vhodného kandidáta pro realizovanou síťovou sondu. Dílčím cílem práce bude realizace filtrovacího (např. iptables, nftables) a detekčního (např. surikata, snort, bro (zeek)) systému. Dalším dílčím cílem bude webové/terminálové rozhraní pro ovládání a vizualizaci dat protékajících sondou. V rámci rozhraní bude možné nastavit filtraci vybraného síťového provozu a analýza pomocí detekčního systému. Dále bude možné zaznamenat protékající datový provoz do formátu pcap a reportovat vybrané informace do logovacích souborů. Poslední částí práce bude návrh a realizace vnější ochrany zařízení, která bude splňovat minimálně IP54 a bude zajištěna doporučená provozních teplota v prostředích od -40°C do +85°C. Výstupem diplomové práce bude síťová sonda, která bude konfigurovatelná z webového nebo terminálového rozhraní. Dílčím výstupem bude testování funkčnosti realizovaných částí sondy (záznam, filtrace a analýza provozu).

DOPORUČENÁ LITERATURA:

[1] DERI, Luca; SPA, NETikos. nProbe: an open source netflow probe for gigabit networks. In: TERENA Networking Conference. 2003. p. 1-4.

[2] Ridho, M. Faqih (2014) Analysis And Evaluation Snort, Bro, And Suricata As Intrusion Detection System Based On Linux Server. Skripsi thesis, Universitas Muhammadiyah Surakarta.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Petr Blažek

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Diplomová práce se zabývá analýzou jednodeskových počítačů, které využívají systém Linux jako operační systém. Dále jsou zkoumaný jednotlivé NIDS systémy a jejich vlastnosti za účelem výběru vhodného kandidáta pro jednodeskový počítač, který má být využit jako síťová sonda sloužící pro analýzu, filtraci a logování síťového provozu. Část práce je zaměřena na vývoj rozhraní, které slouží ke konfiguraci síťové sondy přes webový prohlížeč. Rozhraní umožňuje vykonávat základní operace nad síťovou sondu, která ovlivňují datový provoz nebo specifikují, jaké informace mají být zaznamenány. Následně, došlo k implementaci parserů pro síťové protokoly pomocí knihovny Scappy. Závěr práce obsahuje realizaci krycího obalu pro zařízení dle požadavků IP54.

KLÍČOVÁ SLOVA

Jednodeskový počítač, Linux, Surikata, Analýza a filtrace síťového provozu, logování.

ABSTRACT

Master thesis deals with analysis of single board PC which use Linux as operation system. Analysis of individual NIDS systems and examined their properties for choosing right candidate for single board computer which shall be used as network probe for analysis, filtering and logging of network traffic. Part of the work is aimed on development of a interface which is used for configuration of network probe through the web browser. Web interface allows perform basic operations over network probe which influence network traffic or specify, which information shall be logged. Subsequently network parsers were implemented for network protocols using the Scappy library. The conclusion of the thesis contains the design of the security cover for the device according to the IP54 requirements.

KEYWORDS

Single board computer, Linux, Suricata, analysis and filtering of network traffic, logging.

KLEČKA, Jan. *Síťová sonda pro záznam a analýzu komunikace*. Brno, 2020, 90 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Petr Blažek

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Síťová sonda pro záznam a analýzu komunikace“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Petru Blažkovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Definice síťové sondy může záležet na síťovém manažmentu aplikací, které využíváme. Existují dva typy síťových sond. První jsou softwarové, které jsou zabudované již v síťových monitorovacích nástrojích. Druhý typ je sonda, která je nainstalována na zařízení, které chceme monitorovat. Hlavní cíl síťové sondy je komunikovat s síťovým zařízením a informovat ho o určitých událostech. Sonda posílá/sbírá data z určitých zařízení v aktuálním čase (real-time). V situaci, kdy bylo zjištěno pravidlo, které sonda má detekovat, je vyvolána událost, která informuje oprávněného uživatele.

Sondy využíváme z důvodu zvyšujícího se nebezpečí skrze internetové sítě. S modernější dobou přibývá více možností, jak napadnout nebo infikovat počítačový systém. V aktuálním stavu je mnoho nebezpečných softwaru, které jsou schopny projít skrze uživatelský firewall, který může byt neaktualizovaný. Např. situace, kdy máme v jedné síti více zařízení a jeden z těchto zařízení běží na nepodporovaném OS systému.

Samotná sonda je schopná detekovat i veškerý provoz, který komunikuje v síti. Nejčastěji se využívá přístupu, kdy do sondy tečou všechna data z centrálního/hlavního switche skrze SPAN port (port, na který jsou zasílána všechna data switche). Díky tomuto je sonda schopná identifikovat i situaci, kdy koncové zařízení bylo infikované již před připojením sondy. Nejčastěji se toto dá identifikovat pomocí anomálií, kdy sonda má zaznamenaný určitý model chování. V situaci, kdy zařízení vykazuje odchylky vůči svému modelu, sonda vygeneruje událost a tyto data následně zašle na centrální stanoviště pro kontrolu síťového provozu, nebo také pomocí toho, že infikovaný bod v síti, se pokouší o nadměrnou komunikaci s ostatními zařízeními, i když pro takovou komunikaci není určené.

První kapitola práce se zabývá popisem OS (operační systém) Linux. Je zde uvedeno, jakým způsobem jsou zpracovávány pakety v rámci OS a jednotlivé nástroje, které jde využít pro filtraci provozu. Druhá kapitula popisuje, které systémy lze využít pro filtraci a detekci provozu a je provedeno jejich porovnání. Třetí kapitola obsahuje představení aplikace, která je vytvořena za účelem ovládání síťové sondy pomocí uživatelského rozhraní. Došlo k vytvoření bezpečnostního krytu, který umožňuje umístit zařízení do průmyslového prostředí. Poslední část práce je zaměřena na popis parserů pro síťové protokoly, které jsou implementovány za účelem logování specifických protokolů.

1 Linux

Pod pojmem Linux si lze představit operační systém (OS), stejně jako již dobře známý Windows. V případě, kdy je zmiňován UNIX (operační systém), se dříve hovořilo jako pouze o jádře. V dnešní době je název vztahován na operační systém, který je složen z jádra Linux a operačního systému GNU (rozsáhlá množina svobodného softwaru). Samotná kombinace Linux a GNU (obecně veřejné licence) vytváří pojem, že tento systém je kompletní, postupem času ovšem byly přidávány jednotlivé programy/aplikace, které nyní slouží jako systémové. [1]

Nástroje a aplikace, které jsou implementovány do Linuxu, je nejprve nutné zkompilovat/přeložit do takové podoby, aby byly spustitelné. Nebylo by pro uživatele přívětivé, aby takové úkony musel dělat sám. Pro usnadnění nasazení Linuxu pro uživatele tak vznikly tzv. distribuce, které obsahují všechny nutné aplikace v „balíčku“. Jednotlivé distribuce mohou být např. Ubuntu, Fedora, CentOS.

Linux jádro je hlavní komponentou Linuxového OS a je hlavním rozhraním mezi PC hardwarem a procesy. Komunikuje mezi nimi a spravuje zdroje co nejefektivněji. Jádro je pojmenováno proto, protože existuje v operačním systému a řídí všechny hlavní funkce HW (fyzické zařízení) [2]. Lze si to představit jako pouhý software, který řídí veškerou komunikaci na úrovní HW. Samotné jádro má 4 hlavní funkce:

1. **Správa paměti:** Sleduje kolik paměti se využívá k uložení čeho a kde.
2. **Správa procesů:** Určuje, který proces může využít procesorovou jednotku (CPU), kdy a na jak dlouho.
3. **Ovladače zařízení:** Funguje jako prostředník mezi hardwarem a procesy.
4. **Systémová volání a bezpečnost:** Přijímá požadavky na služby z procesů.

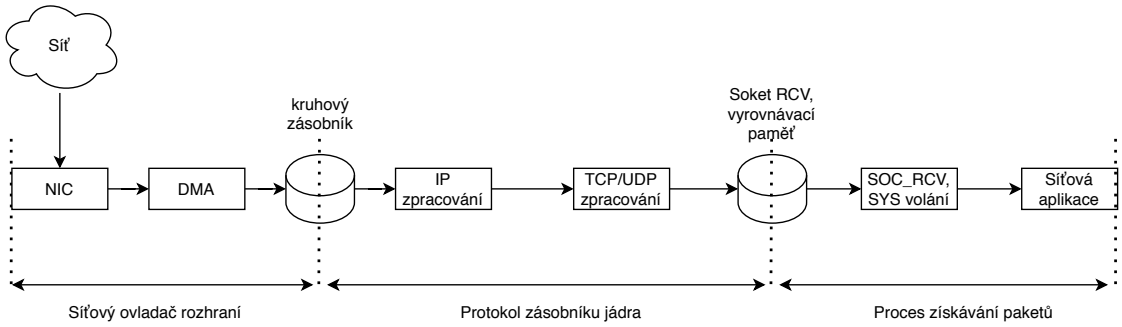
Kód spuštěn systémem běží buď v módu jádra nebo uživatelském módu. V módu jádra má neomezený přístup k HW, zatím co uživatelský mód má omezený přístup k CPU a paměti. Tyto rozdíly tvoří základ, pro některé komplikované operace, jako je oddělení úrovní zabezpečení, vytváření kontejnerů a virtuálních strojů.

Linuxové jádro umožňuje spouštět více programů zároveň. Každý právě běžící program lze poskládat z více nebo jednoho procesu. Z toho vyplývá, že Linux je reprezentován jako víceúlohový systém. Jednotlivé běžící procesy následně mohou být skládány z více podprocesů. Linux je open source software, což znamená, že jeho zdrojové kódy jsou volně k dispozici za dodržení licenčních podmínek. Zdrojové kódy lze nadále libovolně upravovat a šířit. Linux spadá a je šířen pod licencí GPLv2 (obecně veřejná licence verze 2). Veškerý software, který je s Linuxem šířen, je chráněn pomocí licencí. [3]

1.1 Zpracování paketů v linuxu z fyzického média

Samotné zpracování paketů, bez využití bridge a podobných technologií lze chápat z obr. 1.1. Samotný průběh paketů lze rozdělit do 3 částí [4]:

1. Síťový ovladač rozhraní: Paket je přenesen z karty síťového rozhraní do kruhové vyrovnávací paměti.
2. Protokol zásobníku jádra: Paket je přenesen z kruhové vyrovnávací paměti do soketu zvaný „ přijímací vyrovnávací paměť“.
3. Proces získávání paketů: Paket je nakopírován ze soketu do určené aplikace.



Obr. 1.1: Cesta paketů pro aplikaci.

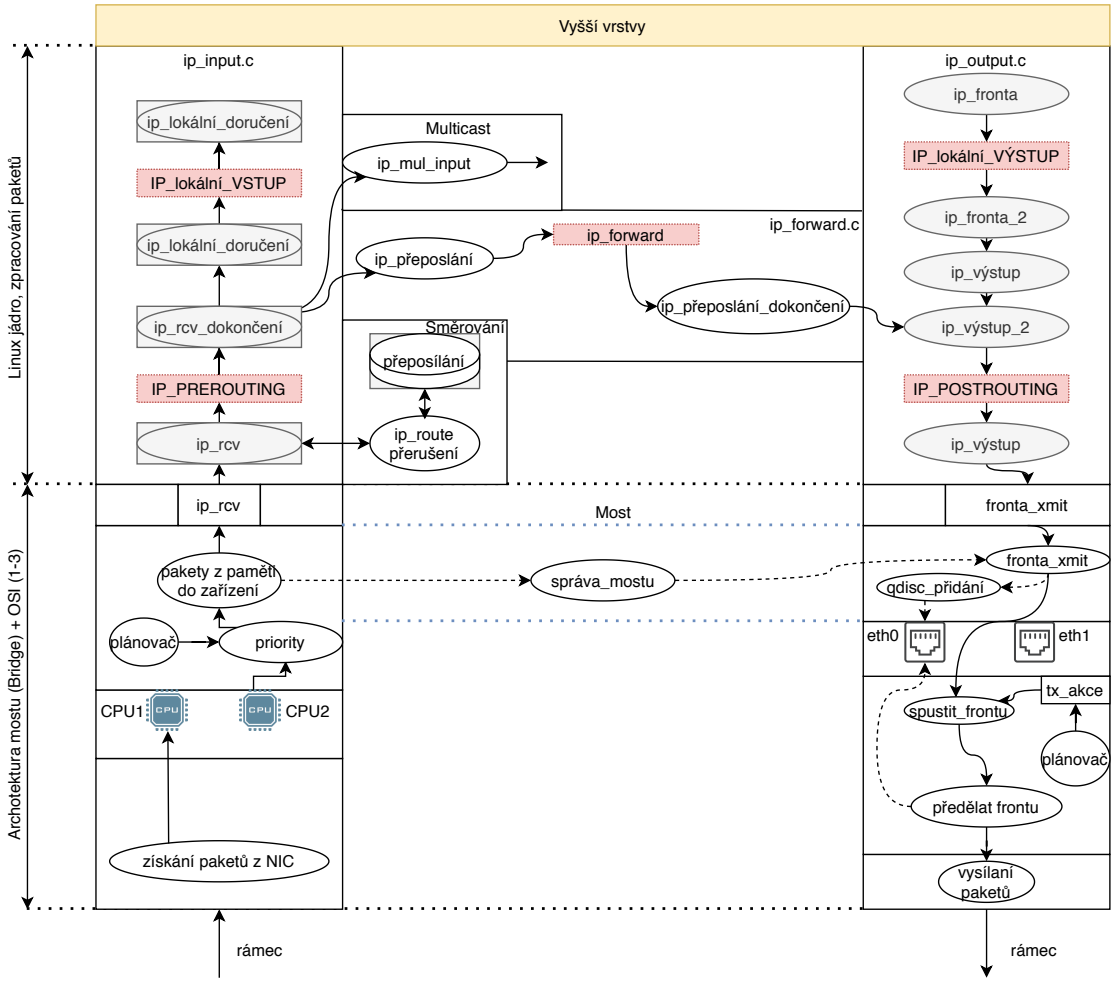
Detailnější přehled pro cestu paketu skrze linux jádro

Pro detailnější postup a analýzu, jak jsou jednotlivé pakety zpracovávány a prochází Linuxovým jádrem lze vidět na obr. 1.2. Obrázek reprezentuje postup paketu od první vrstvy ISO/OSI modelu (model, který popisuje komunikaci mezi dvěma zařízeními) až po vyšší vrstvy.

Při příchodu paketu ovladač síťového rozhraní (NIC) převezme pakety a následně předá CPU. V dalším kroku podle plánovače úloh a využití priorit jsou postupně pakety předávány vyšší vrstvě, kde již probíhá zpracování podle IP (Internet protokol). V situaci, kdy dojde ke zjištění, že paket nemá být zpracovaný daným zařízením, se v bloku IP_ PREROUTING označí a následné blok ip_rcv_dokončení tento paket posílá na správnou cílovou adresu. Kdyby byl paket určený pro dané zařízení, dojde ke dalšímu zpracování paketu pomocí IP_lokální_VSTUP (iptable pravidlo typu INPUT).

V situaci, kdy dochází k přeposlání paketu pomocí IP_FORWARD, cílová stanice příjme paket a přidá ho do dat, připravených ke zpracování. IP_POSTROUTING slouží k úpravě daného paketu (určuje co se s danými daty má stát (aplikace POSTROUTING pravidel). Dále jsou data zpracovávány do front a předané síťovému ovladači.

V situaci kdy mezi vrstvami 1 - 3 ISO/OSI modelu dojde ke zjištění, že paket má být zaslán pomocí mostu do jiné části sítě je tento rámec zpracován viz obr. 1.2 a zpracován do fronty ke zpracování na cílovém zařízení. Detailnější popis významu INPUT, OUTPUT, FORWARD je uveden v podkapitole 1.2.



Obr. 1.2: Detailnější zpracování paketu.

1.2 IPtables

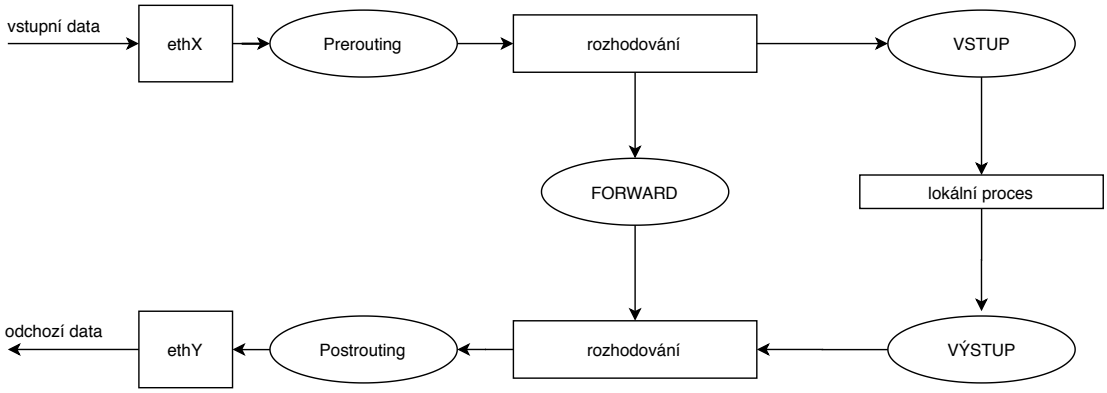
IPtables je nástroj, umožňující pracovat se síťovou komunikací. S využitím tohoto nástroje lze vytvořit seznam pravidel, který restriktivně omezují síťový provoz na daném zařízení.

Dále jsou využity pro údržbu a kontrolu pravidel, filtrování paketů v jádře Linuxu. Lze předdefinovat několik různých tabulek, kde každá tabulka obsahuje řadu předdefinovaných řetězců a může tak obsahovat uživatelem definované řetězce.

Každý řetězec je definován jako seznam pravidel, která mohou odpovídat určité shodě s informacemi v paketu. Každé pravidlo určuje, co dělat s paketem, který

odpovídá shodě. Tomuto se říká „*target*“ neboli cíl, což může být skok na uživatelem definovaný řetězec ve stejné tabulce [5]. Pokud paket, který je právě prohlížen, neodpovídá určitému pravidlu v tabulce, je zkoumáno další pravidlo v pořadí. V situaci, kdy žádné z pravidel neodpovídá paketu, aplikuji se na něj tzv. chain policy [6]. Z hlediska maximální bezpečnosti, by v takové situaci měl být daný paket zahozen.

Paketový filtr (NETFILTER) je implementován do samotného jádra OS Linux. Ovládání tohoto modulu probíhá pomocí IPtabulky, kde můžeme vkládat a odstraňovat jednotlivá pravidla pro síťový provoz. Paketový filtr umožňuje pracovat se třemi seznamy pravidel: INPUT, OUTPUT, FORWARD. Řetězce kontrolují chování pro příchozí spojení. Na následujícím obr. 1.3 je znázorněn princip výše zmíněných pravidel.



Obr. 1.3: Princip IP tables.

Prerouting blok na obr. 1.3 je sada pravidel, která jsou aplikována na příchozi pakety před tím, než jsou zpracovány v routovací tabulce. Využitím tohoto bloku lze modifikovat cílovou adresu Destination NAT (DNAT). Postrouting je opakem Prerouting bloku, kdy dochází k modifikaci paketů, které již prošly routovací tabulkou a můžeme na ně aplikovat pravidla Source NAT (SNAT). INPUT vyjadřuje aplikaci pravidel na vstupní data a OUTPUT zkoumá odchozí pakety.
Pro nastavení veškerého příchozího provozu pomocí implicitního pravidla, můžeme např. využít [7]:

```
iptables -P INPUT DROP
```

Nejpoužívanější příkazy

- -A, –append: Přidá na konec řetězce nové pravidlo.

- -D, –delete: Smaže pravidlo (zadává se buď v přesném tvaru, jak bylo zadáno nebo jeho číslem, které lze získat volbou –lin.).
- -R, –replace: Nahradí pravidlo.
- -I, –insert: Vloží na začátek řetězce nové pravidlo.
- -L, –list: Vypíše všechna pravidla v řetězci. Pokud řetězec není zadán, jsou vypsány všechny řetězce s jejich pravidly.
- -F, –flush: Vyprázdní v řetězci všechna pravidla
- -N, –new-chain: Vytvoření nového řetězce.
- -X, –delete-chain: Smaže vlastní řetězec (nelze odstranit výchozí)
- -P, –policy: Politika (policy) řetězce.
- -E, –rename-chain: Přejmenování vlastního řetězce.

Příklad pravidla

V případě, že chceme vytvořit pravidlo, je potřeba specifikovat: akci, typ pravidla (input, output, forward), a další parametry, jako mohou být např. síťové rozhraní, zdrojová a cílová IP, příjmout nebo zamítnout.

```
iptables -A INPUT -i eth0 -s 192.168.0.1 -j ACCEPT
```

- -A: Přidá pravidlo na konec řetězce.
- INPUT: Pro veškerá data přijaté daným počítačem.
- -i: Specifikuje síťové rozhraní (eth0).
- -s: Specifikuje zdrojovou IP adresu.
- -j: Určuje akci, která má být provedena (accept, drop).

1.3 Nftables

Subsystém linuxového jádra, jehož úkolem je filtrovat síťový provoz. Za vývojem stojí stejní vývojáři jako iptables/netfilter, takže jde pouze o novější verzi iptables. Lze tedy říct, že nftables jsou nástupcem iptables.

Cílem vytvoření nftables je jednodušší a univerzálnější situace, která bude v samotném jádru linuxu vyžadovat méně kódu. Další výhoda tohoto systému je usnadnění práce s jednotlivými pravidly pro uživatele.

Tabulky

Založení nové tabulky lze provést pomocí:

```
nft add table inet filter
```

Tabulky obsahují řetězce, které následně obsahují jednotlivá pravidla. V Nftables nejsou určené žádné výchozí tabulky. Pro jednotlivé tabulky je potřeba specifikovat rodinu adres, pro které se uplatňují. Tyto typy jsou: ip, ip6, inet, arp a bridge. Typ inet dokáže obsluhovat zároveň IPv4 i IPv6.

Řetězce

Účelem pro řetězce je uchovávat pravidla. V Nftables nejsou žádné vestavěné řetězce. Řetězce jsou dvou typů: base a regular. Base řetězce slouží jako vstupní bod a je zavěšen do určitých míst v TCP prostředí (stack) Linuxu. Je místem, kudy přitéká provoz do uživatelem definovaného prostředí řetězců a pravidel. Druhý typ slouží k přesměrování provozu z jiných řetězců. Vytvoření běžného řetězce s názvem webfilter můžeme pomocí:

```
nft add chain inet filter webfilter.
```

Pro každý řetězec je potřeba určit místo ze síťového TCP prostoru Linuxu, ze kterého do něj bude proudit datový tok. Pro IP protokoly lze využít: prerouting, input, forward, output, postrouting.

V situaci, kdy máme vytvořenou tabulku a v ní alespoň jeden řetězec, můžeme tomuto řetězci přidávat jednotlivá pravidla. [8]

```
nft add rule inet filter output ip daddr 8.8.8.8 accept
```

Výsledná konfigurace tabulky, řetězce a přidání pravidla by měla strukturu viz. následující výpis.

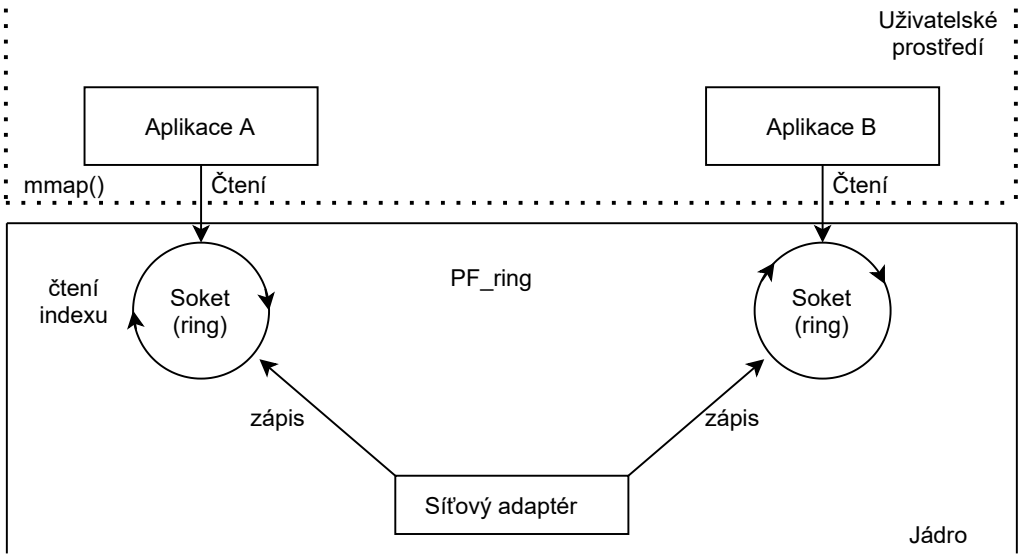
Výpis 1.1: NFtables struktura. Převzato z [8].

```
> nft list table inet filter
table inet filter {
    chain output {
        type filter hook output priority 0; policy accept;
        ip daddr 8.8.8.8 accept
    }
}
```

1.4 Pfring

PF_RING je modul jádra pro správu síťového provozu, který umožňuje efektivní zachytávání paketů, zpracovává pakety a zároveň poskytuje API (aplikační programovací rozhraní) pro zpracování paketů.

Za technikou PF_RING stojí vývojáři projektu ntop.org. Metoda PF_RING otevírá nový typ soketu nazvaný PF_RING. Při otevření soketu se v jádře Linuxu vytvoří vyrovnávací paměť (kruhová paměť - z toho je název PF_RING). Do této paměti se následně ukládají data bezodkladně hned po příchodu dat z ovladače síťové karty. Samotné jádro udržuje odkaz (ukazatel) do vyrovnávací paměti, kde zapisuje veškerá příchozí data. Zároveň je udržován odkaz na místo, ze kterého jsou data čtena z tzv. uživatelského prostředí (user space). Pro čtení dat z „jádra Linuxu“ v uživatelském prostředí se využívá technika MMAP (Memory mapping). Aplikace, které využívají tuto metodu, se vzájemně neovlivňují, protože každý soket vytvořený pro aplikaci využívá samostatnou vyrovnávací paměť. V případě pomalého zpracovávání dat dochází ke ztrátě pouze pro aplikaci s touto vyrovnávací paměti. [9]



Obr. 1.4: Architektura PF_ring.

Výhody:

- Vylepšení výpočetní náročnosti.
- Snížení ztrát paketů.

Nevýhody:

- Samotná aplikace musí být schopna tuto funkci využít.

MMAP - Memory mapping (Zobrazeni do paměti)

Označení pro systémové volání UNIXových OS, pomocí kterého se mapuje soubor do paměti. Jde o rezervování určité části adresního prostoru pro komunikaci s daným zařízením nebo pro přístup do daného souboru. Lze implementovat pouze na OS, které podporují virtuální paměť. [10]

1.5 Netmap

Netmap je framework (softwarová struktura) pro generování a zachytávání paketů z uživatelského prostředí. Architektura je postavená na sdílené paměti. Tato oblast může být přístupná z uživatelského prostoru aplikace a zároveň jádra. Obsahuje vyrovnávací paměti pro všechny pakety spravované rozhraním.

Netmap vystavuje paketové vyrovnávací paměti pro aplikaci a k zahájení přenosu dat využívá standardní systémová volání např. poll(), ioctl() pro inicializaci přenosu dat. Tyto systémová volání pouze aktualizují vyrovnávací paměti paketů a kontrolují platnost dat poskytovaných uživatelskými programy, aby se předešlo pádu aplikace.

Dokuď není aktivní žádná síťová aplikace, ovladač funguje transparentně pro OS a aplikace. Po zapnutí netmap-enabled aplikace, Network interface controller (NIC neboli ovladač síťového rozhraní) je přepnut do speciálního netmap módu. NIC přestane být aktivní pro OS a nedoručuje žádné pakety klasickému OS rozhraní aplikacím. Místo toho jsou pakety doručovány do netmapových specifických datových struktur, kde jsou dostupny pro netmap-enabled aplikaci. V případě vypnutí této aplikace se NIC přepne zpět do transparentního módu. Linux obstarává veškerou práci s pakety v samotném jádru, aby se dodržela bezpečnost na co nejvyšší možné úrovni. [11]

1.6 Porovnání

Kapitola obsahuje porovnání uvedených nástrojů: PF_RING, IPtables, NFtables a Netmap pro stanovení nejvhodnějšího filtračního/směrovacího nástroje. Byly využity informace ze zdrojů, kde docházelo k poměřování parametrů v testovacím provozu.

IPtables vs Nftables

Na první pohled se liší v syntaxi, přepínače se u nftables neinicializují pomocí pomlček. Příkazy lze zapisovat jednoduše za sebe. Větší svoboda pro konfiguraci pravidla, jedno pravidlo může zahrnout i několik výrazů, které jsou čteny z leva do prava. Pravidlo může obsahovat více akcí (např. zalogovat a přejít na jiný řetězec). Pravidla

v nftables lze aplikovat jak na IPv4 tak i IPv6, není potřeba spravovat 2 samostatné firewally. Jedna z posledních hlavních výhod může pro administrátora být, v podpoře integrace složitější datové struktury. Lze vytvářet slovníky, mapy a různé rozsahy. [12]

Podle článku [14], kde byly realizovány testy jako např. DDOS protekce, hledání shod při kombinaci adresy a portu apod. bylo zjištěno, že výkon mezi nftables a iptables je srovnatelný. Dokonce nftables jsou zanedbatelně horší při zmíněných testech, ovšem snaha vývojářů byla soustředit se spíše na funkcionalitu, než výkon. Co se týče škálovatelnosti Nftables posouvají koncept na vyšší úroveň. Rozšiřují seznam datových typů a využití v dalších aplikacích využívajících mapy.

PF_RING vs Netmap

Netmap může být více rizikové řešení, protože nevytváří pro každou aplikaci samostatnou vyrovnávací paměť pro pakety. V případě zahlcení dojde ke zpoždění všech aplikací, které využívají netmap funkci.

V Případě netmapu jsou NIC kruhy odpojeny od TCP/IP hostitelského zásobníku a pakety se přímo vyměují mezi NIC a netmap API.

Podle článku [13] bylo zjištěno, že nevýhoda obou přístupu je obcházení jádra. PF_ring nabízí řešení ve filtrování HW paketů. V případě, že má být poslán TCP paket, musí být správa těchto paketů zajištěno v uživatelském prostoru.

Jeden z rozdílu mezi netmapem a PF_ringem je v tom, že netmap nabízí chráněné využití paměti.

2 Filtrování síťového provozu

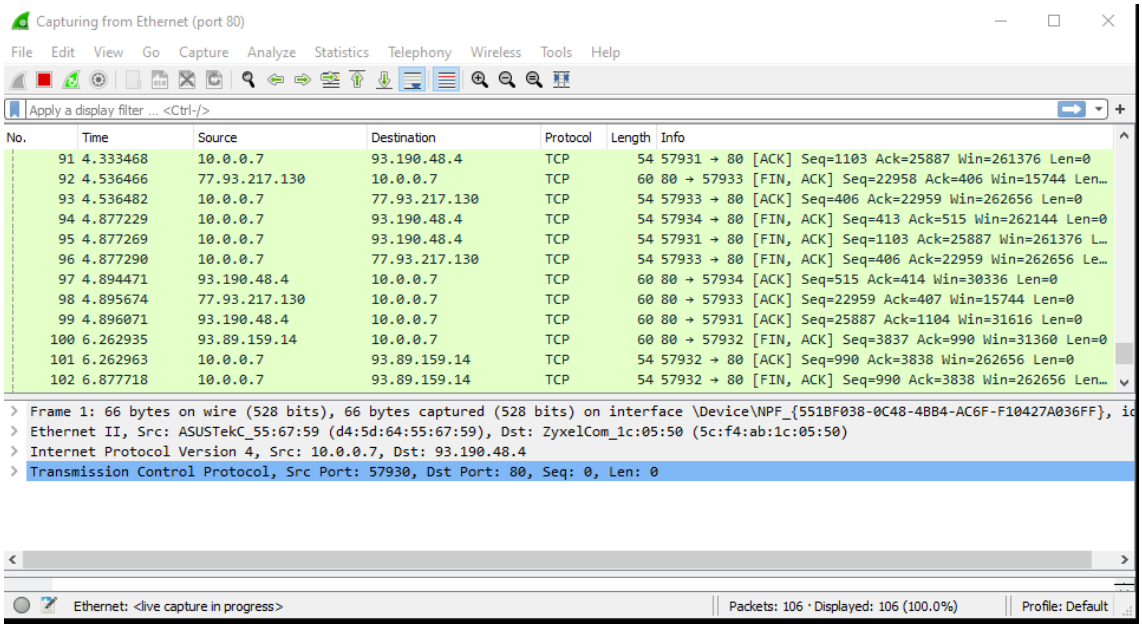
Filtrování síťového provozu je mechanismus rozhodující o tom, jaké pakety jenž dané zařízení monitoruje mají projít dál, odmítnout nebo zahodit. Můžeme to chápat jako určité události, které mají za následek určitou akci:

- **Zahodit** - paket je ignorován a vymazán. Představuje situaci, jako kdyby paket nebyl nikdy přijat.
- **Odmítnout** - firewall odesílá ICMP zprávy tomu, kdo se snažil cílovou adresu kontaktovat s důvodem, proč byl paket odmítnut.
- **Projít** - paket je připuštěn k dalšímu zpracování.

V případě, že máme v síti zařízení, které filtruje síťový provoz, slouží toto zařízení pro zabezpečení provozu např. mezi interními sítěmi. Takové zařízení lze také chápat jako určitý kontrolní bod, kterému přiřazujeme pravidla a je vytvářena komplexní bezpečnostní politika. Filtrovat můžeme podle určitých parametrů, jenž datové provozy obsahují například:

- **protokol:** TCP, UDP, ICMP,
- **port:** specifikuje číslo portu např. 3389 (RDP protokol (Remote Desktop Protocol)). Protokol je využíván pro vzdálený přístup na cílovou stanici,
- **typ paketu:** data, ICMP Echo Request, SYN/ACK, FIN,
- **src ip, dst ip:** Podle specifikace zdrojové nebo cílové IP adresy,
- **síťové rozhraní:** Zvolení ethernet vstupu daného zařízení.

Na obr. 2.1 můžeme vidět filtraci provozu pomocí programu wireshark. Došlo zde k zachycení komunikace s využitím filtru: „port = 80“. [15]



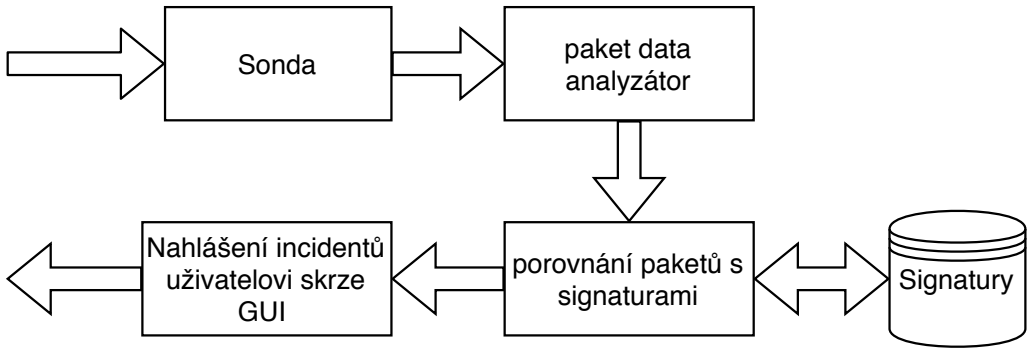
Obr. 2.1: Filtrování provozu.

2.1 Detekce síťového provozu

Síťový systém Intrusion Detection System dále už jen IDS je využíván pro monitorování síťového narušení tzv. síťových útoků a reportování těchto narušení administrátorovi, nebo osobě zodpovědné za síťovou bezpečnost. Odpovědná osoba může následně provést akce ke zmenšení bezpečnostního rizika.

IDS systémy jsou v dnešní době velice důležité, jelikož není možné držet krok se všemi hrozbami, které se neustále vyvíjí a objevují nové zranitelnosti. IDS jsou nástroje využívány administrátory za účelem usnadnění zvládání hrozeb a slabin. Hrozba jsou lidé nebo skupina více lidí, kteří mají dovednosti a možnosti kompromitovat náš PC systém.

Systémy, které jsou založeny na signaturách pro rozpoznávání hrozeb/nebezpečí jsou závislé na udržování známé databáze pro tyto hrozby aktuální. V případě, že nedojde k aktualizaci (přidání) nové signatury danému IDS systému, nebude schopný takový systém upozorňovat na nebezpečí, které může probíhat v interní síti. Účinnost takového systému je pouze taková, jak moc aktuální informace o hrozbách aktualizujeme v databázi pro daný systém. Ovšem s každou novou signaturou, se zvětšuje zátěž pro CPU. Každý paket procházející přes sondu musí být analyzován a je prohledáván, zda se pro něj nenajde shoda z naší databáze signatur [17]. Velice zjednodušeně jsi lze Architekturu IDS systému představit podle následujícího obr. 2.2.



Obr. 2.2: IDS princip.

Paket analyzátor (Paket sniffer)

Hardwarové nebo softwarové (SW) zařízení sloužící pro monitorování síťového provozu. Analyzátor funguje na principu zkoumání jednotlivých proudů datových paketů, které jsou posílány mezi zařízeními v rámci jedné sítě, nebo také mezi různými sítěmi.

2.2 Schopnosti IDS

Následující kapitola byla zpracována ze zdroje [16]. Pod schopnostmi NIDS (síťový systém detekce narušení) systému si lze představit funkcionalitu, kterou provádí. Jako např. identifikace útoků, nahlášení detailů o daném útoku, nebo samotné rozpoznání příznaků v paketech.

Určení signatur útoků

Signatura je určitý „pattern“ nebo-li příznak, kterým se daný útok projevuje. Signatury jsou modelově založené na hlavičkách paketů, za kterými následuje určitý útok. Zahrnuje to např. počet paketů od určitého zdroje (IP) s využitím dalších informací z paketu jako jsou: velikost hlavičky, TTL (doba životnosti), protokol, apod...

Identifikace útoků

Zahrnuje extrakci užitečných informací ze zachyceného provozu. Informace jako zdroj adresa, cílová adresa, typ protokolu, délka záhlaví, zdrojové a cílové porty, atd. Informace jsou následně porovnány s modelovanými signaturami útoku a dochází ke zjištění, zda došlo k útoku.

Nahlášení detailů útoku

Zahrnuje nahlášení útoku administrátorovi. Následně administrátor může vykonat takové akce, aby danému útoku zabránil v budoucnu. Toto nahlašování obsahuje specifikaci detailu útoku jako jsou: zdrojová IP, cílová IP, časové razítko útoku apod.

2.2.1 Snort

Snort je open source síťový detekční systém narušení. NIDS jsou zodpovědné za analýzu síťového provozu a testování každého paketu vůči jednotlivým pravidlům. V případě, že paket odpovídá pravidlu, NIDS systém může událost logovat, poslat upozornění a také vykonat akci jako např. zahodit paket. Poskytuje základní 3 módy.

1. **Sniffer Mode:** Zobrazuje pakety, které se přenáší po síti. Může být nakonfigurován k zobrazení různých typů paketů (TCP, UDP, ICMP) stejně tak jako k zobrazení samotných paketů, ať už záhlaví nebo samotná data.
2. **Packet Logger Mode:** Dovoluje uživateli uložit pakety detekovány z Sniffer módu, aby byly uloženy na harddisk (HDD). Skrze tento mód uživatel může specifikovat pravidla označující, které pakety uložit. Např. uložit pouze pakety vztahující se k určité adrese.

3. **NIDS Mode:** Tento mód je velice podobný módu Packet Logger. Umožňuje ovšem více specifická pravidla, které mají být aplikována na jednotlivé pakety. Aplikována pravidla jsou specifikována nebo zahrnuta v konfiguračním souboru, který je zahrnut jako parametr při zapínání snortu.

Každý z těchto módů má více variant. Jednotlivé varianty mohou být konfigurovány skrze příkazový řádek, nebo jako konfigurační soubor. V případě upozornění v NIDS módu můžeme konfigurovat obsah jednotlivých upozornění, např. kde jsou upozornění ukládána.

Snort komponenty

Snort je složen ze čtyř hlavních částí, které dohromady umožňují Snortu plnit různé režimy. Komponenty, z kterých se Snort skládá, jsou: Dekodér, Preprocessor, Upozorňovací a logovací systém, odchozí modul.

- **Dekodér:** Je zodpovědný za formování paketů, které jsou dále využity jinými částmi systému. Hlavním úkolem modulu je určit, které základní protokoly se v paketu použijí, také určit umístění a velikost dat v paketu. Získané informace se využívají v další komponentě. Také se dívá po anomáliích v hlavičkách (jako je např. neplatná velikost), která může vést k vytvoření upozornění (alert).
- **Preprocessors:** Pracují jako plugin, a jsou schopny upravovat data paketů. Toto umožňuje službám (jako jsou HTTP, FTP) mít odpovídající preprocessor k ověření anomálií specifických pro tuto službu. Jeho úkolem je v konečném důsledku pokusit se o ztížení oklamání detekčního nástroje.
- **Detekovací modul:** Hlavní komponenta, sloužící k detekci v případě, že se objeví „narušení“ v nějakém paketu. K detekci dojde pomocí aplikování jednotlivých pravidel v konfiguračních souborech. V případě, že dojde ke shodě v pravidlu, které je aplikováno na daný paket, je spuštěna akce pro dané pravidlo. Také je zároveň vytvořena událost (alert) a příslušný log pro tuto událost. Informace, které mají být ukládány do logu nebo posílány jako upozornění mohou být nastaveny v konfiguračních souborech.
- **Výstupní modul:** Má za úkol řídit typ generovaného výstupu. Může zahrnovat jednoduché lognutí na databazový server nebo zaslání e-mailů, vytváření XML reportů. [18]

Příklad pravidla

Následující IDS pravidlo (viz výpis 2.1) může být využito např. jak pro Surikatu tak i Snort. V pravidlu si lze všimnout několika částí, které slouží k určitým akcím.

Výpis 2.1: IDS pravidlo.

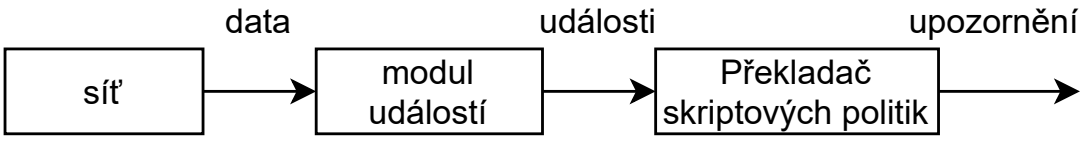
```
alert ip any any -> any any (sid:1000001;msg:"test";
content:"Bezpecnost;")
```

- **alert:** Popisuje typ akce, alert vytvoří upozornění (např. zaslání emailu) v případě shody.
- **ip:** Dovoluje aby pravidlo bylo porovnáváno s jakýmkoliv protokolem (UDP, TCP, ICMP apod...)
- **any any -> any any:** Jakýkoliv zdrojový host a port k jakémukoliv cílovému hostu a portu.
- **sid:** Identifikace pravidla pomocí čísla (speciálního identifikátoru).
- **msg:** Zpráva, která bude zobrazena při vytvoření události.
- **content:** Obsah, na kterém se pravidlo „chytne“, bude vyvolána akce.

2.2.2 Bro (Zeek)

Bro, neboli Zeek (přejmenováno v roce 2018) je IDS systém využívající jak anomálie tak i signatury. Vlastní modul na analýzu provádí převod zachyceného provozu na řadu různých událostí. Událost může být např. zalogování uživatele na FTP server, připojení k web serveru nebo prakticky cokoliv. Výhoda tohoto systému je v tom, co přichází po modulu událostí. Tzv. Policy Script Interpreter je modul, který využívá vlastní jazyk (Bro-Script) a jeho skripty plní dvě funkce. Jedna z těchto funkcí je identifikovat podezřelé činnosti na síti. Druhá hlavní funkcionalita je zapsání takových událostí do logů. Na obr. 2.3 vidíme architekturu systému. Následující body stručně popisují funkcionalitu jednotlivých vrstev znázorněných na obrázků.

Bro je systém zaměřený spíše na síťovou analýzu, vytváří velké množství logů a popisuje jednotlivé události, spíše vhodné pro analytiky. Popisuje detailněji informace, např. o spojení mezi uživateli. [19]



Obr. 2.3: Bro arhitektura.

Modul událostí

Provede několik kontrol integrit k ujištění, že záhlaví paketů jsou správně vytvořena, včetně kontrolního součtu záhlaví IP. V případě, že kontrola selže, potom Bro

vytvoří událost, která indikuje problém a zahodí paket. Dále také redukuje příchozí stream paketů na řadu událostí vyšší úrovně. Události odráží síťovou aktivitu v neutrálních pojmech „popisují, co bylo zaznamenáno, ale ne proč a zda je to důležité“. Např. každý HTTP požadavek se změní na odpovídající událost http_request, která s sebou nese příslušné IP informace (porty, adresy, uri) a také verzi HTTP.

Překladač skriptových politik

Provádí sadu obslužných událostí napsaných ve vlastním skriptovacím jazyce společnosti Zeek. Skripty mohou vyjádřit síťovou bezpečnostní politiku, které kroky podniknout v případě, kdy jsou detekovány různé typy aktivit. Z hlediska více obecného mají funkcionalitu odvození ze vstupního provozu jakékoliv požadované vlastnosti a statistiky.[20]

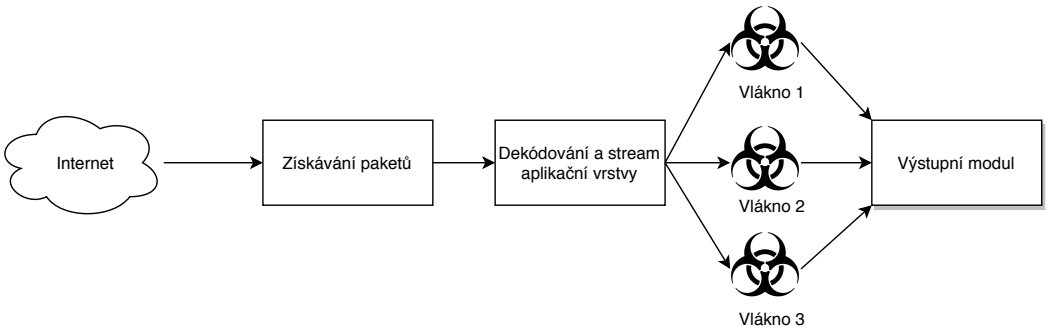
2.2.3 Suricata

Suricata je vysoce výkonný síťový bezpečnostní monitorovací nástroj, který je open source a vlastněn komunitní neziskovou nadací Open Information Security Foundation (OISF), zároveň je touto nadací i vyvíjen. Zdrojové kódy suricaty jsou licensovány pod GNUv2 (General Public License) [21].

Systém je založený na rozpoznávání hrozeb pomocí signatur, v situaci kdy je správně systém nakonfigurován je schopný provádět i real-time inspekci síťového provozu. Vyvolání událostí v případě, že je detekována podezřelá aktivita (v případě shody pravidla s paketem, který je podroben inspekci).

Suricata vlákna

Suricata je schopná běžet na více vláknech. V případě, že máme HW s více CPU nebo jádry, může být tento IDS konfigurován tak, aby rozdělil zátěž na více vláken viz obr. 2.4.



Obr. 2.4: Suricata princip vláken. Převzato z [22].

Lze použít pouze jedno vlákno a zpracovávat pakety jeden po druhém. Toto řešení by fungovalo, ovšem docházelo by zbytečně k velkému časovému zpoždění, pro zpracování velkého množství paketů. Při využití více vláken zvyšujeme výkon Surikaty a zkracujeme dobu potřebnou pro zpracování stejného množství dat v porovnání se zpracováním na jednom vlákně. Surikata se skládá ze čtyř částí.

1. **Získávání paketů:** Modul zodpovědný za čtení paketů ze sítě.
2. **Dekódování a streamování aplikační vrstvy:** Dekóduje pakety a kontroluje aplikaci.
3. **Detekce:** Porovnává signatury a může být spuštěn ve více vláknech.
4. **Odchozí modul:** V tomto modulu jsou zpracovány všechny upozornění.

Suricata také nabízí tzv. runmodes (módy běhu), samotný systém je složen z vláken, modulových vláken a front. Vlákno je totožné procesu, který běží na běžném PC. Vláknový modul lze označit jako určitou část funkcionality. Jeden modul např. slouží pro dekódování paketů, další slouží jako detekční modul a poslední odchozí modul.

Jednotlivé módy běhu jsou: single, worker, autofp. Obecně jako nejvýkonnější je považován mód worker (dělník). V tomto módu se ovladač síťové karty ujistí, že pakety jsou správně rozloženy přes vlákna Suricaty. Dodatečné informace popisující jednotlivé módy běhu lze najít v [23].

2.3 Porovnání systémů

Snort vs Bro/Zeek

Snort je IDS systém, který může pasivně nebo aktivně blokovat síťový provoz. V případě špatné konfigurace je možnost, že to povede k blokování síťového provozu, který je ovšem legitimní. Je založen na pravidlech a rozeznávání hrozeb pomocí signatur. Z toho plyne, že je schopný vytvářet událost (events) tím, že pravidlo spustí určitou akci při rozeznání hrozby v daném paketu.

Bro/Zeek je více pasivní přístup sloužící spíše k síťové analýze. Rozkládá jednotlivé záznamy do mnoha logů jako jsou (spojení, http, ssl, x509) a poskytuje všechny detaily o spojení, které je uskutečněné. Poskytuje více dat k analýze daného provozu a také více informací o tom, co se skutečně děje. Systémy založené na zásadách, jako je Bro interpretují spíše provoz, který vidí. Jedna výhoda Bro systému je, že dekóduje a ukládá na disk všechny soubory, které vidí v síťovém provozu, kontroluje hashe těchto souborů proti černým listinám. [19]

Snort vs Suricata

Jak již bylo zmíněno, Snort i Suricata jsou IDS systémy, založené na signaturách/anomáliích. Největší rozdíl je v tom, že Suricata je novější systém, který již podporuje více-vláknové zpracování narozdíl od Snortu. Pro analýzu sítě není potřeba spouštět více instancí softwaru. Při správném využití více vláknové koncepce lze dosáhnout vyšší přesnost pro detekci hrozeb než u Snortu.

V případě Snortu nezáleží kolik CPU máme, systém nedokáže využívat více CPU/vláken. S využitím jednoho vlákna, dojde při zahlcení systému k nekontrolovatelnému zahazování paketů než u více vláknového systému (Surikata).

Podle [24] lze porovnat jednotlivé hardwarové vytížení v situacích, kdy Snort nebo suricata zpracovává data, nebo běží v módu, kdy čeká na analýzu. Článek popisuje výsledek testování na jedno jádrovém procesoru pro Snort, kde výsledky jsou uvedeny v tabulce 2.1.

Tab. 2.1: Snort rozdělení CPU výkonu.

Analýza	Normalizace	Inspekce paketů
10%	10-20%	70-80%

Následujíci tab. 2.2 představuje, jak instance Snortu vytěžuje HW komponenty ve stavu kdy systém je zatížen/nezatížen. Dle článku, je Snort idealní systém pro střední provoz (okolo 400 Mb/s), více specifických informací o testování HW, na kterém běžel Snort lze najít v samotném článku [24].

Tab. 2.2: Vytížení HW Snort.

HW parametry	V zátěži	Mimo zátěž
CPU	68%	46%
RAM	76.1%	71.7%

Podle analyzovaných informací z předchozích kapitol lze říci, že Surikata je vhodnější na provoz, v řádu jednotek Gb/s. V tab. 2.3 můžeme vidět, jak systém, kde běží Suricata vytěžuje jednotlivé HW prvky.

Tab. 2.3: Vytížení HW Suricata.

HW parametry	V zátěži	Mimo zátěž
CPU	99%	44.4%
RAM	73%	69.9%

Výhoda Surikaty spočívá ve více vláknovém zpracování provozu a lepším rozložení zátěže. Možnosti více upravovat nastavení NIDS systému, podle potřeby zpracování paketů a rovnoměrnému rozložení zátěže.

Tab. 2.4 prezentuje porovnání základních dovednosti/vlastností systémů sloužících jako NIDS. Z tabulky již vyplývá, že Bro se hodí spíše pro analytické účely. Požaduje také velmi mnoho znalostí, aby s ním uživatel mohl pracovat efektivně (převážně se pracuje z logy). Jeho nevýhody vůči ostatním nástrojů jsou však zjevné z tabulky. V případě požadavků na minimálně 1 Gb/s provoz sondy se jeví jako nejvhodnější kandidát nástroj Suricata.

Tab. 2.4: Porovnání IDS systémů

Schopnosti	**Suricata**	**Snort**	**Bro**
Zpracování více vlákny	✓	✗	✗
Anomálie	✓	✓	✓
Signatury	✓	✓	✗
Kompabilitní pravidla	✓	✓	✗
Speciální jazyk	✗	✗	✓
Konfigurační GUI	✓	✓	✗
IPV6	✓	✓	✗
Jednoduchá instalace	✓	✓	✗
Podporované platformy	Windows+Unix	Windows+Unix	Unix
IPS prvky	✓	✓	✗
Nasazení ve vysokorychlostních sítích	✓	✗	✓

3 Síťová sonda

Představuje nástroj pro detekci kybernetických a bezpečnostních událostí v síti. Slouží také pro vyhodnocování událostí a sběr dat. Sonda provádí analýzu síťového provozu v reálném čase. Využívá se pro různé způsoby uplatnění, např. ochrana koncového zařízení, nebo pro analýzu vnitřní sítě, která je následně připojena do internetu. Analyzuje síťovou komunikaci, v níž detekuje známe útoky (signatury), také lze využívat pro detekci podezřelých aktivit, čímž umožňuje rozpoznat neznámé typy útoků. Důležité je, aby sonda při nasazení v síti neovlivňovala protékající data a nedošlo ke zkreslení dat určených k analýze.

Pro nasazení sondy jsou důležité dva faktory. První faktor je zvolený SW, který z velké části rozhoduje, jak bude sonda efektivní. V situaci, kdy máme dostatečný HW, ale špatně nakonfigurovaný SW může dojít k případu, že fyzicky by bylo zařízeno schopné zpracovávat data, ale SW část nebude určena pro takové množství dat viz. kapitola 2.1, kde jsou popsány základy detekce provozu a následně rozebrány jednotlivý zástupci NIDS systémů.

Druhý faktor, který je potřeba sledovat je zvolený HW. Při nedostatečném výkonu zařízení, na kterém běží daný SW může dojít k situaci, že na porty síťového rozhraní bude směrován provoz o takovém objemu dat, že sonda nebude schopna zpracovávat zátěž a začne pakety zahazovat. Také může nastat situace, kdy síťové rozhraní pakety zvládá zpracovávat, ale kapacita RAM paměti nebo výkon CPU jsou nedostatečné a při analýze provozu dochází k takovému přetížení, které může způsobit pád celého systému, na kterém běží samotný NIDS.

3.1 Porovnání HW

V první fázi realizace sondy bylo potřeba najít zařízení, který by bylo vhodné pro následnou implementaci síťové sondy. Byl kladen důraz na určité HW parametry, které by takové zařízení mělo splňovat. Jedny z nejdůležitějších, na které bylo nahlíženo při výběru jsou: procesor, počet síťových portů, síťová karta, RAM, provozní teplota, typ USB konektoru, úložiště, podpora různých OS.

Bylo nalezeno 46 zařízení, které se zdály jako vhodný kandidát pro síťovou sondu. Součet všech zařízení je uveden v příloze v tabulce A. Při zohlednění situace, že sonda by mohla být nasazená i v průmyslovém prostředí, se již počet vhodných zařízení výrazně zmenšil na kandidáty uvedené v tab. 3.1.

Na zařízení, která mají být využita v průmyslu, jsou kladeny větší požadavky ze strany HW oproti klasickým IT zařízením (PC, servery apod...). Většinu průmyslových zařízení můžeme nalézt ve zhoršených podmínkách ať se už bavíme o zvýšené teploty, prašná prostředí nebo prostor s velkou vlhkostí.

Tab. 3.1: HW pro síťovou sondu.

Výpis kandidátů na síťovou sondu			
Zařízení	CPU	Ethernet port	RAM
SBC-350E	Celeron 4305UE	2x	32GB DDR4
SBC-350V	i3-8145UE	2x	32GB DDR4
SBC-350M	i5-8365UE	2x	32GB DDR4
SBC-350P	i7-8665UE	2x	32GB DDR4
SOM-P102D	N4200	1x	8GB DDR4
SOM-P102L	N3350	1x	8GB DDR4
NUC-8665UE	i7-8665UE	2x	32GBDDR4
4X4-4500U	Ryzen 5 4500U	2x	64GB DDR4
MS-98L3-4305UE	Celeron 4305UE	4x	16GB DDR4
conga-IA5/i-E3950	Atom x7-E3950	2x	8GB DDR3l

Při následné analýze jednotlivých NIDS systému, došlo ke zjištění, že pro reálný provoz je potřeba zařízení, s velkou kapacitou RAM (minimálně 8GB pro laboratorní testování) při nasazení Suricaty. Pro využití plného potencionálu je doporučeno mít HW s minimálně 32 GB RAM a dvěma síťovými kartami viz. článek [25]. Na základě těchto požadavků, byl výběr zúžen na výrobce ASRock-Industrial a Congatec AG, který nabízí zařízení specifické pro průmyslová prostředí s dostatečnými požadavky na HW. Tento fakt omezil výběr ze zařízení, které byly nalezeny a považovány za vhodné kandidáty na tab. 3.1. V tabulce 3.1 lze vidět, že první 4 záznamy jsou stejné typy produktu, pouze s rozdílným procesorem. Je zde potřeba ovšem brát i ohled na cenu v poměru k výkonu. Další aspekt, je mít dostatek rozhraní pro připojení externích rozšíření. Jako je např Display Port/HDMI pro případ, že by došlo k výpadku možnosti vzdáleného připojení a nutnosti, se k sondě připojit a konfigurovat koncový bod na místě.

3.1.1 Analyzované zařízení

Následující sekce popisuje vybraná zařízení a jejich HW parametry. Na závěr této kapitoly je shrnut benchmark procesorů, které jsou nabízeny k jednotlivým jednodeskovým počítačům. Porovnání bylo provedeno pomocí stránky <`https://www.cpubenchmark.net`>.

SBC - 350E

Jednodeskový počítač vyráběn firmou ASrock-Industrial. Firmou je nabízen jako jako zařízení, které je možné využít v průmyslových podmínkách (výrobní haly). V tab. 3.2 jsou uvedeny parametry, které byly klíčové při výběru finálního zařízení, na které bude provedena implementace SW.

Hodnota pro RAM paměť je uvedena v celkovém přehledu viz. tab. 3.1. Samotné zařízení může být napájeno zdrojem 9-36V DC (stejnosměrný proud). Ukládání dat na externí HW je umožněno buď pomoci Sata a nebo M2 rozhraní. Pomocí těchto vstupů lze k systému připojit externí HDD pro instalaci systému, uchovávání logů a další operace vyžadující úložný prostor.

Tento typ výrobků je možné zakoupit ve více variantách, např: -350E, -350V, -350M, -350P. Jednotlivé „verze“ obsahují jediný rozdíl, a to v tom, že každá verze má implementovaný rozdílný procesor. Typy procesorů pro jednotlivé verze jsou:

- **350V:** Implementace procesoru i3-8145UE s Intel architekturou a 64bitovými instrukcemi. Procesor disponuje 4 jádry a základní frekvence je 2.2 GHz s „boost“ možností až na 3.9 GHz.
- **350M:** Implementace procesoru i5-8365UE s Intel architekturou a 64bitovými instrukcemi. Procesor disponuje 4 jádry a základní frekvence je 1,6 GHz s „boost“ možností až na 4,1 GHz.
- **350P:** Implementace procesoru i7-8665UE s Intel architekturou a 64bitovými instrukcemi. Procesor disponuje 4 jádry a základní frekvence je 1,7 GHz s „boost“ možností až na 4,4 GHz.

Tab. 3.2: Parametry produktu SBC-350E.

Parametry			
CPU	Počet jader	Frekvence [GHz]	Cena [Kč]
celeron-4305UE	2	2	2 461
Architektura:	Intel	Bity:	64
Síťový adaptér	Počet portů	Spotřeba [W]	Provoz [GB/s]
Intel® I219V	1	0,5	1
Intel® I210AT	1	Neuvedeno	1
USB	2.0	3.0	Typ-C
	2	4	0
Rozměry $[mm]$	160x 101	Provozní teplota	$0 \rightarrow 60\ °C$

Zbytek parametrů zmíněných v tab. 3.2 a 3.1 jsou identické, až na typ síťové karty. SBC-350V je vybavená Intel® I219V a Intel® I210AT. SBC-350M využívá Intel® I219LM a Intel® I210AT. Cena, za kterou jde pořídit zařízeni SBC-350E

se pohybuje kolem 7 360 Kč. U typů výše zmíněných se cena liší v závislosti na použitém CPU.

SOM-P102

Zařízení vyráběno firmou ASrock-Industrial, tato firma je zaměřena především na základní desky a patří mezi špičkové výrobce v průmyslu pro tento typ HW. Základní parametry specifické pro tento výrobek lze vidět v tab. 3.3.

Tab. 3.3: Parametry produktu SOM-P102D.

Parametry			
CPU	Počet jader	Frekvence [GHz]	Cena [Kč]
Pentium N4200	4	1,1 - 2,5	3 703
Architektura:	Intel	Bity:	64
Síťový adaptér	Počet portů	Spotřeba [W]	Provoz [GB/s]
Intel I210AT	1	x	1
USB	2.0	3.0	Type-C
	0	4	0
Rozměry $[mm]$	160x 101	Provozní teplota	$0 \rightarrow 60\ °C$

Zařízení vyžaduje napájení velikosti 12 V DC-In. Ukládání dat je zajištěno pomocí mSata a SATA3 rozhraní s přenosovou rychlostí až 6 Gb/s. Kapacita a typ RAM (náhodný přístup k paměti) jsou reprezentovány v tab. 3.1. Další možné rozhraní jsou např. M.2 a HDMI (možnost připojení zobrazovacího zařízení v případě potřeby, fyzické konfigurace). Firma nabízí pro produkt více variant, které se liší především typem procesoru:

- **P102J:** Celeron J3455 s Intel architekturou a 64bitovými instrukcemi. Základní frekvence 1.5 GHz s možností „boost“ frekvence na 2.3 GHz. Spotřeba 10 W. Disponuje 4 jádry.
- **P102D:** Celeron N4200 s Intel architekturou a 64bitovými instrukcemi. Základní frekvence 1.1 GHz s možností „boost“ frekvence na 2.5 GHz. Spotřeba 6W. Disponuje 4 jádry.
- **P102L:** Celeron N3350 s Intel architekturou a 64bitovými instrukcemi. Základní frekvence 1.1 GHz s možnosti „boost“ frekvence na 2.4 GHz. Spotřeba 4W. Disponuje 2 jádry.

Cena, za kterou lze pořídit zařízení SOM-P102D, je 6 440 Kč.

4X4-4500U

Zařízení vyráběno firmou ASrock-Industrial. Tento typ zařízení nabízí možnost několika variant, která se liší převážně typem využitého CPU. Dostupné varianty produktu jsou:

- **4X4-R1000V:** AMD Ryzen™ Embedded R-Series. Kapacita RAM až 32GB DDR4. Napájení 12V DC.
- **4x4-V2000M:** AMD Ryzen™ Embedded V2718. Základní frekvence 1.7 GHz s možností „boost“ frekvence na 4.15 GHz. Disponuje 8 jádry. 64GB DDR4. Napájení 12 - 19V DC.
- **4x4-V2000V:** AMD Ryzen™ Embedded V2516. Základní frekvence 2.1 GHz s možností „boost“ frekvence na 3.95 GHz. Disponuje 6 jádry. 64GB DDR4. Napájení 12 - 19V DC.
- **4x4-4300U:** AMD Ryzen™ 3 4300U. Základní frekvence 2.7 GHz s možností „boost“ frekvence na 3.7 GHz. Disponuje 4 jádry. 64GB DDR4. Napájení 12 - 19V DC.
- **4x4-V1000M:** AMD Ryzen Embedded V1605B. Základní frekvence 2 GHz s možnosti „boost“ frekvence na 3.6 GHz. Disponuje 4 jádry. 32GB DDR4. Napájení 12 DC.
- **4x4-R1000M:** AMD Ryzen™ Embedded R1606G. Základní frekvence 2.6 GHz s možnosti „boost“ frekvence na 3.5 GHz. Disponuje 2 jádry. 32GB DDR4. Napájení 12 DC.

Tab. 3.4: Parametry produktu 4x4-4500U.

Parametry			
CPU	Počet jader	Frekvence [GHz]	Cena [Kč]
Ryzen™ 5 4500U	6	2.3 - 4.0	Neuvedeno
Architektura:	Zen 2	Bity:	64
Síťový adaptér	Počet portů	Spotřeba [W]	Provoz [GB/s]
Realtek RTL8125BG	1	Neuvedeno	2,5
Realtek R8111FPV	1	Neuvedeno	1
USB	2.0	3.0	Type-C
	0	2	2
Rozměry [mm]	103x 102	Provozní teplota	$0 \rightarrow 60\ °C$

Deska využívá napájení 12-19 V DC v konektoru typu „Jack“. Úložiště pro data může být připojeno pomocí M.2 rozhraní, nebo SATA3, kde rychlost přenosu může

dosahovat až 6 Gb/s. Možnost využít jak DisplayPort tak i HDMI pro připojení externího zařízení, v případě nutnosti konfigurovat HW fyzicky na daném místě bez vzdáleného přístupu.

Porovnání CPU

Byly porovnány typy CPU pomocí webové stránky CPU benchmark. Na obr. 3.1 lze vidět porovnání jednotlivých CPU. Čím vyšší hodnocení má procesor, tím lepších výsledků dosáhl.

Na obr. 3.2 je reprezentováno, jaké hodnocení dostaly jednotlivé procesory, když byly prováděny testy pouze s využitím jednoho vlákna.

Při porovnání jednotlivých zařízení z tab. 3.1 vyplývá, že HW SOM nemá dostatečný výkon pro zvládání provozu vyšší než 1 GB/s zároveň s analýzou provozu. 8GB RAM je minimální kapacita pro laboratorní účely. Z tohoto důvodu je vyřazeno i zařízení CONGA-IA5. Typ MS-98L3-4305UE byl vyřazen kvůli nedostatečnému výkonu procesoru při zvážení situace, že Suricata dokáže zpracovávat data na více vláknech, jsou 2 jádra nedostačující.

Hlavní rozhodování probíhalo mezi typem SBC a 4x4-4500U. Jeden ze zásadních rozdílů je, že na zařízení 4x4-4500U lze instalovat až 64 GB DDR4 RAM a také disponuje USB-C konektorem, který může sloužit k připojení externí síťové karty.

Na základě obrázků 3.1 a 3.2 lze vyvodit, že jako nejlepší kandidát pro sondu se jeví zařízení s AMD Ryzen 5 4500U. Při porovnání situace, že AMD procesor zaostává na vláknovém zpracování vůči Intel Core i7-8665UE pouze nepatrně. Následné porovnání cena/výkon rozhodlo o tom, že jako vhodný kandidát byl vybrán HW typu 4x4-4500U.

Obr. 3.1: Porovnání celkového CPU hodnocení.

Obr. 3.2: Porovnání vůči jednomu vláknu.

3.2 Struktura sondy

Pro implementaci síťové sondy byl zvolen jednodeskový počítač od firmy ASrock-Industrial, konkrétně s označením 4x4-4500U. Disponuje USB konektorem typu C, jenž nabízí připojení výkonné externí síťové karty v případě potřeby zpracovávat provoz z více výstupů. Další výhodou, kterou lze vyzdvihnout oproti většině kandidátu je, že disponuje dvěma síťovými kartami typu Realtek R8111FPV a Realtek RTL8125BG. Detailnější popis nasazeného HW viz. kapitola 3.1.

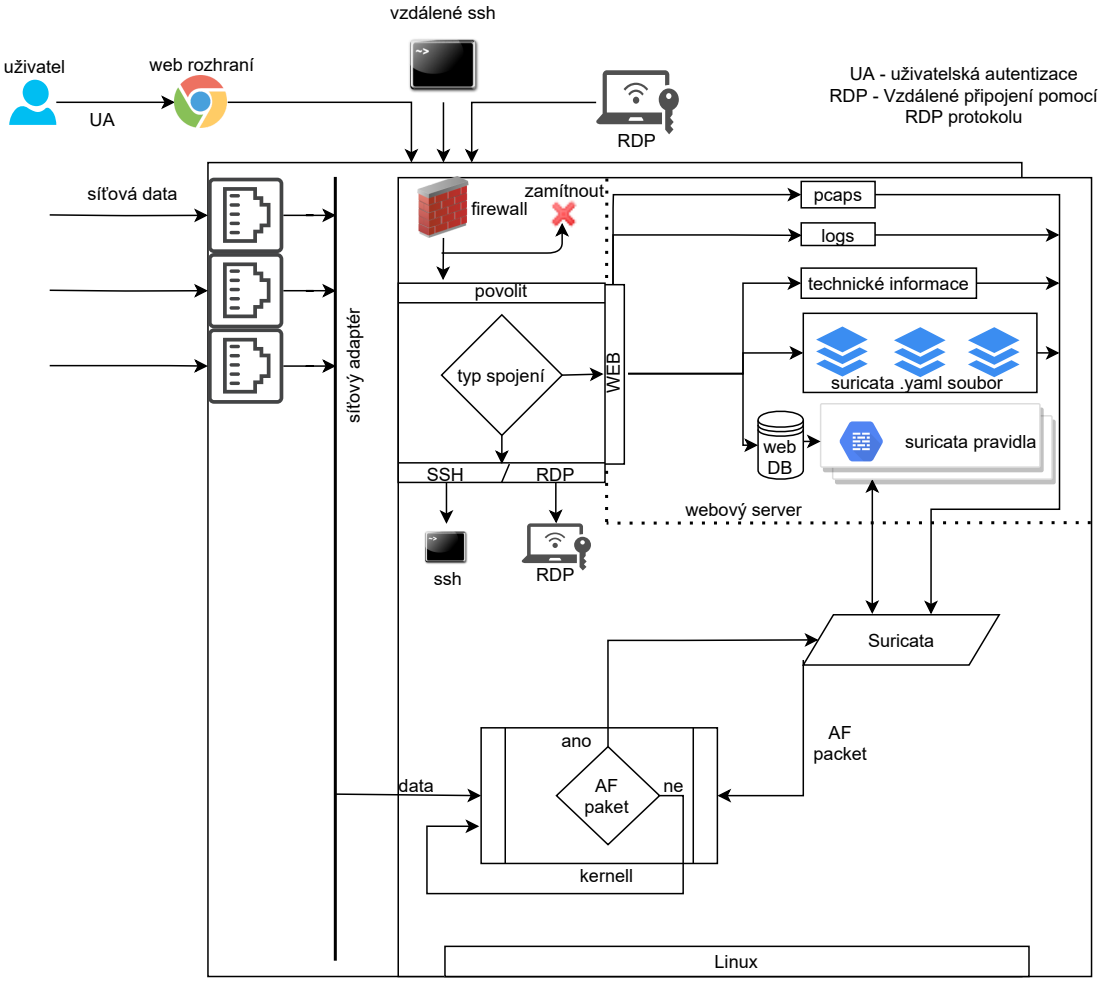
Jako operační systém byl zvolen Linux s distribucí Ubuntu 20.04. Tento OS je zvolen, z důvodu velké flexibility nastavování. Lze libovolně konfigurovat podle potřeby, všechny konfigurační soubory jsou uloženy v .txt podobě, proto je snadná i jejich úprava.

Pro filtrování provozu lze využit IPtables viz. kapitola 1.2, nebo jejich nástupce NFtables, které jsou snadnější na konfiguraci pro uživatele viz. kapitola 1.3. Pomocí pravidel lze specifikovat, která data má zařízení přijmout a dále zpracovat k další analýze. Samotná detekce příznaků již probíhá pomocí implementovaného NIDS systému. Na obr. 3.3 je zobrazena architektura pro síťovou sondu.

Architektura zobrazuje tři možné způsoby, jak se připojit k síťové sondě. První, pomocí webového rozhraní, druhý s využitím SSH připojení pomocí terminálu a poslední typ připojení s využitím vzdáleného připojení pomocí Remote Desktop Protokolu (RDP). Při nasazení SW na nové zařízení, je nutné postupovat podle manuálu, viz příloha C.

Po autentizaci uživatele bude připojení zamítnuto, nebo povoleno. Následně lze konfigurovat jak OS, tak konfigurační soubory Surikaty. Uživatel má vzdálenou kontrolu nad celým systémem. Podle oprávnění může provádět určité modifikace systému.

Pomocí portů na síťových kartách jsou do Linuxu přiváděny pakety a dochází k jejich analýze. V případě, že je nastaven AF paket na Surikatě, jsou veškerá vstupní data směrována z jádra Linuxu do softwarového prostředí Surikaty, kde dochází k aplikaci pravidel na jednotlivé pakety.



Obr. 3.3: Princip komunikace mezi webovou aplikací a síťovou sondou.

3.2.1 Popis implementace zvoleného NIDS

Bylo vybíráno mezi: Snort/Suricata/Bro. Z tab. 2.4 a informací viz. kapitoly 2.2.1, 2.2.2 a 2.2.3 vyplývá, že nejlepším řešením pro NIDS systém je Suricata v případě, že má být sonda nasazena na real-time provoz. Samotná Suricata má prvky i IPS systému, pomocí kterého lze nastavit tyto pravidla:

1. **Povolit:** Provoz odpovídající pravidlům z této kategorie bude reportován.
2. **Blokovat:** Provoz odpovídající pravidlům z této kategorie bude zahozen.
3. **Zahodit:** Pravidla z této kategorie budou ignorována.

Při správném nastavení a využití více vláken, lze efektivněji zpracovávat pakety a analyzovat je na potencionální hrozby. Linux nabízí možnost instalace Surikaty na OS pomocí již připravených apt balíčků. Před samotnou instalaci je potřeba ověřit, zda máme všechny závislosti viz. výpis 3.1, které Surikata využívá.

Výpis 3.1: Suricata instalace nutných balíčků.

```
apt-get install libpcre3 libpcre3-dbg libpcre3-dev\
                build-essential libpcap-dev
                libnet1-dev libyaml-0-2 libyaml-dev\
                pkg-config zlib1g zlib1g-dev
                libcap-ng-dev libcap-ng0 make libmagic\
                -dev libjansson-dev
                libnss3-dev libgeoip-dev liblua5.1-dev\
                libhiredis-dev libevent-dev
                python-yaml rustc cargo
```

V případě, že instalace potřebných závislostí proběhla úspěšně, lze pokračovat v instalaci Surikaty podle příkazů uvedených ve výpisu 3.2.

Výpis 3.2: Instalace Suricaty.

```
sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt-get update
sudo apt-get install suricata
```

Po úspěšné instalaci Surikaty se lze rozhodnout, zda je potřeba mít na zařízení nainstalovaný IPS mód, který využívá IPtables a NFtables pro filtrování provozu. Pro nainstalování nástrojů pro filtraci provozu na OS Linux lze využit příkazu z výpisu 3.3. V rámci diplomové práce byla nainstalována verze Surikaty 6.0.0, kde byla ověřena funkčnost IPS módu. Následně došlo k instalaci Surikaty verze 6.0.1, která byla poslední testovaná verze za účelem filtrováni provozu pomocí IPS módu.

Výpis 3.3: Pro integraci IPtables/NFtables.

```
apt-get install libnetfilter-queue-dev libnetfilter
                -queue1\
                libnetfilter-log-dev libnetfilter-log1\
                libnfnetlink-dev libnfnetlink0
```

3.3 Webové rozhraní

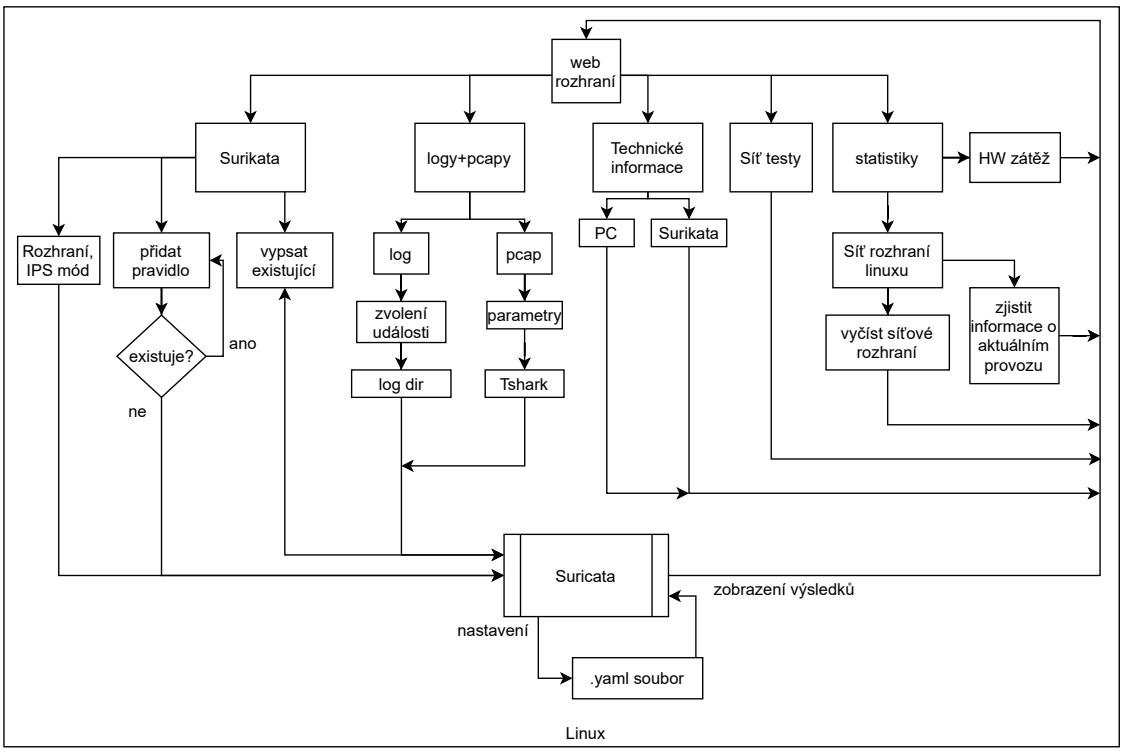
Vizuální stránka softwarove části síťové sondy byla koncipována jako multiplatformní webové rozhraní, pomocí kterého bude moci uživatel provádět konfiguraci a ovládání jako např. přidávat jednotlivé pravidla. Webové rozhraní je složeno z šesti hlavních částí a obsahuje jednotlivé „moduly“ nebo-li aplikace, které budou vykonávat požadovanou funkcionalitu. Pomocí rozhraní bude docházet především ke

komunikaci s Linuxem, vytváření jednotlivých procesů na daném OS a přepisování konfiguračních souborů Surikaty.

Jako programovací jazyk pro webové uživatelské prostředí byl zvolen Python, a pro samotnou implementaci framework Django. Framework využívá princip komunikace na základě MVC (Model-vzhled-kontroler), kde model představuje databázi pro uživatelská data, kontroler slouží pro ovládání jednotlivých prvků a vzhled je grafická prezentace ovládacích prvků (tlačítka, textová pole, apod. . .)

3.3.1 Schéma webového rozhraní

Schéma na obr. 3.4 reprezentuje aplikace, které budou zajišťovat jednotlivé operace síťové sondy. Veškeré nastavení, které uživatel provede, bude propsáno do konfiguračních souborů Suricaty, nebo budou využívat Linux nástrojů jako je Tshark pro zachytávání *.pcap* souborů na určitém síťovém rozhraní s definovanými filtry.



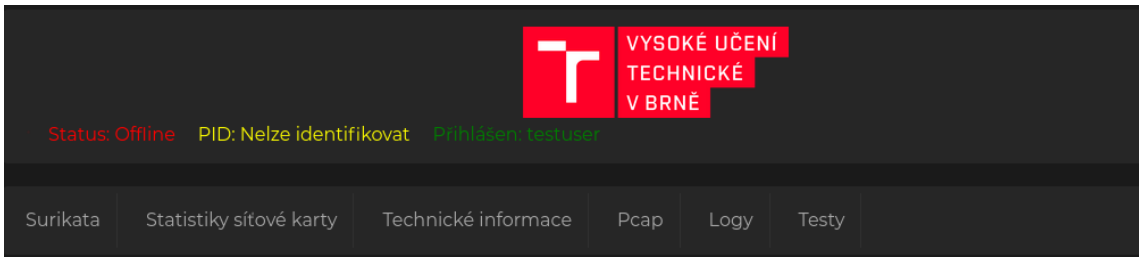
Obr. 3.4: Struktura webového rozhraní.

Na obr. 3.4 lze vidět, jakým způsobem probíhá komunikace mezi jednotlivými moduly. Jedná se o přehled základních kroků, který musí být systémem vykonány po tom, co uživatel vyvolá určitou událost přes webové rozhraní. Na schématu si lze všimnout, že samotná Surikata je konfigurována přes *.yaml* soubor, který slouží jako hlavní konfigurační soubor Surikaty, pro všechna důležitá nastavení. Pro přesnější popis jednotlivých funkcionalit, které webové rozhraní poskytuje je vytvořena

dokumentace pro aplikaci, viz příloha C. Dokumentace specifikuje, jaké informace jsou logovány pro jednotlivé protokoly při analýze provozu. Především pro část, kde dochází k analýze specifických protokolů jako je DLMS, GOOSE apod.

3.4 Aplikace webového rozhraní

Kapitola popisuje podrobně princip jednotlivých modulů, ze kterých je webové rozhraní složeno. Každý modul představuje unikátní aplikaci (viz obr. 3.5 - Surikata, Statistiky, Technické informace, Pcap, Logy, Testy), která zajišťuje funkcionalitu specifické části sondy (filtrace, záznam, logování atd.). Dále je obsahem kapitoly implementace jednotlivých modulů a vzniklé problémy jenž byly potřeba vyřešit.



Obr. 3.5: Navigační menu rozhraní.

3.4.1 Suricata

Slouží pro manipulaci s jednotlivými pravidly založenými na signaturách. Pomocí těchto implementovaných pravidel si lze udělat přehled o tom, jaké události se v síti vyskytují. Podle výsledků z jednotlivých logů lze následně odvozovat určité závěry pro dozorované prostředí. Aplikace disponuje funkcionalitou jako přidat základní pravidlo, výpis jednotlivých pravidel a následně přidat tzv. pokročilé pravidlo nebo nastavení síťového rozhraní (viz příloha obr. B.2). Tyto funkce jsou především pro situaci, kdy Surikata běží v IDS módu. Pro IPS mód je zde vytvořeno ještě nastavení nazvané IPS nastavení, které slouží k přidávání pravidel do iptables. Při správné manipulaci je síťový provoz přesměrován do Surikaty, kde dochází k aplikací odmítnutí/zahození (reject/drop) pravidel na daný jednotlivé pakety. V případě shody jsou drop události ukázané v logovací části Surikaty, která zobrazuje vytvořené událost (fast.log).

Jak již bylo zmíněno, NIDS systém může běžet v IDS nebo IPS módu s využitím sondy jako takzvaného aktivního nebo pasivního prvku. Aktivním prvkem se myslí to, že zařízení bude muset vykonávat filtraci provozu, nebo-li chránit zařízení dále v síti. Pro tohle nastavení zařízení lze využít IDS i IPS mód. Druhý režim je nazvaný pasivní, jelikož zařízení se chová v síti jako bridge a provoz pouze přeposílá na cílové zařízení. Zde je možné logovat a analyzovat provoz bez možnosti použití IPS módu.

Instalovaná verze Surikaty byla ponechána ve výchozí složce a samotné spuštění probíhá v automatizovaném režimu pomocí systémové služby *vut_sonda.service* umístěné v */etc/systemd/system*.

Přidat pravidlo

Základní modul, pomocí kterého lze nadefinovat signaturu, která slouží k účelu, který definoval uživatel viz kapitola 3.2.1. Disponuje možnosti specifikovat základní parametry, které mají být uloženy do DB (databáze) pravidel. Mezi tyto parametry patří např: akce, která má být vykonána při shodě pravidla s daty ve snímaném provozu, protokol pomocí kterého jsou data přenášena (ICMP, UDP), zdrojová síť, zdrojový port, cílový port apod. Systém byl nastaven tak, aby nedovolil uživateli přidat pravidlo s takovým ID (identifikační číslo pravidla), které již v DB existuje. Samotné ID je ve webovém rozhraní prezentováno s názvem SSID, protože zkratka ID je již reprezentována v DB jako PK (primární klíč) pro identifikaci záznamů (řádků v DB). Při přenastavení hlavního klíče v DB by mohlo dojít k inkonzistenci dat. V případě pokusu o přidání pravidla již s existujícím ID, bude uživatel upozorněn pomocí webové rozhraní a může ID nahradit za jiné.

Implementována je také funkce pro vytvoření kopie DB do textového souboru, která může být využita jako export pro uživatelská pravidla. Při využití této funkcionality je ovšem potřeba dávat velký pozor na to, abychom si uvědomily, že soubor formátu *.txt*, kde budou pravidla exportována bude, vždy přepsán pro aktuální stav DB. Všechna pravidla z předchozí verze budou přepsána.

Jedna z dalších funkcionalit, která je důležitá pro aktualizování Suricata pravidel je umožnit uživateli restartovat samotný NIDS. V situaci, kdy je systém restartován, dochází k načítání signatur ze všech souborů, které jsou specifikované v suricata.yaml.

Pro účely testování, došlo k vytvoření souboru s názvem my.rules, který je uložen v adresáři uvedeném ve výpise 3.4.

Výpis 3.4: Cesta k souboru pro uživatelská pravidla.

```
/etc/suricata/rules/my.rules
```

Při snaze o přístup ke konfiguračnímu souboru *my.rules* bylo zjištěno, že Linux vyžaduje vždy heslo pro sudo uživatele. Z tohoto důvodu je potřeba konfigurovat ještě soubor *sudoers*, kde jsou definována bezpečnostní pravidla, jako např. pro jaké uživatele jsou dostupné jednotlivé příkazy apod. Aby byl uživatel schopný využívat veškerou funkcionalitu webového rozhraní je potřeba změnit konfiguraci *sudoers* souboru příkazem *sudovi sudoers*.

Pro konfiguraci souboru je nutné využít příkaz *visudo*, jinak změny nebudou platné. Z domovského adresáře provedeme editaci pomocí příkazu uvedeného ve výpisu 3.5.

Výpis 3.5: Konfigurace /etc/sudoers.

```
sudo visudo sudoers
```

Následně bude otevřen konfigurační soubor, kde jsou nastaveny parametry pro uživatele root. Dál je nutné přidat za zakomentovaný řádek *includedir /etc/sudoers.d* parametr, který odstraní nutnost zadávat heslo pro sudo příkazy. Tento krok provedeme pomocí přidáním řádku z výpisu 3.6.

Výpis 3.6: Přidání konfigurace pro uživatele.

```
username ALL=(ALL) NOPASSWD: ALL
```

Username je uživatel, pro kterého chceme přidat specifické oprávnění. Následně pro dostupnou konfiguraci souboru *my.rules* je nutné ještě přidat souboru potřebná oprávnění, využijeme příkaz uvedený ve výpisu 3.7.

Výpis 3.7: Nastavení oprávnění pro složku.

```
chmod 777 my.rules
```

Po výše zmíněných krocích je možné již při plné funkcionalitě webového rozhraní přidávat pravidla. Na webovém rozhraní je také funkcionalita, která dá příkaz Suricatě, aby prošla konfigurační soubory pro pravidla a zjistila, zda se tam nenachází nové pravidlo.

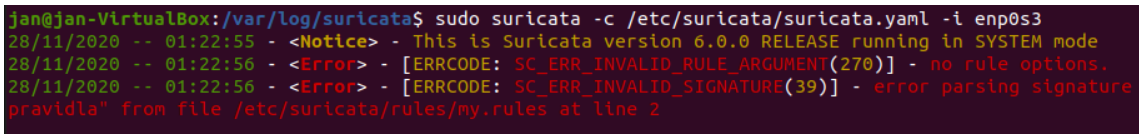
Na obr. 3.6 je uveden příklad, kdy uživatel přidal 2 pravidla. První pravidlo, které detekuje jakékoliv ICMP spojení a druhé, které nemá specifický význam pro detekci a bude hlásit chybu, protože není ve správném formátu.

```
jan@jan-VirtualBox:/etc/suricata/rules$ cat my.rules
alert icmp any any -> any any (msg:"ICMP test"; sid:100000001; rev:1;)
test nefukcniho pravidla
```

Obr. 3.6: Výpis ze souboru pro přidávání pravidel.

Po přidání pravidla je potřebné restartovat Suricatu viz obr. 3.7, kde byl využit příkaz, který zahrnuje i specifikaci síťového rozhraní, na jakém má být systém spuštěn.

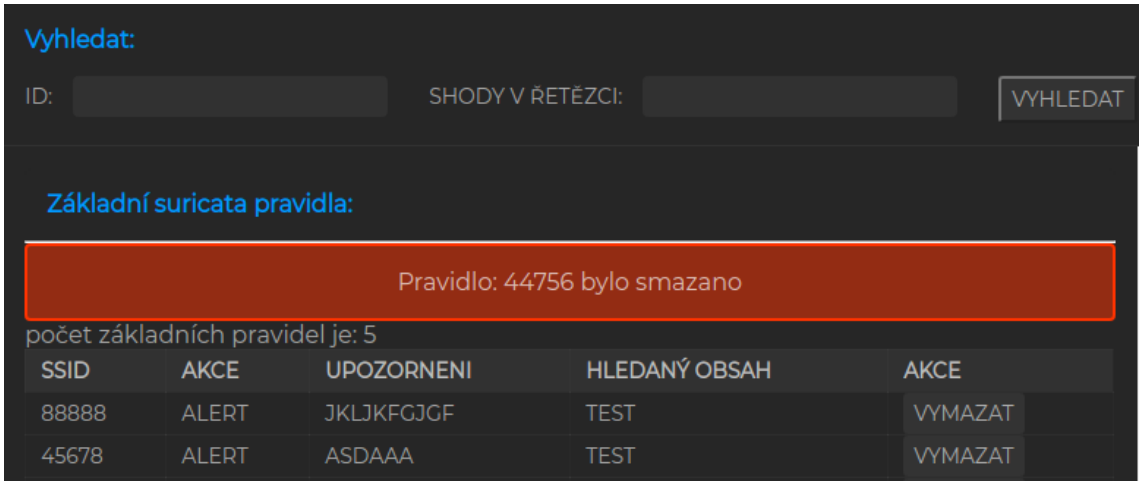
Zároveň lze vidět, že samotná Surikata byla spuštěna i když na obr 3.7 je zobrazena chybová hláška „error parsing signature“. Tento problém je způsoben špatnou syntaxi samotného pravidla. Proto je důležité při přidávání pravidel kontrolovat každý detail, podle dokumentace, jak se pravidla mají zapisovat.



Obr. 3.7: Upozornění na chybu v detekčním pravidle Surikaty.

Výpis pravidel

Funkce modulu, který slouží pro manipulaci se Surikatou. Slouží k vyhledávání a vizualizaci existujících pravidel z DB viz obr. 3.8. Disponuje také možnosti vyhledávat jednotlivé pravidla podle definovaného ID, které je reprezentováno v rámci Surikata pravidel jako identifikátor SSID, nebo podle „shody v řetězci“, která hledá shodu v sloupci „AKCE“. Při vyhledávání podle shody jsou z databáze vráceny všechny pravidla, které obsahují hledanou posloupnost.



Obr. 3.8: Vyhledávání pravidel.

- SSID: Identifikátor pro NIDS pravidlo, které je přidáváno uživatelem. Musí být unikátní.
- AKCE: Zvolení, jaká událost má být provedena s pravidlem v případě shody.
- UPOZORNENI: Uživatel zde vidí, jaké upozornění nastavil pro detekční pravidlo.
- HLEDANÝ OBSAH: Specifická sekvence znaků, která má být hledána v paketu síťového provozu.

Upozornění ve tvaru „Pravidlo:44756 bylo smazáno“ je oznamovací hláška systému. Slouží pro vizualizaci informací o tom, jaké pravidlo uživatel smazal, pomocí tlačítka VYMAZAT ve sloupci AKCE.

Pokročilá pravidla

Slouží především pro uživatele, kteří mají pokročilejší znalost přidávání Surikata pravidel. Jelikož pravidla jsou velice citlivé na chyby. Při špatném formátu pravidlo nebude úspěšně načteno do Surikaty. Z tohoto důvodu, by měl uživatel, který přidá jakékoliv pravidlo zkontrolovat, zda bylo úspěšně načteno pomocí restartování systému a následné kontroly logu. Sekce pokročilého pravidla umožňuje zkopírovat již existující, nebo ověřené pravidlo do textového pole a následně ho pouze uložit do DB pravidel. Není zde potřeba ručně nastavovat jednotlivé parametry, jako bylo v sekci přidání základního pravidla. Grafické rozhraní je uvedeno viz příloha B obr. B.4.

Grafický vzhled vyhledávání je stejný jako na Obr. 3.8. Hlavní rozdíl je v parametru „Shody v řetězci“. Zde není vyhledáváno pomocí sloupce „AKCE“ ale je vyhledávána shoda na daný výraz v rámci celého pravidla.

Nastavení rozhraní

Slouží k ovládání detailnějšího nastavení Surikaty. Umožňuje volit, zda má být NIDS systém využíván v módu IDS nebo IPS pomocí výběrového pole viz příloha obr. B.5. Zjistit, v jakém módu se síťová sonda nachází, lze v hlavičce stránky, kde vedle statusu (Offline/Online) je dále uvedeno IDS/IPS. Modul umožňuje zvolit více rozhraní, na které má být systém spuštěn, stačí podržet klávesu CTRL a vybrat více možností. V případě, že bude využit mód IPS, je důležité zmínit, že není bráno v úvahu rozhraní, které je zvoleno. Surikata musí být spuštěna jiným způsobem než u IDS módu, z tohoto důvodu dochází ke kontrole *suricata.yaml*, jenž slouží jako konfigurační soubor pro IPS mód.

Samotná sonda byla navržena tak, aby šla využívat pro analýzu a filtraci mirorovaného provozu. To znamená, že na síťovém prvku, např. switchi dojde k nastavení, že všechen síťový provoz je duplikován na port, z kterého jsou data přeposílána na síťovou sondu. Lze říci, že v tomto zapojení může být síťová sonda „aktivním prvkem“ a lze využít IPS mód, pro filtraci určitého provozu pomocí IPtables popsané v následující podkapitole „IPS nastavení“.

S využitím možnosti přepnout zařízení do módu bridge je vytvořeno rozhraní s názvem *br0*. Po aktualizaci stránky je přidán mezi rozhraní. Pro kontrolu konfigurace lze využít tlačítko „ZOBRAZIT BRIDGE NASTAVENÍ“. Konfigurace bridge je provedena pomocí nástroje netplan, který využívá *.yaml* soubor v */etc/netplan* adresáři. V případě konfigurace se nesmí konfigurovat přímo tento soubor, ale soubor v umístěny *bash_scripts/bridge_config.txt* v adresáři aplikace. Při využívání bridge není umožněno využívat IPS mód sondy. Další funkcionalita k využití je vypínání a restartování NIDS systému.

Důležité je, aby nedošlo k přejmenování názvu síťového rozhraní pro síťové karty a logicky vytvořený bridge. V případě měnění názvu rozhraní/bridge není zaručeno, jak se SW bude chovat.

IPS nastavení

IPS je implementováno za účelem filtrace nežádoucího provozu a lze ho také využít jako ochranný síťový prvek v případě, že by se provoz dál zpracovával na zařízení umístěným za sondou. Jak již bylo zmíněno v předchozí podsekci, IPS nelze nastavit na sondu v situaci kdy je v módu bridge. Samotná webová aplikace, to sice fyzicky umožňuje, ale žádná IPS pravidla nebudou na síťový provoz aplikovaná. Filtrace lze využít v případě, že na zařízení jsou posílána data ze SPAN portu, která jsou analyzována a zpracována sondou pro síťový provoz. Vizualizace této funkcionality je znázorněna viz příloha obr. B.6.

Manipulace s touto sekcí lze prezentovat jako kdyby uživatel pracoval s terminálovým rozhraním akorát ve webovém rozhraní. Pravidla zde mohou být přidávána do iptables, aby dané iptables pravidlo správně fungovalo je potřeba využít přepínač *-j* s hodnotou *NFQUEUE*, viz výpis 3.8.

Výpis 3.8: Přesměrování provozu do NFQUEUE.

```
iptables -I INPUT -j NFQUEUE
```

Předchozí příklad pro iptables znamená, že budou všechny vstupní pakety pro dané zařízení přesměrovány do fronty NFQUEUE, nebo-li dojde k přesměrování tohoto provozu do Surikaty. Podle pravidel drop/reject v Surikatě, které lze přidávat pomocí webového rozhraní bude provoz filtrován.

Při znalosti iptables lze s rozhraním pracovat jako s příkazovou řádkou. Lze vymazat všechna pravidla, odstranit určité pravidlo ze specifického řetězce (viz výpis 3.9).

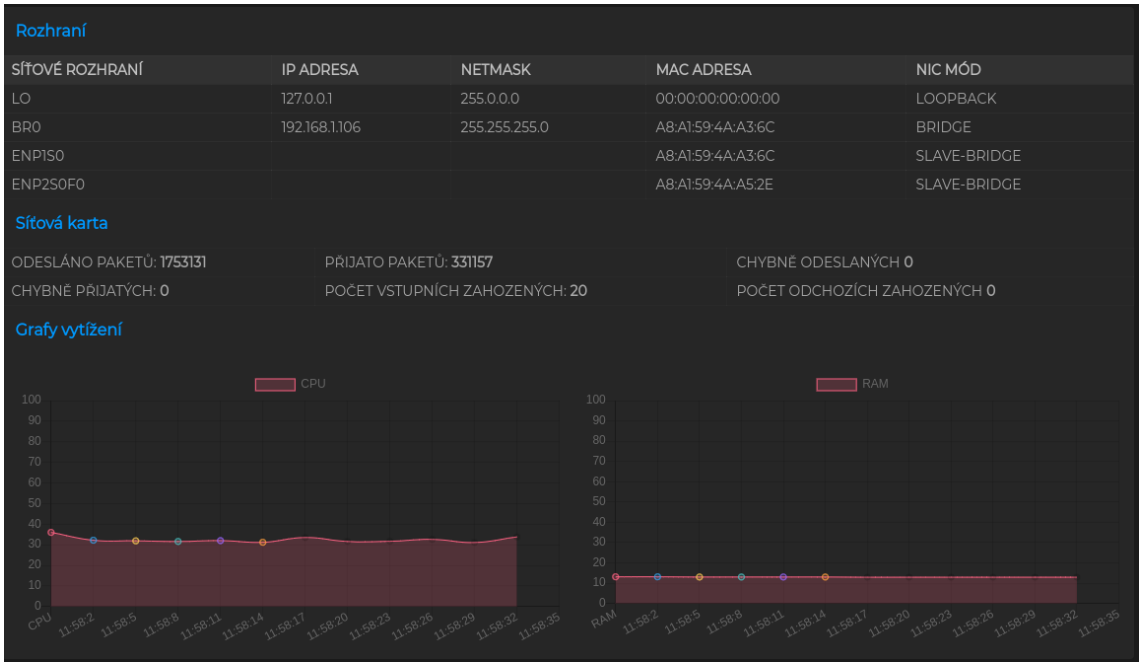
Výpis 3.9: Příklady mazání IPtables záznamů.

```
iptables -F
iptables -D INPUT 1
```

Přepínač *-F* vymaže všechny iptabls pravidla. Pomocí následujícího příkazu s přepínačem *-D* lze odstranit první pravidlo podle indexu ze specifického řetězce. Došlo také, k implementaci několika pravidel zároveň, kdy stačí za každé pravidlo přidat znak „;“, který slouží jako oddělovač. Ve spodní části lze identifikovat všechna pravidla přidané do iptables viz příloha obr. B.6.

3.4.2 Statistiky síťové rozhraní

Modul, který představuje zobrazení informací pro uživatele (viz obr. 3.9). V první tabulce nazvané Rozhraní lze vidět: SÍŤOVÉ ROZHRANÍ, IP ADRESA, NETMASK, MAC ADRESA, NIC MÓD. Síťové rozhraní představuje výpis název síťových karet z daného HW (např. ENP1S0). Sloupec s názvem IP adresa vypisuje pro každou síťovou kartu aktuálně nastavenou IP adresu. Netmask udává rozsah adresního prostoru, který je použit pro síť, lze dopočítat kolik hostů lze umístit do dané sítě. MAC adresa je pevně daná adresa síťové karty. Poslední sloupec NIC MÓD prezentuje informace o tom, v jakém stavu se daná síťová karta nachází. Mohou se objevit tyto stavy: ETHERNET, BRIDGE, LOOPBACK, SLAVE-BRIDGE. Lze si všimnout, že se mezi síťovým rozhraním se objevil záznam s názvem BR0, který byl automaticky přidán do přepnutí sondy do Bridge módu. To znamená, že sonda byla přepnuta do BRIDGE režimu a pouze přeposílá a analyzuje provoz.



Obr. 3.9: Statistiky síťového rozhraní.

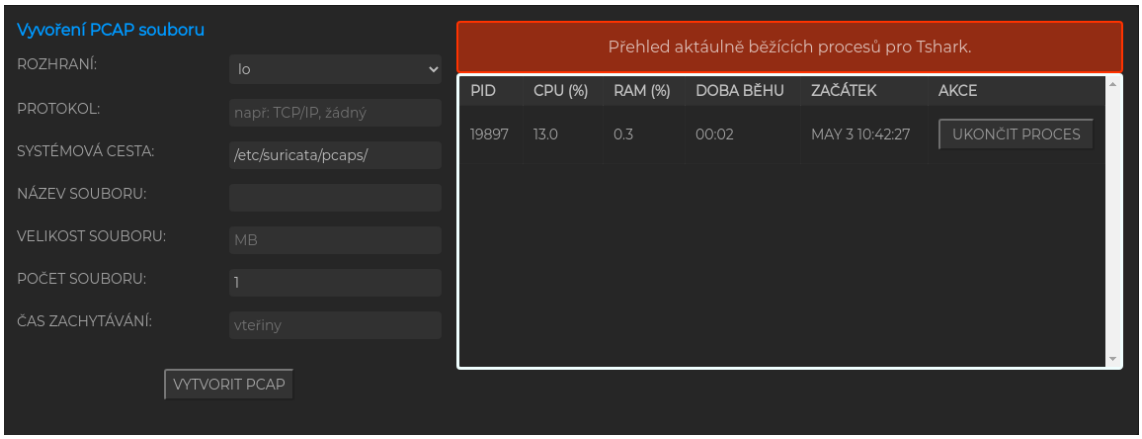
- BRIDGE – Nastavení pro síťovou kartu, při kterém sonda přeposílá síťový provoz na zařízení, bez toho aniž by došlo k modifikaci paketů. Slouží pro záznam a analýzu síťového provozu.
- SLAVE-BRIDGE – Informuje uživatele o tom, které síťové rozhraní jsou přiřazené pod logicky vytvořený bridge. Konfigurace v souboru *bridge_config.txt*, který je umístěn v adresáři aplikace.
- Loopback – Loopback rozhraní pro zařízení.
- Ethernet – Zahrnuje všechny zbyle nastavení pro síťové rozhraní.

Druhá tabulka nazvaná Síťová karta reprezentuje informace i síťovém provozu, např. počet přijatých a odeslaných paketů. Také jsou uvedeny informace o počtu chybně odeslaných a přijatých paketů.

Poslední část s názvem Grafy vytížení slouží pro kontrolu aktuálního vytížení CPU a RAM. Informace jsou aktualizovány v reálném čase v intervalu tří vteřin po dobu otevření záložky „Statistiky síťové karty“. Získávání dat pro grafy vytížení je vytvořeno pomocí API (Aplikační programovací rozhraní). Při správném zadání URL adresy lze aktuální data pro vytížení získat i z externích skriptů/programů. Data jsou prezentována ve formátu JSON.

3.4.3 Pcapy

Modul, jenž je založen na programu Tshark pro zachytávání paketů do *.pcap* souboru na daném zařízení. Samotná komunikace mezi webovým rozhraním a programem Tshark probíhá pomocí podprocesů v Linuxu, kde je pomocí terminálu vytvořen proces, který má za úkol splnit funkcionalitu příkazu. Skrze webové rozhraní je možné zadávat parametry, podle kterých má být *.pcap* soubor vytvořen, viz obr. 3.10.



Obr. 3.10: Zadávaní parametrů pro vytvoření pcapu.

Uživatel může dynamicky volit podle síťových karet (ROZHRANÍ), které jsou k dispozici, na jakém rozhraní bude daný síťový provoz zachytáván. Lze využít např. velikostního limitu, kdy je zadáno, že chceme zachytit pět (POČET SOUBORU) souboru o velikosti 50MB (VELIKOST SOUBORU). Nebo využít časové omezení a nastavit dobu (ČAS ZACHYTÁVÁNÍ), po kterou chceme provoz zachytávat. Název, pod jakým bude *.pcap* soubor vytvořen je určen podle pole NÁZEV SOUBORU. Dále je umožněno i vytváření několika *.pcap* souborů pomocí pole POČET SOUBORU, lze využít např. v kombinaci: VELIKOST SOUBORU = 50, POČET SOUBORU = 5, s příslušným názvem pro soubory. Výsledkem je, že systém vytvoří postupně pět *.pcap* souborů, ve formátu:

```
název_datum_čísloSouboru_časovéRazítko
```

V pravé části obr. 3.10 je vidět tabulka, která reprezentuje jednotlivé informace o každém procesu, který byl vytvořen pomocí programu Tshark. V situaci, kdy dojde k vytvoření příkazu na zachytávání provozu jsou v tabulce zobrazeny příslušné informace o daném procesu, který uživatel vytvořil. Návrh na zachytávání provozu, byl realizován tak, aby mohlo být zachytáváno několik *pcap* souborů zároveň, není nutné čekat, než předchozí proces skončí, aby mohlo dojít k zaznamenávání síťové komunikace pro další *pcap*. Uživatel může díky přehledu ukončit jakýkoliv proces, který je spojen s programem Tshark pomocí tlačítka „Ukončit Proces“.

3.4.4 Technické informace

Část webové aplikace, která popisuje informace o HW na kterém je nasazený NIDS, výpis stručných informací o NIDS systému. V první části lze zjistit informace o architektuře CPU, typu CPU, kapacitě RAM, OS apod.

Druhá část slouží spíše pro ověření, jaká verze NIDS systému je aktuálně využívána, aby bylo jednoduché v případě problému dohledat v dokumentaci řešení na dané problémy. Počet CPU, které NIDS využívá a v případě výchozí instalace Surikaty je uvedena cesta k logům pro případ, že by některý z logů musel být zálohován viz obr. B.8 v příloze.

3.4.5 Logy

Část webového rozhraní sloužící primárně pro analýzu událostí, které se odehrály v systému v určité době. Logy zaznamenávají události pro moduly webové aplikace, typy logů: logování TCP/UDP provozu, Surikata, PCAP. Logy, které vytváří webová aplikace jsou v podobě *.log* souboru a pro každý log je specifikována určitá syntaxe, v jaké je daný log vytvořen.

Pcap logování

Funkcionalita v logovacím modulu aplikace, která slouží pro záznam, kdy uživatel zadal příkaz pro vytvoření pcap souboru pomocí webového rozhraní. Webové rozhraní neumožňuje zásah do logu (editaci), slouží pouze k zobrazení informací, které jsou v patřičném souboru uloženy. Pcap soubor má syntaxi viz výpis 3.10.

Výpis 3.10: Výpis z logů po zadání příkazu na zachytávání soboru.

```
03/02/2021/23:21:37-<INFO>- Zachytávám PCAP:
-i enp0s3 -f "tcp" -w /etc/suricata/pcaps/Spojeni.pcap
-b filesize:5120
```

Jednotlivé parametry zobrazené ve výpisu 3.10. se mění podle toho, jakou kombinaci vstupů uživatel zvolí na webovém rozhraní. Mohou nastat 2 typy zpráv v logu <INFO>, která popisuje úspěšné vytvoření procesu, dochází k zachytávání síťového provozu do souboru. Druhá varianta je <ERROR>, která slouží k tomu, aby došlo k záznamu vstupních parametrů, které uživatel zadal. Nebo situace, kdy nebylo možné takovou kombinací vstupů využít programem Tshark.

Surikata logování

Logování Surikaty je složeno ze čtyř částí, které vyjadřují informace o NIDS systému viz obr. B.9 v příloze. V situaci, kdy nastane nějaký problém spojený se Surikatou, lze tuto část využít pro hledání chyb. Při znalosti jednotlivých logů lze rychle a efektivně zjistit např. zda pravidlo přidané do NIDS systému bylo úspěšně načteno detekčním modulem a je aplikováno na příchozí data.

V případě, že je potřeba logovat mnoho událostí (http, apod.), lze vytváření těchto logů povolit v *Surikata.yaml*. Je potřeba brát v úvahu, že čím více logů bude systém vytvářet, tím větší dopad to může mít na výkon systému.

- **SURICATA.LOG:** Jedná se o log, ve kterém jsou obsaženy všechny informace (typy chyb), které mohou ovlivnit, nebo mít určitým způsobem vliv na správný chod Surikaty (viz. příloha obr. B.9). V tomto logu nejsou záznamy o událostech, ke kterým došlo v rámci pravidel. Při ovladání Surikaty (restart, vypnutí, zapnutí) bude tento log aktualizován pro aktuální nastavení. Je zde vypsán i počet úspěšně/neúspěšně načtených pravidel. Přidané detekční pravidla, které obsahují chybu ve své syntaxi jsou vypsány ve tvaru viz výpis 3.11.

  Výpis 3.11: Záznam o špatně přidaném pravidle.

  ```
  Datum -<Error>-[ERRCODE: SC_ERR_INVALID_SIGNATURE(39)]
  -error parsing signature -> obsah pravidla
  ```

  Z výpisu lze určit, datum, kdy nastal problém a také, že se jedná o chybovou hlášku (-<Error>-). ERRCODE vyjadřuje typ chyby. SC_ERR_ INVALID_ SIGNATURE informuje o chybě v syntaxi pravidla. Jako poslední položka za znakem „->“ je již výpis pravidla, které se nepodařilo načíst.

- **SURICATA-START.LOG:** Obsahuje informace spojené se startem systému (Surikaty) Lze zde zjistit informace např o počtu jader CPU, na jakém rozhraní se snaží systém spustit Suricatu apod. Automaticky vytvářen po zapnutí OS. Nezapisují se zde informace při následné manipulaci (vypnutí/zapnutí) Suricaty ručně pomocí webového prostředí.
- **FAST.LOG:** Prostřednictvím tohoto logu lze zkontrolovat, zda naše nově přidané pravidlo funguje. Slouží jako nejrychlejší způsob ověření, že detekční pravidlo, které je implementované do Surikaty funguje při odpovídajícím provozu. Případně pro kontrolu, jaká pravidla jsou nejčastěji detekována.
- **IMPORT.LOG:** Log, který je vytvořený specificky pro webovou aplikaci. Hlavním cílem je zaznamenat veškeré pravidla, která uživatel přidává v sekci Surikata > přidat pokročile pravidlo > „Choose File“ a následném nahrání souboru do webové aplikace.
  Význam tohoto logu je v tom, že v situaci, kdy uživatel vytvoří *.txt* soubor, který obsahuje pokročilá pravidla pro detekci neví, která pravidla byla úspěšně aplikovaná a která byla odmítnuta. Odmítnutím se myslí, že aktuálně přidávané pravidlo má hodnotu SSID, která již existuje v DB pravidel. SSID je unikátní identifikátor, podle kterého lze vyhledávat pravidla a musí být zaručeno, aby zůstalo v rámci DB unikátní.
  Ve výpisu 3.12. lze vidět, že pravidlo nebylo úspěšně importované do DB pravidel. Z tohoto důvodu nebylo ani přidáno mezi detekční pravidla pro síťový provoz.

Výpis 3.12: Import.log příklad.

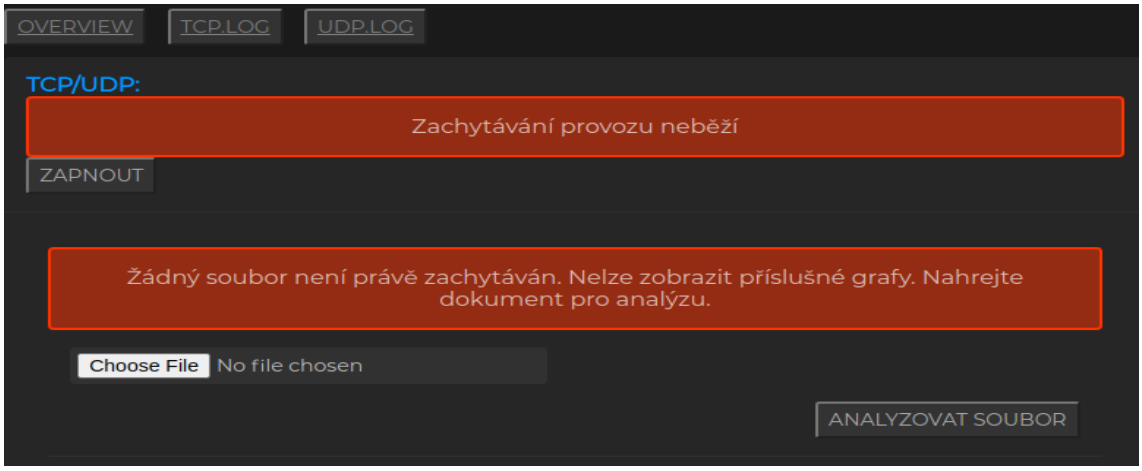
```
01/02/2021 11:36:24-> Pravidlo nebylo přidáno >>
drop tcp HOME_NET any -> EXTERNAL_NET any
(msg:"'ET_TROJAN_Likely"';flow:established,to_server;
classtype:trojan-activity;sid:27000;)
```

  Druhá možnost je, že je splněna podmínka unikátnosti SSID hodnoty. Místo „Pravidlo nebylo přidáno“ uživatel uvidí „Pravidlo přidáno“, po následném restartu Surikaty toto pravidlo bude implementováno a zařazeno mezi aktivní detekční pravidla.

TCP/UDP logování

Představuje záznam komunikace, který je přenášen pomocí TCP/UDP protokolu, které pracují na 4 vrstvě ISO/OSI modelu. Každý záznam „řádek“ představuje unikátní paket v rámci určitého síťového spojení. Jsou zde sledovány pakety ze zdrojového zařízení k cílovému zařízení, které jsou unikátní pro dané spojení. Například,

pro zdrojovou IP adresu s určitým portem na specifickou cílovou IP adresu s určitým portem bude paket v TCP komunikaci s příznakem SYN zaznamenán pouze jednou. V případě, že se změní některá informace v paketu (port, IP, typ přiznaku) je paket zaznamenán znovu. Modul umožňuje analyzovat komunikaci ve dvou stavech - nahrát soubor (Choose File) pro analýzu nebo zapnout analýzu síťového provozu v reálném čase (viz obr. 3.11). Bližší popis je uveden v textu níže.



Obr. 3.11: Způsob analýzy síťového provozu.

Dále modul obsahuje 3 záložky: Overview, TCP.LOG, UDP.LOG, které dávají uživateli obecný přehled o TCP a UDP provozu. Záložka Overview slouží k grafickému zobrazení, ve formě grafů pro TCP a UDP provoz viz graf v příloze B.10. Záložky TCP.LOG a UDP.LOG slouží k detailnějšímu náhledu na záznamy paketů, uživatel vidí specifické informace o IP, portech, typech paketu.

Real-time analýza

Spuštění scriptu, který pomocí knihovny *psutils* vyčítá systémové spojení přes systémový soket. Samotný script je volán v intervalu 2 vteřin. Při využívání tento funkcionality je CPU odebráno vlákno, na kterém běží analýza síťového provozu. Z tohoto důvodu dochází k mírnému poklesu výkonu pro daný HW. Pro každé síťové spojení je pokus o dohledání PID pro tento provoz. V případě spárování procesu s daným síťovým spojením je vytvořen záznam do log souboru s pojmenováním ve tvaru:

Datum_čas_TCP_UDP_traffic.log

Při ukončení real-time analýzy je uložen do příslušného adresáře s datem a časem ukončení. Při každém zapnutí real-time analýzy je vytvořen log, které je možné následně zpětně nahrát a provést tzv. „offline“ analýzu.

Offline analýza

Slouží k nahrání logu ze zachytávání síťové komunikace pomocí tlačítka „Choose File“. Tato funkcionalita umožňuje graficky zobrazit statistiku pro data, viz následující sekce: overview, TCP.LOG, UDP.LOG. V rámci „offline“ jelikož jsou data načítána ze souboru, nedochází k žádné aktualizaci dat. Prezentované informace jsou konečné.

Overview

Slouží jako rozhodovací část, zda má být prováděna analýza na real-time data, nebo bude provedena ze souboru, který je nahrán uživatelem aplikace viz obr. 3.11. U obou případů dojde k zobrazení „síťových grafů“, které reprezentují unikátní komunikaci, která byla zaznamenána, viz příloha B.10. V případě real-time provozu se data aktualizují po každém obnovení stránky v situaci, kdy OS zaznamenal komunikaci, která není duplicitní. Duplicitní znamená, že komunikace musí mít rozdílnou buď zdrojovou IP, cílovou IP, zdrojový port, cílový port nebo u TCP spojení příznak pro paket. Možné příznaky pro TCP spojení jsou uvedeny v dokumentaci pro síťovou sondu v příloze C.

V příloze obr. B.10 zobrazuje příklad analýzy zaznamenané komunikace pro transportní protokol TCP, který je veden přes síťovou sondu. Pro TCP a UDP protokol jsou grafy téměř identické s jediným rozdílem v tom, že graf pro UDP statistiky má navíc parametr nazvaný „nespárováno“. Toto je situace, kdy se knihovně *psutils* nepodařilo získat cílovou IP adresu z paketu.

TCP.LOG a UDP.LOG

Funkcionalita modulu Logy, která slouží k zobrazení unikátních informací i síťové komunikaci. Záznam je vytvořen do souboru s příponou *.log*. Každý řádek představuje unikátní komunikaci v rámci OS. Samotný řádek nepopisuje kompletní komunikaci mezi dvěma stranami (viz výpis. 3.13), ale reprezentuje paket, který byl zaslán mezi komunikujícími uživateli. Každý záznam má předem definovaný formát ve tvaru:

Proto|src_IP|dst_IP|Status|PID|jméno programu|čas zahájení

Výpis 3.13: Výpis TCP.log záznamu.

```
tcp|192.168.1.104:38730|52.26.249.11:443|established|9941|
firefox|
```

U výpisu 3.14 si lze všimnout hodnoty „none“, to značí, že daný paket nemá příznaky pro jednotlivé datagramy jak TCP protokol. Další možnost je, že se objeví znak „?“ v samotném logu, v takovém případě se nepodařilo získat informace. Popis

významu jednotlivých parametrů je detailněji popsán v dokumentaci pro aplikaci. Hlavní rozdíl mezi TCP a UDP logováním je v tom, že TCP obsahuje parametry, které v UDP protokolu nejsou specifikované, např. příznak pro paket.

Výpis 3.14: Výpis UDP.log záznamu.

```
udp|192.168.1.104:46885|216.58.201.68:443|none|18244|
chrome|
```

Scappy přehled

Zobrazuje základní informace o protokolech, pro které byly vytvářeny síťové parsery, nebo byly využité parsery z existující knihovny *python_gen_61850*. V rámci aplikace dochází k dynamické analýze adresáře nazvaného *non_standard_protocols*. Pro každý adresář v *non_standard_protocols* dochází k přidání názvu adresáře do vizuálního rozhraní webové aplikace ve formě tabulky. Každý adresář představuje název pro jednu tabulku, která obsahuje 3 sloupce viz přiloha obr. B.11. Před samotnou tabulkou lze vidět informaci o velikosti adresáře pro logování. Sloupce, které tvoří jednotlivé tabulky jsou:

- POČET LOGŮ – Pro jednotlivé adresáře dochází k zjištění, kolik souborů obsahují. V případě výskytu dalšího adresáře na úrovni logovacích souborů se struktura vnořeného adresáře neanalyzuje a je počítán jako soubor.
- AKTUÁLNĚ ZACHYTÁVÁ – Informace o tom, jestli pro aktuální síťovou analýzu, která probíhá pomocí *scappy* knihovny byl detekován příslušný paket pro specifikované protokoly.
- VELIKOST SLOŽKY – Zjištění kolik místa na disku příslušné adresáře zabírají.

3.4.6 Síťové testy

Slouží pro ověření dostupnosti cílového zařízení v rámci sítě, nebo zjištění, zda zařízení, na kterém běží aplikace má přístup k externím sítím, jako je např. Internet. Aplikace poskytuje 2 typy síťového testu viz příloha obr. B.12.

Ping

Uživatel může zadat cíl, na který má být test proveden buď pomocí využití znalosti IP adresy nebo pomocí domény. Následně je nezbytné nastavit počet paketů, které mají být zaslané. V příkazovém řádku je to využití omezovacího přepínače ping „-c“. Při obdržení informací na výsledné pakety uživatel vidí výsledek testu. Prezentované parametry jsou: ztrátovost, maximální zpoždění, minimální zpoždění,

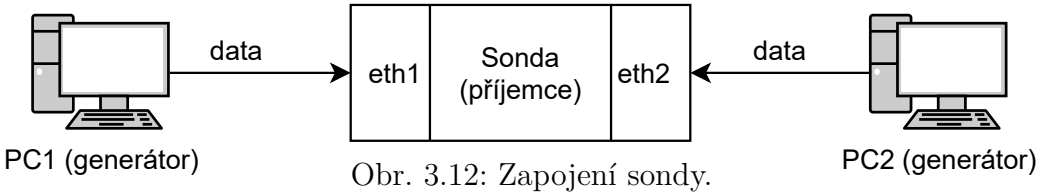
průměrné zpoždění. Výsledek testu je zobrazen na totožné stránce (testy), na které byl zadán požadavek k vykonání daného příkazu. Výsledné parametry jsou zobrazeny uživateli přímo pod polem, kde dochází k zadávání vstupních informací. Oproti klasickému vzhledu aplikace je tato zpráva zvýrazněna červeným blokem, který obsahuje výsledky testu.

Iperf3

Nástroj, který slouží k testování LAN (Local Area Network) a WLAN (wireless LAN) rychlostí a propustností sítě. Pracuje na principu generování provozu ze zařízení chovající se jako klient k zařízení, které je uvedeno jako server. Sonda byla využívána jako server při provedených testech. Jednotlivé výsledky testů lze zaznamenávat i do souborů, v případě správné konfigurace jednotlivých přepínačů. Sondu lze také nastavit jako klient, v situaci, kdy je to vyžadováno. Pomocí parametru *-c* se udává, že zařízení bude v režimu klient, parametr *-s* nastavuje zařízení do režimu server. Při využití přepínače *-s* dochází k vytvoření serveru, který naslouchá na určitém portu pro následující komunikaci ze strany klienta.

3.5 Testování zařízení

Kapitola popisuje 2 základní typy testu, které byly provedeny na síťové sondě. Testy jsou zaměřeny na využití protokolu UDP a TCP. Při tomto testování bylo využito standardu RFC 1242 a RFC 2544, které definují terminologii a metodologii pro dodržení spolehlivého testování síťových parametrů jako jsou např. propustnost, zpoždění či ztrátovost rámců. Samotné testování síťové sondy ovšem nebylo primárně zaměřeno na testování zmíněných parametrů. Hlavním cílem těchto testů bylo zjistit zátěžový limit daného HW, a to především zjištění vytížení CPU a RAM při provozu na síťových kartách [26]. Topologie pro testovací scénář je znázorněn na obr. 3.12. Schéma zapojení HW prvků bylo totožné pro všechny prováděné testy. Nebraly se v úvahu žádné ztráty na přenosových médiích, protože délka propojovacích kabelů byla v rámci desítek centimetrů. Eth1 síťová karta je na síťové sondě označována jako *enp2s0f0* a eth1 jako *enp1s0*.



Obr. 3.12: Zapojení sondy.

3.5.1 UDP testování

Testování probíhalo naráz ze 2 zařízení zároveň na cílovou síťovou sondu. Každé zařízení posílalo data pomocí UDP protokolu na síťové rozhraní sondy. První zařízení (PC1) s OS Ubuntu 20.04 a druhé zařízení (PC2) s Windows 10.

Testování CPU,RAM v maximální zátěži

S využitím nástroje hping3 na PC1 došlo k testování síťové sondy pomocí zahlcení síťového rozhraní UDP pakety. Pro nástroj hping3 bylo využito příkazu:

hping3 192.168.11.1 –udp -d 5000 –flood

Přepínač *-d* vyjadřuje velikost payloadu v Bajtech a přepínač *–flood* umožnil odesílat v průměru 94 840 paketů/s ze zařízení PC1 na síťové rozhraní eth1 (generátoru).

Na PC2 byl využit nástroj „WAN Killer“ od společnosti Solar Winds [1]. Důvodem využití rozdílného nástroje oproti PC1 je ten, že PC2 s OS Windows nepodporuje nástroj hping3. Dalším důvodem byla snaha využít rozdílných nástrojů pro generování provozu, za účelem eliminace případných nedostatků vybraného nástroje. Při zadání cílové IP adresy bylo potřeba v grafickém rozhraní zvolit:

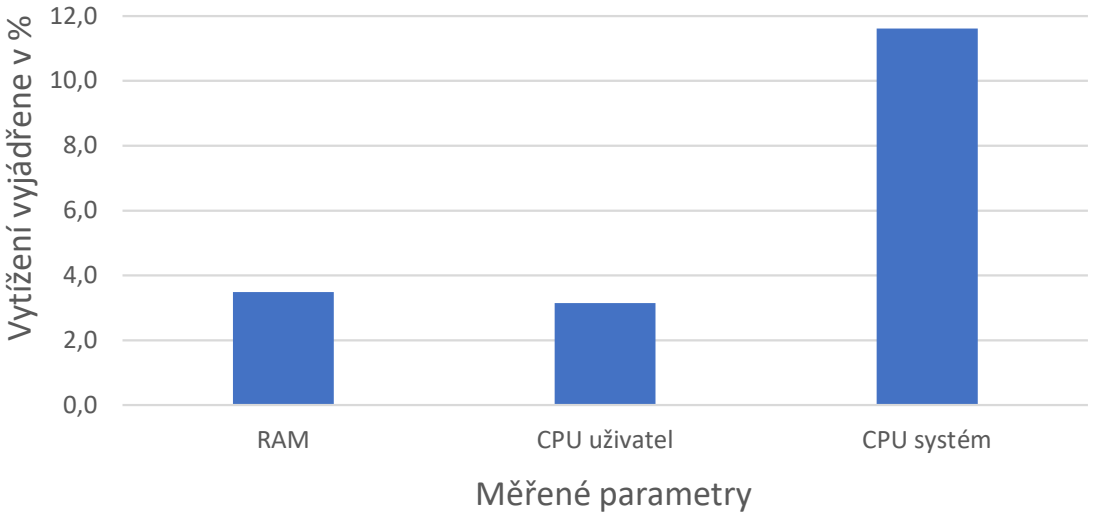
- Protokol – UDP
- Šířka pásma – nastavit na 125 MB/s, což odpovídá hodnotě 1000 Mb/s
- Velikost paketů – 15 872 B
- Počet paketů – 9 924 paketů/s

Výsledky testu jsou prezentovány v grafu na obr. 3.13. Test byl vykonáván po dobu 30 s. Po tuto dobu byly hodnoty zachytávané externími skripty na samotném OS síťové sondy. Pro nalezení maximální možné hodnoty paketů/s generované z PC2 s využitím síťové karty *Killer™ E2200 Game Networking LAN* na rozhraní eth2 síťové sondy byla provedena série testů, kde se postupně zvyšoval počet paketů/s na generovaném zařízení, do okamžiku, kdy program hlásil chybovou hlášku, že daný počet paketů/s při nastavené dané velikosti nedokáže vygenerovat.

Osa Y vyjadřuje průměr hodnot v procetench získaných za dobu, po kterou byl test prováděn. V grafu jsou uvedeny hodnoty pro:

- RAM - Popisuje kolik procent RAM kapacity je využito.
- CPU uživatel - Vyjadřuje, kolik procent z celkového zatížení CPU využívají procesy, běžící pod přihlášeným uživatelem.
- CPU systém - Vyjadřuje, kolik procent z celkového zatížení CPU využívají procesy, které jsou označené jako systémové.

[1] https://www.solarwinds.com/engineers-toolset/use-cases/traffic-generator-wan-killer



Obr. 3.13: UDP flood prezentace výsledků.

Při využití obou testů zároveň pro test síťové sondy bylo dosaženo výsledku viz graf na obr. 3.13. Graf popisuje, že i když obě síťové karty (eth1 a eth2) jsou zatěžovány maximálním možným provozem, dochází k minimálnímu vytížení systémových prostředků.

Následný test s využitím nástroje iperf3 a protokolem UDP bylo testováno, jakého rozdílu zpoždění (jitter) a ztrátovosti paketů bude dosaženo při maximální zátěži síťových karet. Síťová sonda byla konfigurována jako server pomocí příkazu:

```
iperf3 -s
```

Klientskou stanici zde reprezentuje PC1 s využitím příkazu:

```
iperf3 -u -c dst_IP
```

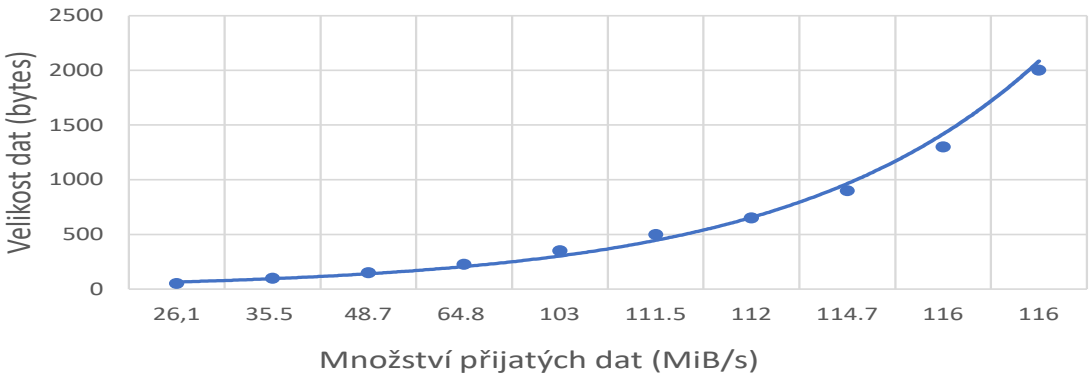
Vzhledem k omezení síťových karet PC1 a PC2 na 1 000 Mb/s, nebylo možné vygenerovat větší provoz jak 116,1 MB/s z jednoho zařízení. Výsledná ztrátovost z testu byla 0% a rozdíl zpoždění 0,084 ms ze strany příjemce (serveru).

Výsledek testu jednoznačně ukazuje, že UDP provoz nevytěžuje systémové prvky významným způsobem při provedených testech. Pro větší vytížení HW prostředků, by muselo dojít buď k zapnutí např. systémových služeb, nebo přidání mnoha NIDS pravidel, které by byly aplikovány na každý UDP paket.

Testování síťových karet sondy

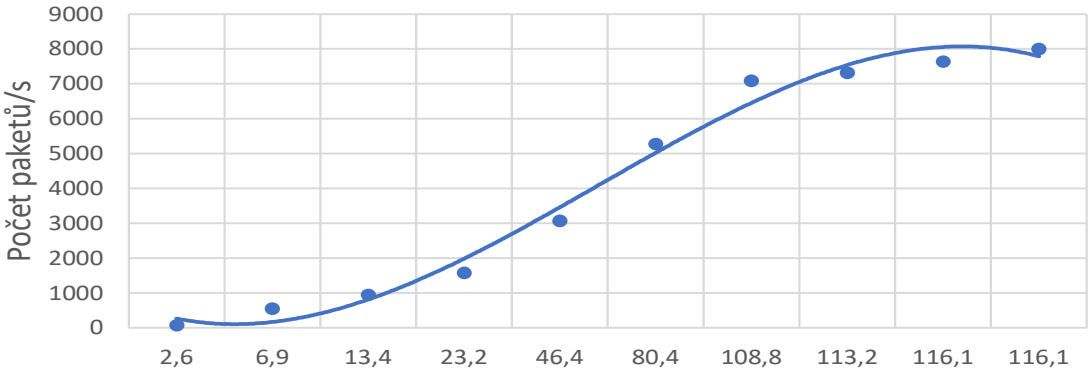
V rámci testování síťové sondy došlo k ověření, jakým způsobem budou síťové karty reagovat na příchozí provoz UDP paketů. Na obr. 3.14. a 3.15 jsou prezentovány výsledky nástroje hping3 a WAN Killer pro generování UDP paketů na síťové rozhraní sondy.

Na PC1 s využitím nástroje hping3 a využitím přepínače *-d*, který nastavuje velikost paketů došlo, k modifikaci velikosti paketů. V testu, ve kterém byl využít nástroj hping3 (viz obr. 3.14) bylo generování paketů zahájeno na velikosti 50 B. Následně tato hodnota byla zvětšována až do doby, než bylo dosaženo situace, kdy modifikace velikosti neměla vliv na příchozí provoz na sondě. Následné navyšování velikosti pro payload nebylo konstantní, byly využité kroky zvětšení o 75/150/300/400/700 B. Maximální možná velikost pro daný paket dosáhla 2000 B. Při zvyšování velikosti nad tuto hodnotu již nedocházelo k obdržení většího množství dat na sondě. Rychlost zasílání paketů byla v případě různých velikostí jednotlivých paketů různá. Při velikosti 50 B bylo vytvářeno průměrně 305 000 paketů/s. Na konci testu, při generování paketů o velikosti 2 000 B počet průměrně přijatých paketů byl 116 000/s.



Obr. 3.14: Hping3 test.

Druhý test prováděný z PC2, který byl prováděny přes eth2 síťovou kartu využíval program „WAN Killer“ pro generování UDP paketů. Test využíval principu generování velkého UDP paketu a následnou manipulaci s počtem zaslaných paketů za vteřinu. Velikost paketu byla nastavena na 15 875 B pro celý průběh testu a rychlost generování paketů postupně zvyšována až nad hodnotu 8 000 paketů/s.



Obr. 3.15: „WAN Killer“ test.

Při porovnání výsledků z obr. 3.14 a 3.15, si lze všimnout, že u programu „WAN Killer“ došlo dosažení k velmi podobné hodnotě přijatého provozu jako u nástroje

hping3, hodnota se lišila pouze o 0,1 MB/s. V případě programu „WAN Killer“ byla šířka pásma využita na 100% při generování přibližně 8 000 UDP paketů o velikosti 15 875 B. Při následném navyšování počtu paketů/s nad 8 000 docházelo k zahlcení HW generátoru, tedy PC2.

Z tohoto výsledku lze konstatovat, že obě síťové karty zvládají 116 MB/s provoz pro protokol UDP. Chybějících 9 MB/s pro dosažení hodnoty 125 MB/s nebylo dosaženo ani v datovém toku směrem od sondy k generátoru provozu, lze tedy usuzovat, že se jedná o hlavičky/výrobní nedostatek nebo SW limit sondy. Od sondy ke generátoru provozu (PC1) byl zaznamenán síťový provoz o nejvyšší hodnotě 500 B/s.

Tabulka 3.5 zobrazuje přehledně nastavení parametrů pro jednotlivé testy. Sloupce CPU a RAM zobrazují vytížení HW prostředků pro situaci, kdy oba testy založené na protokolu UDP, byly spuštěny a síťová sonda byla testována. U sloupce počet paketů/s znak „»“ znamená, že na začátku testu se začínalo na generování množství X a končilo na Y paketů/s. Stejný význam má značka i ve sloupci Velikost paketu v B (mění se pouze velikost, ne rychlost).

Tab. 3.5: Tabulka parametrů pro testy.

Rozhraní	Nástroj	Velikost paketu v B	Počet paketů/s	Max. rychlost MB/s	CPU	RAM
eth1	*hping3*	50 >>2 000	305 000 >>116 000	116	16 %	3,7 %
eth2	*WLAN Killer*	15 875	50 >>9 000	116,1		

3.5.2 TCP měnění velikosti okna

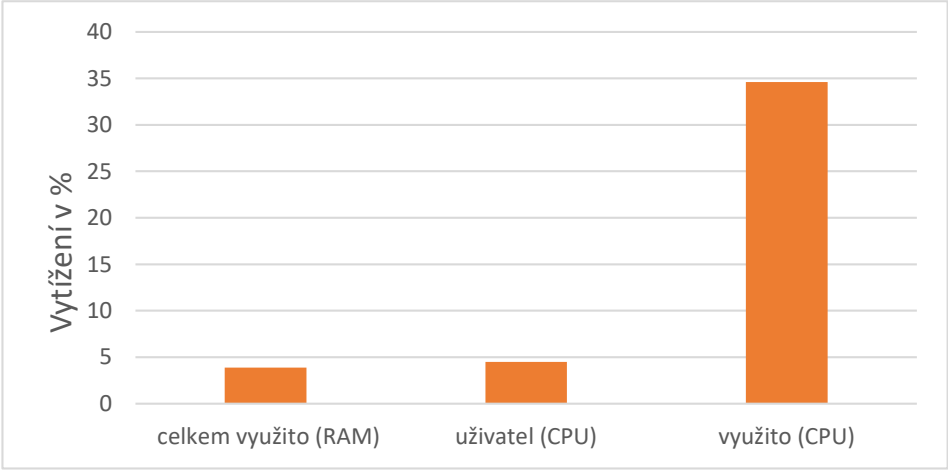
Test, který je založen na protokolu TCP. S využitím nástroje ipref3 (viz kapitola 3.4.6) je generován TCP provoz, ve kterém došlo k modifikaci tzv. „tcp okna“, které slouží k navýšení velikosti přenášených dat až na velikost 65 535 B. Pro nastavení lze využít technik jako je „klouzavé okno“, pro dynamické navyšování této hodnoty a optimalizaci TCP protokolu.

Pro generování provozu na obou zařízeních byl využit příkaz:

ipref3 -c IP_addr_rozhraní -b 1000 M -w 65535 -M 150 -n6000 M

- **-b** – Nastavení šířky pásma na 1000 Mb/s.
- **-w** – Nastavení velikosti TCP okna na 65 535 B.
- **-M** – Přenastavení TCP segmentu na specifickou velikost. Výchozí velikost TCP protokolu pro ethernet je 1 460 B. Častější segmentace způsobí vyšší vytížení CPU. Hodnota -M 150 nastavuje TCP segment na velikost 150 B.
- **-n** – specifikace kolik vyrovnávacích pamětí „bufferů“ má být přeneseno před tím, než je test ukončen. Přepisuje výchozí časový parametr (10 vteřin) pro ukončení testu. Nastavuje ukončení testu po přenesení dat o velikosti 6 000 MB.

Na obr. 3.16 lze vidět, že oproti testování pomocí UDP protokolu (viz obr. 3.13) je výrazně vyšší celkové vytížení CPU s využitím maximálního možného výkonu generátorů.



Obr. 3.16: TCP flood prezentace výsledků.

Oproti testu UDP flood, zde nebylo možné generovat provoz 230 MB/s ale pouze provoz směrem k sondě přibližně 180 MB/s. Samotný TCP protokol, vrací potvrzovací informace pro jednotlivé pakety (sekvenční číslo). Z toho důsledku zařízení posílalo zpětně informace jednotlivým zdrojům komunikace. Zpětná komunikace ze sondy na jednotlivé generátory provozu dosahovala velikosti 20 MB/s. Rozdíl 30 MB/s, které zde nejsou dosaženy může být způsoben vyšší náročností generování TCP paketů pro jedno z méně výkonných zařízení.

3.5.3 Testování HW prostředků při využívání Surikaty

Test využívá princip zahlcení zařízení pomocí TCP paketů viz kapitola 3.5.2. Do souboru detekčních pravidel bylo přidáno pravidlo viz výpis 3.15, které slouží pro detekci TCP provozu.

Výpis 3.15: Pravidlo využité pro testování.

```
alert tcp any any -> any any (msg:"SURICATA_TCP";
sid:46666;)
```

Zátěž byla generována jak s PC1, tak PC2 (obr. 3.12.). Samotný test lze rozdělit do několika fází, kde každá fáze trvá po dobu 60 vteřin a následné se přidává další zátěž pro sondu.

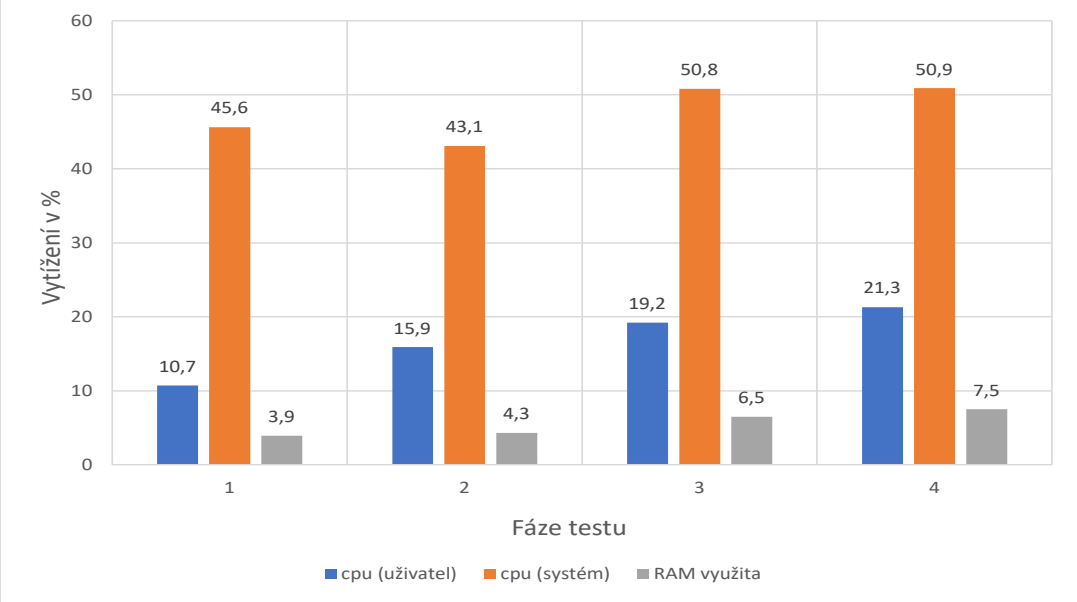
- Počáteční fáze, kde dochází generování maximálního provozu z jednotlivých PC pro TCP protokol. Na cílovém zařízení je zapnuta pouze Surikata s aplikovaným pravidlem pro detekci TCP protokolu. Dále také Python web server z důvodu, aby bylo možné mít spuštěnou službu v */etc/systemd/system*, která se stará o chod webové aplikace s názvem *vut_sonda.service*.

- Druhá fáze spočívá v zapnutí internetového prohlížeče, což simuluje situaci, kdy by sonda byla využívaná a uživatel potřeboval manipulovat se sondou přes webové rozhraní.
- V třetí fázi dojde k zachytávání pcap souboru na síťovém rozhraní.
- Čtvrtá fáze aplikuje funkcionalitu, sloužící pro analyzování TCP/UDP provozu. Je spuštěn externí skript, který běží na samostatném vláknu (dojde k zmenšení výkonu CPU). Zároveň byla data o provozu zapisována do souboru v intervalu 2 vteřin.

Výsledek testu je reprezentován jako graf s jednotlivými fázemi. Jak již bylo zmíněno, po každých 60 vteřinách je přidána další fáze testu, až do doby, než sonda bude využívat maximální možnou funkcionalitu, kterou nabízí. Hodnoty pro CPU, RAM a síťové rozhraní uvedené v grafu byly vypočítány jako aritmetický průměr z celého záznamu, který byl pořizován v intervalu 1 s. Výsledná hodnota v každé fázi testu je vytvořena z 60 testovacích hodnot. Výsledky jsou reprezentovány viz obr. 3.17 a 3.18.

Na obr. 3.17. lze vidět výsledky z testování pro CPU a RAM. Graf je rozdělen do 4 části, podle principu testovacího scénáře. Hodnoty byly získané za pomocí využití shell skriptů, které byly na cílovém zařízení spuštěny pro každou fázi testu.

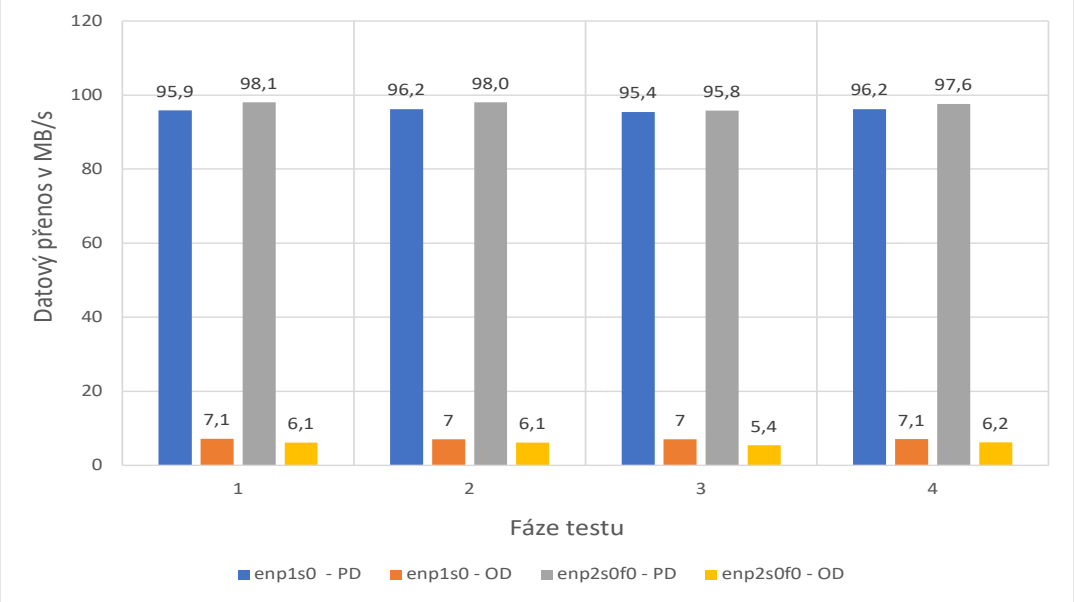
Při porovnání CPU z obr. 3.17. pro jednotlivé fáze lze vidět, že uživatel postupně spotřebovává více procesorového výkonu v závislosti na zapínaní/využívání funkcionalit sondy. Výkon CPU, který uživatel využívá, se z hodnoty 10,7 % dostal více než na dvojnásobek v poslední fázi testu. Při porovnání využití výkonu CPU mezi uživatelskými procesy a systémovými lze vidět zásadní rozdíl. Každá zátěž vzniklá přes webové rozhraní je vázána na uživatelské procesy.



Obr. 3.17: Zatížení CPU a RAM při jednotlivých fázích testů.

Samotná RAM v rámci testování nebyla výrazně vytěžována. Procesy, které byly vytvářené neměli zásadní rozdíl na alokování RAM paměti. V případě, že by v Surikatě bylo aplikovaných více detekčních pravidel, mohlo docházet k většímu alokování jednotlivých výpočetních prostředků z důvodu procházení každého příchozího paketu na větší množství detekčních pravidel. Jedna z dalších možností, jak by mohlo dojít k alokování většího množství výpočetního výkonu pro jednotlivé procesy by bylo v situaci, kdy uživatel zapne logování pro více protokolů zároveň. Např. logovat protokoly: HTTP, DNS, FTP, SSH.

Při testování výkonu zařízení došlo zároveň k zaznamenávání provozu na obou síťových rozhraní zařízení. Výsledky jsou prezentovány na obr. 3.18. Zde jsou reprezentovány datové přenosy pro každou fázi testu a pro obě síťové karty. Zkratka PD (přijaté data) slouží pro informování uživatele o tom, kolik MB/s na daném rozhraní síťová sonda přijímala. Druhá zkratka OD (odeslaná data) vyjadřuje informaci jakou rychlostí byly na TCP pakety zasílány odpovědi, vyjádřena v MB/s. Lze konstatovat, že každé síťové rozhraní v jednotlivých fázích testu přijímalo v průměru stejný síťový provoz v MB/s, až na malé rozdíly, které mohly být způsobeny tím, že byly využité dva různé HW s dvěma odlišnými SW generátory provozu.



Obr. 3.18: Zatížení síťových rozhraní při jednotlivých fázích testů.

3.5.4 Shrnutí testů

Při testování jednotlivých síťových rozhraní eth1 a eth2 bylo zjištěno, že nezáleží jaké zařízení nebo nástroj generuje daný provoz. Obě síťové rozhraní jsou určené pro 1 Gb zátěž a zvládnou téměř totožný síťový provoz o velikosti 116 MB /s.

V rámci testování pomocí UDP a TCP došlo ke zjištění, že při fragmentaci TCP paketů je síťová sonda více vytížená, než u zpracování UDP datagramů. Při sledo-

vání CPU došlo k vytížení většímu až o trojnásobek u zpracovávání TCP provozu (vytížení bylo 35%). Datový provoz poklesl téměř o 50 MB/s vůči UDP protokolu, kde bylo dosaženo zpracovávání dat o velikosti 230 MB/s. Při 2 Gb/s provozu se zapnutím jednotlivých funkcionalit síťové sondy (webové rozhraní, zachytávaní pcapů, analýzu provozu, zapnutí NIDS systému) bylo vytížení CPU maximálně na 50% výkonu a RAM paměť dosáhla hodnoty 20% vytížení. Jednotlivé testy ukazují, že se jedná o lineární nárůst spotřebovaného výkonu vzhledem k přidávání zátěže na síťovou sondu.

Z testu lze vyvodit závěr, že síťová sonda je schopná efektivně zpracovávat objem dat větší než 1 Gb/s. V případě připojení externí síťové karty např. do USB-C portu, který podporuje vyšší datové přenosy, by bylo teoreticky možné zpracovávat i datový provoz okolo 2 až 2,5 Gb/s.

3.6 Návrh krycího obalu pro síťovou sondu

Na sondu musí být vytvořena vnější ochrana (obal), který bude splňovat minimálně podmínky standardu s označením IP54, protože zařízení může být umístěno do míst s nepříznivými podmínkami. Pro zachování správné funkčnosti elektronického zařízení a také ochraně před nepříznivými vlivy jako je působení prachu, vody a proniknutí nežádoucích těles, je potřeba správně navrhnout ochranu. Standard IP udává odolnost elektrického zařízení pro vniknutí cizích těles a také proti vniknutí kapalin. Zmíněná ochrana je tvořená pomocí označení „IP“, za kterým následují dvě číslice. První číslice popisuje ochranu před vniknutím cizích předmětů viz tabulka 3.6 a nebezpečným dotykem. Druhá číslice vyjadřuje stupeň ochrany před vniknutím vody viz tab. 3.7.

Tab. 3.6: Tabulka stupně ochrany proti nebezpečnému dotyku.

Stupeň	**Nebezpečným dotykem**	**Vniknutím cizích předmětů**
IP 0x	bez ochrany	bez ochrany
IP 1x	dlaní (>5x5 cm)	velkých
IP 2x	prstem (>12,5x12,5 mm)	malých
IP 3x	nástrojem (>2,5 mm)	drobných
IP 4x	nástrojem, drátem (>1 mm)	velmi drobných
IP 5x	jakoukoliv pomůckou	prachu částečně
IP 6x	jakoukoliv pomůckou	prachu úplně

Tab. 3.7: Tabulka stupně ochrany proti vniknutí vody.

Stupeň	**Vniknutím vody**
IP x0	bez ochrany
IP x1	Chráněno proti kapající vodě 1+0,5 mm za minutu.
IP x2	Chráněno proti kapající vodě 3+0,5 mm za minutu (náklon max 15°).
IP x3	Chráněno proti vodní tříšti (v úhlu 60° vertikálně).
IP x4	Chráněno proti stříkající vodě (všechny úhly).
IP x5	Chráněno proti tryskající vodě.
IP x6	Chráněno proti intenzivně tryskající vodě.
IP x7	Chráněno proti ponoření do vody (30 min, max 1 m).
IP x8	Chráněno proti potopení do vody (neomezený čas).
IP x9	Chráněno proti tryskající vysokotlaké teplé vodě.

Parametry zohledněné při výběru

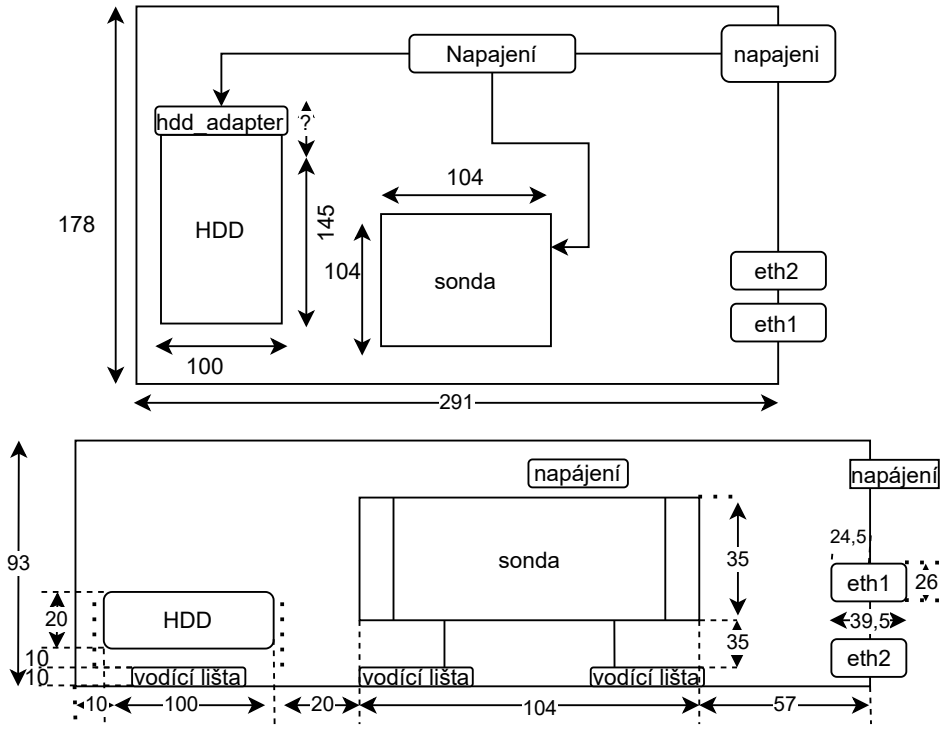
Při vybírání ochrany záleželo na následujících parametrech. Stupeň IP ochrany, materiál obalu, jenž může být zhotoven buď z plastu, nebo kovu. Co se týče průmyslových krytů, jsou především zhotoveny z hliníku. V dnešní době jsou ovšem vyráběny i kryty z plastu, které splňují i standardy IP67 a více. Dále byla zohledněna pořizovací cena krytu, aby odpovídala hodnotě zařízení, které chceme chránit a zároveň je potřeba, aby za tuto cenu bylo dosaženo požadované ochrany. K samotnému přístupu k zařízení, mohou být využity 2 základní způsoby. První způsob je realizován, pomocí šroubování dvou kusů krytu do sebe, toto zajišťuje spolehlivost v těsnění a pevnosti. Nevýhoda může být v době, kterou trvá rozmontovat kryt a následně je až umožněn přístup k elektronice. Druhý způsob využívá principu jednoduchého otevírání na principu kufru. Jsou zde 2 protikusy, které do sebe po zatlačení zapadnou a je tak docíleno uzavření krytu. Velikost úložného prostoru, který je důležitý pro instalaci jednotlivých HW komponent do prostoru krytu.

Výběr krytu

Prvotní krok nutný, pro výběr správné velikosti krytu bylo zjištění kolik vnitřní plochy bude vyplněno při instalaci HDD a sondy. Po následném zjištění velikosti jednotlivých parametrů byl vybrán ochranný kufr, který disponuje vnitřními rozměry 291x 178x 93. Samotný návrh, jak jednotlivé prvky v sondě budou umístěny a popis jejich rozměrů je viz obr. 3.19.

Po zvážení parametrů jako jsou cena, IP ochrana, rozměry kufru vyšel jako nej-

lepší kandidát ochranný kufr značky NANUK model 909[2]. Obr. 3.19 v horní části ukazuje, jak budou HW prvky umístěny a ve spodní části lze vidět půdorys (pohled zepředu), z důvodu zjištění, jaká výška bude dosažena, při implementaci HW.



Obr. 3.19: Návrh krytu.

Veškerý HW, který má být umístěn bude přidělán na nevodivé lišty, umístěné do krytu. Samotné ethernetové konektory byly vybrány tak, aby měly co nejmenší přesah přes stěnu kvůli využitelnému prostoru.

Na obr. 3.20 je ukázáno, jak síťová sonda vypadá uvnitř krycího obalu, který byl navržen a následné sestrojen. Teplotní rozsah, pro který může být kryt využit v prostředí je -29 °C až 60 °C. Tyto teploty jsou uvedeny výrobcem jako maximální teplotní rozsah, v kterém lze kryt využít. Na obr. 3.21 lze vidět ochranný kryt z vnějšího pohledu.

[2]https://www.profikon-trezory.cz/odolny-kufr-model-909-cerny-p1686-75

Obr. 3.20: Pohled na uložení HW v bezpečnostním obalu.

Obr. 3.21: Pohled na bezpečnostní obal z hora.

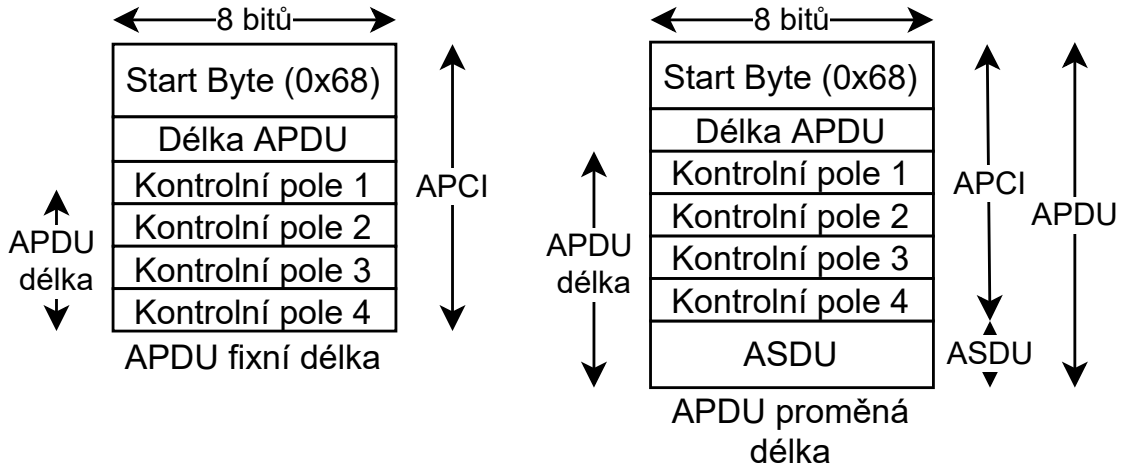
4 Parsery pro síťové protokoly

S využitím programovacího jazyka Python a knihovny *Scappy* došlo k vytváření parserů pro síťové protokoly jako jsou např. DLMS (Device Language Message Specification) a IEC 60870-5-104. Princip vytváření jednotlivých parserů je postaven na práci s daty, které paket obsahuje. S využitím programu Wireshark v kombinaci s dokumentací pro daný protokol lze postupně dohledat, jakým způsobem je L7 hlavička strukturovaná.

Došlo k rozšíření knihovny s názvem *python_gen_61850*, která je implementována do síťové sondy. Z této knihovny jsou využité zdrojové kódy pro detekci GOOSE a SV (Sample Value) protokolů. Knihovna byla implementována do vytvořené webové aplikace a došlo k vytvoření logovacího systému pro specifikované data v jednotlivých protokolech.

4.1 IEC 60870-5-104

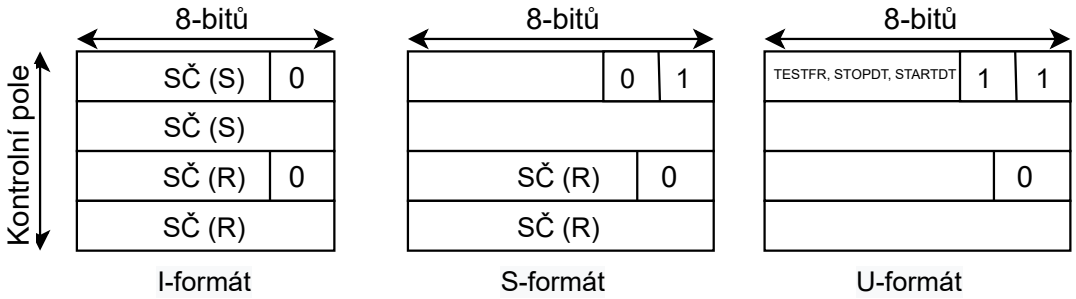
Standard pro dálkové ovládání zařízení a systémů v sítích založených na TCP/IP komunikaci. Definuje strukturu, v jaké mohou být informace protokolu přenášeny. V samotné struktuře protokolu nás zajímá zkratka APCI (Informace o řízení aplikačního protokolu). Každý APCI začíná Bajtem 0x68, který následuje 8-bitů dlouhý APDU (Aplikační datová jednotka protokolu) a čtyřmi 8-bitovými kontrolními poly, ve kterých jsou prezentovány informace o sekvenčních číslech viz obr 4.1.



Obr. 4.1: IEC struktura protokolu. Převzato z [27].

Na obr 4.1 lze vidět, že paket může mít fixní nebo proměnou délku, kde je navíc pole ASDU, které obsahuje data aplikační služby. Formát rámce je určen posledními dvěma bity prvního kontrolního pole (CF1). Standard definuje základní 3 typy formátů, viz obr. 4.2.

- I-formát: Vždy obsahuje ASDU. Kontrolní pole indikují směr zpráv. Prezentovány jsou zde dvě 15-bitové sekvenční čísla, kterým je postupně zvedaná hodnota o 1 pro každý APDU v každém směru.
- U-formát: Využíván k vykonávání dohlížecích funkcí. Vždy má konstantní délku. Vždy je složeno pouze z jednoho APCI.
- S-formát: Provádí se k nečíslovaným řídicím funkcím. Vždy má pevnou délku. Skládá se pouze z jednoho APCI. Lze zaslat pouze jeden z těchto rámců: TESTFR (Test Frame), STOPDT (Stop Data Transfer), STARTDT (Start Data Transfer) v daný okamžik.



Obr. 4.2: Formáty paketu.

Na obr. 4.2 lze vidět 3 strukturu jednotlivých paketů. SČ vyjadřuje zkratku pro sekvenční číslo. Hodnota R popisuje stranu (reciever), S udává sekvenční číslo od odesílatele.

V případě, že je ASDU část přítomná, jsou extrahovány data s názvy: TypeID, SQ, CauseTX, QA, Addr. Tyto informace vypovídají detailněji o jednotlivém objektu a lze určit, z jakého důvodu došlo k zaslání zprávy [27]. Tyto informace budou reprezentovány u logů s rámcem typu I.

Na základě dovednosti rozpoznat protokol IEC 60870-5-104. Lze využít knihovnu *Scappy* pro následně logování provozu. Samotný systém (kernel) tento síťový provoz nerozpozná, jelikož to není běžný protokol a parser není implementovaný v OS. Možnost, jak zjistit, že protokol je využíván na daném zařízení, by bylo za pomocí zachytávání provozu do *.pcap* souboru. Následné implementovat do Wiresharku rozšíření, které by parsovalo data.

Vytvořený parser je schopný logovat provoz v následujícím formátu, kde první řádek (záznam) slouží o popisu informací, pro jednotlivé data oddělené znakem „|“. Formát zápisu je:

proto| src_IP:src_port| dst_IP:dst_port| typ_ramce

Ukázka záznamu logovacího souboru pro protokol IEC 60870-5-104 je zobrazen ve výpisu 4.1.

Výpis 4.1: Ukázka z logů pro protokol 104.

```
IEC-104| 192.168.11.248:2404|192.168.11.111:56185 |
U-frame
```

4.2 DLMS

Protokol určen pro předkládání objektů do formátu zprávy, která je následně odeslána. Zabezpečení komunikace mezi klientem a serverem. Představuje také způsob, jak se připojit k zařízení „MeterCom Pro“, lze zjišťovat informace o aktuálním stavu zařízení, vyčítat informace o dodávkách elektrické energie, nebo upgrade firmwarů. Patří do normy IEC 62056, kde patří mezinárodní standardy a specifikace pro DLMS/COSEM (Companion Specification for Energy Metering). COSEM obsahuje specifikace, které definují aplikační vrstvy protokolu DLMS [28]. Došlo ke zjištění, že DLMS protokol využívá 2 typy formátu pro přenos zprávy. První typ je určen pro TCP/IP sítě, druhý je využit pro přenos dat u sériových linek. Typy těchto přenosů jsou:

1. Wrapper: Určen pro komunikaci v TCP/IP síti. Výhoda využití je ve velikosti zprávy. Stejná zpráva v Wrapperu má menší délku než totožná zpráva přenášena pomocí HDLC. Nezajišťuje integritu. Na obr. 4.3 lze vidět složení paketu pro Wrapper.

Obr. 4.3: Formát paketu přenášen pomocí Wrapperu.

- verze: Označuje verzi wrapperu, nejčastěji využívána verze 0x01.
- zdr_addr: Zdrojová adresa.
- cil_addr: Cílová adresa.
- délka dat kolik B dat bude přenášeno.
- data: samotná data přenášena Wrapperem.

Ukázka z DLMS logu je prezentována ve výpisu 4.2. Logovací soubor je vytvářen podle následující předlohy:

DLMS:Wrapper| zdr_adr:sport| cil_adr:dport| srcAddr| dstAddr| Délka dat| Request type|

Parametry srcAddr a dstAddr jsou adresy jednotlivých objektů, které spolu komunikují. Request type identifikuje, jaký požadavek byl poslán v rámci dané zprávy.

Výpis 4.2: Ukázka z logů pro protokol DLMS (Wrapper).

```
Wrapper| 10.0.0.140:63115| 10.0.0.12:4061| 16|
17404| 13| GetRequest|
```

2. HDLC: Využívá se u sériových linek. Zajišťuje integritu. Samotný paket je složen z jednotlivých polí viz obr. 4.4.
   - flag: Bajt, který je specifikován pomocí hexadecimální hodnoty 7E, jak na začátku rámce tak i na samotném konci. Pomocí tohoto návěstí lze určit, že se jedná o DLMS protokol přenášený pomocí HDLC.
   - frame format: Typ formátu + informace o velikosti dat.
   - cil_adr: Cílová adresa.
   - zdr_adr: Zdrojová adresa.
   - control: Pomáhá k zajištění pořadí zpráv.
   - hcs: Kontrolní součet pro HDLC.
   - data: APDU.
   - fcs: Kontrolní součet celého rámce.
   - flag: Označuje konec zprávy, využita hodnota 7E stejně jak pro začátek rámce.

flag	frame format	cil_adr	zdr_adr	control	hcs	data	fcs	flag
1B	2B	1B	1B	1B	2B	nB	1B	1B

Obr. 4.4: Formát paketu přenášen pomocí HDLC

Způsob jakým je vytvářen logovací soubor pro DLMS přenášené přes HDLC je pomocí následující šablony:

DLMS:HDLC| src_IP:sport| dst_IP:dport| srcAddr| dstAddr| dstLSAP| srcLSAP| LLC Quality| Request Type|

Ukázka zaznamenaného logu pro HDLC je ve výpisu 4.3.

Výpis 4.3: Ukázka z logů pro protokol DLMS (HDLC).

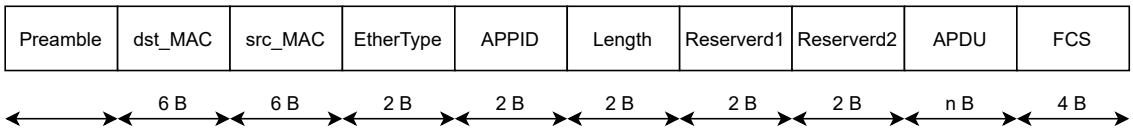
```
DLMS:HDLC over TCP| 10.0.0.11:4060| 10.0.0.140:63114|
 0| 33| 52| 226| 60| GetResponse|
```

4.3 GOOSE

Objektově orientovány protokol implementující časově kritický přenos událostí, jako je ochrana elektrického zařízení mezi zařízeními IEC 61850. Norma IEC 61850 definuje dvě skupiny komunikačních služeb. Klient-server a Peer-to-peer, kde druhý typ komunikace slouží k tzv „Generic Substation Event Services („GSE“). GSE jsou přidružené k časově kritické činnosti, jako je rychlá a spolehlivá komunikace. Jedna ze zpráv jsou zprávy GOOSE, které slouží k vysílání zprávy přes LAN síť.

Formát zprávy

GOOSE zpráva je spojena s třemi vrstvami ISO/OSI modelu, tyto vrstvy jsou: fyzická, linková a aplikační vrstva. Na úrovni linkové vrstvy je GOOSE protokol zabalen v 802.3 ethernet rámci viz obr. 4.5.



Obr. 4.5: Goose formát. Převzato z [29].

- dst_MAC – Multicástová cílová MAC adresa pro GOOSE.
- src_MAC – Unicastová zdrojová MAC adresa, která identifikuje odesílající zařízení.
- EtherType – Indikuje GOOSE protokol (0x88b8).
- APPID (2 bytes) – Nazyvané taky jako aplikační ID, které identifikuje aplikaci, která obdrží zprávu.
- Length – Délka GOOSE zprávy.
- Reserved1 – Rezervované pole 1.
- Reserved2 – Rezervované pole 2.
- APDU – APDU, které obsahuje detailnější informace o zprávě. Detailní popis APDU lze nalézt v [29]. Startovací byte je hodnota 0x61 (1 B). Celková délka je zjištěna od hodnoty 0x61 + Length .
- FCS: – Chybový detekční kód.

Více informací k protokolu a formátu APDU lze nalézt v [29]. Ukázku z logovacího souboru je prezentována ve výpisu 4.4. Formát zprávy je tvořen podle následujících dat:

GOOSE » | src_MAC| dst_MAC| APPID| Length| gocbRef| timeAllowedToLive| datSet| goId| t| stNum| sqNum| testNum| confRevNum| ndsCom| numDataSetEntries| allDataLength| Data1|

Výpis 4.4: Ukázka z logů pro protokol GOOSE.

```
00:21:c1:53:29:ad| 01:0c:cd:0 1:00:03| 4| 189| b'AA1J1Q01
A4LD0/LLN0GOCurrent'| 11000| b'AA1J1Q01A4LD0/LLN0CMMXU1'|
b'AA1J1Q01A2LD0/LLN0.Current'| (1603723438, 3401740580)|
1|973| 0| 100| 0| 9| 57|34565|
```

4.4 Sampled Values

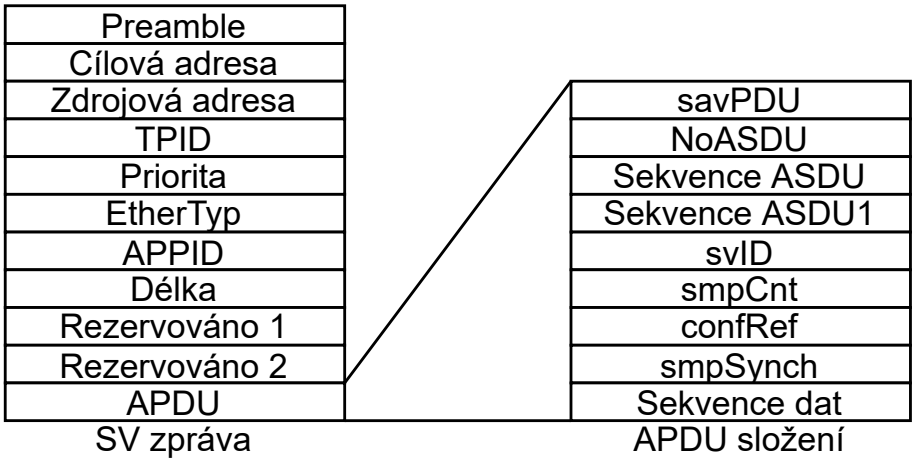
Standard IEC 61850 definuje protokol zvaný Sampled Values (SV). Komunikace protokolu je založena na typu komunikace pomocí dvou stran. První strana je zveřejňovatel (publisher) a druhá odběratel (subscriber). Protokol je především používán pro výměnu informací mezi jednotkami nazvanými Merging Units a IEDs (Inteligentní elektronické zařízení) nad Ethernet vrstvou.

Princip komunikace je založen na tom, že zveřejnovatel pravidelně zasílá zprávy v přesně daných časových intervalech. Časový interval závisí dvou faktorech. První je frekvence signálu a druhý je vzorek na periodu (Samples Per Periode). Např, když frekvence signálu je 50 Hz a SPP má hodnotu 80 tak časový interval bude 1/50/80. Všechny zprávy jsou publikovány pod určitým tématem (topic). Odběratel obdrží všechny zprávy, ale filtruje a parsuje pouze ty, které jsou pro něj adekvátní podle určitého tématu [30]. Pole, která jsou využitá pro logování jsou:

- Cílová adresa – Cílová MAC adresa.
- Zdrojová adresa – Zdrojová MAC adresa.
- APPID – ID aplikace.
- Délka – Délka zprávy.
- svID – SV identifikátor, uživatelem jedinečný identifikátor řetězce použitý pro předplatné.
- smpCnt – Index SV zprávy.
- confRev – Konfigurační revize.
- smpSynch – Definuje synchronizační mechanismus hodin použitých pro poslání SV zprávy. Může nabývat hodnot:
  - 0 – Žádná
  - 1 – Lokální
  - 2 – Vzdálená
- Sekvence dat – Posloupnost naměřených hodnot napětí a proudu, jak je uvedeno v kódování sekvence dat.

Pro aplikování parserů nebylo do detailu protokol zkoumán, došlo pouze k analýze, jaký má zpráva formát a vybrání určitých části pro vytváření *.log* souborů. Na

obr. 4.6 lze vidět architekturu SV zprávy. Pro parser byla využita jak SV zpráva, tak i údaje z APDU části.



Obr. 4.6: Formát SV zprávy. Převzato z [30].

SV protokol má následující formát pro logovací soubor:

IEC61850 Sampled Values » | src_MAC| dst_MAC| APPID| Length| noASDU| seqASDU| svId| smpCnt|confRev| smpSynch| PshMeas1Length|

Výpis 4.5: Ukázka z logů pro protokol SV.

```
Tue May 11 22:06:44 SampledValue | 00:21:c1:53:29:ef| 01:0
c:cd:04:00:02| 16384| 112| 1| 97| b'AA1J1Q01MU0103'| b'AA1
J1Q01MU0103'| 1630| 100| 0| 64|
```

Závěr

Cílem diplomové práce byla analýza jednodeskových počítačů. Návrh síťové sondy pro analýzu síťového provozu, realizace filtrovacího a detekčního systému. Vývoj a implementace ovládacího konfiguračního rozhraní přes internetový prohlížeč pro konfiguraci zařízení.

V úvodu práce je popsán OS Linux, který byl zvolen jako vhodný pro implementaci následného filtrovacího mechanismu a detekčního NIDS. Dále byl shrnut princip průchodů paketů skrze jádro Linuxu. Následně jsou rozebírány filtrační mechanismy IPtables a Nftables, které umožňují zamezit provoz pomocí pravidel pro jednotlivá vstupní/výstupní data. Dále byly analyzované jednodeskové počítače na základě vybraných HW parametrů. Následně došlo k porovnání jednotlivých zařízení, aby bylo možné rozhodnout, které z těchto kandidátů bude nejvhodnější pro implementaci.

Část práce je zaměřena na popis jednotlivých NIDS systému jako jsou Suricata, Snort, Bro, kde je proveden rozbor architektury těchto systémů a jednotlivých části, ze kterých jsou složeny. Následně bylo provedeno porovnání na základě odborné literatury, která popisuje jejich výkon v definovaném/daném objemu protékajícího provozu a zatížení HW prvků.

Další částí práce byla implementace uživatelského rozhraní, které je realizováno pro jednoduchou dostupnost ve webovém prohlížeči. Rozhraní se skládá z modulů, které zajišťují analýzu, filtraci a záznam síťového provozu.

Dále proběhlo testování síťových karet na vybraném HW a sledování využití CPU a RAM v situaci, kdy byl na zařízení generován datový provoz od velikosti 2 Gb/s. Při testování byli použity protokoly TCP a UDP a bylo zjištěno, že sonda má dostatečný potenciál pro použití dvou externích síťových karet s přenosovými rychlostmi až 2,5 Gb/s. Výsledný provoz, který by byl analyzován by mohl nabývat až 5 GB/s.

Pro ochranu HW části sondy byl navržen bezpečnostní kryt pro síťovou sondu, který chrání zařízení před vlivy prostředí, které na sondu mohou působit. Náročným prostředím se myslí takové, kde mohou působit např. vlivy počasí (teplota, vlhkost, déšť). Síťová sonda může být využita v teplotách od 0 °C do 60 °C podle specifikací od výrobce HW. Došlo ke snížení rozsahu požadovaných teplot, v kterých HW může být využit z důvodu cena/výkon a možností připojit externí zařízení pomocí USB-C.

Poslední část práce byl návrh a realizace SW části, kde dochází k detekci dodatečných protokolů jako jsou DLMS, SV, GOOSE, IEC 60870-5-104. Při detekci zmíněných protokolů pomocí vytvořeného parseru v knihovně *Scappy* dochází k získání potřebných dat z paketů a následnému logování vybraných informací.

Literatura

[1] Seminární práce na téma LINUX | Jan Václavík. Jan Václavík | [online]. Copyright © Jan Václavík 2014 [cit. 12.10.2020]. Dostupné z: <http://janvaclavik.cz/seminarni-prace-na-tema-linux/

[2] What is the Linux kernel?. [online]. Copyright ©2020 Red Hat, Inc. [cit. 12.10.2020]. Dostupné z: <https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel

[3] O'Reilly Media Technology and Business Training [online]. Copyright © 2001, O [cit. 14.10.2020]. Dostupné z: <https://www.oreilly.com/openbook/debian/book/ch01_02.html

[4] Wenji Wu, Matt Crawford *The Performance Analysis of Linux Networking* Dostupné z: <https://indico.cern.ch/event/408139/contributions/979737/attachments/815628/1117588/CHEP06.pdf

[5] Linux man page. Linux Documentation [online]. Dostupné z: <https://linux.die.net/man/8/iptables

[6] VUT v Brně: Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací. Najbr Ondřej *Adaptivní Linuxové firewally, geografický firewalling [online]. Brno, 2009 [cit. 2020-10-14].* Dostupné z: <http://hdl.handle.net/11012/2793

[7] IptablesHowTo - Community Help Wiki. Official Ubuntu Documentation Dostupné z: <https://help.ubuntu.com/community/IptablesHowTo

[8] *nftables: správa pravidel v našem firewallu* Root.cz - informace nejen ze světa Linuxu [online]. Copyright © 1998 [cit. 15.10.2020]. Dostupné z: <https://www.root.cz/clanky/nftables-sprava-pravidel-v-nasem-firewallu/

[9] VŠB v Ostrava: Lukáš Kuna *Možnosti analýzy IP toků v OS Linux* Dostupné z: <http://wh.cs.vsb.cz/sps/images/a/a0/Kuna-IPFlows-Linux.pdf

[10] MMAP. The Open Group Publications Catalog [online]. Copyright © 2001 [cit. 17.10.2020]Dostupné z: <https://pubs.opengroup.org/onlinepubs/9699919799/functions/mmap.html

[11] *Comparison of frameworks for high-performance packet IO.* ResearchGate | Find and share research [online]. Copyright © 2008 [cit. 15.10.2020]. Dostupné z: <https://www.researchgate.net/publication/301405281_Comparison_of_frameworks_for_high-performance_packet_IO

[12] *nftables: linuxový firewall s moderními vlastnostmi* - Root.cz. Root.cz - informace nejen ze světa Linuxu [online]. Copyright © 1998 [cit. 17.10.2020]. Dostupné z: <https://www.root.cz/clanky/nftables-linuxovy-firewall-s-modernimi-vlastnostmi/

[13] Netmap and PF_RING - DNA. Blog for and by my students, current and future... [online]. Dostupné z: urlhttps://sgros-students.blogspot.com/2013/12/netmap-and-pfring-dna.html

[14] Cloud Developer Tutorials and Software from Red Hat | Red Hat Developer [online]. Dostupné z URL: <https://developers.redhat.com/blog/2017/04/11/benchmarking-nftables

[15] TU v Ostravě: DSpace VŠB-TUO [online]. Copyright ©v [cit. 07.10.2020]. Dostupné z URL: <https://dspace.vsb.cz/bitstream/handle/10084/119127/CZY017_FEI_N2647_2601T013_2017.pdf?sequence=1

[16] Detecting and preventing attacks using network intrusion detection systems. ResearchGate | Find and share research [online]. Copyright © 2008 [cit. 07.10.2020]. Dostupné z URL: <https://www.researchgate.net/publication/41845843_Detecting_and_preventing_attacks_using_network_intrusion_detection_systems>.

[17] (PDF) Research in Intrusion-Detection Systems: A Survey. ResearchGate | Find and share research [online]. Copyright © 2008 [cit. 12.10.2020]. Dostupné z: <https://www.researchgate.net/publication/2612139_Research_in_Intrusion-Detection_Systems_A_Survey

[18] *The Snort Intrusion Detection System* InfoSec Blog. InfoSec Blog [online]. Copyright © 2017 Kevin Almansa. Powered by [cit. 14.10.2020]. Dostupné z: <https://kevinalmansa.github.io/ids/ips/Snort/#:~:text=Snort%20is%20an%20open%20source,such%20as%20dropping%20the%20packet.

[19] *Suricata vs Snort vs Bro (Zeek) AT&T Cybersecurity. AlienVault is Now AT&T Cybersecurity* [online]. 2020 [cit. 14.10.2020]. Dostupné z: <https://cybersecurity.att.com/blogs/security-essentials/open-source-intrusion-detection-tools-a-quick-overview

[20] Introduction — *Zeek User Manual v3.2.2.* [online].Copyright 2019, The Zeek Project [cit. 14.10.2020]. Dostupné z: <https://docs.zeek.org/en/current/intro/

[21] *What is Suricata* — Suricata 4.1.0-dev documentation. [online]. Copyright © Copyright 2016, OISF [cit. 15.10.2020]. Dostupné z: <https://suricata.readthedocs.io/en/suricata-4.1.3/what-is-suricata.html

[22] AlienVault is Now AT&T Cybersecurity [online]. Dostupné z: <https://cybersecurity.att.com/blogs/security-essentials/suricata-ids-threading-capabilities-overview

[23] *Runmodes* — Suricata 4.1.0-dev documentation. [online]. Copyright © Copyright 2016, OISF [cit. 15.10.2020]. Dostupné z URL: <https://suricata.readthedocs.io/en/suricata-4.1.3/performance/runmodes.html>.

[24] *Suricata vs snort* - Technologie - bibliothequer. Dissertations, mémoires, rapport de stage exemples (gratuite en ligne) -bibliothequer.com [online]. Copyright ©2019 bibliothequer.com [cit. 15.10.2020]. Dostupné z URL: <https://bibliothequer.com/technologie/suricata/>.

[25] *Suricata IDS* - An overview of threading capabilities | AT&T Cybersecurity. AlienVault is Now AT&T Cybersecurity [online]. Copyright © Copyright 2020 [cit. 19.11.2020] Dostupné z URL: <https://cybersecurity.att.com/blogs/security-essentials/suricata-ids-threading-capabilities-overview#:~:text=.

[26] Bradner S. a McQuaid J. *RFC 2544 - Benchmarking Methodology for Network Interconnect Devices* Dostupné z URL: <http://www.faqs.org/rfcs/rfc2544.html

[27] *IEC-104* - Faculty of Information Technology [online]. Copyright © [cit. 30.03.2021]. Dostupné z URL: <https://www.fit.vut.cz/research/publication-file/11570/TR-IEC104.pdf

[28] *DLMS* - 4.1 DLMS - Uživatelský manuál 2N® MeterCom PRO. 2N Manuals Dashboard [online]. Dostupné z URL: <https://wiki.2n.cz/mcpum/latest/cs/4-konfigurace/4-1-dlms.

[29] MATOUŠEK, P. *Description of IEC 61850 Communication.* [online]. Copyright 2018© [cit. 22.04.2021]. Dostupné z URL: <https://www.fit.vut.cz/research/publication-file/11832/TR-61850.pdf>

[30] Typhoon HIL Documentation *Description of IEC 61850 Communication.* [online]. Copyright 2018© [cit. 22.04.2021]. Dostupné z URL: <https://www.typhoon-hil.com/documentation/typhoon-hil-software-manual/References/iec_61850_sampled_values_protocol.html>

Seznam symbolů, veličin a zkratek

API	Aplikační programovatelné rozhraní
DB	Databáze pravidel
DNAT	Destination Network Adress Translation - cílový síťový adresní překlad
NIC	Network interface adapter - Ovladač síťového rozhraní
TCP	Transmision Control Protocol
UDP	User Datagram Protocol
OS	Operační systém
ID	Identifikační číslo
IP	Internet protokol
IDS	Intrusion Detection System - systém detekce narušení
TTL	Time To Live - doba životnosti
GNU	Obecně veřejné licence
NAT	Network Adress Translation - síťový adresní překlad
OS	Operační systém
SNAT	Source Network Adress Translation - zdrojový síťový adresní překlad
PK	Primární klíč
NIDS	Network Intrusion Detection System - Síťový systém detekce narušení
HDD	Harddisk
NIC	Network Interface Controller - Ovladač síťového rozhraní
DMA	Direct Memory Access - Přímý paměťový přístup
DC	Direct current - stejnosměrný proud
RAM	Random acess memory - náhodný přístup k paměti

Seznam příloh

A Kompletní seznam kandidátů na síťovou sondu

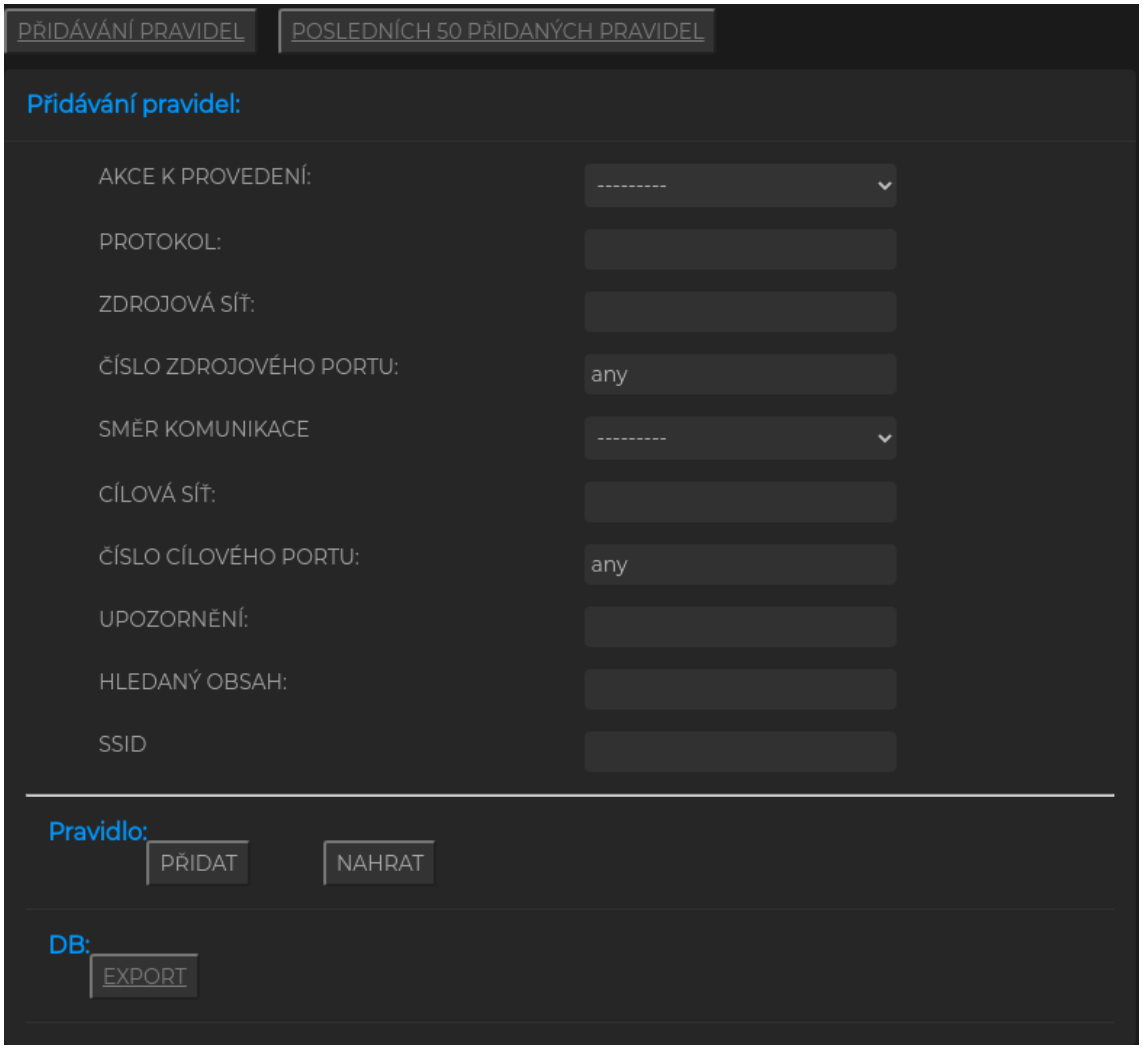
Tab. A.1: Výpis nalezených zařízení na síťovou sondu.

Zařízení	RAM	CPU
ODROID-XU4	2Gbyte LPDDR3	Cortex™-A15 Cortex™-A7
Banana PI M3	2 GB LPDDR3	Allwinner A83T ARM Cortex-A7 Octa-Core
ASUS Tinekr Board S	2GB DDR3	ARM Cortex-A17
UP-CHT01-A20-0464-A11	4GB DDR3L	Atom x5-Z8350
NVIDIA Jetson	4 GB 64-bit LPDDR	ARM Cortex-A57
ODYSSEY - X86J4105864	8GB LPDDR4	Intel Celeron J4105
PICO-WHU4-A11-0002	DDR4 16B	i3-8145ue
LattePanda V1.0	4GB DDR3L	Intel Atom x5-Z8350
UPS-APLP4-A20-08128	8GB DDR4	Intel® Pentium™ N4200
NanoPi R2S	1GB DDR4 RAM	Cortex-A53
Libre Computer	4GB LPDDR4	2 ARM Cortex-A72 4 ARM Cortex-A53
Newport GW6404	2GB DDR4	Cavium OcteonTX™ Dual Core /Quad Core
Orange Pi 4B	4GB LPDDR4	Dual-core Cortex-A72 Quad-core Cortex-A53
PC Engines APU.4D4	4GB DDR3	G series GX-412TC
APU PC Engines apu2e4	4GB DDR3	G series GX-412TC
MinnowBoard Turbot Dual Ethernet Quad Core Board	2GB DDR3L	Intel® Atom™ E3845
ADL120S	Až 32GB DDR4	Core i7-6700TE
Axiomtek CAPA841 3.5	až 8GB DDR3L	Intel Atom E3845
PCM-9366N-S2A1E	8GB DDR3L	Intel® Pentium™ N4200
MIO-5373U-U7A1	Až 32GB DDR4	i7-8665UE
computer j1900	2- 4GB DDR3	j1900
atom D2550 industrial mini itx firewall router	až 4GB DDR3	Atom D2550/N2600/N2800 dual core processors

Zařízení	RAM	CPU
industrial motherboard Skylake-U i3i5i7	až 32GB DDR4	Skylake-U Soc i3/i5/i7
6 LAN firewall router itx motherboard	až 8GB DDR3	I7-4500U
MS-98L3	až 16GB DDR4	i7-8665UE/i5-8365UE
SBC8018	128MByte DDR SDRAM 128MByte NAND Flash	ARM926
ESPRESSObin SBUD102 V5 64 Bit	1 až 2 GB DDR3	dual core ARM Cortex A53
imx6 Rex Module	4GB DDR3	i.MX6 QuadPlus
SBC-350	Až 32GB DDR4	i7-8665UE, i5-8365UE i3-8145UE,CELERON-4305UE
SOM-P102	8GB DDR4	Apollo Lake SoC Processor

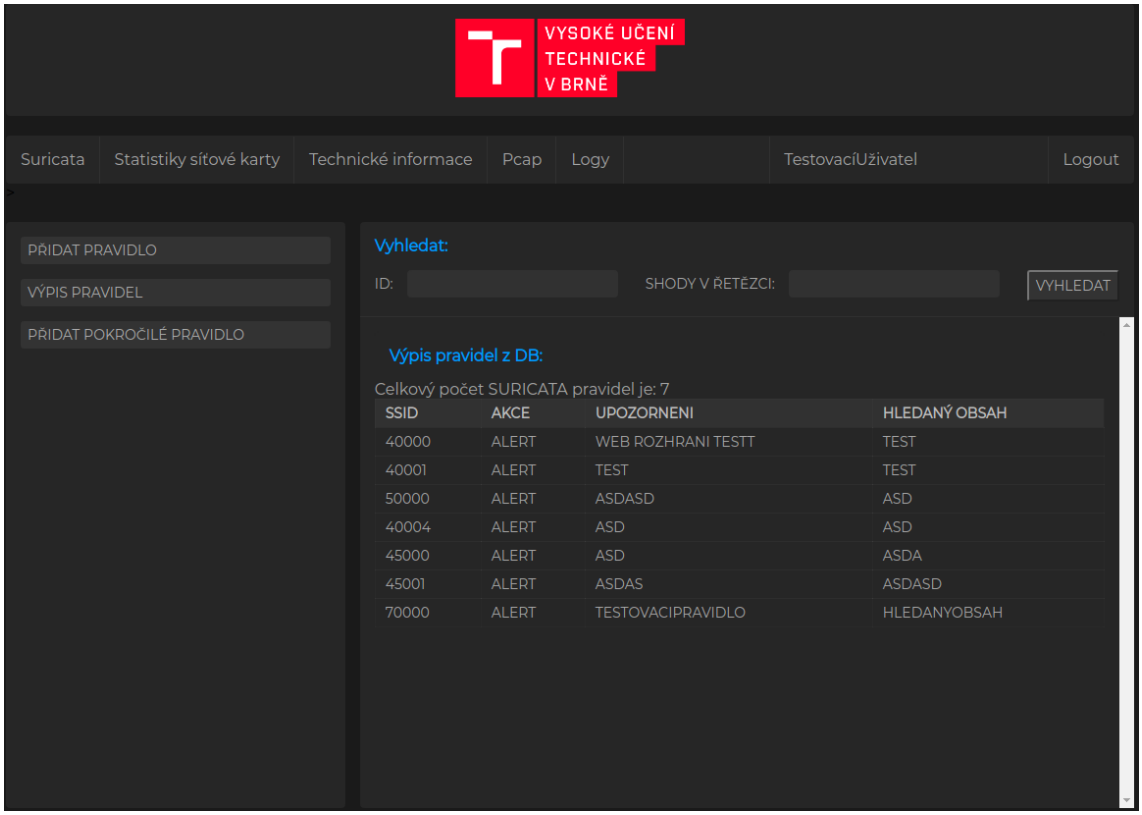
B Ukázky GUI rozhraní pro vybrané moduly.

Kapitola, která obsahuje grafický přehled o webovém rozhraní pro uživatele. Je zde prezentován grafické rozhraní, které uživatel ovládá.

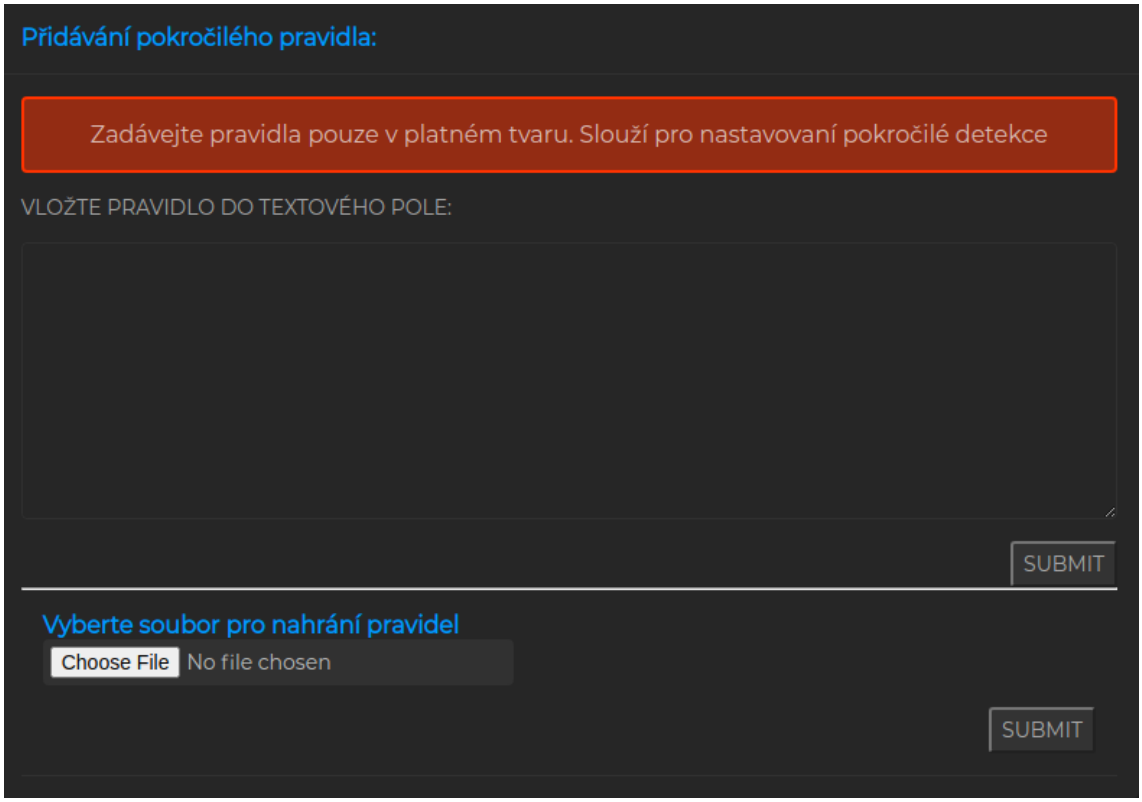


Obr. B.1: Přidáváni pravidel.

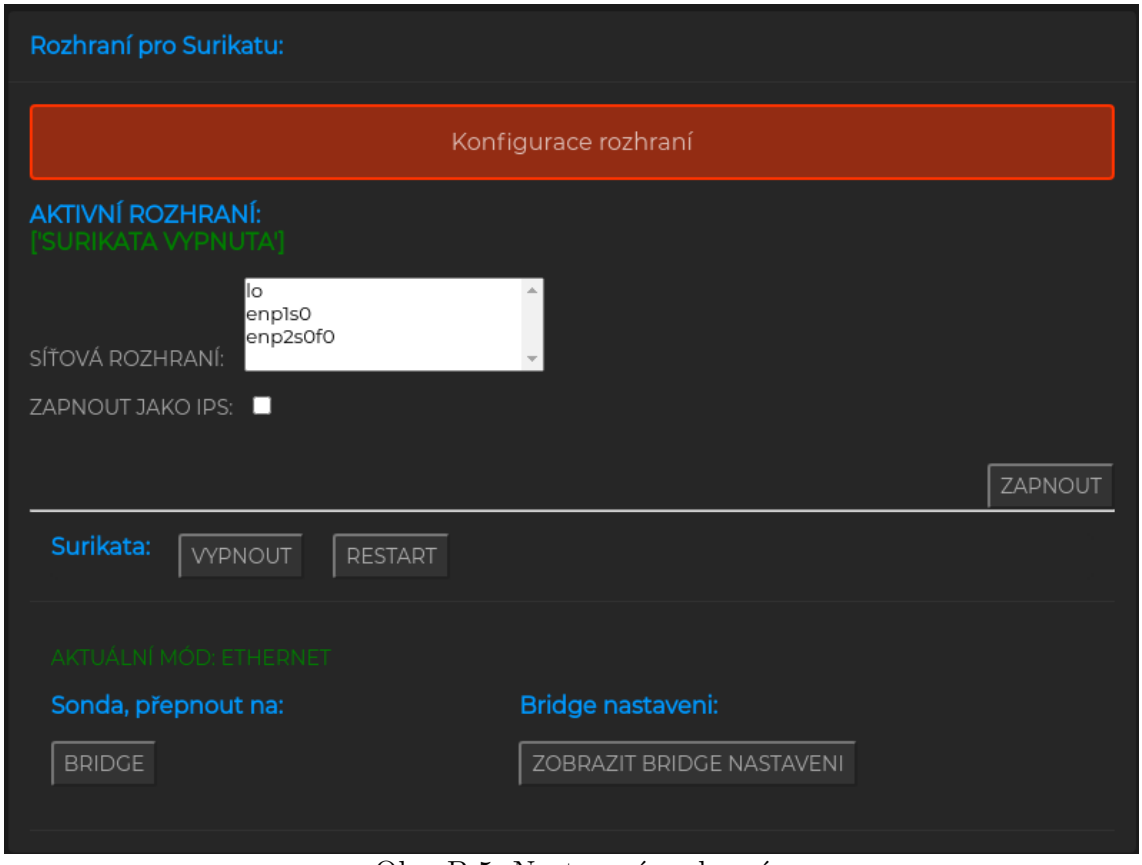
Obr. B.2: Přehled nastavení pro Surikatu.



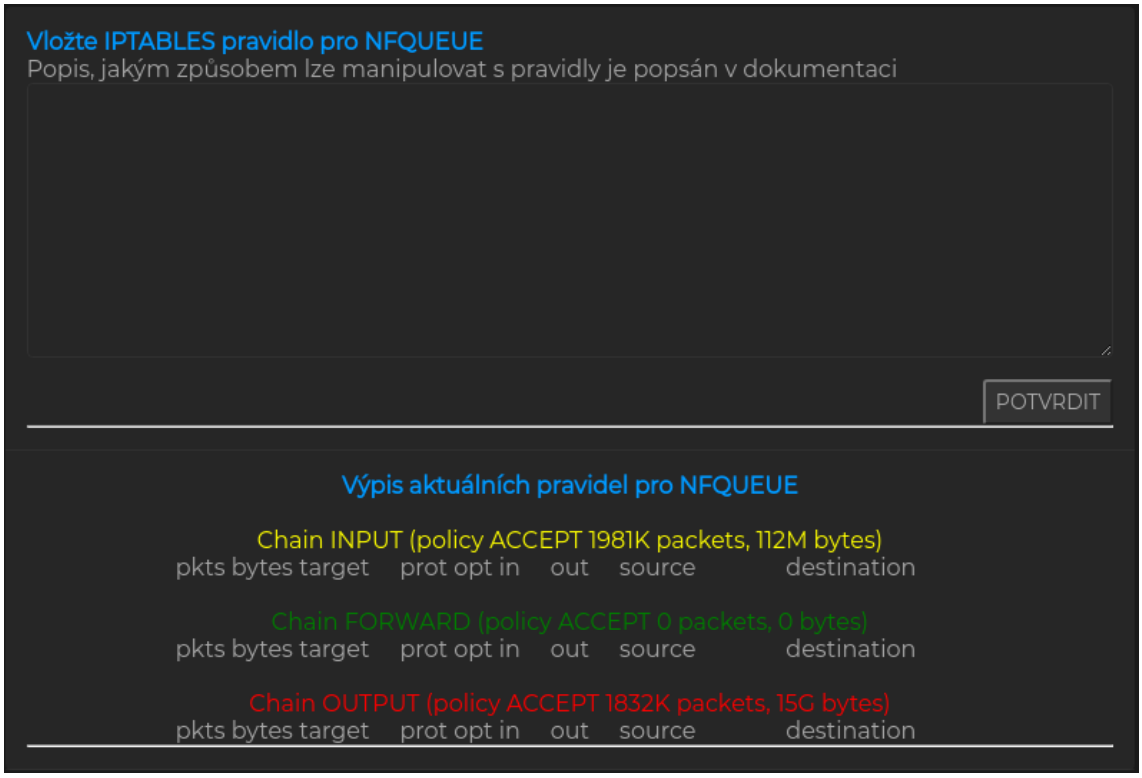
Obr. B.3: Vyhledávání pravidel.



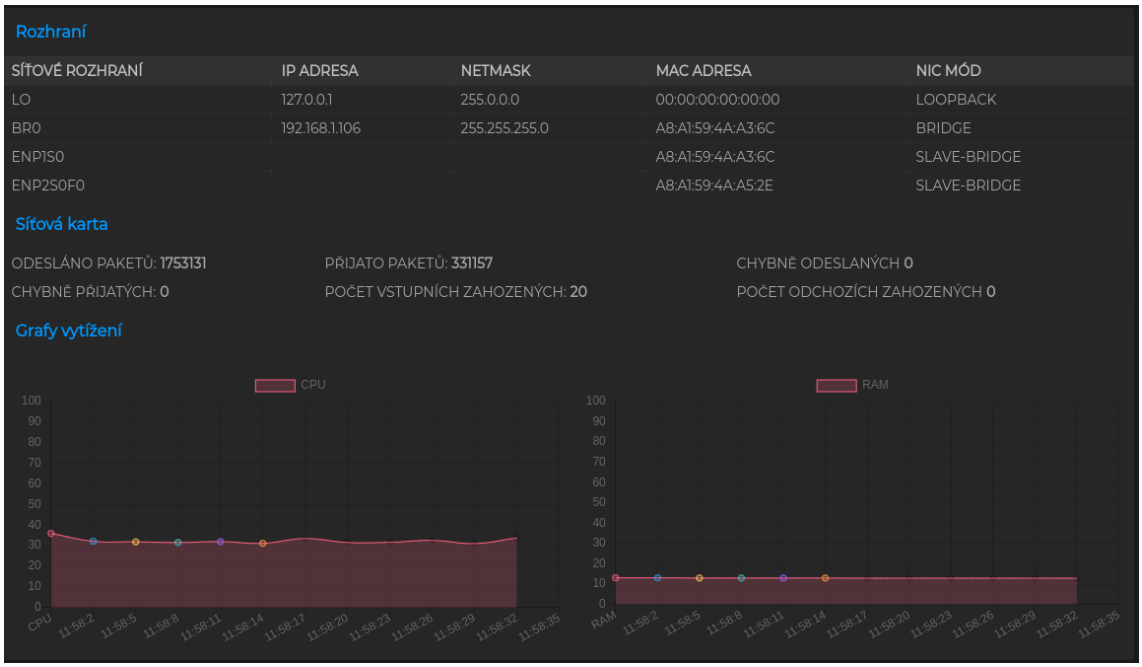
Obr. B.4: Pokročilé přidávání.



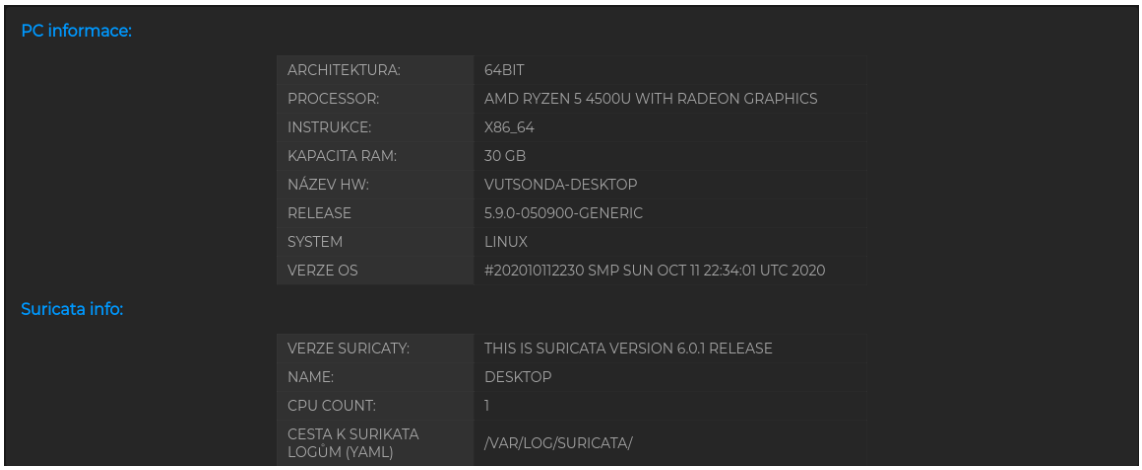
Obr. B.5: Nastavení rozhraní



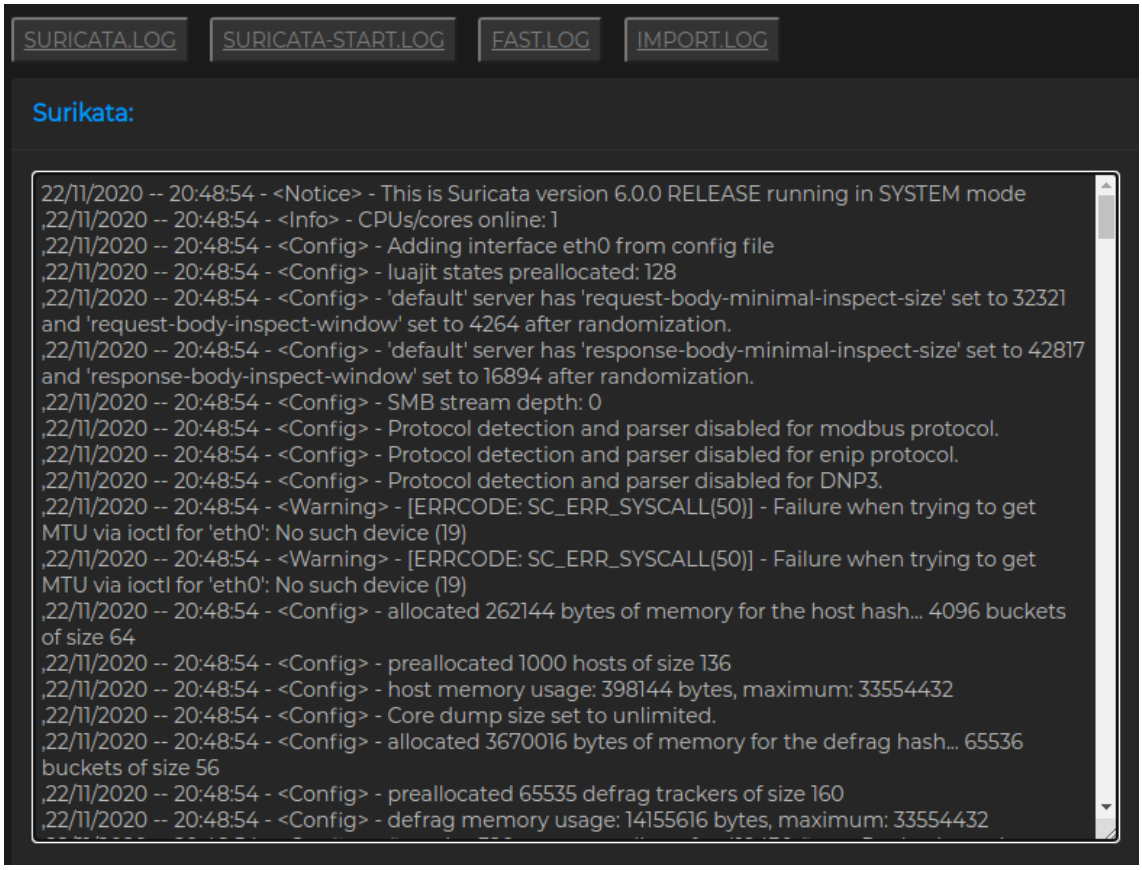
Obr. B.6: IPS nastavení



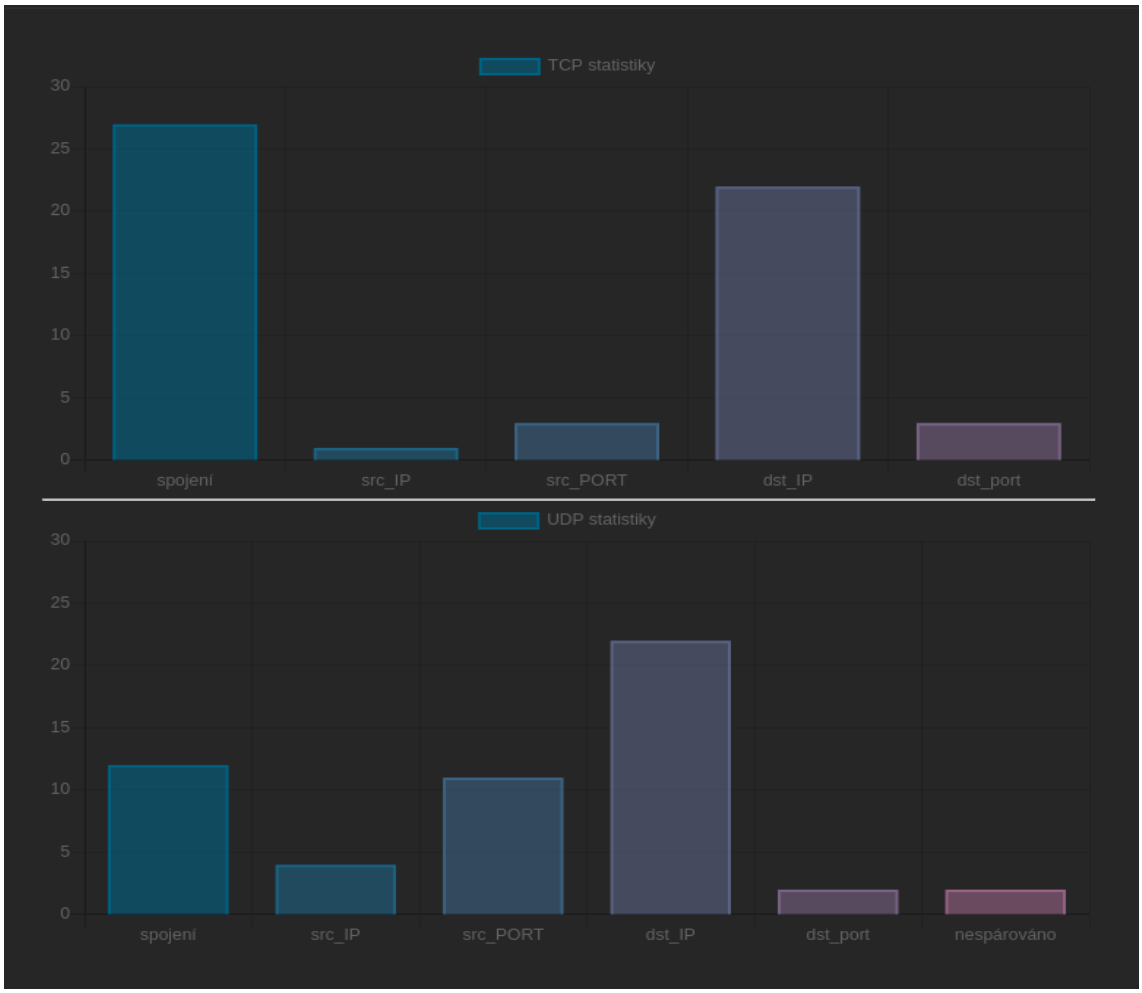
Obr. B.7: Síťové statistiky přehled.



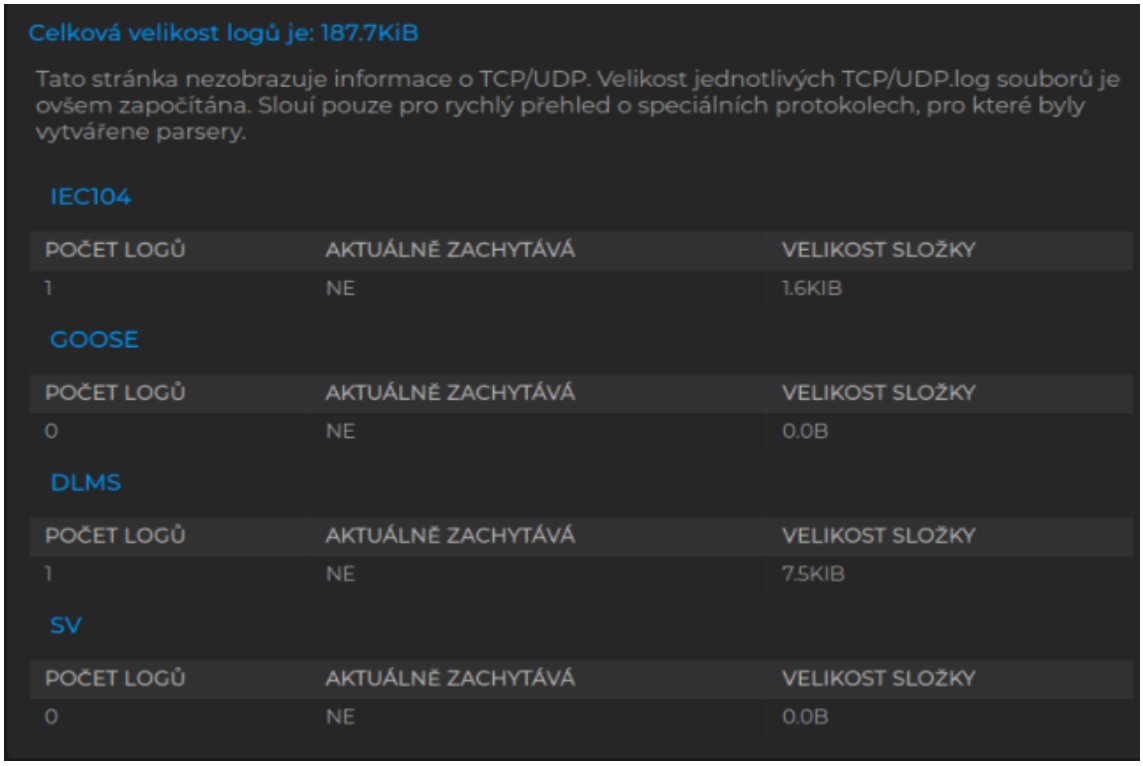
Obr. B.8: Technické informace o zařízení.



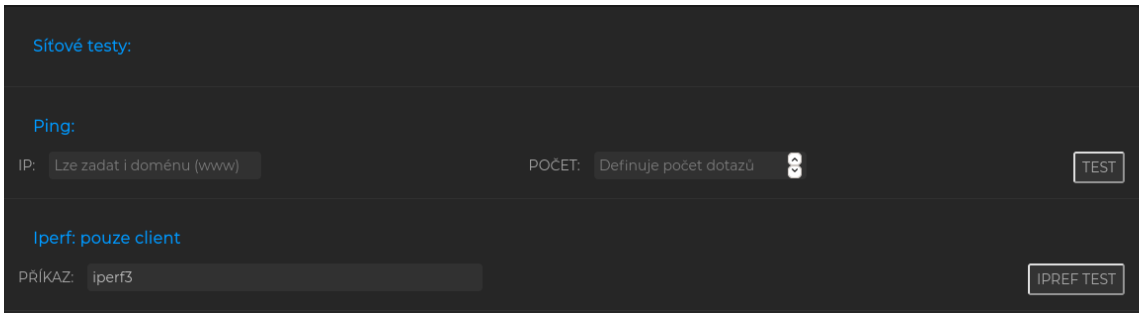
Obr. B.9: Surikata.log.



Obr. B.10: Síťová komunikace grafy.



Obr. B.11: Scappy přehled.



Obr. B.12: Síťové testy.

C Návod k webovému rozhraní

Pro získání návodu bez nutnosti stahovat zdrojové kódy lze využít následující odkaz:
<https://drive.google.com/drive/folders/1vjmhhh-UC7jkVNc5EcQwych-0nje2Jsg?usp=>
Adresář s aplikací obsahuje zdrojové kódy a návody k použití/instalaci aplikace. Pro získání zdrojových kódu je nutné kontaktovat vedoucího práce, který má přístup k github adresáři.

- <https://github.com/Saurus119/SitovaSonda_Klecka>