VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

AKCELERACE DETEKCE BEZPEČNOSTNÍCH HROZEB V SÍTI

ACCELERATED DETECTION OF NETWORK SECURITY THREATS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ADAM PIECEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN KUČERA
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Piecek Adam**
Obor: Informační technologie
Téma: **Akcelerace detekce bezpečnostních hrozeb v síti**
Accelerated Detection of Network Security Threats
Kategorie: Počítačová architektura

Pokyny:

1. Seznamte se se systémy IDS pro detekci bezpečnostních hrozeb v síti, a s konceptem softwarově definovaného monitorování.
2. Proveďte analýzu statistických vlastností síťového provozu v akademické síti CESNET2.
3. Navrhněte využití konceptu softwarově definovaného monitorování pro urychlení funkce vybraných IDS aplikací.
4. Implementujte systém dle návrhu a proveďte experimenty k ověření dosažených výsledků.
5. Diskutujte možnosti dalšího urychlení.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kučera Jan, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

prof. Ing. Lukáš Sekanina, Ph.D.
vedoucí ústavu

Abstrakt

Bakalářská práce se zabývá akcelerací systémů IDS (Intrusion Detection System) pro detekci bezpečnostních hrozeb v síti. Hlavním cílem práce je návrh na využití konceptu softwarově definovaného monitorování (Software Defined Monitoring, SDM) k urychlení činnosti IDS aplikací a to s ohledem na možnost jejich následného nasazení pro analýzu vysokorychlostních sítí. Navržený systém je implementován a následně vyhodnocen pro dvě vybrané open-source aplikace – Snort a Suricata. Nad rámec zadání je pro systém Suricata implementována i nativní podpora rozhraní SZE2 pro příjem paketů s cílem dosažení ještě vyššího urychlení při použití akcelerační síťové karty. V práci jsou dále analyzovány a porovnány dvě varianty konceptu. První varianta využívá hardwarově akcelerovanou verzi SDM. Druhá varianta je potom založena na plně softwarové implementaci principu SDM. Obě varianty jsou následně vyhodnoceny z hlediska dosažených výsledků a výkonových parametrů celého systému před a po akceleraci.

Abstract

This bachelor's thesis deals with the acceleration of IDS (Intrusion Detection System) for detection of security threats in networks. The main goal of the thesis is a proposal to use the Software Defined Monitoring (SDM) concept to accelerate the activity of IDS applications with a regard to their subsequent deployment for high-speed network analysis. The proposed system is implemented and subsequently evaluated for two selected open-source applications – Snort and Suricata. Over and above the task, native support for the SZE2 interface for packet acquisition is also implemented for the Suricata system in order to achieve even faster acceleration using an accelerated network interface card. Two alternatives of the concept are further analysed and compared in the thesis. The first alternative uses the hardware-accelerated version of SDM, while the second alternative is based on full software implementation of the SDM principle. Both alternatives are then evaluated in terms of achieved results and performance parameters of the entire system before and after the acceleration.

Klíčová slova

Systémy pro detekci bezpečnostních hrozeb, IDS, softwarově definované monitorování, SDM, bezpečnost, monitorování, vysokorychlostní sítě, hardwarová akcelerace.

Keywords

Instrusion Detection System, IDS, Software Defined Monitoring, SDM, Security, Monitoring, High-speed Networks, Hardware Acceleration.

Citace

PIECEK, Adam. *Akcelerace detekce bezpečnostních hrozeb v síti.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kučera Jan.

Akcelerace detekce bezpečnostních hrozeb v síti

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Kučery. Další informace mi poskytli Ing. Viktor Puš, Ph.D. a Ing. Lukáš Kekely. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Adam Piecek
15. května 2017

Poděkování

Chtěl bych hlavně poděkovat vedoucímu práce Ing. Janu Kučerovi za jeho odbornou pomoc a velmi cenné rady při psaní této práce. Chtěl bych také poděkovat lidem ze sdružení CESNET za spolupráci a poskytnuté informace. V neposlední řadě děkuji rodině za podporu po celou dobu mého studia.

Obsah

1 Úvod **2**

2 Systémy pro detekci bezpečnostních hrozeb **4**
2.1 Klasifikace detekčních systémů 5
2.2 Vybrané detekční systémy 8
2.2.1 Snort 8
2.2.2 Suricata 10
2.2.3 Bro 11

3 Softwarově definované monitorování **13**
3.1 Firmwarová část systému 14
3.2 Softwarová část systému 16

4 Konceptuální návrh **19**
4.1 Charakteristické vlastnosti na síti 20
4.2 Analýza síťových toků z pohledu detekce incidentů 22

5 Implementace **25**
5.1 Komunikace s řadičem SDM 25
5.2 Nativní podpora rozhraní SZE2 27
5.3 Softwarová varianta SDM 28

6 Dosažené výsledky **30**
6.1 Prostředí pro provádění měření 30
6.2 Snort a Suricata s rozhraním libpcap 32
6.3 Suricata s rozhraním SZE2, softwarová varianta 35
6.4 Shrnutí výsledků měření, diskuze 39

7 Závěr **41**

Literatura **43**

Přílohy **47**

A Obsah CD **48**

Kapitola 1

Úvod

V posledním dvacetiletí jsme svědky rychlého rozvoje informačních technologií. V moderní společnosti se stávají různé druhy zařízení a služeb běžnou až nezbytnou součástí našich životů. Jsou to ale právě počítačové sítě, které umožňují uživatelům komunikovat či sdílet informace mezi sebou napříč celým světem. Rychlost přenášených dat na velkých páteřních linkách a v datových centrech může být až několik tisíckrát větší než rychlost dnes běžného domácího internetového připojení. Vývoj rychlosti počítačových sítí se přitom nutně musí promítnout i do oblasti bezpečnosti a monitorování těchto sítí. Vysokorychlostní sítě totiž představují výzvu pro bezpečnostní odvětví především při ochraně před počítačovými útoky. Každoročně vzrůstá počet útoků nejen na vládní instituce, ale také na velké nadnárodní společnosti [6], což jen potvrzuje nutnost se bezpečnostní situaci věnovat. Jedním z používaných nástrojů pro bezpečnostní monitorování sítí jsou systémy pro detekci bezpečnostních hrozeb (Intrusion Detection System, IDS).

Systémy IDS umožňují detailní pohled na síťový provoz. Provádí analýzu provozu nejen na základě hlaviček paketů, jako je tomu například v paketových filtrech (firewallech), ale dokáží detekovat daleko sofistikovanější útoky, díky detailní analýze paketů (DPI). Při analýze se totiž zaměřují také na aplikační (L7) vrstvu a přenášená uživatelská data. Hledají se například známé vzory, které vypovídají o nežádoucích aktivitách. Podobně jako antivirové systémy využívají některé typy IDS ke své činnosti databázi vzorů známých bezpečnostních incidentů. Úspěšnost zachycení útoku potom závisí na aktuálnosti databáze vzorů. Systémy IDS jsou však z důvodu provádění detailních analýz pomalé a pro použití na vysokorychlostních sítích nedosahují dostatečné výkonnosti. Tento problém je možné řešit určitou formou hardwarové akcelerace. Takovou možnost nabízí například koncept softwarově definovaného monitorování (Software Defined Monitoring, SDM), který se zaměřuje na problematiku hardwarové akcelerace bezpečnostního monitorování vysokorychlostních sítí. SDM umožňuje předzpracování síťových dat na akcelerační síťové kartě a aplikaci nabízí flexibilní softwarově řízenou ztrátu příchozích informací. Díky tomuto přístupu lze na síťové kartě zpracovat až 80 % veškerého síťového provozu, který není z hlediska detekce incidentů relevantní, a detailní analýzu následně zaměřit na zbylých 20 % provozu.

Tato práce se zabývá návrhem na využití konceptu softwarově definovaného monitorování k akceleraci právě IDS aplikací. Konkrétně byla v rámci práce navržena a implementována podpora konceptu SDM pro open-source IDS systémy Snort a Suricata. Pro určení vlivu použití akcelerační síťové karty na dosažené zrychlení je implementace pro systém Suricata provedena ve dvou variantách. První varianta využívá hardwarově akcelerovanou verzi konceptu SDM. Druhá varianta je potom založena na plně softwarové implementaci principu SDM. Nad rámec zadání je pro systém Suricata navíc implementována také nativní

podpora rozhraní SZE2 pro dosažení ještě vyššího urychlení. Závěrem práce je vyhodnocen přínos realizovaného řešení a obě varianty jsou porovnány z hlediska výkonových parametrů celého systému před a po akceleraci.

Práce je rozdělena do několika kapitol shrnujících postup při řešení zadaného problému. Kapitoly 2 a 3 obsahují teoretický úvod k navazující praktické části práce. V kapitole 2 je nejprve uvedena základní charakteristika systémů pro detekci bezpečnostních hrozeb v síti, dále je rozebrána klasifikace systémů podle různých kritérií a následně jsou popsány tři konkrétní nástroje – Snort, Suricata a Bro. V rámci následující kapitoly 3 je potom vysvětlen princip softwarově definovaného monitorování. V kapitole 4 je rozebrán vlastní návrh řešení, jak koncept SDM v kombinaci se systémy IDS vhodně využít. Tato kapitola obsahuje také shrnutí charakteristických vlastností síťového provozu využitelných pro nasazení konceptu SDM (podkapitola 4.1) a výsledky analýzy, která zjišťuje množství detekovaných incidentů v závislosti na počtu paketů analyzovaných na začátku každého síťového toku (podkapitola 4.2). Detaily implementace navrženého systému jsou popsány v kapitole 5. V rámci kapitoly jsou rozebrány obě varianty implementace, jak varianta využívající hardwarově akcelerovanou verzi, tak varianta založená na plně softwarové implementaci principu SDM. Je zde popsán také postup implementace nativní podpory rozhraní SZE2 pro systém Suricata. Kapitola 6 se věnuje popisu použitého prostředí, výkonnostnímu měření vytvořeného systému a následnému vyhodnocení dosažených výsledků. Závěrečná kapitola 7 shrnuje obsah a celkové výsledky bakalářské práce.

Kapitola 2

Systémy pro detekci bezpečnostních hrozeb

V této kapitole se nachází první část teoretického rozboru, který tvoří základ pro praktickou část práce popsanou v následujících kapitolách. V rámci kapitoly jsou charakterizovány systémy pro detekci bezpečnostních hrozeb, je zde uvedena jejich klasifikace podle různých kritérií a následně jsou popsány tři konkrétní volně dostupné nástroje – Snort, Suricata a Bro. Pokud není uvedeno jinak, základní charakteristika těchto systémů vychází z následujících prací [35, 48, 33].

Systémy pro detekci bezpečnostních hrozeb (Intrusion Detection System, IDS) jsou bezpečnostní systémy, které umožňují sledování síťového provozu, případně činnosti operačního systému s cílem detekce a hlášení podezřelých aktivit představujících potenciální narušení zabezpečení sítě nebo operačního systému. IDS je schopno odhalit probíhající útoky a v rámci nastavené bezpečnostní politiky reagovat na nežádoucí aktivity, i když nutně nemusí představovat přímé ohrožení. IDS je systém pasivní povahy. V případě detekce ohrožení bezpečnosti pouze podává hlášení administrátorovi, případně nadřazenému systému, a přitom nijak nezasahuje do jakýchkoliv síťových služeb, eventuálně procesů.

Systém IDS je velmi důležitým nástrojem pro zajištění bezpečnosti počítačové sítě. Při jeho absenci může na síti bez vědomí uživatelů proběhnout několik opakovaných útoků a proniknutí. Bez vědomí o existenci proniknutí potom může vzniknout dojem, že není důvod přidávat další bezpečnostní opatření. Samotný systém IDS ale nemůže zabránit proniknutí. V případě úspěšného útoku IDS nicméně poskytne velmi důležité informace o tom, že proniknutí vůbec proběhlo a který cíl byl napaden. Na základě typu útoku lze potom později odvodit, jak velké škody byly napáchány a jak jim pro příště zabránit. V lepším případě dá IDS okamžitě na vědomí, že detekovalo dílčí kroky vedoucí k proniknutí, tudíž hlášení upozorní síťové administrátory, aby včas zasáhli a zabránili tak škodám dříve než nastanou. IDS ale poskytuje také informace o neúspěšných pokusech útočníka. Vede tak ke zvýšení bdělosti v následujícím období a zároveň poskytuje zpětnou vazbu o tom, na které typy proniknutí je bezpečnostní systém odolný. Systémem IDS může být přitom speciálně vyvinuté hardwarové zařízení se softwarovým řízením, nebo software, který je schopen za běhu spolupracovat s operačním systémem běžícím na osobním počítači případně serveru.

Obdobné systémy IPS (Intrusion Prevention System) se dají pokládat za variantu IDS systémů. Kromě samotné pasivní detekce dále umožňují na hrozbu či událost aktivně reagovat např. filtrací a tím zablokovat škodlivý či nežádoucí datový provoz. Při nasazení a konfiguraci aktivního systému pro blokování síťového provozu je ale třeba zajistit, aby při detekci

nedocházelo (případně jen v omezené míře) k tzv. false positives (mylné označení spojení za nebezpečné).

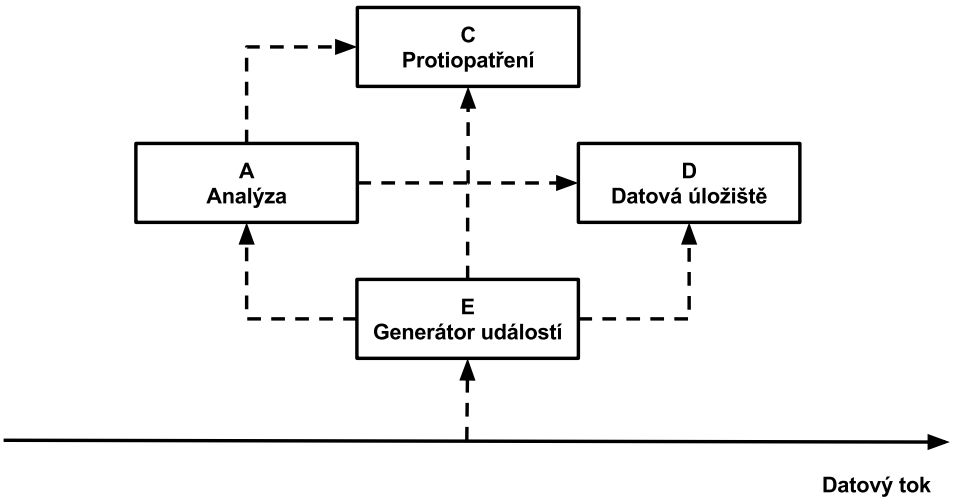
Obecnou strukturu a způsob činnosti systémů IDS, respektive IPS systémů, popisuje model CIDF (The Common Intrusion Detection Framework). Jeho prvky můžeme pozorovat napříč různými implementacemi IDS. Základní model je znázorněn na obrázku 2.1. Model definuje čtyři základní bloky systému IDS:

Generátor událostí (blok E). Generátor událostí (event generators) sleduje dění na datovém toku, provádí analýzu protokolu a zaznamenané informace předává dalším blokům na zpracování.

Analýza (blok A). Nástroje pro analýzu (analysis engines) aplikují definovaná pravidla, například detekci signatur. Na základě výsledku analýzy potom ukládají stavové informace do bloku D nebo vyvolají spuštění příkazů v bloku C.

Datová úložiště (blok D). Datová úložiště (storage mechanism) představují úložiště dat ve formě databází či logů. Musí jít o rychlý systém, aby nedošlo k zahlcení IDS. Modul řeší např. ukládání informací o stavu síťového spojení pro potřeby rekonstrukce TCP.

Protiopatření (blok C). Protiopatření (countermeasures) může představovat například vynucené uzavření síťového spojení, ať už na základě přímé komunikace s firewallem nebo jiné obdobné funkcionality charakteristické pro IPS. Systémy IDS se podílí na protiopatření pouze formou hlášení.



Obrázek 2.1: *Model CIDF.*

2.1 Klasifikace detekčních systémů

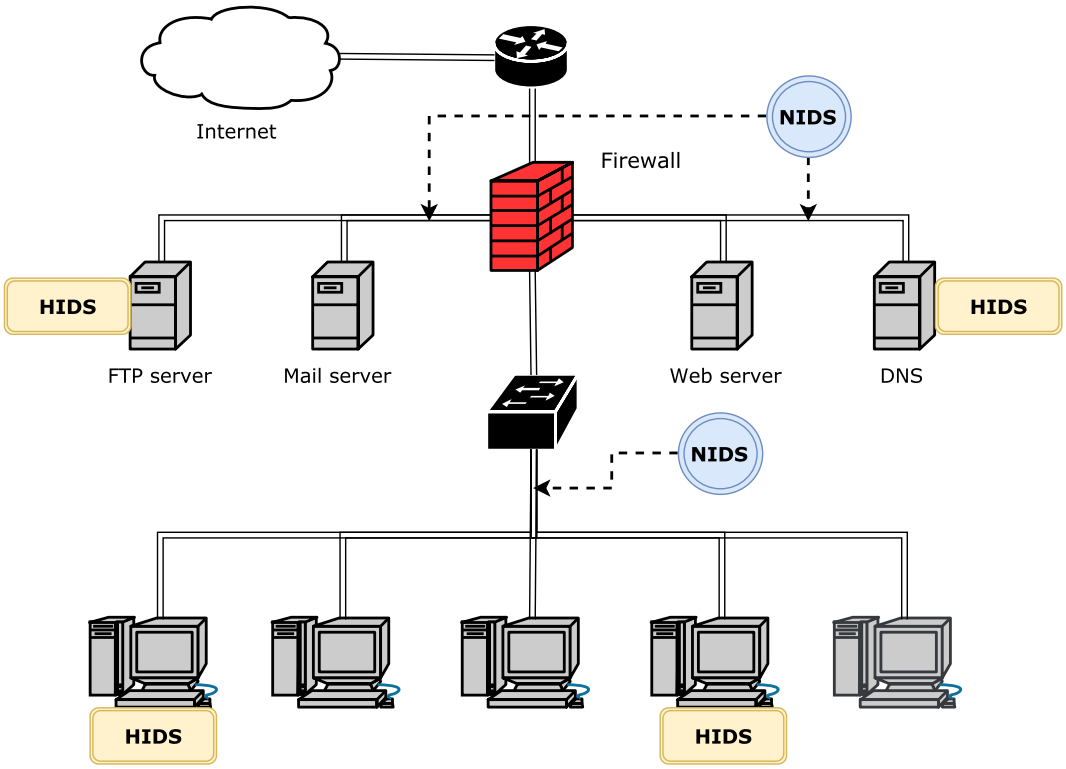
Systémy pro detekci bezpečnostních hrozeb lze klasifikovat podle různých kritérií. Prvním zde uvedeným kritériem je klasifikace systémů IDS podle použité techniky detekce. V rámci tohoto dělení se rozlišují dvě skupiny. Jsou jimi systémy pro vyhledávání vzorů a systémy pro detekci anomálií. Část této kapitoly popisující klasifikaci podle užívaných technik byla čerpána z prací [48, 29, 35].

Systémy pro vyhledávání vzorů. Systémy pro vyhledávání vzorů (Signature Based Setection Systems) jsou nejtypičtější podobou IDS. Tyto systémy v každém příchozím paketu provádějí vyhledávání známých vzorů uložených v databázi. V případě nalezení shody je odesláno hlášení správci nebo se událost jinak zaznamená. Vzory jsou různě složité a typicky kontrolují jeden určitý protokol nebo konkrétní aplikační vrstvu. Účinnost tohoto typu systémů roste, pokud se navíc ukládá kontext probíhající komunikace. S tím se však musí řešit také problém, jak dlouho se bude kontext nedokončené komunikace udržovat v paměti. V případě brzkého zahození kontextu se detekce útoku nemusí podařit. Na druhou stranu dlouhodobé udržování kontextu má zase zásadní vliv na výkonnost systému.

Tyto systémy jsou přímo závislé na aktuálnosti databáze vzorů stejně jako např. antivirové programy. Množství vzorů se může pohybovat v řádech několika tisíc, což může při analýze vysokorychlostního síťového toku způsobit zahazování u vyrovnávacích pamětí. Nevýhodou těchto systémů jsou tedy nároky na zdroje a nebezpečí zahlcení samotného systému při DoS útoku. Za výhodné lze považovat to, že systémy mají za sebou mnoho let vývoje a mohou využívat zjištění antivirových systémů, které používají podobné techniky detekce.

Systémy pro detekci anomálií. Systémy pro detekci anomálií (Anomaly Base Intrusion Detection Systems) jsou schopny odhalit podezřelé aktivity, které za běžných situací na síti neprobíhají. Pro popis běžného charakteru sítě se využívá statistických analýz či se definují pravidla nebo heuristiky. Výsledkem jsou definované meze, kdy jejich překročení znamená pravděpodobnou anomálii. Hlavní tedy je co nejlépe vystihnout normální provoz na dané síti a poté sledovat její vlastnosti pomocí mezí, jejichž překročení vyvolá hlášení. Pokud je normální provoz na síti definován špatně, nastavené meze budou často překračovány a roste podíl falešných hlášení. Anomálie se sledují z komplexní situace provozu na komunikační síti, v rámci síťového toku, nebo i uvnitř paketu např. obsahem netypických hodnot v hlavičkách. Problematické může být hledání anomálií, kde nevypočitatelné chování uživatelů má velký dopad na charakter sítě. Nevýhodou je, že z detekované anomálie se těžko vyvozuje záměr útočníka (pokud se ovšem opravdu o útok jedná a nejde o pouhou neškodnou anomálii např. ve smyslu odchylky od statistických hodnot). Významnou výhodou těchto systémů je, že jsou schopny detekovat nové neznámé incidenty, které třeba ještě nejsou obsaženy v databázi známých vzorů, takže se obě tyto techniky detekcí vzájemně doplňují.

Druhým kritériem klasifikace systémů pro detekci hrozeb je jejich rozdělení podle způsobu, respektive místa, nasazení. Na základě způsobu nasazení lze systémy IDS rozdělit do dvou kategorií. Do první kategorie se řadí detekční systémy, které získávají informace z hostitelského systému. Jedná se o tzv. Host-based IDS (HIDS). Druhá skupina, systémy, které získávají informace z počítačové sítě, se potom nazývají Network-based IDS (NIDS). Oba typy systémů znázorňuje obrázek 2.2. Část této kapitoly o klasifikaci IDS podle způsobu nasazení čerpá z prací [48, 29, 49, 27].



Obrázek 2.2: *Příklad umístění HIDS a NIDS v síti.*

Host-based IDS (HIDS). Systém HIDS je v rámci sítě umístěn přímo na daném zařízení, kterým je typicky server nebo pracovní stanice (workstation). Jak je znázorněno na obrázku 2.2, tento typ systému může běžet na dvou koncových stanicích nebo dále na serverech poskytujících služby FTP a DNS. V podobě softwarového agenta monitoruje operační systém, ukládá data do logovacích souborů nebo generuje hlášení. Čím je monitorování hostitelského systému důkladnější, tím je náročnější na výpočetní zdroje a představuje určitou režii. Díky integraci s hostitelským systémem se HIDS zaměřuje na aplikační vrstvy síťového modelu u běžících procesů. HIDS samotný běží na aplikační úrovni, má tedy přímý přístup k datům přenášeným síťovými a transportními protokoly přes knihovny operačního systému. To může být výhodné například při detekování ohrožení důvěrnosti nebo integrity šifrovaného spojení. Na druhou stranu pokud útočník získá úplnou kontrolu nad zařízením, pak je v jeho moci HIDS vypnout a zamést za sebou stopy v logovacích souborech. Nevýhodou těchto systémů je potom problematická správa a údržba více takových zařízení využívajících své HIDS. Problém se potom ještě více komplikuje v případě správy velké komunikační sítě s více druhy operačních systémů a konfigurací.

Network-based IDS (NIDS). Systém NIDS se obvykle skládá z dalšího hardwarového zařízení (sondy) připojeného do sítě. Zařízení je vybaveno síťovou kartou nastavenou v promiskuitním režimu a separátním rozhraní, na němž je spuštěna softwarová část IDS. Může jít také o speciální hostitelskou aplikaci, která pomocí speciálních knihoven přistupuje k síťovému rozhraní nezávisle na operačním systému. Zařízení NIDS bývá umístěno na strategickém místě v topologii sítě. Na obrázku 2.2 je například zařízení umístěno na lince připojující koncové stanice nebo linkách připojujících jednotlivé servery.

Systém NIDS bývá někdy označován jako tzv. packet-sniffer, jelikož pracuje v pasivním režimu a má přístup k síťovým datům na nejnižší úrovni. Používá se pro získání komplexních informací pro přehled o počítačové síti, dokáže detekovat anomálie a aplikovat hledání vzorů na různých segmentech počítačové sítě, nikoli pouze na datech určených pro vybraný počítač. Pokud se stíhá analyzovat celý procházející síťový provoz, benefitují z této ochrany všechna zařízení připojená k síti. Toto je však právě v případě vysokorychlostních linek problematické. IDS totiž provádí analýzu provozu nejen na základě hlaviček paketů, ale zaměřují se také na aplikační (L7) vrstvu a přenášená uživatelská data.

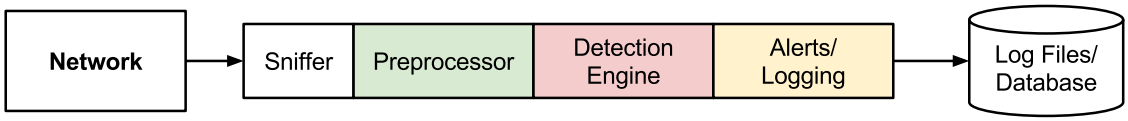
Aby detekce probíhaly úspěšně, je potřeba uvnitř systémů NIDS implementovat mechanismy pro skládání fragmentovaných datagramů a mechanismy pro ukládání kontextu spojení. V ideálním případě by měl být seskládaný síťový tok na straně NIDS stejný jako u chráněných zařízení tak, aby detekce probíhala nad validními daty. Nevýhodou NIDS je, že nemá žádné informace o stavu počítačů připojených k síti. Komunikující aplikace totiž posílají data ve vlastní režii, což může způsobit nekonzistentní stav na straně IDS a útočník toho může využít. Dalším problémem je analýza zabezpečených spojení například pomocí technologií IPsec nebo VPN.

2.2 Vybrané detekční systémy

Následující sekce se věnuje třem volně dostupným implementacím NIDS. Snort, Suricata a Bro jsou nejrozšířenější a nejčastěji zmiňované open-source implementace IDS, které se využívají v praxi. Jsou předmětem mnoha článků a prací [23, 38, 46, 28, 44, 22, 26] věnujících se tomuto tématu. Všechny tři zmíněné systémy jsou přitom součástí Security Onion [4], což je speciální linuxová distribuce určená pro potřeby monitorování počítačové sítě, síťovou bezpečnost a správu logování.

2.2.1 Snort

Snort je v základu open-source paketový sniffer (provádí čtení paketů ze sítě a zobrazuje je v konzoli), paketový logger (provádí záchyt paketů z IP sítě a ukládá je na disk), ale také IDS [23]. Systém umožňuje analýzu síťového provozu a detekci různých typů útoků na základě definovaných vzorů (pravidel). Přímo pro Snort přitom existuje spousta kvalitních a denně aktualizovaných databází pravidel. Příkladem mohou být pravidla od organizace Sourcefire's Vulnerability Research Team (VRT, Talos [47]), pravidla EmergingThreats [39], případně pravidla od početné komunity [43]. Snort umožňuje použití i tzv. Shared Object Rules (SO rules) [41]. Jedná se o pravidla, která jsou zapsána ve specializovaném nízkoúrovňovém jazyce podobném jazyku `C` a jsou dostupná v podobě předkompilovaných knihoven.



Obrázek 2.3: *Schéma architektury Snort.*

Na obrázku 2.3 je znázorněna architektura systému Snort. Pakety ze sítě (Network) jsou zachytávány v rámci komponenty Sniffer. Následující barevně znázorněné součásti (Preprocessor, Detection Engine a výstupní modul Alerts/Logging) jsou potom rozšiřitelné formou

pluginů. Silnou stránkou Snortu je právě zasazení pluginů do vnitřní architektury, jejímž rozšířením se systém profiluje podle potřeby [38]. Pluginy v preprocesorové části (Preprocessor) provádí analýzu a dekódování příchozích dat na úrovni hlaviček paketů. Pluginy v detekční části (Detection Engine) aplikují jednotlivá pravidla a zabývají se samotnou detekcí incidentů. Výstupních pluginů (Alerts/Logging) je k dispozici také celá řada. Určují další způsob zpracování hlášení, zaznamenání či jejich vizualizaci. Výstupní informace Snortu mohou tak být zapisovány do databáze či logovacích souborů.

Aktivní vývoj od roku 1998 přinesl Snortu pověst standardního IDS nástroje, jenž byl v minulosti několikrát nasazen a testován. Počátkem roku 2017 se přitom objevila nová verze Snortu 3.0 (Snort++) [42], která prodělala od Snortu 2.x značný pokrok. Kromě výkonnostních vylepšení vnitřních komponent a dalších funkcí, byla přidána hlavně možnost vícevláknového zpracování paketů. S touto aktualizací tak vyvstává otázka, jak si v budoucnu Snort 3.0 výkonnostně povede při srovnání s IDS Suricata, která byla od počátku budována se zaměřením na vícevláknové zpracování. V době psaní této práce byl Snort 3.0 k dispozici pouze v nestabilní a stále vývojové alpha verzi, tudíž jsou výsledky zmíněné v podkapitole 6.2 provedeny na starší a stabilní verzí 2.9.

S jednovláknovou verzí Snortu 2.x byla v minulosti provedena řada výkonnostních měření a porovnání [38, 22, 40]. Obecně se dá říci, že systém v závislosti na počtu detekčních pravidel dokáže spolehlivě detekovat hrozby v síťovém provozu při rychlostech do 500 Mb/s. Jako jednovláknová aplikace je však vysoce optimalizovaný, nespotřebovává oproti jiným IDS takové množství paměti RAM a výkonu procesoru. Díky jednovláknovému zpracování nemusí řešit problém se synchronizací a komunikací mezi vlákny. Podle [34] je pro nasazení na vysokorychlostní síti do 10 Gb/s potřeba použít výkonnou hardwarovou síťovou kartu, která umožňuje rozdělení datového toku na menší toky. Každý z nich je pak třeba odeslat na jiné vstupní rozhraní a zpracovat jinou instancí Snortu. Při konfiguraci Snortu pro příjem paketů s využitím např. rozhraní `PFRING` [36] je potom možné, aby na serveru nad jedním fyzickým rozhraním běželo několik procesů Snort souběžně. S rostoucím počtem spuštěných instancí Snortu se však vyskytuje problém se správou jednotlivých konfiguračních souborů a výsledných logů.

Pro doplnění následuje příklad konkrétního pravidla a popis jeho jednotlivých položek.

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (
        msg: "ET TROJAN Likely Bot Nick in IRC (USA +..)";
        flow: established, to_server;
        flowbits: isset, is_proto_irc;
        content: "NICK";
        pcre: "/NICK.*USA.*[0-9]3,/i";
        classtype: trojan-activity;
        reference: url,doc.emergingthreats.net/2008124;
        sid: 2008124
        rev: 2;
)
```

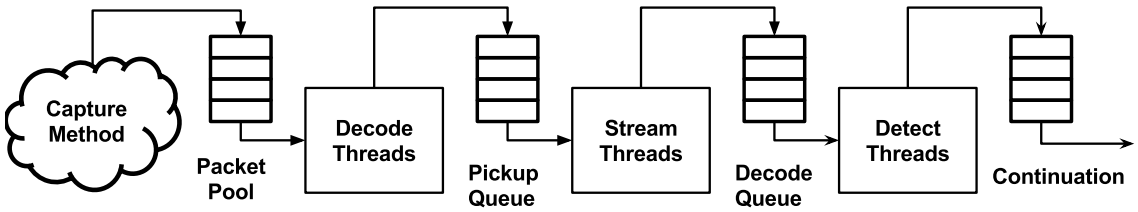
Každé pravidlo se dělí na tři hlavní části. Akce (červená barva) určuje způsob dalšího zpracování, jestliže jsou podmínky v druhé (modrá) a třetí (fialová) části pravidla splněny. Akce `drop` v případě, kdy je IDS nastaveno do režimu IPS, přijatý paket zahodí. Nejčastější typ akce je však `alert`, jenž vyvolá pouze hlášení detekované události. V hlavičce pravidla (modrá barva) je na prvním místě položka značící protokol (`tcp`), následuje zdrojová IP adresa (`HOME_NET`), zdrojový port (`any`), směr analyzované komunikace (`->` nebo `<>`),

cílová IP adresa (`EXTERNAL_NET`) a port (`any`). Proměnné `HOME_NET` a `EXTERNAL_NET` se nastavují v konfiguračním souboru a definují IP prefixy vnitřní a vnější sítě. Klíčové slovo `any` použité v ukázce pravidla značí kteroukoli validní hodnotu. Fialová část popisuje především vlastní obsah pravidla, vyhledávaný vzor a dodatečné informace. Klíčové slovo `msg` označuje textový popisek pravidla, který je k vidění v logovacím souboru. Pro upřesnění, ke které situaci v komunikaci se pravidlo vztahuje, slouží `flow` a `flowbits`. Hledaný vzor je vyjádřen v položkách `content` a `pcre`. Závažnost detekovaného incidentu a třída, ke které se incident řadí, je v položce `classtype`. K podrobnějšímu dohledání informací o incidentu slouží položka `reference`. Klíčová slova `sid` a `rev` popisují dodatečné informace k pravidlu, identifikační číslo a číslo revize [7].

2.2.2 Suricata

Suricata je open-source IDS a IPS využívající pro detekci techniku vyhledávání vzorů v paketech [20]. Suricata je charakteristická svou architekturou, jenž byla už od počátku vývoje přizpůsobená k vícevláknovému zpracování paketů pro dosažení vysoké výkonnosti. Díky tomu se pro analýzu sítě s rychlostí nad 1 Gb/s nemusí spouštět více nezávislých instancí programu, jako tomu je např. u Snortu nebo Bro. Až na drobné výjimky podporuje Suricata stejný formát pravidel jako Snort. Pro zdroj pravidel lze tak použít stejné databáze (VRT [47] a Emerging Threats [39]).

Při srovnání se Snortem lze Suricatu považovat za nástroj, který nabízí podobné možnosti, avšak díky vícevláknovému zpracování poskytuje výrazně vyšší výkonnost. Suricata ale přitom nabízí také řadu zcela nových funkcí. Příkladem může být extrakce souborů (tzv. file carving) z HTTP případně FTP komunikace, podpora vyhledávání vzorů s využitím grafického akcelerátoru CUDA, nebo možnost nastavení rozložení zátěže na jednotlivá jádra CPU. Zároveň v průběhu vývoje přebírá užitečné vlastnosti z jiných IDS nástrojů. Například dostupný skriptovací engine `LuaJIT` umožňuje vytvoření flexibilního uživatelského skriptu, který se spouští při přijetí každého paketu a který slouží k popsání složitějších detekčních pravidel. Skripty se dají použít také při logování k definování formátu výstupu [22, 45].



Obrázek 2.4: *Zřetězené zpracování paketu v sytému Suricata.*

Na obrázku 2.4 je zjednodušeně znázorněna část zřetězeného vícevláknového zpracování paketů v systému Suricata. Zpracování probíhá ve čtyřech krocích: Zachycení paketu ze síťového rozhraní (capture method). Tato funkce je implementovaná ve více modulech, které podporují různá rozhraní pro záchyt paketu např. `pcap`, `Netmap` nebo `PFRING`. Analýza a dekódování hlaviček (decode threads). Sestavení síťového toku (stream threads). Analýza dle definovaných pravidel a generování hlášení (detect threads).

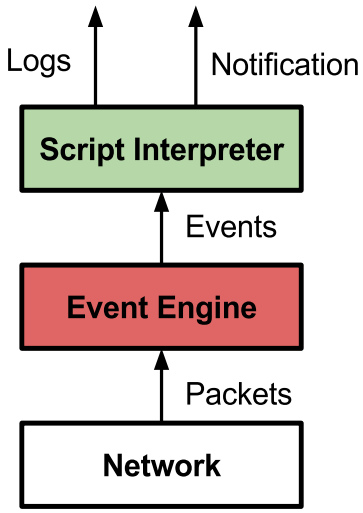
Funkcionalitu Suricaty lze dále rozšiřovat pomocí uživatelských modulů. S pomocí modulů lze přidávat např. nová rozhraní pro záchyt paketů, podporu nových protokolů nebo lze definovat vlastní funkci, která se vykoná nad každým přijatým paketem.

Díky vícevláknovému zpracování a vysokému výkonu se Suricata z vybraných IDS (Snort, Bro) nejvíce hodí k analýze vysokorychlostní sítě. Podle [26] má Suricata při využití vícevláknového zpracování výrazně vyšší přesnost detekce incidentů než Snort, i když za cenu mírně vyšších nároků na CPU. Na rozdíl od nezávislých instancí Snortu je totiž třeba řešit synchronizaci a komunikaci mezi vlákny, což představuje určitou režii. Díky vyšší propustnosti při vícevláknovém zpracování je také vyšší pravděpodobnost detekce incidentu. Při zahlcení systému jsou totiž pakety nekontrolovatelně zahazovány, což se při vysokých rychlostech u jednovláknového zpracování děje častěji.

2.2.3 Bro

The Bro Network Security Monitor, zkráceně Bro, je open-source nástroj pro analýzu síťového provozu. Používá se pro síťová měření, forenzní analýzu, testování, ale i jako IDS. Bro nabízí uživateli framework, jenž je svým přístupem k analýze zcela odlišný od typických IDS. Detekční pravidla jsou zapsaná ve speciálním jazyce `Bro's scripting language`. Tento interpretovaný typovaný jazyk tak nabízí při psaní uživatelských skriptů a detekčních metod daleko větší flexibilitu než pravidla používaná pro Snort a Suricatu. K extrakci hlaviček paketů lze využít zabudované high-level funkce, které pokrývají analýzu jak aplikačních, tak i nižších vrstev. K dispozici jsou i další specializované funkce umožňující např. měření výkonnosti detekčního systému, či framework a knihovny pro komunikaci s externími nástroji [24, 46, 28, 37].

Základní architektura systému Bro je znázorněna na obrázku 2.5. Bro se skládá ze dvou hlavních komponent – Event Engine a Script Interpreter. Event Engine provádí analýzu síťového toku (Network) zahrnující především extrakci položek z hlaviček paketů. Na základě analýzy se následně vytváří tzv. události (Events). Komponenta Script Interpreter provádí správu sady skriptů napsaných v jazyce Bro a spustí obsluhu události, pokud je v rámci skriptu pro danou událost zaregistrován odpovídající event-handler. Provedenou akcí může být vytvoření notifikace (Notification), uložení informací do logu (Logs), výpočet statistik či spuštění jiné externí aplikace. Skripty rozšiřují funkcionalitu systému stejně jako u modulárního programování [28].



Obrázek 2.5: *Hlavní architektura Bro.*

Systém Bro nepodporuje nativní vícevláknové zpracování, avšak nabízí režim pro rozdělení zátěže mezi procesorová jádra či dokonce mezi fyzická zařízení. Režim se nazývá Bro-Cluster a celý tento systém lze ovládat pomocí nástroje BroControl. Dokument [44] popisuje návrh, jak využít tento systém pro analýzu sítě s rychlostí až 50 Gb/s. Základní myšlenkou je rozložit datový tok na menší části a související úseky síťové komunikace rozesílat jednotlivým analyzačním procesům běžícím na více zařízeních. K tomu jsou potřeba dvě komponenty. První pro rozložení, agregování a filtrování síťového provozu pro jednotlivá zařízení (servery). Druhá komponenta nasazená na každém zařízení rozdělující síťová data pro více běžících procesů Bro. Podle dokumentu běžel tento systém celkově na pěti serverech.

Kapitola 3

Softwarově definované monitorování

Obsah této kapitoly vychází z prací [32, 30, 31]. Softwarově definované monitorování (Software Defined Monitoring, SDM) je systém pro hardwarovou akceleraci síťových aplikací sloužících k bezpečnostnímu monitorování počítačové sítě. SDM je schopno sledovat jednotlivé síťové toky[1] a na nich aplikovat softwarem definované akce. Hardwarová část provádí určité předzpracování příchozích dat na úrovni jednotlivých toků a uživatelské aplikaci běžící v softwaru tímto poskytuje jemnou softwarově řízenou ztrátu příchozích informací ze sítě. Díky tomuto přístupu umožňuje SDM dosažení vysoké propustnosti zpracování příchozích síťových dat připojených linek až do celkové rychlosti 100 Gb/s.

Základní předzpracování síťových dat na akcelerační síťové kartě má stěžejní význam při monitorování vysokorychlostních sítí. V případě standardního přístupu k monitorování, kterým je použití serverové platformy a běžné síťové karty, probíhá veškeré zpracování paketů plně softwarově v rámci operační paměti. Tento model umožňuje monitorovat sítě s přenosovou rychlostí do 10 Gb/s. Při použití stejného přístupu na vysokorychlostní síti s rychlostí 100 Gb/s a více vznikají problémy s výkonností. Veškerý zachycený provoz se totiž posílá skrze sběrnici do operační paměti počítače, kde jsou pakety následně zpracovávány procesorem. Zde vzniká problém kvůli vytížení sběrnice PCI Express a také kvůli procesoru, který se musí zabývat jak parsováním paketu, tak pokročilými operacemi uživatelské aplikace. Zatížení sběrnice lze v tomto případě zlepšit, pokud by přes ni posílaly data jen nezbytně nutná pro monitorování. Ve většině případů totiž stačí k monitorování informace nacházející se v záhlaví paketů. Těchto informací je v porovnání s celým paketem výrazně méně. Data je tedy vhodné předzpracovat, čímž se zmenší velikost přeposílaných dat přes sběrnici. Díky předzpracování síťových dat v hardwaru se zároveň uvolní softwarové prostředky. Z tohoto poznatku částečně vychází i princip SDM. Provedením extrakce údajů ze záhlaví přímo na hardwarové akcelerační síťové kartě se totiž může přes systémovou sběrnici přenášet daleko menší množství dat. Procesor se zbaví zátěže při analýze paketů a extrakci údajů a rovnou pracuje nad užitečnými daty. Lze přitom pokračovat dále a do hardwaru přesunout např. i provádění agregací a výpočtů spojených s NetFlow statistikami o celém síťovém toku.

[1] Síťový tok je posloupnost paketů majících společnou vlastnost a zároveň procházejích určitým bodem pozorování za daný časový interval. Společnými vlastnostmi se zde míní např. zdrojová a cílová IP adresa, zdrojový a cílový port, typ transportního protokolu [25].

Hlavním a nejdůležitějším rysem SDM je však softwarem řízená kontrolovaná ztráta příchozích informací na úrovni síťových toků. Průběh činnosti SDM je následující. Neznámé toky jsou ve výchozím stavu odesílány k softwarovému zpracování. Softwarová bezpečnostní aplikace zpracuje příchozí síťová data a rozhodne, jakým způsobem má hardware předzpracovávat další pakety tohoto toku. Pošle tedy požadavek skrze softwarový řadič do hardwaru v podobě příslušného pravidla pro daný síťový tok. Následující příchozí pakety uvedeného toku jsou pak do softwaru odeslány patřičně předzpracované. Možnosti předzpracování jsou zkrácení paketu, extrakce informací ze záhlaví paketů, nebo agregace NetFlow statistik k celému toku. Pokud bezpečnostní aplikace usoudí, že je nutné dále detailněji analyzovat síťový tok, pravidlo nepošle a nechá hardware nadále odesílat celé pakety. Naopak pokud usoudí, že není potřeba se síťovému toku dále věnovat, pošle pravidlo informující hardware, aby další pakety příslušného síťového toku nepřeposílal (zahazoval). Právě díky možnosti dynamicky zahazovat zbylé části síťového toku v hardwaru mohou být softwarové prostředky efektivně využity pro detailnější a pokročilejší analýzu vybraných síťových toků až do úrovně aplikačního protokolu (HTTP, FTP, ...).

Měření síťového provozu v [30] přitom ukazuje, že i malý počet těch nejtěžších toků přenáší vzhledem k ostatním tokům velmi vysoké množství všech přijímaných paketů (jedná se o tzv. heavy-tailed rozložení), takže i relativně malý počet vložených pravidel ovlivní zpracování 80 % až 90 % všech přijímaných paketů.

SDM se skládá ze dvou částí. Firmwarová část provádí jednotlivé typy předzpracování na akcelerační síťové kartě při plné rychlosti linky (tzv. wire-speed). Softwarová část potom řídí firmware a slouží jako prostředník při komunikaci softwarové aplikace s firmwarem.

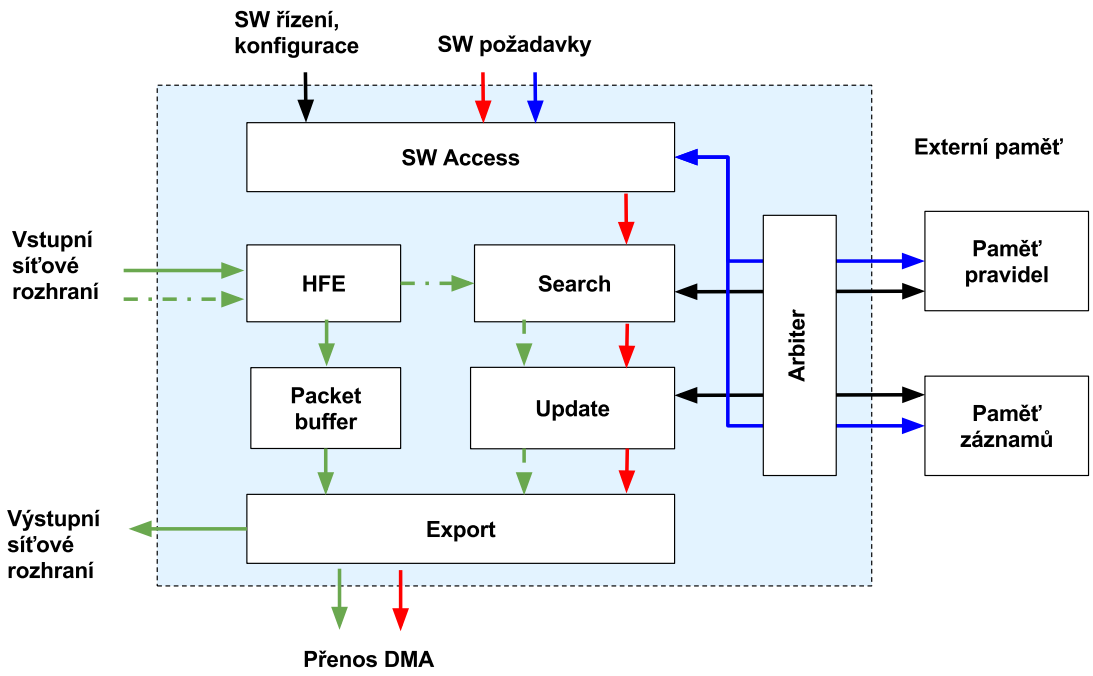
3.1 Firmwarová část systému

Firmware SDM je implementován na akcelerační síťové kartě osazené čipem FPGA. Programovatelná hradlová pole (Field-programmable Gate Array) jsou rekonfigurovatelné prefabrikované čipy, které se skládají z obecných programovatelných bloků a konfigurovatelných propojení mezi nimi. Po naprogramování realizují různé uživatelsky definované kombinační a sekvenční logické obvody. Hlavní výhodou této technologie je rychlý návrh a vývoj aplikace, kterou lze díky rekonfiguračním vlastnostem FPGA později pohotově upravovat a opravovat tak odhalené chyby v návrhu obvodu. Funkce obvodu jsou popisovány prostřednictvím jazyků pro popis hardware (Hardware Description Language, `HDL`) např. `VHDL` nebo `Verilog`.

Firmwarová část SDM je implementována v jazyce `VHDL` nad platformou NetCOPE, která poskytuje jednotné rozhraní nad hardwarovými prvky síťové karty. Díky využití této platformy je firmware SDM snadno přenositelný mezi různými akceleračními kartami z rodiny COMBO, protože komunikační rozhraní definované infrastrukturou NetCOPE je vždy stejné. Platforma tak poskytuje jádru SDM vstupní, výstupní síťová rozhraní a rozhraní k externí paměti, která je umístěná na síťové kartě.

Schéma architektury firmwaru SDM je uvedeno na obrázku 3.1, kde světle modrá ohraničená oblast zobrazuje samotné jádro firmwaru. Jádro SDM zpracovává dva druhy požadavků – požadavky vzniklé na základě příchodu paketu na vstupní síťové rozhraní (zelená šipka) a požadavky softwarového řadiče (červená šipka).

Přes vstupní síťové rozhraní jsou přijímány příchozí pakety. V rámci platformy NetCOPE nad nimi přitom byla už dříve provedena řada kontrol (ověření limitů velikosti rámce, kontrola CRC a MAC adresy). K těmto paketům jsou přenášeny také odpovídající metadata o příjmu (přesná časová značka příchodu rámce, identifikace fyzického vstupního rozhraní,



Obrázek 3.1: *Schéma architektury firmwaru SDM.*

velikost příchozích dat). Počáteční zpracování dat ze vstupního síťového rozhraní probíhá v jednotce HFE (Header Field Extractor). Pakety se v této jednotce analyzují. Po extrahování důležitých částí se extrahovaná data předávají k dalšímu zpracování jednotce Search (přerušovaná zelená šipka) a zároveň se původní analyzovaný paket odloží do FIFO paměti (Packet Buffer). Jednotka Search podle identifikace síťového toku vyhledá příslušné pravidlo v externí paměti (černá šipka). Pokud je pravidlo nalezeno, je předáno spolu s extrahovanými daty jednotce Update, která provede operaci podle instrukce obsažené v pravidlu. Operace aktualizuje záznam o příslušném síťovém toku v externí paměti. To v případě Net Flow zahrnuje zvýšení čítače paketů a bytů, aktualizaci koncové časové značky a logického součtu TCP příznaků. Extrahovaná data a pravidlo jsou předány do bloku Export. Ten vyzvedne původní přijatý paket z paměti FIFO a podle přeposlaného pravidla rozhodne, zda má paket zkrátit a odeslat do softwaru, nebo paket odeslat celý. Jednotka Export je také schopna zaslat do softwaru pouze extrahovaná data ze záhlaví. Původně přijatý paket může být přitom souběžně poslán na některá z výstupních síťových rozhraní. Pokud jednotka Search pravidlo nenajde, z bloku Export se odešle do softwaru paket celý, aby mohl software učinit svou analýzu a případně vytvořit nové pravidlo.

Jednotka SW Access na základě požadavků od softwarového řadiče spravuje pravidla v externí paměti (modrá šipka). Zpracovávané příkazy od softwarového řadiče jsou přidání nebo odebrání pravidla a příkaz exportu záznamu. Přidávání a odebírání pravidel z paměti se přitom provádí atomicky a neovlivňuje vyhledávání prováděné jednotkou Search. Správa paměti je řešena formou hašovací tabulky, která využívá algoritmu kukaččího hašování. Kolize zapříčiněná vložením pravidla se vždy řeší částečnou reorganizací položek v tabulce. Požadavky na exportování záznamu od softwarového řadiče do softwaru mají několik variant: exportování záznamu, exportování záznamu se současným vynulováním čítačů, či odstranění pravidla. Odstraněním pravidla se zároveň provede vynulování záznamu.

Odesílání dat do softwaru, ať už je to celý paket nebo záznam o toku, se realizuje přímým přístupem do operační paměti (DMA) a využívá se přitom prostředků platformy NetCOPE.

3.2 Softwarová část systému

Řadič SDM představuje hlavní řídící prvek celého systému. Řadič na základě požadavků od uživatelské aplikace zadává firmwaru pokyny pro předzpracování jednotlivých síťových toků z vysokorychlostní sítě. Uživatelským aplikacím přitom poskytuje jednoduché rozhraní pro ovládání a komunikaci s firmwarem. Aplikace má možnost si prostřednictvím řadiče určit, do jaké míry chce daný síťový tok předzpracovat. Na výběr má tyto možnosti předzpracování:

Žádné — přijímané rámce jsou k softwarovému zpracování uživatelskou aplikací posílány celé, zcela beze změny. Tato varianta je ve většině případů také výchozí možností pro zpracování neznámých síťových toků.

Částečné — k softwarovému zpracování uživatelskou aplikací jsou přeposílané už extrahované informace z hlaviček rámců ve formě tzv. unifikovaných hlaviček (UH).

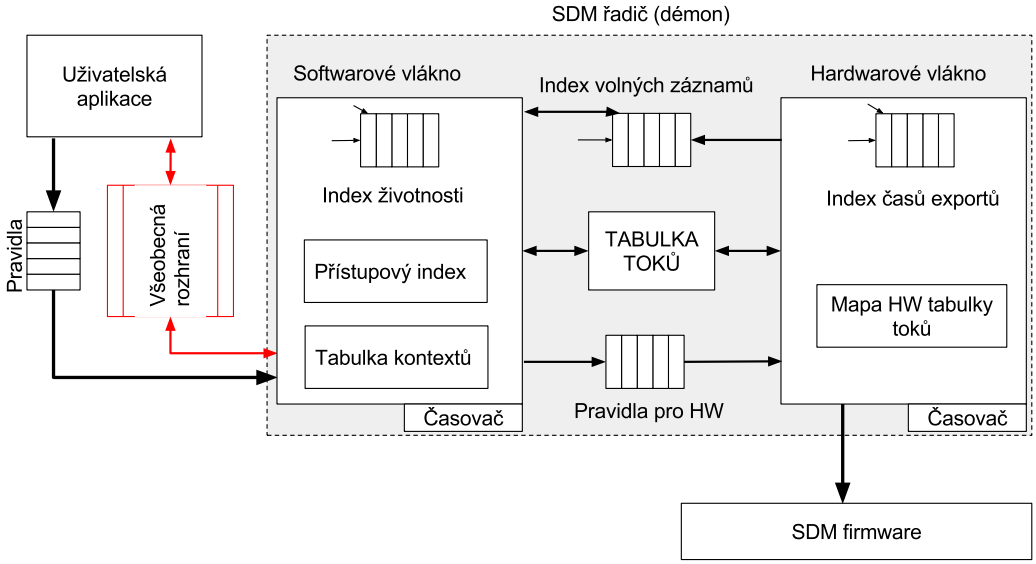
Úplné — informace o rámcích jsou firmwarem úplně agregovány do záznamu o toku. Například formát záznamu NetFlow obsahuje čítač počtu paketů, čítač počtu bytů, časovou značku začátku a konce síťového toku a logický součet TCP příznaků.

Zahazování — pakety daného síťového toku se v rámci firmware zcela zahazují, protože z pohledu uživatelské aplikace už nejsou pro další analýzu zajímavé.

Samotný přenos dat mezi hardwarem a softwarem uskutečňují ovladače a knihovny platformy NetCOPE, nad kterou řadič funguje. Bez vědomí uživatelské aplikace řadič dále automaticky vykonává režijní operace nad firmwarem. Z počátku řídí inicializaci pravidel, poté do jisté míry také kontroluje proces samotného přidávání, odebírání a dalších změn konkrétních pravidel ve firmwarové části systému. Řadič dále zajišťuje pravidelné exportování záznamů o tocích. V případě úplného předzpracování příchozích dat odkazuje každé pravidlo na adresu konkrétního záznamu o toku, ke kterému se vztahuje. Řadič proto musí sledovat také obsazení těchto záznamů ve firmwarové části v externí paměti, protože při přidávání nových pravidel potřebuje vědět, které záznamy jsou aktuálně volné a mohou být znovu použity. Z tohoto důvodu si řadič uchovává také informace o obsazenosti paměti záznamů.

Řadič SDM je schopen obsluhovat více softwarových aplikací najednou. Z tohoto důvodu umožňuje také agregovat jejich požadavky. Například pokud první softwarová aplikace žádá informace o síťovém toku ve formě unifikovaných hlaviček (UH) a druhá aplikace požaduje pakety toku zcela zahazovat, výsledkem agregace potom bude přeposílání UH.

Pokud má mít aplikace možnost vytvářet nová pravidla pro SDM, je třeba tuto aplikaci u řadiče nejprve zaregistrovat jako zdroj pravidel. Každá aplikace si při komunikaci s řadičem vytvoří takzvaný kontext, ke kterému přiřadí množinu neměnných pravidel definující základní chování pro nové neznámé síťové toky. Kontexty jednotlivých aplikací se od sebe liší jednoznačným identifikátorem. Aplikace registrované jako zdroje pravidel potom mají možnost přidávat či rušit pravidla v existujících kontextech. Řadič pravidla přijímá, udržuje, agreguje a následně aplikuje do firmwaru. Řadič po vypršení určitého času od zavedení pravidla zaručuje zneplatnění pravidla. Pravidlo však může zneplatnit také sama aplikace.



Obrázek 3.2: *Schéma řadiče SDM.*

Řadič je implementován formou démona běžícího na pozadí. Uživatelské aplikace s démonem komunikují přes specializovanou knihovnu `libsdm`, která využívá prostředky meziprocesové komunikace poskytované operačním systémem. Schéma SDM řadiče je na obrázku 3.2. Na schématu jsou zobrazena dvě vlákna provádějící činnost zpracování požadavků. Softwarové vlákno (levá část) se stará o příjem, zpracování a agregaci pravidel či jiných příkazů od uživatelské aplikace. Hardwarové vlákno (pravá část) se potom stará o komunikaci s firmwarem. Zařizuje konfiguraci, nahrávání anebo rušení pravidel ve firmwaru. Po vypršení časovače dále pravidelně vybavuje export záznamů o tocích. Mezi vlákny je umístěna tabulka toků (uprostřed), v níž jsou uložené záznamy o všech tocích, které v dané chvíli řadič spravuje. Jedná se o toky, u kterých je aktuálně ve firmwaru nahrané příslušné pravidlo. Zároveň se v tabulce uchovávají toky, u kterých se sleduje splnění určité podmínky (např. překročení meze počtu paketů), než může být toto pravidlo vloženo do firmwaru. Tabulka toků je realizovaná jednoduchým polem. Pro rychlé hledání v tabulce se spravuje několik typů indexů:

Přístupový index se získá z identifikátoru toku a použije se k vyhledání onoho toku.

Index volných záznamů ukazuje na neobsazené místo v tabulce toků.

Index životnosti umožňuje přístup k záznamům, kterým vypršela doba platnosti.

Index časů exportu se použije pro export záznamů daného toku z firmwaru.

V případě přidávání pravidla je postup následující. Aplikace připraví odpovídající parametry a zavolá funkci z knihovny `libsdm`. Požadavek na přidání pravidla se vloží do fronty a čeká na vyjmutí softwarovým vláknem. Softwarové vlákno po vyjmutí nejprve použije tabulku kontextu k vytvoření základní struktury záznamu, jenž se použije v tabulce toků. Potom pomocí přístupového indexu ověří, zda už je záznam přítomný v tabulce toků. Pokud není, využije se indexu volných záznamů k alokaci nového záznamu. K nově vytvořenému záznamu je vytvořen index životnosti a uložen v seznamu. Dále se záznam vloží

do tabulky toků a uloží se jeho odpovídající vyhledávací index. Jestliže se má pravidlo odeslat do firmwaru, dá o tom softwarové vlákno vědět hardwarovému vláknu. Hardwarové vlákno následně odešle pravidlo do firmwaru a ohlídá si, že bylo firmwarem přijato. Jestliže vkládané pravidlo vyžaduje pozdější exportování záznamu, uloží se navíc informace v mapě HW tabulky toků a zabezpečení pravidelného exportu se dovrší uložením indexu času exportu.

Kapitola 4

Konceptuální návrh

V předchozích kapitolách 2 a 3 byly charakterizovány systémy pro detekci bezpečnostních hrozeb v síti a byl vysvětlen princip softwarově definovaného monitorování. Spolupráce těchto dvou systémů by měla umožnit zvýšení celkové výkonnosti IDS aplikací a otevřít tak cestu k detailnější analýze síťového provozu při vyšších rychlostech.

Základní myšlenka využití systému SDM k akceleraci IDS aplikací vychází z následujícího předpokladu. Pokud by systémům IDS postačovalo k detekování většiny incidentů analyzovat vždy pouze prvních n paketů každého síťového toku, bylo by možné velmi efektivně využít právě systém SDM ke zpracování (respektive zahození) zbývající části toku, která by tak z hlediska detekce incidentů nebyla relevantní. Tím by se systém IDS oprostil od nutnosti každý síťový tok dále detailně analyzovat, čímž by se výrazně snížila zátěž celého systému. Z bližšího pohledu by tento přístup vedl především ke snížení vytížení CPU. Při využití akcelerační síťové karty s podporou konceptu SDM, kde lze zahození zbývajících paketů toku realizovat v hardwaru, by se tento přístup kladně projevil i na zatížení systémové sběrnice PCI Express. Tím pádem by se celkově zvýšila maximální propustnost systému a umožnilo by to tak jeho nasazení na rychlejší síti. Díky využití konceptu SDM se tak v případě zahlcení systému předejde nekontrolované ztrátě příchozích dat, což povede ke zvýšení úspěšnosti detekce bezpečnostních incidentů. Tato kapitola si v dalších částech klade za cíl zjistit především následující:

- Jak velké jsou síťové toky přenášené na vysokorychlostní síti. Velikostí síťového toku se zde přitom míní počet paketů daného toku.
- Jaký je podíl objemu provozu těžkých síťových toků vzhledem k celkovému objemu přenášených dat (paketů) na vysokorychlostní síti.
- Kolik paketů ze začátku síťového toku stačí analyzovat k úspěšné detekci incidentu, respektive kolik paketů postačuje analyzovat k detekci stejného počtu událostí jako v případě analýzy celého síťového toku.

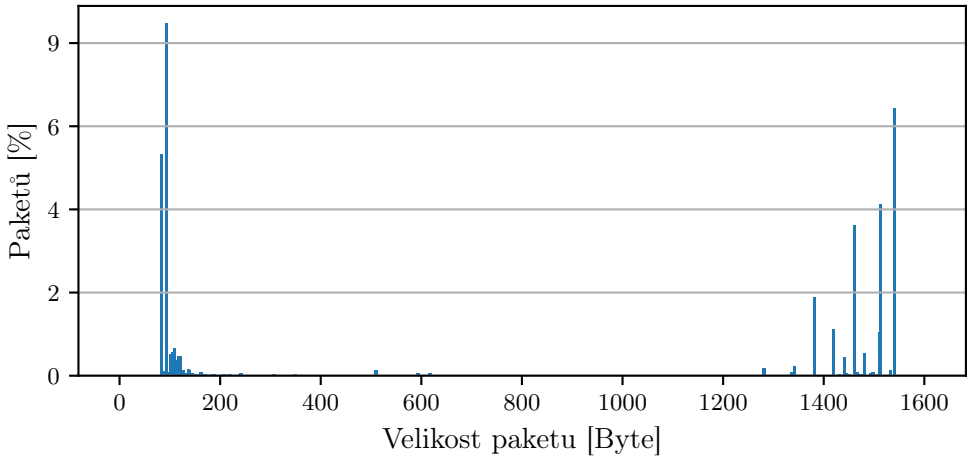
Kapitola se skládá ze dvou částí. První část obsahuje výsledky měření charakteristických vlastností vysokorychlostní sítě se zaměřením na délku síťových toků (ve smyslu počtu paketů) a objemové vlastnosti nejtěžších toků na síti. Tyto vlastnosti jsou totiž velmi důležité z hlediska využitelnosti konceptu SDM. Druhá část kapitoly se následně zabývá analýzou síťových toků z pohledu detekce incidentů. Cílem je ověření počátečního předpokladu, že pro úspěšnou detekci většiny incidentů pomocí IDS stačí zpracovat vždy pouze prvních

n paketů každého síťového toku. Jsou zde tak prezentovány výsledky analýzy nad zachyceným provozem z vysokorychlostní sítě, která zjišťuje právě množství detekovaných incidentů v závislosti na počtu paketů, které jsou na začátku každého síťového toku analyzovány.

4.1 Charakteristické vlastnosti na síti

Cílem měření je ověření charakteristických vlastností síťového provozu, které umožňují využití konceptu softwarově definovaného monitorování k akceleraci IDS aplikací. Měření bylo prováděno na akademické síti CESNET2 [13] na optické lince mezi Brnem a Vídní, která spojuje síť sdružení CESNET s rakouskou sítí ACOnet [15]. Přenosová kapacita linky je 2 x 10 Gb/s v obou směrech. Jednotlivá měření byla provedena v průběhu pracovních dnů v týdnu od 3. do 5. dubna 2017 v době od 8:00 do 17:00 hodin. To znamená v čase, kdy je linka nejvíce vytížená. Samotné měření bylo realizováno nástrojem `FlowMon Exporter` od společnosti Flowmon Networks [16]. Architektura tohoto nástroje přináší možnost přidání vlastních pluginů pro získání vstupních síťových dat, vlastního způsobu jejich zpracování a exportu vytvořených záznamů o tocích. FlowMon Exporter interně vytváří a spravuje záznamy o tocích, jejichž informace lze v uživatelských pluginech využít. Pro potřeby měření vlastností sítě byl tak vytvořen vlastní plugin. Při implementaci byla využita funkce histogramu z knihovny `GSL` (GNU Scientific Library) [17]. Výsledná výstupní data z pluginu byla potom zpracována vlastním skriptem v programovacím jazyce `Python` a zobrazena v grafech pomocí knihovny `matplotlib`.

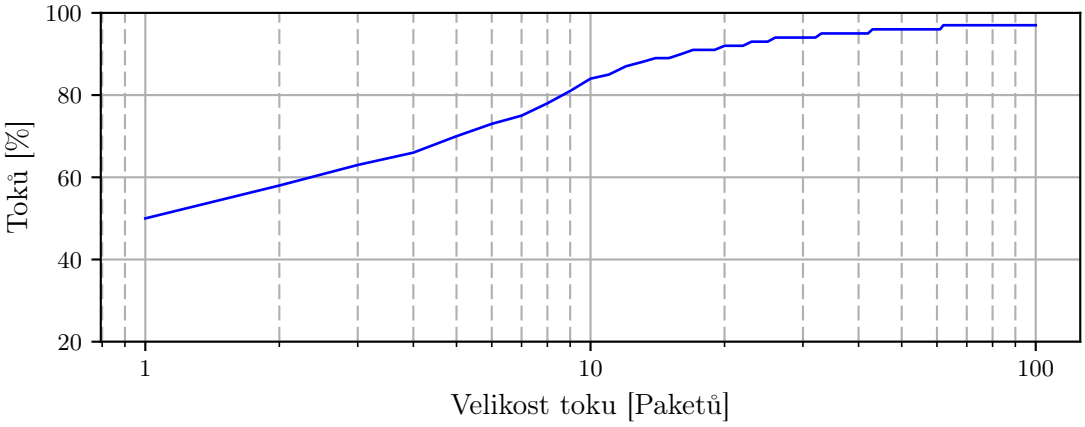
Graf na obrázku 4.1 nabízí pohled na distribuci délek paketů v testované síti. Na ose x je v histogramu vynesena velikost paketu v bytech. Na ose y je potom procentuální četnost paketů odpovídající délky. IP datagram je přenášen prostřednictvím ethernetového rámce, jehož datové pole nabývá velikosti v rozsahu od 46 do 1500 bytů. Na první pohled lze v grafu vidět, že dominantní délky paketů se pohybují kolem hodnot 100 a 1500 bytů.



Obrázek 4.1: *Rozložení hustoty výskytu paketu podle velikosti.*

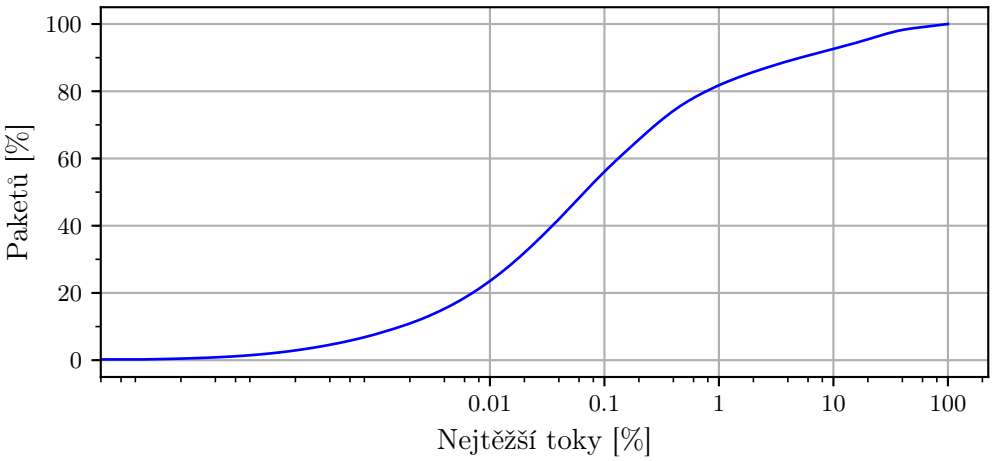
V grafu na obrázku 4.2 je dále zobrazena distribuční funkce velikosti toků na měřené lince. Na ose x je znázorněna velikost síťového toku v podobě celkového počtu paketů. Osa y potom vyjadřuje procentuální podíl z celkového počtu síťových toků na síti. Měřítko na ose x je přitom logaritmické. Z grafu vyplývá, že polovina všech zachycených toků je

tvořená jediným paketem. Necelá pětina toků na síti má potom počet paketů větší než deset.



Obrázek 4.2: *Distribuční funkce velikosti toků.*

V grafu na obrázku 4.3 je také znázorněna distribuční funkce se zaměřením na těžké toky. Osa x vyjadřuje, kolik procent těžkých toků se podílí na příjmu všech paketů. Osa y potom vyjadřuje procentuální podíl paketů z celkového počtu přijatých paketů. Měřítko na ose x je přitom logaritmické. Z výsledků měření vyplývá, že jen jedno procento nejtěžších toků obsahuje až 80 % všech přijatých paketů.



Obrázek 4.3: *Heavy-tailed rozložení počtu paketů v toků.*

Uvedené výsledky měření ukazují výhody využití konceptu SDM. Pokud by se například k analýze předávalo systému IDS vždy jen prvních dvacet paketů ze začátku každého síťového toku, bylo by plně analyzováno 90 % všech síťových toků, protože právě takový podíl toků nepřesahuje délku dvaceti paketů (viz graf na obrázku 4.2). Zbylých 10 % síťových toků by se potom pokládalo za toky těžké, neboť tyto toky mají délku větší než dvacet paketů. Po dvaceti paketech by se další pakety těžkých toků zpracovávaly zcela autonomně pouze v hardwaru. Tímto způsobem by tak hardware systému SDM zpracoval (respektive zahodil) až 90 % z celkového počtu příchozích paketů (viz graf na obrázku 4.3).

Naopak detailní softwarové analýze v rámci systému IDS by bylo podrobeno jen 10 % všech paketů. Selektivním analyzováním pouze desetiny z celkového množství příchozích paketů se tak výrazně zvýší propustnost systému, protože se významnou měrou snižuje zatížení CPU i zatížení systémové sběrnice. Z hlediska charakteru síťového provozu, tzn. rozložení velikosti a objemových vlastností toků na síti, se tak jednoznačně potvrzují výhody použití konceptu SDM. Aby mohl celý takto navržený systém spolehlivě fungovat, musí se ještě ověřit počáteční předpoklad, to že systému IDS k úspěšnému detekování většiny incidentů stačí analyzovat např. jen prvních dvacet paketů každého síťového toku. Toto je úkolem následující části kapitoly.

4.2 Analýza síťových toků z pohledu detekce incidentů

Cílem této podkapitoly je ukázat, že pro úspěšnou detekci většiny incidentu v síťovém provozu stačí analyzovat vždy jen prvních n paketů každého síťového toku.

Na stejné testované síti, na které probíhala analýza základních charakteristik síťového provozu v předchozí kapitole 4.1, byl proveden záchyt síťového provozu do souboru `pcap`. Záchyt síťových dat probíhal přibližně 15 minut a na lince s průměrnou rychlostí 86 094 paketů za sekundu tak bylo uloženo celkově 77 484 934 paketů s průměrnou velikostí 724 bytů o celkovém objemu 55 GB dat. Dohromady bylo zachyceno 1 944 321 síťových toků. Soubor `pcap` musel být dále předzpracován, protože oba použité systémy Suricata i Snort neumí zpracovat zapouzdření dat do protokolu TRILL, který se na síti CESNET2 běžně vyskytuje. Před navazujícími experimenty a měřením se proto musely rámce zapouzdřené tímto protokolem extrahovat.

Jako zdroj pravidel pro měření byla zvolena databáze EmergingThreats (ET) [39]. Pro samotnou analýzu zachycených dat byl potom vybrán systém Suricata, protože na rozdíl od systému Snort dosahuje díky vícevláknového zpracování vyšší výkonnosti (viz kapitola 2.2.2). Systém Suricata umožňuje provést offline analýzu `pcap` souboru, ve kterém je síťový provoz uložen. Díky tomu lze určit celkový očekávaný počet detekovaných incidentů pro následující experimenty. Pro měření byly v konfiguračním souboru dále nastaveny proměnné `HOME_NET` a `EXTERNAL_NET` na hodnotu `any`. V rámci tohoto nastavení se tak nijak neomezuje rozsah IP adres a tím i počet toků, nad kterými je prováděna vlastní detekce. Je tak detekováno mnohem více incidentů. Na druhou stranu tím dochází k výrazně vyššímu zatížení systému, protože se podmínka v hlavičce pravidla snáze splní, tudíž se mnohem častěji aplikuje obsah samotného pravidla. Tento režim konfigurace bude s výhodou použit i dále (viz kapitola 6) v rámci provádění online detekce v reálném čase.

Pro zjištění, kolik paketů ze začátku síťového toku stačí k úspěšnému detekování hrozby, jsem ve frameworku `PcapPlusPlus` [3] vytvořil pomocný program. Ten přečte vstupní `pcap` soubor se zachyceným síťovým provozem a podle parametrů vytvoří nový soubor, ve kterém ponechá vždy jen prvních n paketů každého síťového toku. Zbývající pakety daného toku se neuloží. Vzniklé `pcap` soubory s různým počtem paketů v síťových tocích jsem následně podrobil offline analýze systémem Suricata.

Výsledky provedené analýzy jsou shrnuty v tabulce 4.1. V prvním sloupci *Typ incidentu* se vyskytují skupiny incidentů, jenž se týkají určité bezpečnostní oblasti (DNS, DOS, TROJAN). V detekčním pravidlu se tato informace vyskytuje v rámci klíčového slova `msg`. Ve druhém sloupci *Průměr* jsou k typům incidentů doplněny údaje o průměrném počtu paketů, které je nutné analyzovat, aby se dotyčný typ incidentu detekoval. Sloupec *Maximum* popisuje, kolik je potřeba k danému typu incidentu analyzovat paketů, aby se všechny incidenty detekovaly. Sloupec *Priorita* v tabulce vypovídá o míře závažnosti daného typu

detekovaného incidentu. Prioritu lze přitom v rámci pravidla zadat explicitně nebo hodnota vychází z nastavení proměnné `classtype` [14].

Tabulka 4.1: *Agregace vybraných detekovaných incidentů.*

Typ incidentu	Průměr	Maximum	Priorita
ET CINS	1	1	střední
ET CNC	1	1	vysoká
ET DNS	4	4	vysoká
ET DOS	1	1	střední
ET MALWARE	7	9	vysoká
ET P2P	5,5	105	vysoká
ET POLICY	10,7	80	střední
ET SCAN	1,9	4	střední
ET TFTP	29,4	65	vysoká
ET TROJAN	3	5	vysoká

Následuje obecný popis jednotlivých skupin vybraných incidentů uvedených v tabulce 4.1 a příklady konkrétních útoků, které Suricata v zachyceném síťovém provozu našla.

ET CINS: Tyto pravidla pocházejí z organizace CINS [21]. CINS zaznamenává nedůvěryhodné či nechvalně známé IP adresy, jejichž komunikace by měla být blokována. K detekci stačilo analyzovat jeden paket v každém síťovém toku.

ET CNC: Pravidla detekují komunikaci webových robotů nebo možnou nákazu malwarem vyznačující se komunikací s C&C serverem [19]. Pravidla vycházejí z veřejně známých nebezpečných C&C serverů a charakteristik komunikace známého malware [8]. K detekci stačilo analyzovat také jediný paket.

ET DNS: Jedná se o pravidla detailně analyzující protokol DNS. Konkrétním příkladem detekce může být neobvyklé nastavení příznaků v hlavičce DNS paketu [10]. Průměrně je potřeba analyzovat čtyři pakety v každém toku.

ET DOS: Detekce DoS útoků [9], konkrétně amplifikačního útoku využívajícího zranitelnosti protokolu NTP [18] pro synchronizaci času. Opět jde incident rozpoznat v prvním paketu.

ET MALWARE: Skupina pravidel, která slouží pro detekci napadení koncové stanice malwarem. Příkladem může být detekce malware, který využívá chyby v prohlížeči Internet Explorer. Průměrně bylo třeba k detekci analyzovat prvních sedm paketů v tocích.

ET P2P: Detekování nežádoucího spojení klient-klient. Průměrně bylo třeba analyzovat prvních šest paketů v tocích.

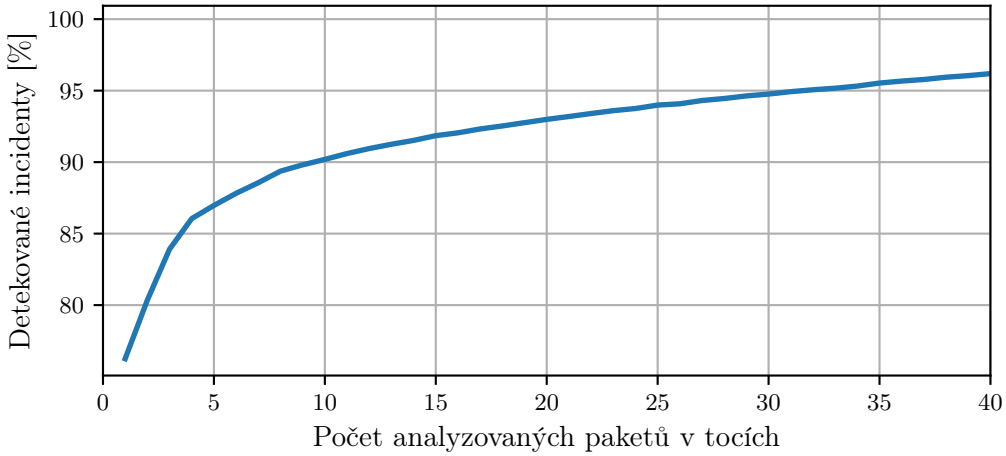
ET POLICY: Detekování bezpečnostní slabiny na straně uživatelské aplikace. Příkladem je detekce zastaralé a zranitelná verze softwaru Java 1.8.x. Průměrně bylo nutné analyzovat jedenáct paketů.

ET SCAN: Detekce skenování systému. Příkladem takového pravidla je také detekce útoku hádání hesel (výchozích hesel) pro přístup k SSH. Průměrně stačilo k detekování incidentu analyzovat první dva pakety.

ET TFTP: Analýza protokolu TFTP. Detekovaný incident se týká čtecích požadavků. Průměrně bylo potřeba analyzovat až třicet paketů v každém síťovém toku.

ET TROJAN: Detekce síťového trojského koně. Příkladem může být odpověď DNS severu informujícím o nealokovaném adresovém prostoru. Pro detekci incidentu stačilo v tocích analyzovat první tři pakety.

Graf na obrázku 4.4 se zaměřuje na počet všech detekovaných incidentů bez ohledu na jejich závažnost, které Suricata v zachyceném síťovém provozu úspěšně nahlásila. Graf ukazuje distribuční funkci počtu detekovaných incidentů vzhledem k počtu analyzovaných paketů. Osa x vyjadřuje právě počet analyzovaných paketů ze začátku síťového toku. Osa y potom vyjadřuje procentuální podíl úspěšně detekovaných incidentů při analýze jen dané části síťového toku. Z grafu je patrné, že např. pro detekování 90 % všech incidentů stačí analyzovat pouze prvních deset paketů v každém síťovém toku, což jen potvrzuje počáteční předpoklad pro úspěšnou akceleraci IDS aplikací prostřednictvím systému SDM.



Obrázek 4.4: *Distribuční funkce počtu detekovaných incidentů.*

Tato kapitola věnující se konceptuálnímu návrhu shrnula důležité charakteristické vlastnosti sítě a výsledky analýzy síťových toků z pohledu detekce incidentů. Grafy na obrázcích 4.2, 4.3 a 4.4 ukázaly, že kombinace IDS s konceptem SDM přinese IDS aplikacím možnost zvýšení celkové výkonnosti. Díky nasazení systému SDM se tak sníží zátěž CPU a systémové sběrnice, neboť IDS bude muset analyzovat mnohem méně paketů. Systém SDM totiž dokáže efektivně odfiltrovat těžké toky, které obsahují výrazný podíl celkového počtu všech přijatých paketů a které nejsou z hlediska detekce incidentů nadále relevantní. Díky vyšší výkonnosti se tak předejde nekontrolované ztrátě příchozích dat a zlepší se i úspěšnost detekce bezpečnostních incidentů.

Navržený systém může v budoucnu pracovat i adaptivně. Pokud systém IDS nasazený na síti stíhá analyzovat všechny pakety a všechny síťové toky, není akcelerace prostřednictvím SDM nutná. V případě nárůstu zatížení lze potom akceleraci aktivovat. Na grafu na obrázku 4.4 je vidět, že při použití SDM, kdy by se analyzovalo jen prvních 30 paketů, by sice z důvodu nasazení SDM klesl počet detekovaných incidentů o 5 %, na druhou stranu by se ale zabránilo právě nekontrolovatelnému zahazování vstupních dat, které by zajisté vyústilo k daleko většímu poklesu úspěšnosti detekce, více viz kapitola 6.

Kapitola 5

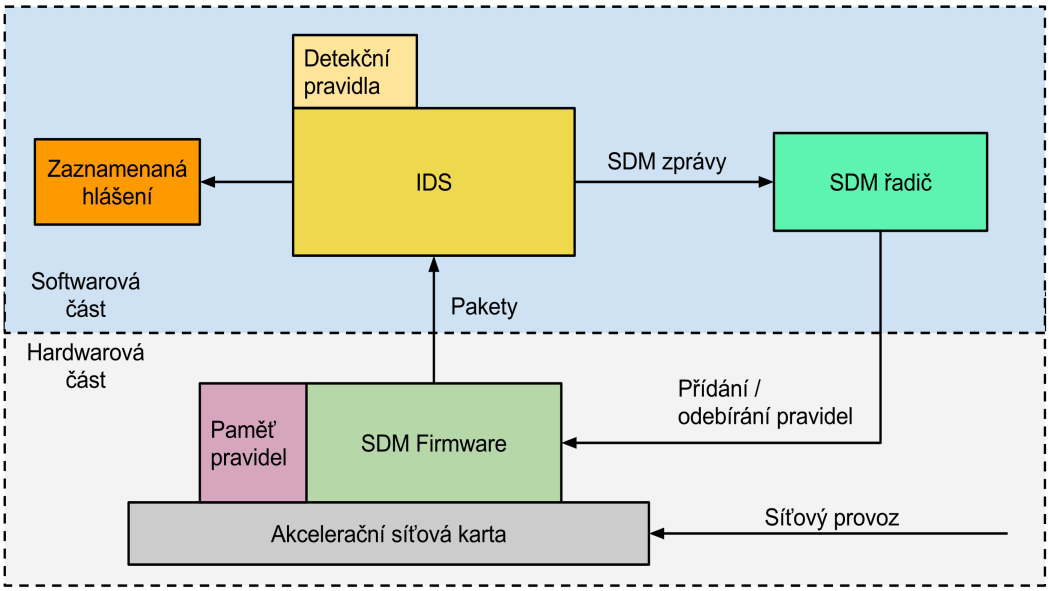
Implementace

Tato kapitola se zabývá implementací navrženého systému pro dvě IDS aplikace – Snort a Suricata. V první části kapitoly je popsána varianta využívající hardwarově akcelerovanou verzi systému SDM a implementace založená na knihovně `libpcap`. Tato varianta systému však dosahovala celkově velmi nízké propustnosti, viz výsledky měření v podkapitole 6.2. Z tohoto důvodu tak byla nad rámec zadání pro systém Suricata pro dosažení vyšší výkonnosti implementována nativní podpora rozhraní SZE2. Podrobnosti k implementaci podpory rozhraní SZE2 jsou v druhé části této kapitoly. Poslední část kapitoly je věnována fungování a implementaci softwarové varianty konceptu SDM. Softwarová varianta byla implementována za účelem dalšího srovnání s hardwarově akcelerovanou variantou z hlediska vlivu na dosažení urychlení. Od hardwarově akcelerované varianty se ta softwarová liší pouze tím, že jsou principy SDM implementovány čistě softwarově.

5.1 Komunikace s řadičem SDM

Pro hardwarově akcelerovanou variantu je charakteristické využití FPGA akcelerační síťové karty s firmwarem podporujícím koncept SDM. Na obrázku 5.1 je zobrazena architektura tohoto řešení. Zjednodušeně se může celý systém rozdělit na softwarovou (horní polovina obrázku) a hardwarovou část (dolní polovina obrázku). Softwarové části systému dominuje činnost IDS. Úkolem hardwarové části je předzpracování síťového provozu v rámci firmwaru SDM, který je řízen softwarovým SDM řadičem. Síťový provoz je přijímán akcelerační síťovou kartou (v dolní části obrázku). Firmware SDM se dále podle pravidla v paměti rozhoduje, zda má zpracovávaný paket zahodit nebo přeposlat do IDS k analýze. Ve výchozím stavu se paket vždy přepošle. IDS poté učiní analýzu pomocí detekčních pravidel a výsledky zaznamená ve formě hlášení. Informace z hlavičky každého zpracovaného paketu však ještě ve formě zprávy přepošle řadiči SDM (pravá část obrázku). Řadič si paket započítá do svých vnitřních struktur (viz kapitola 3.2) a zkontroluje překročení tzv. threshold hodnoty. Tato hodnota popisuje hranici počtu paketů, po kolika se síťový tok pokládá za těžký. Jakmile tedy tok překročí určitý počet paketů, řadič nainstaluje pravidlo do firmwaru SDM, aby další pakety toho toku zahazoval. Po vypršení zvoleného časového limitu tzv. timeoutu se předpokládá, že síťový tok již skončil a pravidlo se z paměti řadiče a firmwaru SDM zase odebere.

Pro realizaci hardwarově akcelerované varianty se muselo v IDS nejdříve zajistit přístup k rozhraní SZE2 pro příjem paketů. Firmware SDM, respektive akcelerační karty COMBO, využívají toto proprietární rozhraní k DMA přenosům síťových dat do operační paměti.



Obrázek 5.1: *Princip systému s hardwarově akcelerovanou variantou SDM.*

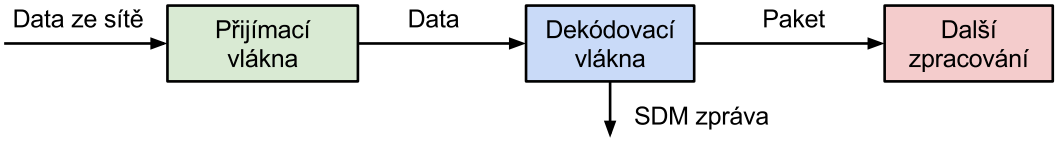
Uživatelským aplikacím je následně poskytováno softwarové SZE2 rozhraní pro přístup k přeneseným datům. Systémy Snort i Suricata však implicitně podporují pouze knihovnu `libpcap`, která poskytuje standardní rozhraní pro čtení paketů ze sítě. Toho se dá ale využít. Za účelem kompatibility s řadou aplikací je totiž pro SZE2 k dispozici také upravená knihovna `libpcap`, která umožňuje přijímat data ze sítě i prostřednictvím akceleračních karet COMBO. IDS systémy Snort a Suricata jsou tak schopny k příjmu dat ze sítě používat upravenou verzi `libpcap` knihovny. Dále bylo potřeba pro realizaci řešení zajistit především napojení systému na řadič SDM.

Knihovna `libsdm` poskytuje komunikační rozhraní mezi řadičem SDM a uživatelskou aplikací (v tomto případě systémem IDS). Inicializace rozhraní se provádí voláním funkce `sdm_init`. Dále je třeba voláním funkce `sdm_start_context` založit tzv. kontext, jenž definuje základní chování SDM při přijetí nových toků. Při inicializaci se rovněž musí otevřít komunikační kanál k řadiči funkcí `sdm_open`, která alokuje potřebné datové struktury pro komunikaci. Jakmile jsou data hlavičky paketu dekódována, lze formou zprávy odeslat informace o paketu řadiči SDM. Zpráva se odesílá voláním funkce `sdm_send`. SDM zpráva se skládá z identifikátoru toku, příkazu k přidání pravidla do paměti SDM, odpovídající akce určující způsob předzpracování (zahození) dat a informace o tom, ke kterému kontextu zpráva patří. Identifikátor toku se potom skládá ze zdrojové a cílové IP adresy, portů, protokolu, označení verze IP a čísla rozhraní, na kterém byl paket přijat. Do firmwaru SDM se pravidlo neinstaluje okamžitě, ale až po překročení tzv. threshold hodnoty (viz dříve).

První možností pro implementaci komunikace s řadičem SDM bylo vytvoření zcela nezávislé aplikace – `rules_inserter`, která poběží souběžně s IDS. Aplikace načítá data z rozhraní SZE2, analyzuje a extrahuje informace z hlaviček paketů a předává tyto informace řadiči SDM. Výhodou tohoto řešení je, že není třeba zasahovat do zdrojového kódu IDS a provádět poté rekompilaci. Nevýhodou tohoto přístupu je skutečnost, že analýza a extrakce dat z paketů se provádí duplicitně v IDS i v aplikaci `rules_inserter`. Toto řešení bylo použito pro systém Snort. Pro systém Suricata bylo implementováno řešení už dříve znázorněné na obrázku 5.1, kde zdrojem zpráv pro řadič SDM je přímo systém IDS.

Na obrázku 5.2 je zobrazen příjem vstupních dat ze sítě, jejich dekódování a odeslání zprávy řadiči SDM, tak jak je zpracování implementováno v rámci architektury systému Suricata. Implementace byla vytvořena na základě návodu pro vývojáře [2], který popisuje způsob, jak vytvořit vstupní uživatelský plugin pro příjem a zpracování paketů. Hlavní uživatelem registrované callback funkce využívají dvě skupiny vláken. Úkolem první skupiny vláken (Přijímací vlákna) je co nejrychleji získat paket ze vstupního rozhraní a předat jej k dalšímu zpracování. Každé přijímací vlákno si přitom musí otevřít vlastní rozhraní pro příjem dat ze sítě. Druhá skupina vláken (Dekódovací) přijaté pakety dekóduje a naplní své vnitřní struktury potřebnými informacemi.

Přijímací vlákna nebylo třeba v tomto případě nijak měnit, protože původní callback funkce implicitně podporuje rozhraní knihovny `libpcap`. Při inicializaci dekódovacích vláken bylo naopak potřeba pro každé vlákno otevřít vlastní komunikační kanál k řadiči SDM, protože tyto vlákna jsou použita právě pro posílání zpráv řadiči. Po dekódování paketu je tedy přidána funkce, která využije získané extrahované informace z paketu k vytvoření zprávy pro řadič SDM.



Obrázek 5.2: *Schéma pluginu pro systém Suricata.*

5.2 Nativní podpora rozhraní SZE2

Rozhraní SZE2 slouží k příjmu paketů ze sítě prostřednictvím akcelerované síťové karty COMBO. Jsou dvě možnosti, jak k tomuto rozhraní přistupovat. Buď za využití standardizované knihovny `libpcap` nebo přímo. Nativní rozhraní SZE2 přitom na rozdíl od upravené knihovny `libpcap` umožňuje příjem paketů ve vícevláknovém režimu. Díky vícevláknovému zpracování se tak po implementaci přímé podpory SZE2 předpokládá možnost výrazně rychlejšího příjmu dat ze strany IDS. Nativní podpora SZE2 rozhraní byla tedy implementována jako alternativní callback funkce pro přijímací vlákna v rámci už vytvořeného vstupního uživatelského pluginu (viz výše).

Každé procesní vlákno si může k rozhraní SZE2 otevřít vlastní datový kanál pomocí funkce `szedata_open`. Počet přijímacích vláken může být přitom až tolik, kolik je k dispozici dostupných DMA kanálů pro přenos dat ze síťové karty do RAM. Počet dostupných DMA kanálů lze zjistit voláním funkce `szedata_ifaces_available`. Správným nastavením parametrů funkce `szedata_subscribe3` lze dále zajistit, aby každému přijímacímu vláknu příslušel právě určitý DMA kanál. Takovéto využití vícevláknového přístupu zaručuje rychlý přísun vstupních dat do systému. Na rozhraní SZE2 je přijímán nejen samotný paket, ale také tzv. hlavička SZE. Hlavička obsahuje další informace a metadata, která vyplní firmware na COMBO kartě při příjmu paketu ze sítě. Následuje popis z čeho se hlavička SZE skládá a jaké položky obsahuje.

Velikost rámce a velikost hlavičky SZE: První pole udává součet velikosti SZE hlavičky a velikosti následujících dat v bytech. Druhé pole potom udává velikost samotné SZE hlavičky, viz obrázek 5.3. Hodnoty slouží k posunu ukazatelů v rámci přijímacího kruhového bufferu v RAM.

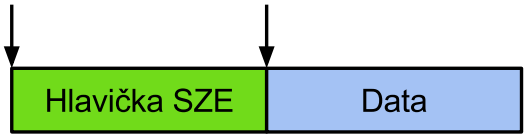
Číslo vstupního síťového rozhraní: Číslo označuje konkrétní vstupní síťové rozhraní na akcelerační síťové kartě COMBO, na kterém byl paket ze sítě přijat.

Číslo DMA kanálu: Číslo určuje, přes který kanál DMA byl paket do RAM přenesen.

CRC haš: Předpočítaný údaj, který slouží k identifikaci síťového toku na základě IP adres, portů, protokolu a vstupního rozhraní pro potřeby některých aplikací.

Typ dat: Položka popisuje formát dat za SZE hlavičkou. Data mohou mít podobu paketu, zkráceného paketu, UH hlavičky obsahující extrahované informace z hlavičky paketu nebo flow UH hlavičky obsahující agregované informace o síťovém toku.

Časová značka: Informuje o času přijetí paketu na akcelerační síťové kartě s nanosekundovou přesností. První 32-bitová část pole udává čas přijetí standardně v sekundách od 1. 1. 1970. Druhá 32-bitová část pole potom udává čas příjmu paketu jako počet nanosekund uplynulých od začátku této sekundy.



Obrázek 5.3: *Rámec SZE.*

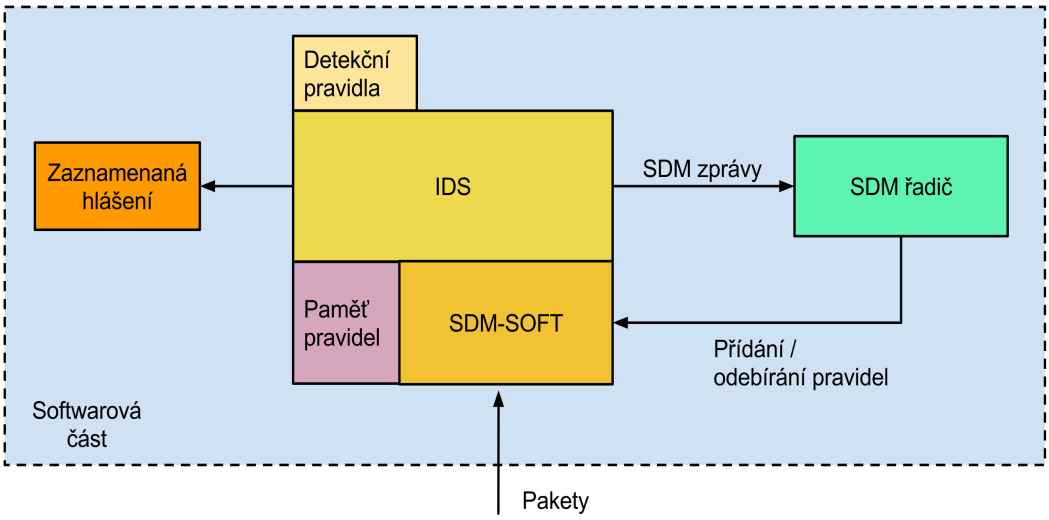
Pro získání SZE rámce je k dispozici funkce `szedata_read_next`. Získat hlavičku a příslušná data, jejichž obsahem je paket lze přes funkci `szedata_decode_packet`. Jak ilustruje obrázek 5.3, tato funkce dekóduje rámec SZE a vrátí dva ukazatele (znázorněné na obrázku šipkami). Z ukazatele na SZE hlavičku se potom získá číslo vstupního rozhraní a časová značka. Následně se extrahovaná metadata společně s druhým ukazatelem předají IDS systému k dalšímu zpracování.

5.3 Softwarová varianta SDM

Základní myšlenku konceptu SDM o předzpracování (zahazování) těžkých toků lze aplikovat jak v hardwaru, tak v softwaru. Softwarová varianta tohoto principu je přitom z ekonomického hlediska výrazně levnější alternativou, protože akcelerační síťová karta s FPGA může být finančně značně nákladná.

Na obrázku 5.4 je znázorněna architektura této varianty implementace. Příchozí data ve formátu SZE2 jsou v rámci IDS a komponenty SDM-SOFT dekódována. Dále se na základě paměti pravidel ověřuje, zda se má daný paket dále analyzovat či nikoli. Ve výchozím stavu, tzn. pokud se příslušné pravidlo nenajde, se analýza provádí. Stejně jako u hardwarově akcelerované varianty se potom SDM řadiči přepošle zpráva o nově příchozím paketu. SDM řadič je opět implementován jako nezávislý program, který běží na pozadí. V rámci softwarové varianty je pouze nahrazen jeden z jeho modulů, jenž původně sloužil ke komunikaci s firmwarem. Místo toho probíhá komunikace s komponentou SDM-SOFT. Komunikační rozhraní je řešeno stejně, jako v případě zasílání SDM zprávy řadiči. Knihovna `libsdm` pro tyto účely poskytuje sdílenou FIFO frontu. Frontu tedy plní SDM řadič příkazy pro přidání či odebrání pravidla z paměti pravidel (fialová část), podobně jako tomu bylo

v předchozí hardwarově akcelerované variantě. Ostatní funkcionalita SDM řadiče je ponechána beze změny. Komponenta SDM-SOFT spravuje paměť pravidel, ve které přidává či odebírá pravidla. Na základě těchto pravidel se následně rozhoduje, jestli se paket zahodí či nikoli. Proces je podobný, jako tomu bylo v předchozí variantě, kde se pravidla instalovala do firmwaru SDM.



Obrázek 5.4: *Princip softwarové varianty SDM.*

Paměť pravidel je implementována jako hašovací tabulka [1], která podporuje vícevláknový výlučný přístup ke konkrétnímu řádku tabulky. Jestliže v paměti pravidel existuje pravidlo pro právě přijímaný paket, nastaví se tomuto paketu v rámci systému Suricata příznaky `PKT_NOPACKET_INSPECTION` a `PKT_NOPAYLOAD_INSPECTION`. Příznaky zabrání, aby se nad paketem aplikovaly jakékoliv detekční mechanismy. Tímto způsobem se tak simuluje zahození příchozího paketu. Paket sice musí být částečně zpracován, ale i tak se tím výrazně šetří dostupné prostředky. V komponentě SDM-SOFT pracuje samostatné a zcela nezávislé vlákno, které zpracovává pouze požadavky od SDM řadiče, a do paměti pravidel pravidla přidává nebo je naopak odebírá. Dekódovací vlákna v rámci celého systému z této paměti pouze čtou. Do paměti pak zapisuje pouze vlákno z komponenty SDM-SOFT.

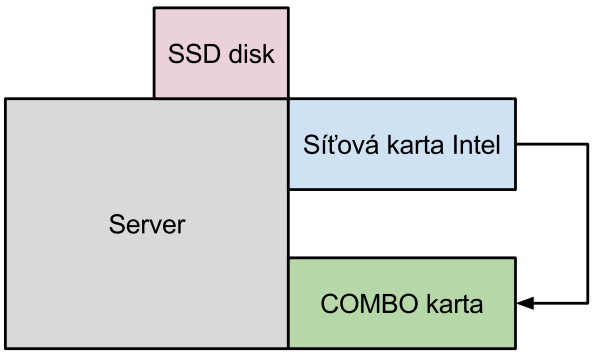
Kapitola 6

Dosažené výsledky

Kapitola popisuje dosažené výsledky získané při měření vytvořeného systému. Porovnány jsou výsledky jak hardwarově akcelerované varianty využívající koncept SDM, tak verze s plně softwarovou implementací principu SDM. Hlavními sledovanými údaji byla propustnost, která je spočítána jako podíl zahozených paketů na vstupních síťových bufferech, a dále počet incidentů detekovaných systémem IDS. Před samotnými výsledky měření podkapitola 6.1 nejprve popisuje použité prostředí pro provádění experimentů a jak se pro potřeby měření přehrává zachycený síťový provoz na vysoké rychlosti. Podkapitola 6.2 se zaměřuje na oba IDS systémy Snort a Suricata s implementací využívající knihovnu `libpcap`. Měření v podkapitole 6.3 probíhalo už jen nad systémem Suricata, ve kterém byla doplněna nativní podpora rozhraní SZE2 a implementována softwarová varianta principu SDM. Protože počet detekčních pravidel má velký vliv na celkovou výkonnost IDS, byla Suricata změřena nejprve se všemi pravidly z použité databáze a potom v druhém případě pouze s vybranými pravidly detekujícími jen určitý typ incidentu (detekce malware).

6.1 Prostředí pro provádění měření

V této podkapitole je popsáno použité prostředí pro prováděná měření, jejichž výsledky jsou dále rozebrány v následujících podkapitolách 6.2 a 6.3. Je zde také vysvětleno, jakým způsobem je v rámci experimentů řešeno přehrávání uloženého síťového provozu dostatečnou rychlostí tak, aby se dosáhlo výkonnostních limitů IDS.



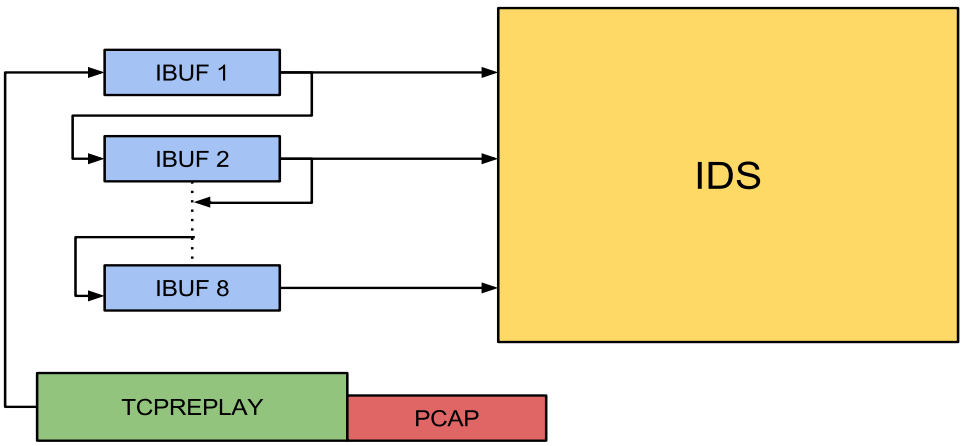
Obrázek 6.1: *Připojené periferie k měřícímu serveru.*

Na obrázku 6.1 je schématicky znázorněno použité měřící prostředí. Zachycený síťový provoz uložený v podobě `pcap` souboru na disku SSD se přehrával a ve smyčce následně ze sítě opět zachytával v rámci stejného serveru, na kterém následně probíhala také analýza nově zachycených dat prostřednictvím systému IDS. Veškerá měření byla realizována na serveru Supermicro X9DRG-QF [5] se dvěma 8-jádrovými procesory Intel Xeon E5-2650 pracujícími na frekvenci 2.6 GHz a operační pamětí o velikosti 64 GB. Dále byla použita akcelerační síťová karta COMBO-80G s čipem FPGA Xilinx Virtex-7 a firmwarem SDM. Vstupní síťové rozhraní akcelerační karty bylo použito v konfiguraci 8x10 Gb/s, což umožňuje příjem dat s celkovou maximální rychlostí 80 Gb/s. Pro generování síťových dat byla potom použita síťová karta Intel X520 se dvěma 10 Gb/s porty.

Přehrávání uloženého síťového provozu se provádělo pomocí nástroje `tcpreplay`. Po přehrání souboru se následně sledovalo, kolik paketů bylo celkově zpracováno, respektive zahozeno na vstupním síťovém rozhraní z důvodu zatížení systému, a také kolik incidentů bylo systémem IDS úspěšně detekováno. Každý z těchto údajů byl následně vyhodnocen pro různé rychlosti přehrávání síťového provozu.

Při přehrávání uloženého sítového provozu je potřeba dosáhnout vysokých rychlostí, které by simulovaly vysokorychlostní přenos. Limitující je ovšem rychlost čtecí operace z paměťového média, kde je zachycený provoz uložen. Rychlost čtení z pevných disků se přitom pohybuje v rozmezí 80-160 MB/s, avšak SSD paměti nabízejí rychlejší čtení dat přibližně až 550 MB/s, proto byl pro měření zvolen právě tento typ paměťového média.

Systém Suricata s implementovanou nativní podporou rozhraní SZE2 byl však dostatečně výkonný na to, aby stíhal analyzovat přehrávaný provoz z SSD paměti bez ztráty paketů. Pro zjištění výkonnostních limitů bylo tedy nutné rychlost přehrávaného provozu dále zvýšit hardwarovou replikací. Na obrázku 6.2 je znázorněn princip replikace. Provoz, který se přijme na prvním vstupním rozhraní lze konfigurací firmwaru dále přeposlat na výstupní rozhraní. Vhodným vnějším propojením optických kabelů se provoz z prvního výstupního rozhraní následně přivede na druhé vstupní rozhraní. Toto propojení se aplikuje tolikrát, kolikrát je třeba provoz replikovat. Při správném zapojení optických kabelů stačí před každým měřením jen správně nakonfigurovat firmware. Takováto replikace dat na akcelerační síťové kartě zajistí, aby se provoz přehrávaný například rychlostí 7 Gb/s po dvojité replikaci znásobil na rychlost 14 Gb/s. Přes kanály DMA sice potečou v jednu chvíli dvakrát stejná data, ale dosáhne se tím ve výsledku vyšších rychlostí přehrávání.



Obrázek 6.2: *Replikace na vyrovnávacích bufferech IBUF.*

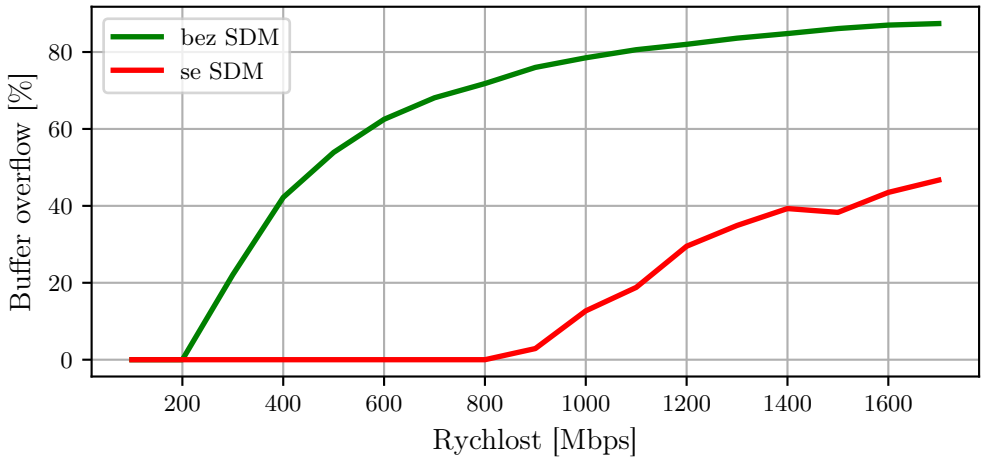
Systém IDS o replikaci musí nutně vědět a patřičně s ní pracovat, jinak by ve vstupních datech viděl zdvojnásobené délky síťových toků a navíc by se změnila očekávaná posloupnost komunikačních kroků např. v případě stavového protokolu TCP. Z toho důvodu je potřeba v IDS provést úpravy tak, aby bylo schopné nezávisle rozeznat původní data od dat vzniklých replikací. Kopie původního síťového toku se tedy musí rozlišovat kromě pětice (IP adresy, porty, protokol) ještě dalším identifikátorem. Využít se dá položka v hlavičce SZE, která vyjadřuje číslo vstupního rozhraní, na němž byl paket přijat. Jakmile síťová data v rámci firmwaru opustí vyrovnávací buffer vstupního síťového rozhraní (IBUF), doplní se do hlavičky SZE právě identifikační číslo bufferu. Původní data jsou tak například označena číslem jedna, jednou replikovaná číslem dva atd. Na základě této šestice potom IDS identifikuje jednotlivé kopie síťového toku a může je zpracovat nezávisle. V přijímaném provozu ve výsledku zůstanou délky toků stejné a znásobí se jen jejich počet.

Podporu pro rozeznávání, ze kterého vstupního rozhraní paket pochází, lze přidat pouze do varianty systému Suricata s nativní podporou SZE2. V případě získávání dat z knihovny `libpcap` není totiž možné přímo přistoupit k SZE hlavičce, tudíž nelze rozlišit, ze kterého rozhraní paket pochází. Tím však nevzniká žádný problém, protože s použitím knihovny `libpcap` je celková výkonost IDS nižší než u varianty s nativní podporou SZE2, takže není třeba replikaci uvažovat. Podpora pro rozeznávání je implementována tak, že číslo vstupního rozhraní z hlavičky SZE se přidalo jako klíč při výpočtu haše ve zdrojovém souboru `flow-hash.c`. Původní a replikovaný tok se tak ukládá na různých místech v hašovací tabulce toků. Suricata si tedy nespojuje původní síťový tok s replikovaným a vnímá je odlišně.

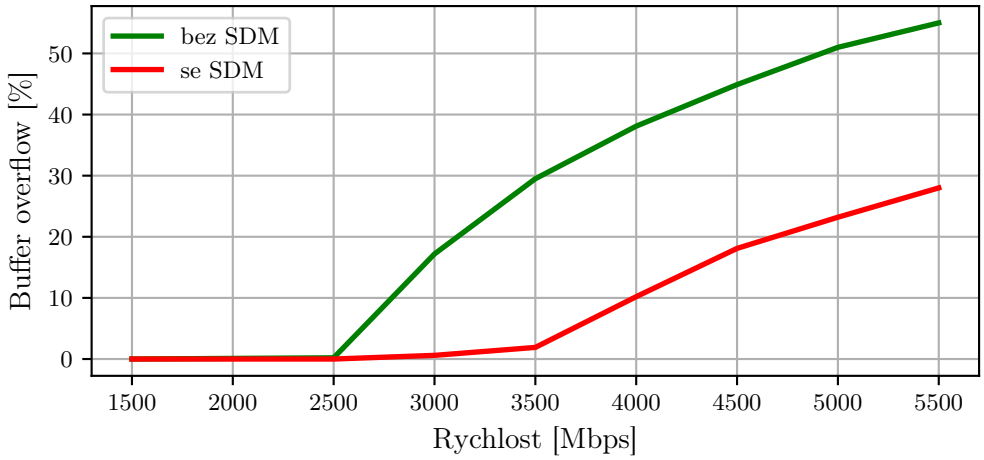
6.2 Snort a Suricata s rozhraním libpcap

První provedené měření sledovalo vliv nasazení konceptu SDM na urychlení IDS aplikací. Oba systémy Snort a Suricata přitom přijímaly pakety pomocí knihovny `libpcap`. Suricata v tomto měření však používala jen jedno přijímací vlákno, protože toto rozhraní nepodporuje použití více vláken. Pro přehrávání síťového provozu byl použit už dříve zachycený provoz z reálné sítě o velikosti 55 GB. Při nastavení hodnoty threshold parametru bylo snahou docílit toho, aby nedocházelo k selhání instalace nového pravidla do hardwarové paměti pravidel. Parametr timeout byl ponechán na výchozí hodnotě 300 s.

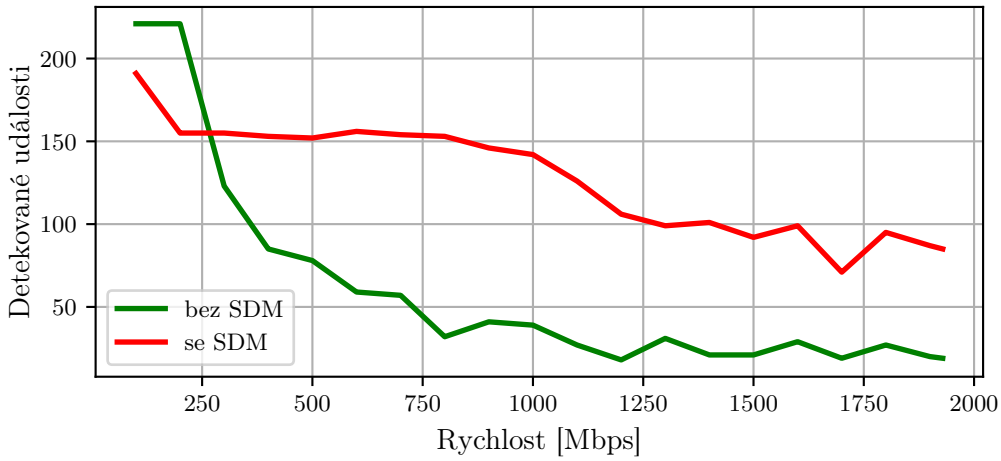
Výsledky měření propustnosti jsou pro systém Snort znázorněné v grafu na obrázku 6.3 a pro systém Suricata v grafu na obrázku 6.4. Na ose x je uvedena celková rychlost přehrávaného provozu. Na ose y je potom v procentech vyjádřeno, kolik paketů bylo na vstupních síťových bufferech zahozeno z důvodu zaplnění kapacity bufferu. Zelená křivka označuje vždy výsledky daného IDS systému bez akcelerace pomocí SDM. Červená křivka ukazuje naopak průběh při použití SDM. Při porovnání grafů 6.3 a 6.4 jde vidět, že Suricata byla na tom s propustností daleko lépe než Snort. Systém Suricata, přestože používal daleko více detekčních pravidel, nestíhal zcela zpracovávat všechny příchozí pakety až od rychlosti 2500 Mb/s, kdežto systém Snort dosahoval svého limitu už při rychlosti 200 Mb/s. Oba grafy zároveň ukazují pozitivní vliv při použití SDM. Díky akceleraci byl schopen systém Snort analyzovat až 800 Mb/s rychlý provoz. Celkové zrychlení bylo tedy 400 %. Systém Suricata s mírným zahazováním zvládal analyzovat provoz o rychlosti 3500 Mb/s, což je tak zrychlení přibližně o 40 %. Grafy na obrázcích 6.5 a 6.6 zobrazují počet detekovaných incidentů při různých rychlostech. Díky akceleraci prostřednictvím SDM se detekovalo v obou případech i více incidentů. V případě systému Snort se při rychlosti 800 Mb/s detekovalo třikrát více incidentů. V případě systému Suricata se při rychlosti 3500 Mb/s detekovalo až o 25 % incidentů více.



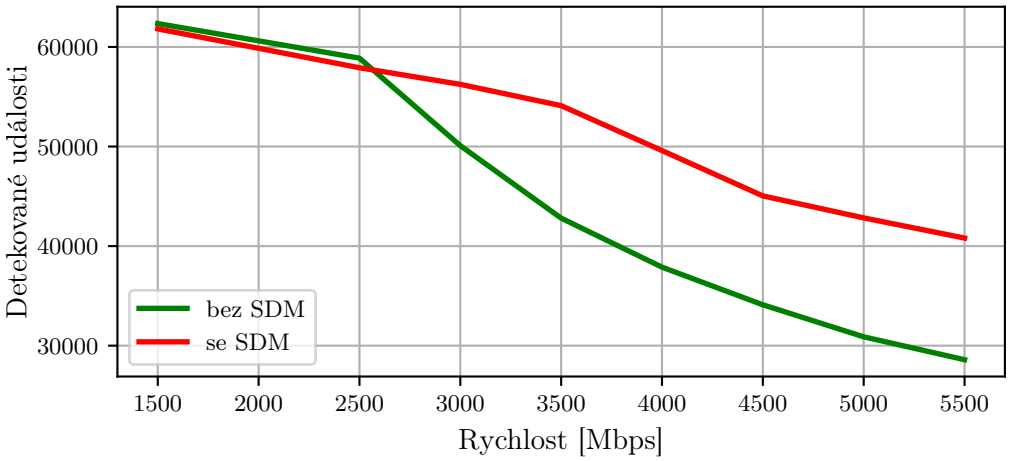
Obrázek 6.3: *První měření propustnosti systému Snort.*



Obrázek 6.4: *První měření propustnosti systému Suricata.*



Obrázek 6.5: *Počet úspěšně detekovaných incidentů pro sytém Snort.*



Obrázek 6.6: *Počet úspěšně detekovaných incidentů pro systém Suricata.*

Grafy výsledků měření doprovází vždy tabulka popisující konkrétně nastavené parametry měření. Pro první provedené měření se jedná o tabulku 6.1. Dále následuje krátké vysvětlení záhlaví tabulek a diskuze nad vlivem konkrétních hodnot a parametrů zvolených pro jednotlivá měření.

Pravidla. V tomto sloupci tabulky je uveden počet použitých detekčních pravidel. Čím více detekčních pravidel systém IDS pro analýzu při vysokých rychlostech používá, tím více se na vstupních síťových bufferech typicky zahazuje, protože se buffery zaplňují rychleji než je IDS schopno pakety z bufferů odebírat. Čím více detekčních pravidel je použito, tím více je systém IDS zatížen při analýze každého jednotlivého paketu.

Threshold. Při spouštění řadiče SDM ve formě démona běžícího na pozadí se také nastavuje parametr threshold. Jak bylo uvedeno již dříve, tato hodnota říká po kolika paketech se má daný síťový tok pokládat za těžký a po kolika paketech se mají další pakety toho toku při zpracování ve firmwaru SDM zahazovat. Čím nižší je hodnota threshold parametru, tím méně paketů se v IDS analyzuje, což zvyšuje celkovou výkonnost systému. Na druhou stranu se ale může snižovat počet úspěšně detekovaných incidentů. S nižším parametrem threshold se navíc rychleji zaplňují paměti ve firmwaru a řadiči SDM a tím dochází častěji k selhání při přidání pravidla. Vhodně nastavený threshold musí být tedy kompromisem mezi požadavky SDM a IDS na výkon a přesnost detekce. V SDM je snaha maximálně využít hardwarovou paměť ve firmwaru, aby byl co nejlepší poměr dat zpracovaných v hardwaru a softwaru. Na druhou stranu z hlediska IDS je snaha o co nejvyšší hodnotu threshold parametru, aby se mohly detekovat i incidenty vyžadující analýzu většího počtu paketů ze začátku síťového toku.

Tabulka 6.1: *Parametry prvního měření.*

IDS	**Pravidla** [počet]	**Threshold** [paketů]	**Timeout** [sekund]	**Replikace** [x krát]	**PCAP soubor** [GB]
Snort	8545	8	300	0	55
Suricata	12442	25	300	0	55

Timeout. Stejně jako parametr threshold tak i timeout se nastavuje při spuštění SDM démona. Hodnota timeout se uvádí v jednotkách sekund a říká, jak dlouho mají být pravidla pro těžké toky uloženy v paměti firmwaru SDM. Čím nižší je tato hodnota, tím menší je pravděpodobnost zaplnění paměti pravidel. Na druhou stranu se opakovaně do IDS posílají toky, které časově přesáhnou nastavený timeout, protože po smazání pravidla z SDM se každý takový tok pokládá za nový.

Replikace. Hardwarová replikace přehrávaného síťového provozu probíhá vhodným propojením vstupních a výstupních síťových rozhraní na akcelerační síťové kartě, jak bylo vysvětleno dříve na obrázku 6.2 viz podkapitola 6.1. Hodnota v tabulce udává, kolik je zapojeno vstupních síťových bufferů, tzn. tolikrát se síťový provoz při měření replikuje.

PCAP soubor. Pro přehrávání síťového provozu byly k dispozici dva `pcap` soubory se zachycenými testovacími daty z reálné sítě. První zachycený soubor s celkovým objemem dat 55 GB byl použit už v rámci kapitoly 4.2 při analýze charakteristických vlastností sítě a síťových toků z hlediska detekce incidentů. Druhý soubor s objemem dat 200 GB byl potom na stejné lince zachycen dodatečně pouze pro potřeby výkonnostních měření vytvořeného systému. Výhodou využití menšího 55 GB souboru je, že se soubor při čtení uloží v RAM paměti, tudíž je potom umožněno přehrávání síťového provozu na vyšších rychlostech, než kdyby se data pomalu načítala z SSD disku. Výhodou použití většího 200 GB `pcap` souboru je naopak to, že i při vyšších rychlostech se síťový provoz přehrává dostatečně dlouhou dobu tak, aby se projevil vliv SDM i z dlouhodobějšího hlediska a podíl hardwarově zpracovaných dat se vzhledem k nastavené hodnotě thresholdu a timeoutu ustálil.

6.3 Suricata s rozhraním SZE2, softwarová varianta

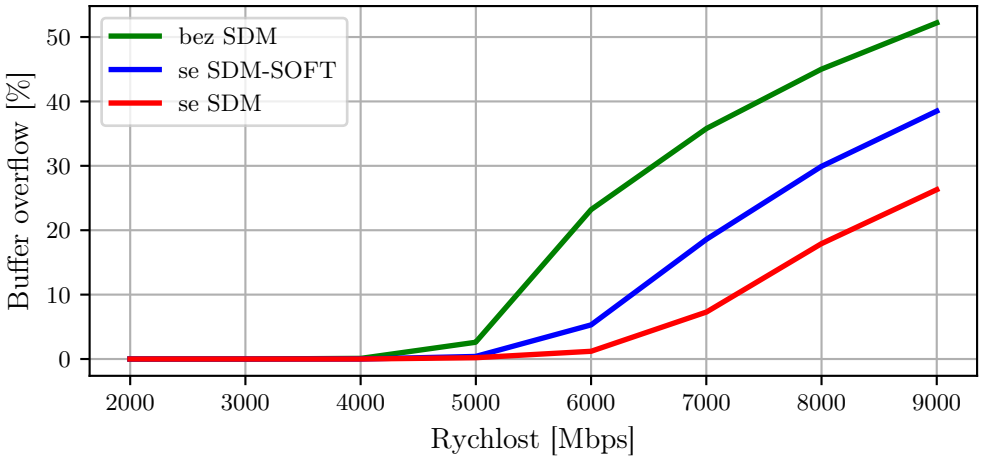
Druhé a třetí měření bylo provedeno až po rozšíření systému Suricata. Do systému byla přidána nativní podpora rozhraní SZE2 a pro možnost dalšího srovnání byla implementována také plně softwarová varianta konceptu SDM. K následujícím grafům byla tak přidána modrá křivka, která znázorňuje výsledky měření právě plně softwarové implementace principů SDM (SDM-SOFT).

Systém Suricata byl pro potřeby druhého měření opět velmi výrazně zatížen vysokým počtem pravidel z databáze EmergingThreats (ET) [39]. Při tomto měření se přehrával větší ze souborů 200 GB se zachyceným síťovým provozem z reálné sítě. Stejně jako u předchozího měření byla zvolena hodnota pro threshold parametr tak, aby byla optimálně využita hardwarová paměť pravidel. Parametr timeout pro odstranění neaktivních toků byl opět ponechán na výchozí hodnotě 300 s. Pro potřeby rychlejšího přehrávání zachyceného síťového provozu se přitom musel původní provoz celkem šestkrát replikovat. Veškeré parametry druhého měření shrnuje i tabulka 6.2. Samotné výsledky druhého měření jsou potom zachyceny na dvou grafech na obrázcích 6.7 a 6.8.

Tabulka 6.2: *Parametry druhého měření.*

IDS	Pravidla [počet]	Threshold [paketů]	Timeout [sekund]	Replikace [x krát]	PCAP soubor [GB]
Suricata	13642	20	300	6	200

Graf na obrázku 6.7 znázorňuje dosaženou propustnost systému před a po akceleraci. Na ose x je stupnice rychlostí přehrávání provozu. Osa y vyjadřuje procentuální podíl zahozených paketů na vstupních síťových bufferech. Zelená křivka ukazuje průběh bez akcelerace pomocí SDM. Červená křivka naopak zobrazuje průběh při použití SDM. Modrá křivka vykresluje použití softwarové implementace SDM. Oproti předchozímu měření, kde se pakety přijímaly přes rozhraní `libpcap`, se nyní s využitím nativního rozhraní SZE2 zvýšila celková propustnost. Maximální rychlost bez zahazování a bez použití SDM je 4 Gb/s. Při nasazení SDM se hranice s nulovým počtem zahozených paketů posunula jen na rychlost 5 Gb/s. Pokud bychom ale tolerovali určitou ztrátovost paketů, rozdíl akcelerované a neakcelerované varianty se s vyšší rychlostí dále zvětšuje. Například při rychlosti 7 Gb/s zůstává u hardwarově akcelerované varianty míra zahazování stále pod 10 %, zatímco pro variantu bez SDM je ztrátovost více jak 35 %. Také softwarová implementace principu SDM přináší zajímavé výsledky. Na první pohled je vidět, že se křivka pohybuje v oblasti mezi zelenou a červenou křivkou. Dalo by se říci, že tak přináší systému Suricata zrychlení až o 1 Gb/s.

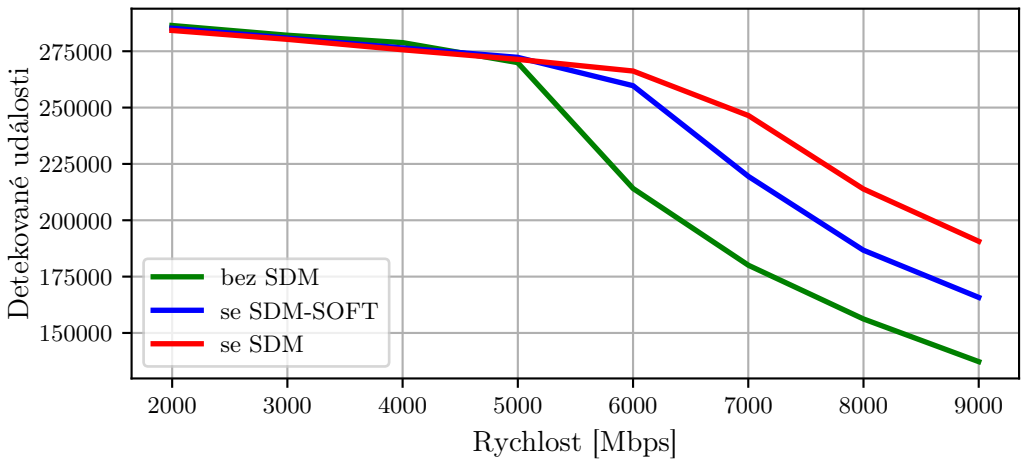


Obrázek 6.7: *Druhé měření propustnosti systému Suricata s velkou sadou pravidel.*

Graf na obrázku 6.8 se zaměřuje na počet detekovaných incidentů. Na ose x je vynesena rychlost přehrávaného síťového provozu v jednotkách Mb/s. Na ose y je potom znázorněn počet úspěšně detekovaných incidentů nahlášených systémem IDS. Z grafu je patrné, že i když se pakety do rychlosti 4 Gb/s vůbec nezahazovaly, přesto (avšak velmi mírně vzhledem k počtu celkově detekovaných událostí) klesá počet detekovaných incidentů. Systém Suricata pravděpodobně častěji autonomně uvolňoval paměťové prostředky určené k uchování kontextu komunikace. Tento jev může být zřejmě ovlivněn také použitým způsobem generování testovacích dat, kdy síťový provoz zachycený při určité rychlosti z reálné sítě je následně v rámci prováděných měření přehráván rychlostí zcela jinou. Z tohoto důvodu tak dochází k deformování časových vlastností původních síťových toků a systém Suricata na něj dle konfigurace může reagovat například častějším uvolňováním stavu komunikace.

Od rychlosti 5 Gb/s lze v případě varianty bez SDM vidět výrazně větší propad v počtu detekovaných incidentů. To proto, že od této rychlosti se výrazně více paketů na vstupních síťových bufferech nekontrolovatelně zahazuje z důvodu přetížení systému. Hardwarově akcelerovaná i softwarová varianta s SDM v tomto případě po dvacátém paketu filtrovaly zbylé pakety těžkých toků. Jak bylo vidět v grafu na obrázku 6.7, zlepšila se tím propustnost a umožnilo to tak úspěšně analyzovat provoz i při vyšších rychlostech. Počet detekovaných

incidentů se stoupající rychlostí tak neklesal právě také díky menší míře nekontrolovatelného zahazování dat na vstupu.



Obrázek 6.8: *Druhé měření počtu úspěšně detekovaných incidentů sytému Suricata s velkou sadou pravidel.*

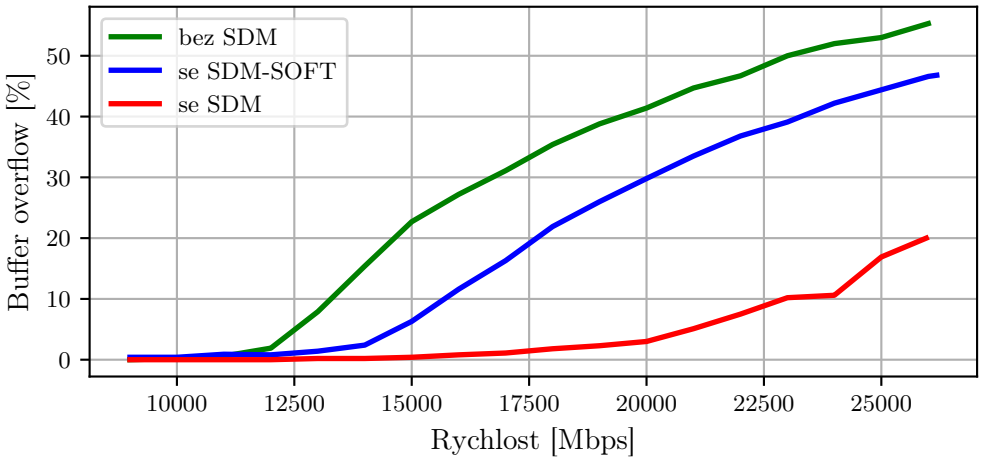
U reálného nasazení systému IDS pro monitorování sítě by se podle doporučení [12, 11] vybrala pouze některá pravidla detekující určitý typ incidentu a podle způsobu nasazení také pouze pravidla relevantní pro chráněnou síť. Proto je pro třetí měření počet detekčních pravidel experimentálně snížen na 976. Veškerá vybraná pravidla se přitom zaměřují pouze na detekci malware.

Pro přehrávání síťového provozu byl dále za účelem rychlejšího přehrání s co nejmenší mírou replikace zvolen menší 55 GB soubor. Přestože se ve třetím měření přehrává provoz daleko vyšší rychlosti, parametr vypovídající o míře replikace má nižší hodnotu (4) než tomu bylo v předchozím druhém měření (6). To je z toho důvodu, že menší 55 GB soubor díky dostatečně velké kapacitě RAM (celkově 64 GB) zůstane při opakovaném čtení v paměti. Přehrávaná data se potom nemusí znovu pomalu načítat z SSD, síťový provoz se tak může přehrávat na vyšších rychlostech a není třeba data tolikrát replikovat.

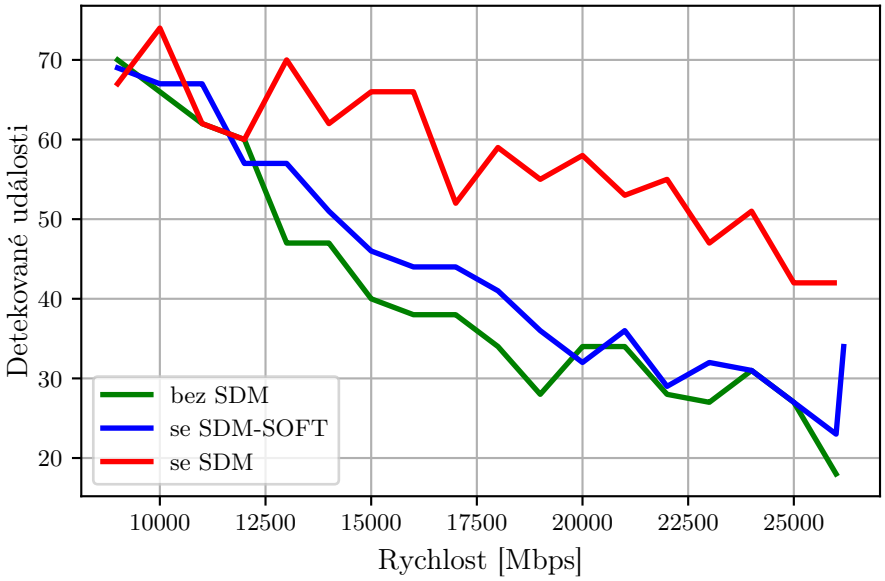
U třetího měření byl experimentálně nastaven jak parametr threshold tak i parametr timeout. Pro detekci incidentů vypovídajících o přítomnosti napadení malwarem bylo v rámci offline analýzy souboru `pcap` zjištěno, že pro detekci všech incidentů tohoto typu stačí analyzovat vždy jen prvních 10 paketů ze začátku každého síťového toku. Parametr threshold byl tedy z pohledu IDS a z hlediska spolehlivosti detekce nastaven na nejnižší možnou hodnotu 10. V této souvislosti byla poté snaha vhodně nastavit i parametr timeout. Čím je hodnota timeoutu nižší, tím dříve se smaže instalované pravidlo z paměti SDM firmwaru či v případě softwarové varianty z hašovací tabulky. Pro měření zvolená hodnota 10 sekund tak zajistila, aby nedocházelo k častému selhání při instalaci pravidla z důvodu enormního

Tabulka 6.3: *Parametry třetího měření.*

IDS	**Pravidla** [počet]	**Threshold** [paketů]	**Timeout** [sekund]	**Replikace** [x krát]	**PCAP soubor** [GB]
Suricata	976	10	10	4	55



Obrázek 6.9: *Třetí měření propustnosti sytému Suricata s pravidly detekujícími malware.*



Obrázek 6.10: *Třetí měření počtu úspěšně detekovaných incidentů sytému Suricata s pravidly detekujícími malware.*

zaplnění paměti. Zároveň nižší hodnota parametru timeout umožní při přehrávání menšího 55 GB souboru, díky častějšímu odstraňování pravidel z paměti, náhled na chování SDM z dlouhodobějšího hlediska. Souhrn všech parametrů měření poskytuje tabulka 6.3. Samotné výsledky provedeného měření jsou potom znázorněny v navazujících grafech na obrázcích 6.9 a 6.10.

V grafu na obrázku 6.9 je sledována propustnost systému vzhledem k rychlosti přehrávaného provozu v rozmezí od 10 do 25 Gb/s. Díky menšímu počtu detekčních pravidel umožňovala varianta bez použití SDM (zelená křivka) analyzovat síťový provoz bez zahazování paketů až do rychlosti 10 Gb/s. Hardwarově akcelerovaná varianta (červená křivka) je potom schopná bez zahazování paketů analyzovat síťový provoz až do rychlosti 15 Gb/s. Pokud bychom tolerovali ztrátovost paketů přibližně 10 %, dosahuje systém v porovnání

s neakcelerovanou variantou výrazného zrychlení a dokáže pracovat až při rychlosti 22 Gb/s. Oproti předchozímu druhému měření kopíruje tentokrát softwarová varianta (modrá křivka) mnohem více variantu bez použití SDM (zelená křivka). Pro doplnění je k dispozici také graf na obrázku 6.10, který se zabývá počtem detekovaných incidentů pro různé rychlosti. Na grafu je vidět, že křivky mají několik zřetelných vrcholů, což se u druhého měření na obrázku 6.8 nevyskytuje. To je způsobeno hlavně tím, že celkový počet detekovaných incidentů je z důvodu nižšího počtu použitých pravidel daleko menší.

6.4 Shrnutí výsledků měření, diskuze

V této kapitole byly popsány výsledky celkem tří měření, které charakterizují dosažené zrychlení a celkově přínos akcelerace IDS pomocí SDM. Už v prvním měření se ukázalo, že systémy Snort i Suricata dosahovaly díky SDM lepších výsledků. V druhém měření se dále zlepšil výkon systému Suricata díky implementaci nativního rozhraní SZE2. Vliv nasazení SDM ale nebyl zas tak velký, jelikož umožnil IDS systému analyzovat jen o 2-3 Gb/s rychlejší provoz. Příliš velký počet použitých detekčních pravidel totiž způsobí výrazné zvýšení složitosti analýzy síťového provozu a potom ani použití SDM nedokáže výkon IDS tolik zlepšit. Jak se ukázalo ve třetím měření s nižším počtem detekčních pravidel, je akcelerace pomocí SDM daleko více znát. Systém IDS s přibližně tisícem detekčních pravidel zaměřenými na malware dokáže díky SDM analyzovat až o 7-9 Gb/s rychlejší provoz. Vliv na dosažené zrychlení mělo zřejmě také to, že před samotným třetím měřením proběhlo nejprve několik pokusů s cílem nalezení vhodných hodnot pro oba parametry timeout a threshold.

V druhém i třetím měření je dále vidět, že i softwarová varianta SDM dosahuje o něco lepších výsledků než varianta zcela bez akcelerace. Na základě těchto poznatků tedy stojí úvahu, jestli by implementace softwarové varianty SDM, která není závislá na použití specializované akcelerační karty, nemohla být v budoucnu standardní a uživatelsky nastavitelnou součástí IDS systémů.

Jak je zmíněno už v podkapitole 6.3, u reálného nasazení systému IDS pro monitorování sítě by se podle doporučení [12, 11] vybrala pouze některá pravidla detekující určité typy incidentů. Mimo jiné by také konfigurační proměnné `HOME_NET` a `EXTERNAL_NET` byly podle způsobu nasazení nastaveny na relevantní hodnoty odpovídající chráněné síti. Aktuální konfigurace použitá při měřeních způsobuje zvýšení zátěže systému, protože se vyhodnocuje každé pravidlo. Pravidla totiž mohou být vyřazena z analýzy už na základě definované IP adresy či rozsahu adres, čímž se zrychlí vyhledávání pravidla, takže se dají předpokládat ještě lepší výkonnostní vlastnosti, než jaké jsou demonstrovány v grafech. Testovací síťová data pocházela z hraniční linky sítě CESNET2, kde např. i kvůli tranzitnímu provozu nelze jednoznačně určit prefixy chráněné sítě, proto se pro všechna měření v rámci této práce vždy používala obecná hodnota `any`.

Systém Suricata poskytuje celou řadu dalších uživatelsky nastavitelných parametrů, které mohou mít pozitivní vliv na celkovou výkonnost systému. Jedná se například o podrobné nastavení počtu vláken v jednotlivých částech architektury, určení poměru rozložení vláken na jednotlivá jádra CPU, velmi detailní nastavení velikosti alokovaných pamětí nebo také času pro sestavování segmentů síťového provozu (reassembling), využití pokročilých možností grafického akcelerátoru CUDA aj. V této práci bohužel nezbyl prostor na hlubší analýzu vlivu konfigurace těchto pokročilých parametrů a nalezení tak zcela optimálního nastavení systému. Pro provedená měření tak byly použity výchozí, respektive standardně doporučené hodnoty těchto parametrů.

Kromě optimálního nastavení parametrů lze dále uvažovat také o vylepšení způsobu vyhodnocení dosažených výsledků. Ačkoliv jak pro generování tak pro následnou analýzu síťového provozu byl vyhrazen nezávisle vždy jeden z procesorů, samotné měření by se dalo zpřesnit, pokud by systém IDS s akcelerační síťovou kartou a SDM běžel na vyhrazeném serveru. Pro generování síťového provozu pomocí nástroje `tcpreplay` by potom mohl být vyhrazen druhý server, tak aby generování síťového provozu bylo zcela odděleno a v žádném případě nedocházelo k nežádoucímu ovlivňování výsledků měření. Při přehrávání zachyceného síťového provozu při vyšších rychlostech docházelo dále k časovým deformacím síťových toků. Síťové toky se tak při přehrávání staly z časového hlediska kratší, než jaké ve skutečnosti byly. Možným řešením je zvětšovat zatížení systému pouze replikací síťového provozu nikoliv zvyšováním rychlosti přehrávání zachycených dat. To na druhou stranu samo o sobě povede k různému počtu detekovaných incidentů v každém kroku měření.

Základní myšlenku akcelerace IDS aplikací pomocí SDM lze i nadále vylepšovat. Vylepšit lze například dosavadní mechanismus identifikace paketů, které nejsou z hlediska bezpečnostní analýzy tak důležité. Dosavadní přístup zahazuje pakety těžkých toků, tzn. pakety jejichž počet v rámci toku překročí určitou hranici. Alternativně lze například implementovat to, aby IDS informovalo řadič SDM a vynutilo zahazování dalších paketů určitého toku v momentě, kdy je v daném toku úspěšně detekována alespoň jedna událost. Lze totiž předpokládat, že v rámci většiny toků se bude vyskytovat nejvýše jeden incident a zbylé pakety jsou potom z hlediska další analýzy irelevantní. Mohl by se například vytvořit také mechanismus určité formy adaptivní akcelerace. Pokud by IDS samo o sobě stíhalo analyzovat všechny příchozí pakety, SDM by bylo neaktivní, takže by se spolehlivě detekovaly všechny potencionální incidenty. Až v případě nárůstu síťového provozu, kdy by IDS přestalo stíhat, by se teprve akcelerace pomocí SDM aktivovala. Následně by už docházelo k řízenému zahazování paketů na straně SDM, což by přineslo výrazně větší počet úspěšně detekovaných incidentů, než kdyby se pokračovalo v nekontrolovaném a náhodném zahazování na straně IDS. Další mechanismus, který by mohl vylepšit dosavadní systém, je adaptivní nastavení parametru threshold. Parametr by se nenastavoval staticky při spuštění SDM démona, ale dynamicky by se měnil za běhu tak, aby byla maximálně využita hardwarová paměť pravidel a aby se prostřednictvím IDS analyzoval co největší počet paketů. Tím by se udržoval co nejlepší poměr mezi úspěšností detekce a počtem paketů zpracovaných v SDM.

Kapitola 7

Závěr

Cílem bakalářské práce byla akcelerace systému IDS pro detekci bezpečnostních hrozeb v síti. Mnou navržený systém využívá k akceleraci koncept softwarově definovaného monitorování a je implementován a vyhodnocen pro dvě vybrané IDS aplikace – Snort a Suricata. Pro systém Suricata jsem implementoval i nativní rozhraní SZE2 a dva návrhy využití konceptu SDM. První varianta využívá hardwarově akcelerovanou podporu SDM na akcelerační síťové kartě s FPGA. Druhá varianta potom implementuje principy SDM v softwaru. Obě varianty jsou vzájemně porovnávány. Při vyhodnocení dosažených výsledků se přitom sledovala především celková propustnost systému a počet úspěšně detekovaných incidentů před a po akceleraci.

Na počátku jsem se v rámci řešení bakalářské práce nejprve důkladně seznámil se systémy IDS pro detekci bezpečnostních hrozeb v síti. Z dostupných informací a zdrojů jsem rozebral výhody a nevýhody tří nejrozšířenějších open-source IDS implementací – Snort, Suricata a Bro. Z teoretického pohledu se pro analýzu vysokorychlostních sítí jako nejvhodnější jeví systém Suricata, neboť jako jediný podporuje možnost vícevláknového zpracování paketů, což má zajisté zásadní vliv na celkovou výkonnost systému. Pro potřebu návrhu akcelerace jsem se dále seznámil s principy a možnostmi softwarově definovaného monitorování. Zjistil jsem si detaily současné implementace firmwaru SDM a teoretický rozbor jsem završil nastudováním funkcí softwarového řadiče, který je řídícím prvkem celého systému.

Pro vytvoření konceptuálního návrhu řešení akcelerace jsem se v prvé řadě zaměřil na analýzu vysokorychlostní sítě. Ověřil jsem charakteristické vlastnosti vysokorychlostní počítačové sítě, které jsou využitelné a významné pro nasazení konceptu softwarově definovaného monitorování. Nejvýznamnější vlastností je přitom zjištění, že pro síť je charakteristické tzv. heavy-tailed rozložení síťových toků. To znamená, že velmi malé procento nejdelších toků na síti má zároveň velmi výrazný podíl na celkovém počtu přenášených paketů. SDM dokáže takovéto těžké toky efektivně filtrovat a tím snížit zátěž CPU a systémové sběrnice. Dále jsem ověřil, že pro úspěšnou detekci většiny incidentů systémem IDS stačí analyzovat vždy pouze začátky síťových toků. Tím jsem ověřil počáteční předpoklad pro úspěšnou akceleraci IDS aplikací prostřednictvím SDM.

Pro první ověření funkčnosti jsem do systémů Snort a Suricata podle návrhu nejprve implementoval komunikaci s řadičem SDM. Pro příjem paketů ze sítě jsem použil rozhraní knihovny `libpcap` a následně získal výsledky. Díky použití SDM bylo v obou systémech patrné zlepšení propustnosti i zvýšení počtu úspěšně detekovaných incidentů. Protože lepšího výkonu dosahoval systém Suricata, pokračoval jsem v práci na rozvoji právě tohoto systému. Implementoval jsem nativní rozhraní SZE2 a také softwarovou variantu konceptu SDM. Poté ale vznikl problém s dostatečně rychlým přehráváním uloženého síťového pro-

vozu, jenž by ukázalo přínos akcelerace pomocí konceptu SDM. Důvodem byla limitující rychlost čtení z paměti, kde byl provoz uložen. Problém jsem vyřešil replikací provozu na vstupním síťovém rozhraní na akcelerační síťové kartě. Systém Suricata jsem následně upravil tak, aby dovedl rozlišovat původní a replikované síťové toky.

Druhým měřením jsem sledoval propustnost a počet detekovaných incidentů systému Suricata v případě zatížení všemi detekčními pravidly. S tolerancí ztrátovosti příchozích paketů 10 % na vyrovnávacích bufferech se u hardwarově akcelerované varianty při rychlosti 7 Gb/s propustnost zlepšila přibližně o 26 % a počet úspěšně detekovaných incidentů se zvýšil přibližně o 37 %. Pro třetí měření jsem vybral pouze některá detekční pravidla, téměř tisíc pravidel zaměřených jen na detekci malware. Pro nižší počet použitých pravidel má akcelerace daleko větší vliv. S tolerancí zahazování příchozích paketů 10 % na vyrovnávacích bufferech se u hardwarově akcelerované varianty při rychlosti 22,5 Gb/s propustnost zlepšila přibližně o 40 % a počet úspěšně detekovaných incidentů se zvýšil o 78 %.

Navržený systém je možné dále rozšiřovat a optimalizovat. Jednou z možností je implementace adaptivní akcelerace, kdy by se parametr threshold měnil dynamicky v průběhu analýzy síťového provozu podle potřeby. Dále se nabízí možnost implementovat nový mechanismus filtrování paketů síťového toku, jakmile je v daném toku detekován první incident. Způsob vyhodnocování výsledků lze dále vylepšit díky možným změnách měřicího prostředí a vhodnějším a systematickým nastavením parametrů pro SDM. Ve spolupráci se sdružením CESNET se v současné době připravuje také publikace výsledků práce na konferenci IEEE INFOCOM 2018.

Literatura

[1] *Fast hash filter*. [cit. 2017-04-25]. Dostupné na URL: https://github.com/CESNET/Nemea-Framework/tree/master/common/fast_hash_filter

[2] *Packet Acquisition API*. [cit. 2017-04-23]. Dostupné na URL: https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Packet_Acquisition_API

[3] *PcapPlusPlus*. [cit. 2017-04-22]. Dostupné na URL: https://github.com/seladb/PcapPlusPlus

[4] *Security Onion*. [cit. 2017-04-25]. Dostupné na URL: https://securityonion.net/

[5] *Supermicro X9DRG-QF*. Dostupné na URL: http://www.supermicro.nl/products/motherboard/Xeon/C600/X9DRG-QF.cfm

[6] *World's Biggest Data Breaches and Hacks*. [cit. 2017-04-23]. Dostupné na URL: http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/

[7] *Writing Snort Rules*. [cit. 2017-05-06]. Dostupné na URL: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node27.html

[8] *Command and control (malware)*. Wikimedia Foundation, 2001-. Dostupné na URL: https://en.wikipedia.org/wiki/Command_and_control_(malware)

[9] *Denial-of-service attack*. Wikimedia Foundation, 2001-. Dostupné na URL: https://en.wikipedia.org/wiki/Denial-of-service_attack

[10] *Checking Multiple Bits in a Flag Field*. 2008. [cit. 2017-04-21]. Dostupné na URL: http://blog.talosintelligence.com/2008/08/checking-multiple-bits-in-flag-field_29.html

[11] *New ET Users Guide*. 2013. [cit. 2017-05-08]. Dostupné na URL: http://doc.emergingthreats.net/bin/view/Main/NewUserGuideo

[12] *What Every IDS User Should Do*. 2013. [cit. 2017-05-08]. Dostupné na URL: http://doc.emergingthreats.net/bin/view/Main/WhatEveryIDSUserShouldDo

[13] *CESNET: e-infrastruktura pro vědu, výzkum a vzdělávání*. 2016. [cit. 2017-04-15]. Dostupné na URL: https://www.cesnet.cz/

[14] *General Rule Options: classtype*. 2016. [cit. 2017-04-20]. Dostupné na URL: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node31.html#Snort_Default_Classifications

[15] *ACOnet: Austrian Academic Computer Network*. 2017. [cit. 2017-04-15]. Dostupné na URL: https://www.aco.net/?L=1

[16] *Flowmon: Driving Network Visibility*. 2017. [cit. 2017-04-15]. Dostupné na URL: https://www.flowmon.com/cs

[17] *GSL - GNU Scientific Librar*. 2017. [cit. 2017-04-15]. Dostupné na URL: https://www.gnu.org/software/gsl/

[18] *Hackers Spend Christmas Break Launching Large Scale NTP-Reflection Attacks*. 2017. [cit. 2017-04-21]. Dostupné na URL: https://www.symantec.com/connect/blogs/hackers-spend-christmas-break-launching-large-scale-ntp-reflection-attacks

[19] *Known Bot Command and Control Rules*. 2017. [cit. 2017-04-20]. Dostupné na URL: http://doc.emergingthreats.net/bin/view/Main/BotCC

[20] *Suricata Open Source IDS/ IPS / NSM engine*. 2017. [cit. 2017-04-05]. Dostupné na URL: https://suricata-ids.org/

[21] *What is CINS?* 2017. [cit. 2017-04-20]. Dostupné na URL: https://sentinelips.com/cins/

[22] ALBIN, E. *A comparative analysis of the Snort and Suricata intrusion-detection systems*. Diplomová práce. Naval Postgraduate School, 2011. Dostupné na URL: http://www.dtic.mil/dtic/tr/fulltext/u2/a552115.pdf

[23] BEALE, J., BAKER, A., ESLER, J., aj. *Snort: IDS and IPS Toolkit*. Jay Beale's open source security series, Syngress, 2007. ISBN 9781597490993. Dostupné na URL: https://books.google.cz/books?id=M9plZZxJB_UC

[24] BRO *Frameworks - Bro 2.5 documentation*. 2014. [cit. 2017-04-11]. Dostupné na URL: https://www.bro.org/sphinx/frameworks/index.html

[25] CLAISE, B. *Cisco Systems NetFlow Services Export Version 9* [RFC 3954 (Informational)]. October 2004. Dostupné na URL: http://www.ietf.org/rfc/rfc3954.txt

[26] DAY, D., BURNS, B. A Performance Analysis of Snort and Suricata Network Intrusion Detection and Prevention Engines. *ICDS 2011, The Fifth International Conference on Digital Society*. 2011. Ročník 2011: str. 187. ISSN 2308-3956. Dostupné na URL: http://www.thinkmind.org/download.php?articleid=icds_2011_7_40_90007

[27] FERNANDEZ, M., PORRES, I., FÅK, V. *An Evaluation of current IDS*. Dizertační práce, Institutionen för systemteknik, 2008.

[28] GUSENICA, L. *Detekce malware v síťovém provozu [online]*. 2015 [cit. 2017-03-14]. [cit. 2017-03-28]. Dostupné na URL: http://is.muni.cz/th/396457/fi_b/

[29] HANK, A. *Detekce narušení počítačové sítě*. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií, 2006. Dostupné na URL: http://www.fit.vutbr.cz/study/DP/BP.php?id=5828

[30] KEKELY, L. *Hardwarová akcelerace aplikací pro monitorování a bezpečnost vysokorychlostních sítí*. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií, 2012. Dostupné na URL: http://www.fit.vutbr.cz/study/DP/DP.php?id=15490

[31] KEKELY, L., PUŠ, V., KOŘENEK, J. Software Defined Monitoring of Application Protocols. *Proceedings of the 33rd IEEE International Conference on Computer Communications*, INFOCOM 2014. Toronto, Canada, 2014, s. 282–284. ISBN 978-80-214-4694-6.

[32] KUČERA, J. *Aplikačně specifický procesor pro stavové zpracování síťových dat*. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií, 2013. Dostupné na URL: http://www.fit.vutbr.cz/study/DP/BP.php?id=16018

[33] LARSEN, R. *Slow Port Scanning with Bro*. Dizertační práce, Department of Computer Science and Media Technology Gjøvik University College, 2013. Dostupné na URL: https://brage.bibsys.no/xmlui/bitstream/handle/11250/144028/RLarsen_2013.pdf?sequence=1&isAllowed=y

[34] LOCOCO, M. *Capacity Planning for Snort IDS*. 2011. [cit. 2017-03-28]. Dostupné na URL: http://mikelococo.com/2011/08/snort-capacity-planning/

[35] NORTHCUTT, S., ZELTSER, L. *Inside network perimeter security*. Sams Publishing, 2005. ISBN 0672327376.

[36] NTOP *PF RING: High-speed packet capture, filtering and analysis*. [cit. 2017-04-05]. Dostupné na URL: http://www.ntop.org/products/packet-capture/pf_ring/

[37] PAXSON, V. Bro: A System for Detecting Network Intruders in Real-time. *Comput. Netw.* Prosinec 1999. Ročník 31, č. 23-24: s. 2435–2463. ISSN 1389-1286. Dostupné na URL: http://dx.doi.org/10.1016/S1389-1286(99)00112-7

[38] PIHELGAS, M. *A Comparative Analysis of Open-Source Intrusion Detection Systems*. Dizertační práce, Tallinn University of Technology, 2012. Dostupné na URL: https://www.etis.ee/File/DownloadPublic/538ba91a-476c-4573-b35d-440c7cf37f3c?name=Mauno_Pihelgas-A_Comparative_Analysis_of_OpenSource_Intrusion_Detection_Systems.pdf&type=application%2Fpdf

[39] PROOFPOINT *Emerging Threats.net Open rulesets*. 2017. [cit. 2017-03-27]. Dostupné na URL: https://rules.emergingthreats.net/

[40] RØDFOSS, J. T. *Comparison of Open Source Network Intrusion Detection Systems*. Dizertační práce, Oslo University College, 2011. Dostupné na URL: https://www.duo.uio.no/bitstream/handle/10852/8951/Rodfoss.pdf?sequence=1&isAllowed=y

[41] SNORT *Shared Object Rules*. 2017. [cit. 2017-03-29]. Dostupné na URL: https://www.snort.org/faq/shared-object-rules

[42] SNORT *Snort++ Alpha 4 Available Now!* 2017. [cit. 2017-03-25]. Dostupné na URL: http://blog.snort.org/2017/03/snort-alpha-4-available-now.html

[43] SNORT *What are Community Rules?* 2017. [cit. 2017-04-05]. Dostupné na URL: https://www.snort.org/faq/what-are-community-rules

[44] STOFFER, V., SHARMA, A., KROUS, J. *100G Intrusion Detection*. 2015. [cit. 2017-03-26]. Dostupné na URL: http://commons.lbl.gov/download/attachments/120063098/100GIntrusionDetection.pdf

[45] SURICATA *Suricata Manual*. [cit. 2017-04-01]. Dostupné na URL: http://suricata.readthedocs.io/en/latest/

[46] SVOBODA, J. *Analýza síťového provozu metodou hloubkové inspekce paketů [online]*. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno, 2014 [cit. 2017-03-14]. Dostupné na URL: http://is.muni.cz/th/250890/fi_m/

[47] TALOS *Talos*. 2017. [cit. 2017-04-01]. Dostupné na URL: http://www.talosintelligence.com/

[48] ČERNÝ, M. *Analýza nedostatků IDS systémů*. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií, 2008. Dostupné na URL: http://www.fit.vutbr.cz/study/DP/BP.php?id=7839

[49] ŠUMSKÝ, D. *Systémy detekce průniku [online]*. Diplomová práce. Masarykova univerzita, Fakulta informatiky, Brno, 2006 [cit. 2017-03-06]. Dostupné na URL: http://is.muni.cz/th/51653/fi_m/

Přílohy

Příloha A

Obsah CD

```
/
├── app/
│   ├── context_founder/
│   ├── PcapSplitter/
│   └── rules_inserter/
├── CLI_tisk/
├── grafy/
├── incidenty/
├── mereni/
├── sdm_daemon/
├── suricata/
│   └── src/
├── text/
├── bp-xpiece00.pdf
└── README.txt
```

Adresář `app/` obsahuje pomocné aplikace, které byly implementovány pro potřeby bakalářské práce. Aplikace `context_founder` slouží pro založení kontextu u řadiče SDM a využívala se u měření, kde IDS nevyužívalo akceleraci SDM. Aplikace `PcapSplitter` přečte vstupní `pcap` soubor a z každého síťového toku zkopíruje určitý počet n paketů, které následně uloží do nového `pcap` souboru. Aplikace musí běžet ve frameworku `PcapPlusPlus` a v práci byla použita v podkapitole 4.2. Aplikace `rules_inserter` byla poprvé zmíněna v podkapitole 5.1. Jak už bylo zmíněno, aplikace načítá data z rozhraní SZE2, analyzuje a extrahuje informace z hlaviček paketů a předává tyto informace řadiči SDM.

Adresář `CLI_tisk/` obsahuje skripty, jenž poskytují v příkazovém řádku rychlý náhled na proběhlé měření a další drobné funkce.

Adresář `grafy/` obsahuje skripty v jazyce `Python`, které vytvořily grafy v této práci.

Adresář `incidenty/` obsahuje skripty, jenž pomohly při analýze síťových toků z pohledu detekce incidentů.

Adresář `mereni/` obsahuje skripty, které spouští samotné měření a zároveň ukládají užitečné informace do logů.

Adresář `sdm_daemon/` obsahuje modul pro softwarově implementovanou verzi SDM. Modul nahrazuje komunikaci s firmwarem SDM za komunikaci s komponentou SDM-SOFT (viz podkapitola 5.3).

Adresář `suricata/` obsahuje rozšiřující moduly a úpravy některých zdrojových souborů v systému Suricata. Implementována je zde hardwarová i softwarová podpora pro koncept SDM a také nativní podpora rozhraní SZE2.

Adresář `text/` obsahuje zdrojové soubory textu bakalářské práce.

Soubor `bp-xpiece00.pdf` obsahuje text bakalářské práce ve formátu `pdf`.

Soubor `README.txt` obsahuje informace o adresářové struktuře a pokyny ke zprovoznění.