Jihočeská univerzita
Přírodovědecká fakulta
Ústav aplikované informatiky

Výzkum a implementace open-source řešení IDS/IPS systému jako ochranu proti aktuálně probíhajícím hrozbám v prostředí velkého poskytovatele internetových služeb

Autor: Bc. Tomáš Starý
Vedoucí práce: Ing. Rudolf Vohnout, Ph.D.

České Budějovice 2019

Bibliografické údaje

Starý T., 2019: Výzkum a implementace open-source řešení IDS/IPS systému jako ochranu proti aktuálně probíhajícím hrozbám v prostředí velkého poskytovatele internetových služeb [Research and implementation of open-source IDS / IPS system solutions as protection against current threats in a large web host provider environment. Master Thesis, in Czech.] - 86 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic

Anotace

Cílem diplomové práce je vytvořit řešení IDS/IPS systému pomocí platformy Docker, které bude sloužit k filtraci nebezpečného/nechtěného síťového provozu v prostředí velkého poskytovatele internetových služeb. Filtrace bude probíhat na základě modifikovaných původních bezpečnostních pravidel a pravidel zcela nových. Na základě nových pravidel dojde ke komparaci situace současné a předešlé.

Annotation

The aim of the thesis is to create an IDS/IPS system solution using the Docker platform, which will be used to filter unsafe/unwanted network traffic in a large web host provider environment. Filtration will be based on modified original safety rules and completely new rules. Based on the new rules, the current and the previous situation will be compared.

Klíčová slova

IDS/IPS, Docker, Kontejner, Image, Bezpečnostní pravidla

Keywords

IDS/IPS, Docker, Container, Image, Security rules

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

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

V ..... dne ..... Podpis autora

Poděkování

Rád bych poděkoval panu Ing. Rudolfu Vohnoutovi, Ph.D. za umožnění zabývat se tímto zajímavým tématem, za odborné rady a celkovou korekturu práce. Také bych rád poděkoval své přítelkyni a rodině za velkou podporu při tvorbě práce. V neposlední řadě bych rád poděkoval panu Mgr. Josefu Grillovi a ostatním pracovníkům z firmy WEDOS Internet, a.s., bez kterých by byla práce jen těžko realizovatelná.

Obsah

Úvod

Při vývoji aplikací je v dnešní době stále větší zájem o možnou přenositelnost výsledné aplikace na různé platformy. Možnost aplikaci spustit na zařízení s operačním systémem Windows, macOS a s různými linuxovými distribucemi. Možnost spustit aplikaci na různých typech zařízení od virtuálních serverů, přes běžné počítače až po cloudové řešení. Všechny tyto možnosti lze zajistit pomocí open-source platformy Docker, která umožňuje výslednou aplikaci zabalit do docker obrazu a spustit ji oddělenou od prostředí, ve kterém má být spuštěna, v docker kontejneru. Tato vlastnost má hned několik výhod. Všechny potřebné vývojové knihovny a závislosti jsou součástí obrazu a pokud je potřeba aplikaci odstranit, je smazán celý kontejner. Po aplikaci nezůstávají žádné zbytkové soubory. Změny, které jsou provedeny uvnitř kontejneru, jsou po smazání kontejneru ztraceny. V případě potřeby uložení provedených změn lze vytvořit obraz nový z obrazu předešlého, což vývojáři dává možnost verzování jednotlivých vývojových fází aplikace.

Jednou z možných skupin aplikací, kterou lze převést do prostředí platformy Docker, je skupina IDS/IPS systémů. IDS/IPS systémy jsou v dnešní době nepostradatelnou součástí každé vetší počítačové sítě. Zajišťují upozornění na nelegitimní a neautorizovanou síťovou komunikaci a v případě IPS módu také okamžitou automatickou reakci na detekovaný problém. V prostředí velkého poskytovatele hostingových služeb je IDS/IPS systém samozřejmostí. Nicméně jeho typ a podoba se může u každé společnosti výrazně lišit. IDS/IPS systémy lze rozdělit do několika skupin podle typu použití a metody detekce. Podle typu použití se IDS/IPS systémy dělí na uzlově orientované systémy, na síťově orientované systémy a na hybridní řešení, které v sobě kombinuje obě předchozí varianty. Podle metody detekce je možno IDS/IPS systémy rozdělit na dvě hlavní skupiny. První skupinou jsou systémy založené na detekci anomálií a druhou skupinou jsou systémy založené na detekci pomocí pravidel. Pro kontrolu síťového provozu lze použít bezpečnostní pravidla defaultní, která byla vytvořena různými bezpečnostními agenturami nebo vytvořit pravidla vlastní. Pravidla vlastní jsou určena pro případ, že se v počítačové síti vyskytuje provoz, pro který nebyla doposud vytvořena defaultní bezpečnostní pravidla. Tvorba nových bezpečnostních pravidel je závislá na konkrétním IDS/IPS systému. Každý IDS/IPS systém má svou syntaxi pro psaní nových bezpečnostních pravidel a také své různé metody detekce, které lze do pravidel zanést. [2], [4], [9], [13], [15], [16]

1. Metodika

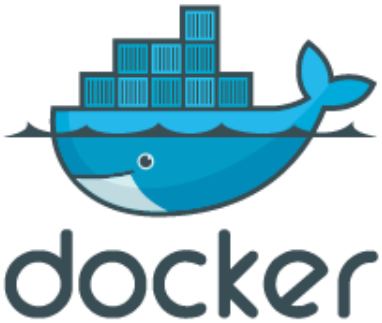
Prvotní fází diplomové práce bylo seznámení se s platformou Docker. Jak tato platforma funguje a jak je ji možno použít ve firemním prostředí společnosti WEDOS Internet, a.s. Po této prvotní fázi došlo k prozkoumání dalších potřebných částí, kterými byly IDS/IPS systémy. Pomocí IDS/IPS systémů je možno redukovat či úplně odstranit nebezpečný/nepotřebný síťový provoz, jenž firmu postihuje. I když společnost již jeden IDS/IPS systém ve svém síťovém prostředí používala, bylo umožněno pro účel této práce vybrat systém znovu, a to na základě pevně stanovených požadavků. Z tohoto důvodu byly vybrány čtyři open-source IDS/IPS systémy, u kterých byly specifikovány jejich vlastnosti. Na základě kritérií stanovených společností WEDOS Internet, a.s. došlo k výběru jednoho z těchto IDS/IPS systémů a tím byl systém Suricata. Výsledně vybrané řešení se na základě stanovených kritérií jevilo jako nejlepší možnou volbou, a navíc se jednalo o systém, se kterým měla firma již zkušenosti. Po vybrání konkrétního IDS/IPS systému byly prozkoumány části dokumentace s bezpečnostními pravidly, jenž po vytvoření výsledného řešení tvořily jednu z hlavních rolí této diplomové práce.

Tvorba výsledného řešení IDS/IPS systému Suricata v prostředí Docker byla rozdělena do několika verzí, přičemž každá verze přidávala konkrétní funkcionalitu či opravovala nedostatky verze předešlé. Vývoj a testování probíhalo na testovacím prostředí virtuálního serveru a až finální verze byla přesunuta na firemní servery společnosti WEDOS Internet, a.s. Po nasazení finálního řešení bylo zapotřebí použít některá pravidla z již používaného firemního IDS/IPS. Nicméně tato pravidla bylo nutno rozdělit na dvě části. První část se zaměřovala na defaultní pravidla a část druhá na pravidla dříve vytvořená samotnou společností. Dalším krokem bylo specifikovat, jaká pravidla byla stále používána a jaká nikoli a skutečně používaná pravidla přenést do řešení nového.

Poslední částí diplomové práce bylo vytvoření zcela nových pravidel odpovídajících na současné hrozby postihující zákazníky společnosti. Na základě nových pravidel byl proveden test jejich účinnosti spočívající ve srovnání stavu před a po jejich nasazení.

2. Platforma Docker

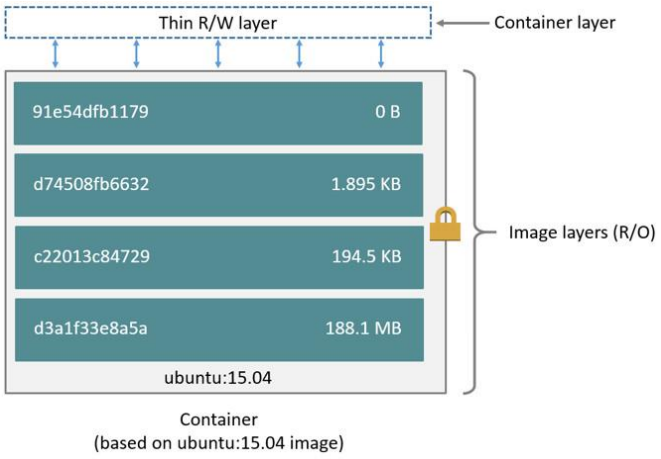
Docker byl poprvé představen na konferenci pro vývojáře v jazyce Python, zakladatelem a generálním ředitelem firmy dotCloud panem Solomonem Hykes. Tato konference se konala ve městě Santa Clara 15. března 2013. Do této doby se jednalo o neznámou platformu, kterou používala mimo firmu dotCloud pouze malá skupina lidí. Po konferenci se zájem o platformu značně rozšířil. Z tohoto důvodu se vývojáři rozhodli vydat platformu pod open-source licencí a umožnit volnou distribuci. Během jednoho roku se pojem Docker velice rozšířil mezi odbornou komunitou. Nicméně mnoho lidí si stále nebylo jisto, o jakou platformu se jedná a jak přesně funguje. [1], [2]



Obrázek č. 1 Logo platformy Docker [3]

Platforma Docker je open-source nástroj pro vývoj a spouštění aplikací nezávislých na prostředí, na kterém mají být provozovány. Aplikace jsou přenosné mezi zařízeními a mohou být spuštěny na počítačích, virtuálních počítačích či serverech, v cloudovém prostředí a také v datacentrech, nezávisle na operačním systému. Aby mohla být softwarová aplikace spuštěna, často musí kromě svého kódu obsahovat také systémové a běhové knihovny a závislosti potřebné pro spuštění. Platforma Docker vývoj aplikací zjednodušuje a umožňuje všechny tyto potřebné části aplikace zabalit do jedné jednotky zvané Docker image. Docker image je možno vytvořit z takzvaného souboru Dockerfile. Soubor Dockerfile obsahuje příkazy pro tvorbu vlastního obrazu a měl by být umístěn přímo v kořenovém adresáři aplikace. Pro vytvoření docker image ze souboru Dockerfile slouží příkaz „`docker build`“. Docker image je složen z jedné nebo více vrstev. Tyto vrstvy jsou napojeny na jednotlivé kroky vedoucí k vytvoření samotného obrazu. [2] Vrstvy jsou navzájem propojeny a jsou seřazeny nad sebou. Jednotlivé vrstvy

obsahují pouze rozdíly od vrstev předchozích. Všechny vrstvy jsou v režimu read-only, kromě poslední, která je přidána až po spuštění cílového kontejneru. To zajišťuje jednu z hlavních vlastností platformy Docker. Pokud dojde k vytvoření funkčního obrazu, lze se k tomuto obrazu vždy vrátit i po havárii kontejneru. Schéma vrstev kontejneru je zobrazeno na následujícím obrázku č. 2. Na obrázku je patrné rozdělení vrstev obrazu a kontejneru. Kontejner je v tomto příkladě založen na obrazu operačního systému Ubuntu 15.04 a obsahuje několik dalších vrstev image, které jsou pouze pro čtení. Nad tyto vrstvy je na závěr přidána vrstva kontejneru, s kterou je možno nadále pracovat v režimu read/write.

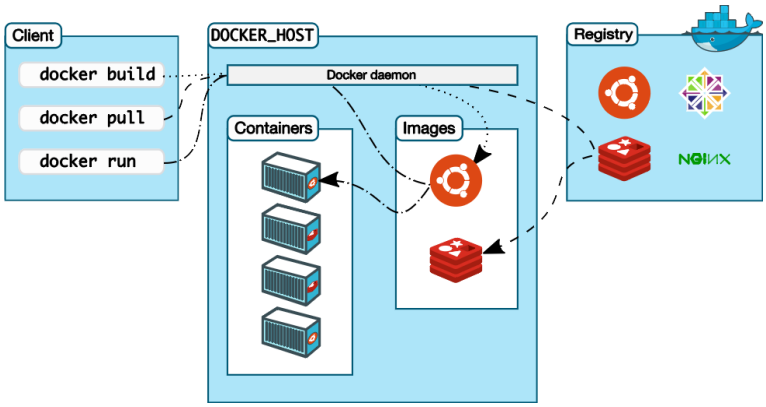


Obrázek č. 2 Zobrazení vrstev kontejneru [4]

Všechny změny, které jsou provedeny v kontejneru, jsou zapisovány do nejvrchnější vrstvy. Pokud dojde ke smazání kontejneru, všechny změny, které byly v kontejneru provedeny, budou také smazány. Nicméně obraz, ze kterého byl kontejner spuštěn, zůstane nezměněn. Vzhledem k tomu, že při každém spuštění nového kontejneru se vytvoří vlastní vrstva v režimu read/write a všechny změny jsou zapisovány právě do této vrstvy a nikoli do vrstev vytvořených obrazem, lze stejný obraz používat pro tvorbu více kontejnerů, které jsou na sobě nezávislé. V případě nutnosti uchovat změny, které byly provedeny přímo v kontejneru, je možné použít perzistentní úložiště. Perzistentní úložiště je složka či soubor, které se nacházejí přímo na hostitelském zařízení a jsou připojeny přímo do vnitřku kontejneru. Do kontejneru může být připojeno několik perzistentních úložišť a také lze jedno perzistentní úložiště sdílet najednou s více kontejnery. [4]

2.1. Architektura

Platforma Docker se zdá být velice složitá vzhledem k jejím možnostem. Nicméně jejím základem je jednoduchý model klient/server. Model platformy Docker je složen nejméně ze dvou částí. První částí je klient a druhou částí je server/démon. Část server se stará o spuštění a správu Docker objektů. Mezi tyto objekty patří Docker obrazy, kontejnery, sítě a úložiště. Klient je ze strany uživatele používán k tomu, aby sdělil serveru, jaký úkon je potřeba vykonat. Provoz Docker démona může být realizován na více fyzických serverech. V důsledku toho může klient využít více serverů ve stejný čas. Další částí, která je volitelná, jsou registry, které slouží pro uložení Docker obrazu a doplňujících informací o těchto datech. Příkladem Docker registru je Docker Hub, který je veřejně dostupný a je v počátečním nastavení určen jako výchozí registr. Ovšem je možné používat i soukromé registry. Docker obrazy lze stahovat z registru ale lze je do registru také nahrávat. Veškerá komunikace je řízena klientem, ale pokud Docker server požádá o přímou komunikaci s obrazovými registry, je přímá komunikace Docker serveru umožněna. Za řízení serverů jsou zodpovědní klienti a servery se naopak starají o správu kontejnerových aplikací. Model klient/server je znázorněn na obrázku č. 3.



Obrázek č. 3 Model klient/server v prostředí Docker [5]

Od běžných aplikací s modelem klient/server se Docker model trochu liší. Klient a server používají stejný binární kód, namísto toho, aby jej měly oddělený. Části klient a server jsou dostupné ihned po nainstalování Docker prostředí. Spuštění Docker serveru/démona je velice jednoduché. Lze jej přirovnat k jednoduchému spuštění kontejneru s parametrem „-d“, který zajistí, aby se kontejner choval jako démon a registroval příchozí komunikaci. Zpravidla bude mít hostitel platformy Docker spuštěn jeden Docker démon, který se bude

starat o vytvořené kontejnery a klient pomocí příkazového řádku bude určovat co má server vykonávat. [1], [5]

2.1.1. Docker objekty

Při práci s prostředím Docker jsou využívány takzvané Docker objekty. Jak bylo výše zmíněno, mezi tyto objekty patří například docker image, kontejner a služby. Jednotlivé Docker objekty budou popsány níže.

Docker Image

Docker image je šablona v read-only módu, která obsahuje jednotlivé kroky pro vytvoření Docker kontejneru. Docker image je zpravidla založen na jiném obrazu, který je použit jako základ a nad tento základní obraz jsou uloženy uživatelské úpravy potřebné pro výslednou aplikaci. Například pro vytvoření nového Docker image lze použít základní obraz se systémem Ubuntu a následně přidat ještě instalaci webového či databázového serveru s přesnou konfigurací pro výslednou aplikaci. Obrazy lze tvořit vlastní, případně lze použít nějaký veřejný registr například Docker Hub a stáhnout obraz již někým vytvořený. Pro vytvoření vlastního Docker image slouží Dockerfile, který určuje postup pro vytvoření obrazu a jeho spuštění. Jak bylo již popsáno v úvodní kapitole, každý krok v souboru Dockerfile vytváří jednu read-only vrstvu. V případě jakékoli modifikace souboru Dockerfile a následnému vytvoření obrazu dochází k úpravám pouze vrstvy, u které v DockerFile došlo k modifikaci. Tato vlastnost značně urychluje vytvoření modifikovaného obrazu. [5]

Docker kontejner

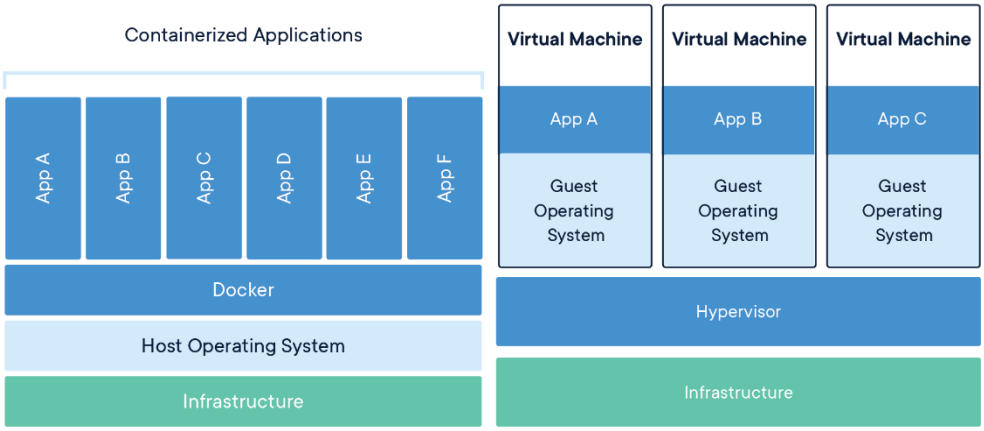
Docker kontejner je možno označit jako běžící instanci konkrétního obrazu. Kontejner lze připojit do jedné či více sítí a také umožňuje připojení jednoho či více úložišť. Z již vytvořeného kontejneru a jeho aktuálního stavu se dá také vytvořit nový Docker obraz. Každý kontejner je v počátečním nastavení velice dobře izolovaný od hostitelského zařízení či jiných kontejnerů. Nastavení izolace lze nicméně libovolně měnit a umožnit tak přístup kontejneru k sítím, úložištím a dalším subsystémům jiných kontejnerů či hostitelského zařízení. Výsledné nastavení a podoba kontejneru je definována základním obrazem a parametry při jeho spouštění. Všechny změny, které jsou provedeny uvnitř kontejneru a nejsou uloženy na perzistentní úložiště, jsou odstraněním kontejneru ztraceny. [5]

Services

Docker services neboli služby, vytváří skupiny složené z kontejnerů a Docker démonů. Celou skupinu si lze představit jako skupinu, jenž má více manažerů (Docker démonů) a více pracovníků (kontejnerů). Pomocí služeb lze nastavit cílový stav, například počet duplicit v určitý okamžik. Služby jsou nastaveny tak, aby v počátečním nastavení vyrovnávaly výkon napříč všemi pracovními uzly. [5]

2.2. Platforma Docker vs. virtuální stroje

Platforma Docker umožňuje běh kontejnerů na aplikační vrstvě řídící zabalení samotného kódu aplikace a všech potřebných závislostí. Na jednom hostitelském zařízení může být spuštěno několik kontejnerů a navzájem sdílet jádro operačního systému, přičemž každý kontejner je spuštěn jako izolovaný proces. Kontejnery jsou mnohem méně náročné na velikost oproti virtuálním strojům. Virtuální stroje jsou abstrakcí fyzického hardware. Pomocí virtualizace lze rozdělit jeden fyzický hardware na několik virtuálních strojů. Aby bylo možné na fyzickém hardware pracovat s virtuálními stroji je nutné mít nainstalován hypervizor. Každý virtuální stroj obsahuje svůj vlastní operační systém a aplikace. [6] Na následujícím obrázku č. 4. je patrné, jaké jsou rozdíly mezi platformou Docker a virtuálními stroji. První vrstva je jak pro platformu Docker, tak i virtuální stroje totožná. Jedná se vrstvu fyzického hardware (infrastrukturu). Může se jednat o klasický počítač či server. V případě platformy Docker to může být dokonce i virtuální stroj. Druhou vrstvou je vrstva operačního systému. Pro platformu Docker je tato vrstva nutná. Mezi hlavní operační systémy, kde je možné prostředí Docker provozovat, patří různé linuxové distribuce, MacOS a Windows. U virtuálních strojů je tato vrstva přítomna pouze v případě, že se jedná o virtualizaci, která je nainstalována až na konkrétním operačním systému. Pokud se jedná o virtualizaci, která je spuštěna přímo na fyzickém hardware, tak vrstva s operačním systémem, jak je patrné na obrázku, chybí a je použita rovnou vrstva s hypervizorem. Mezi virtualizaci instalovanou přímo na fyzický hardware patří například Hyper-V od firmy Microsoft nebo VMware ESX. Vrstva s hypervizorem slouží u virtuálních strojů pro jejich řízení a komunikaci s fyzickým hardware. [7], [8]



Obrázek č. 4 Rozdíly mezi virtuálními stroji a platformou Docker [6]

Místo vrstvy s hypervizorem obsahuje platforma Docker vrstvu Docker. Jedná se o Docker démona, který je spuštěn na operačním systému hostitelského zařízení a slouží pro řízení kontejnerů. Poslední vrstvou u platformy Docker jsou samotné kontejnery obsahující výsledné aplikace, běhové a systémové knihovny a potřebné závislosti pro spuštění dané aplikace. Ve schématu virtuálních strojů jsou poslední vrstvou samotné virtuální stroje. Nicméně oproti platformě Docker musí mimo samotné aplikace, běhové knihovny a závislosti pro běh těchto aplikací obsahovat také samostatný operační systém, na kterém jsou tyto aplikace nainstalovány. [7], [8]

2.3. Výhody a nevýhody platformy Docker

Jako každá technologie má i platforma Docker své výhody a nevýhody. V této práci lze nalézt výhody a nevýhody v bodovém seznamu. Po vyjmenování bodového seznamu budou jednotlivé výhody a nevýhody podrobněji popsány.

Výhody:

- Úspora na nákladech
- Standardizace
- Kompatibilita napříč platformami
- Rychlé nasazení
- Podpora ze strany poskytovatelů cloudových řešení
- Izolace
- Bezpečnost

Nevýhody:

- Omezení rychlosti
- Složitost perzistentních úložišť
- Grafické prostředí běžících aplikací
- Kompatibilita

První výhodou platformy Docker je úspora na nákladech. K úspoře dochází při spouštění stejných aplikací v prostředí Docker, které spotřebovávají méně hardwarových prostředků. Omezení výdajů vynaložených na infrastrukturu, může vést také k omezení pracovníků, jenž danou infrastrukturu spravují a k opětovné úspoře financí. Druhou výhodou je standardizace. Platforma Docker poskytuje vývojové, testovací a produkční prostředí. Z již běžícího kontejneru je možno vytvořit novou verzi Docker image. V takovém případě je možné jednotlivé Docker image označovat verzemi a v případě problémů se vracet ke starším funkčním verzím obrazů. Jedná se o velice rychlé testování a nasazení výsledné aplikace. Další výhodou je kompatibilita napříč platformami. Prostředí Docker je nezávislé na prostředí, kde se nachází (notebook, server, virtuální stroj). To umožňuje redukovat čas strávený na nastavení jednotlivých pracovních stanic vývojářů. Rychlost nasazení je jednou z hlavních výhod. Velká rychlost při vytváření kontejneru je způsobena tím, že není nutné zavádět operační systém stejně jako je tomu v prostředí virtuálních strojů. Jednou z velice důležitých výhod je také podpora ze strany poskytovatelů cloudových řešení. Mezi hlavní poskytovatele lze zařadit Amazon Web Services (AWS) a také Google Compute Platform (GCP). Dalšími cloudovými prostředími, kde je Docker platformu možné provozovat, jsou například Microsoft Azure a OpenStack. Prostředí Docker lze také nainstalovat na virtuálních strojích vytvořených pomocí programu VirtualBox od společnosti Oracle. Předposlední výše uvedenou výhodou je izolace. Kontejnery jsou v základním nastavení od sebe navzájem izolované a také izolují aplikaci běžící v kontejneru od hostitelského zařízení. Platforma Docker také provádí oddělení zdrojů pro jednotlivé kontejnery, což zaručuje, že jednotlivé aplikace využijí pouze zdroje, které jim byly přiděleny. V případě odstranění aplikace dochází ke smazání celého kontejneru a v hostitelském zařízení nezůstávají žádné dodatečné soubory spojeny se smazanou aplikací. Poslední výše zmíněnou výhodou je bezpečnost. Vzhledem k předešlé výhodě je bezpečnost zaručena právě pomocí izolace.

Izolace kontejnerů navzájem zamezuje komunikaci kontejnerů mezi sebou a je tak jednodušší kontrolovat správu provozu. [9]

První nevýhodou, v již uvedeném bodovém seznamu, je omezení rychlosti. Rychlost je stejně jako u virtuálních strojů pomalejší oproti běhu přímo na fyzickém hardware. Nicméně ve srovnání s virtuálními stroji je na tom platforma Docker podstatně lépe. Omezená rychlost je způsobena tím, že na jednom fyzickém stroji běží více kontejnerů a ty si mezi sebou musí rozdělit dostupné zdroje jako je například síťové rozhraní. Z výhody „izolace“ bohužel plyne i jedna nevýhoda, kterou je složitost perzistentních úložišť. Jak již bylo zmíněno, po smazání kontejneru všechna data, která nebyla uložena na perzistentním úložišti, zmizí. Proto je nutné se zaměřit na zálohování a obnovu dat. U většího počtu kontejnerů může být tento problém poměrně komplexní. Předposlední nevýhodou v seznamu je grafické prostředí běžících aplikací. Platforma Docker byla vždy zaměřena především na serverové aplikace a nikoli na aplikace s GUI. I když je možné použít nástroj X11, který umožňuje vytvářet grafické rozhraní uvnitř kontejneru, není ani toto řešení problému zcela ideální. Poslední uvedenou nevýhodou je kompatibilita. Kompatibilita patří i mezi výhody. Nicméně zde se nejedná o kompatibilitu přímo platformy Docker ale některých nástrojů, které jsou s platformou spojeny. Firmy podporující tuto platformu samozřejmě vytváří i své produkty, které nemusí vždy být kompatibilní s produkty jiných podporujících společností. Typickým příkladem je software OpenShift. Tento software je možné použít pouze s orchestračním nástrojem Kubernets. [10], [11], [12]

3. IDS/IPS

IDS neboli Intrusion Detection System slouží pro identifikaci neautorizované a nelegitimní síťové činnosti pomocí sady nástrojů, metod a zdrojů. Na rozdíl od firewallu IDS systémy nejsou určeny pro zamezení přístupu ale k upozornění, že se někdo o neautorizovaný přístup pokusí. IDS pracují na třetí vrstvě síťového modelu TCP/IP. Sběr analyzovaných dat probíhá za pomoci pasivních senzorů umístěných na regulačních bodech. Na základě předem vytvořených pravidel dochází k analýze jednotlivých packetů a pokud dojde ke shodě je vygenerováno upozornění, které je zapsáno do logů. Na základě těchto logů může dojít k odpovídající reakci na vzniklý problém.

IPS (Intrusion Prevention Systems) je vyvinut z předešlého systému IDS. Na rozdíl od samotného IDS, které poskytuje pouze detekci nelegitimního provozu a generování upozornění, umožňuje IPS systém provádět aktivní reakci na identifikovaný problém bez nutnosti zásahu správce sítě. Tato reakce spočívá v blokaci dané komunikace, či naprostém zamezení příchozích či odchozích packetů z konkrétní IP adresy. Tento typ detekce se zapojuje do sítě sériově, tak aby veškerá komunikace procházela přes tuto kontrolu. Pro úspěšné fungování systému je nutná správná konfigurace, aby nedocházelo k častým planým upozorněním a nesprávné blokaci komunikace. Stejně jako u IDS lze IPS systémy rozdělit na uzlově orientované (HIPS) a síťově orientované (NIPS). Rozdělení druhů IDS je popsáno v následující kapitole. [13], [15]

3.1. Druhy IDS systémů

IDS se dají rozdělit do tří skupin. První skupinou jsou uzlově orientované systémy detekce narušení HIDS, druhou skupinou jsou síťově orientované systémy detekce narušení NIDS a poslední skupinou je kombinace předešlých dvou skupin.

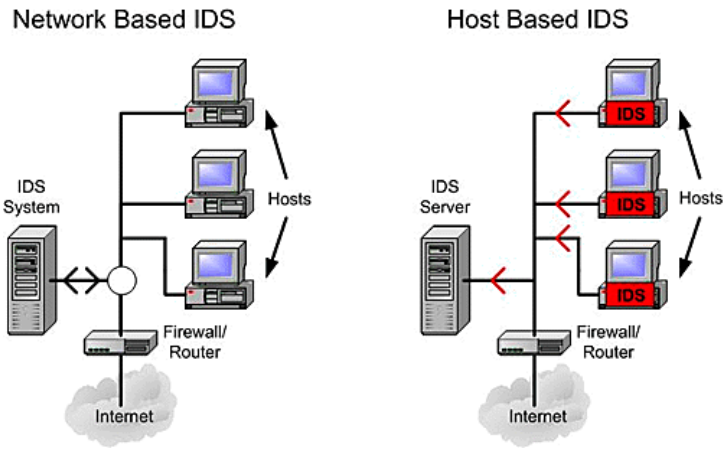
3.1.1. HIDS

Uzlově orientované systémy detekce narušení jsou instalovány na klientských stanicích jako specifický software, jenž umožňuje skenování systémových aktivit a událostních logů. Po proskenování dochází k porovnání s bezpečnostní databází obsahující nebezpečné události a pokud je nalezena shoda dochází k vygenerování upozornění. Jelikož je software instalován přímo na klientské stanici je přímo závislý na operačním systému. Z toho vyplývá problém s komplikovanější a dražší implementací, jelikož na

každém operačním systému dochází k různému nastavení. HIDS jsou vhodné pro detekci interního napadení. Umožňují rychlou detekci lokálních útoků na klientských stanicích a tím pádem zabránění napadení celé sítě. [13]

3.1.2. NIDS

Síťově orientované systémy detekce narušení se ve většině případů umísťují do sítě sériově, čímž se liší od předešlého druhu HIDS. Umístění obou druhů systému je znázorněno na obrázku č. 5. Z tohoto obrázku je patrné, že NIDS je umístěno v bodě, kudy proudí veškerý provoz do sítě. Komunikace je posílána na analýzu prostřednictvím větvení sítě či zrcadlení portů. Následně dochází k důkladné rekonstrukci komunikace, na které je provedena analýza. Při objevení nelegitimního provozu dochází k hlášení systému a vygenerování upozornění.



Obrázek č. 5 Umístění network a host based IDS [14]

3.1.3. Hybridní řešení

Hybridní řešení kombinuje obě metody, HIDS sloužící pro odhalení nebezpečných událostí na uzlovém systému a NIDS, jenž slouží ke kontrole a sledování síťové komunikace. Hybridní řešení poskytuje vyšší zabezpečení celé počítačové sítě. Nicméně je velice komplikované na implementaci a následnou správu sítě. [13], [15]

3.2. Metody detekce IDS/IPS

Metody detekce, jenž používají IDS/IPS systémy, se rozdělují na dvě základní třídy. Na detekci založenou na pravidlech a detekci založenou na anomáliích.

3.2.1. Detekce založená na pravidlech

Detekce založená na pravidlech, taky známá jako detekce zneužití, pracuje na principu analýzy shromážděných dat a jejich následném porovnání se záznamy obsahující popis již objevených útoků. Toto řešení je podobné jako u antivirových programů, které obsahují databázi známých škodlivých kódů. Funkčnost detekce útoků je pevně spjata s kvalitou používané databáze pravidel. Pokud je použita kvalitní databáze pravidel, tak téměř nedochází ke špatnému vyhodnocení útoků. Toto řešení je často využíváno jako základní metoda detekce u systému IDS/IPS. [13], [16]

3.2.2. Detekce založená na anomáliích

Detekce anomálií, také označována jako profilově orientovaná detekce, se od ostatních metod liší tím, že je definován soubor činností, které jsou zakázány, ale také činností, které jsou povoleny. Správce systému musí předem definovat běžný provoz na síti, do kterého se řadí například využívané protokoly, obvyklá velikost packetů a jejich četnost. Systém poté sleduje odchylky od stanoveného typického chování sítě a na základě zjištění jiného chování detekuje probíhající útoky. Tento princip umožňuje odhalit i neznámé typy útoků, které prozatím nebyly do bezpečnostních databází přidány. Nevýhodou tohoto řešení je problematické nastavení typického chování na síti. Pokud je typický stav sítě stanoven špatně, vznikají takzvané False Positive, jenž jsou špatně vyhodnoceny jako nestandardní chování. Pokud se takovýchto případů objevuje velké množství, je nutné provést znovu nastavení typického síťového provozu.

Detekce anomálií a její pojetí spadá do následujících zásadních skupin: behaviorální, provozní vzory a protokoly. Behaviorální analýza vyhledává odchylky v síťovém chování, jenž je známé a standardizované. Takovéto chování lze například vidět mezi vzájemným vztahem packetů a tím, co je v daný okamžik posíláno počítačovou sítí. Protokolová analýza odhaluje porušení síťových protokolů či využití chyb protokolů pomocí definic v RFC.

Pomocí detekce anomálií je možné shromažďovat jak statistická data o systému, tak i jeho typickém chování. Příkladem statistických dat je například hodnota UDP provozu na serveru, která nikdy nepřesáhne 20 %. Zatímco typické chování ukazuje například, že uživatel 123 typicky neposílá žádná data pomocí protokolu FTP mimo firemní síť. [13], [16]

3.3. Běžně používané IDS/IPS systémy

Co IDS/IPS systémy jsou, jaký mají účel, jaký je v nich rozdíl a další otázky byly zodpovězeny v přechozí kapitole číslo 3. IDS/IPS. Tato kapitola se bude věnovat nejznámějším IDS/IPS systémům a následně jejich možnému využití ve firmě WEDOS Internet, a.s. Pro tento účel byly vybrány čtyři IDS/IPS systémy Suricata, Snort, OSSEC a Bro.

3.3.1. Suricata

Suricata je bezplatný open-source software poskytující rychlý a robustní systém pro odhalení síťových hrozeb. Jedná se o poměrně nový bezpečnostní software, který byl ve stabilní verzi vydán neziskovou nadací OISF v roce 2010. Navzdory ranné verzi se software velice rychle vyvíjel, a to i za pomoci počítačové komunity, která se zaměřila na bezpečnost, použitelnost a efektivitu. V roce 2017 byla vydána již verze 4.0.1, a v poslední čtvrtině roku 2019 došlo k vydání zcela nová verze 5.0.0. Tento IDS/IPS systém je dostupný na operačních systémech Linux, Windows, MacOS a dalších platformách.

Obrázek č. 6 Logo systému Suricata [17]

Suricata zprostředkovává detekci síťových útoků v reálném čase v režimu IDS, ve kterém monitoruje probíhající komunikaci a na základě pravidel zjišťuje možné hrozby. Pro rychlé zabránění probíhajícího útoku lze využít režim IPS, ve kterém jsou hrozby ihned

eliminovány. Dalšími režimy systému Suricata jsou NSM neboli bezpečnostní monitoring sítě a režim offline zpracování zachycených packetů (pcap).

Detekce síťových hrozeb je realizována na základě výkonných a obsáhlých pravidel, jenž dovolují provádět komplexnější analýzu síťových útoků. Za použití standartních formátů vstupů a výstupů jako jsou Yaml a Json lze provést analýzu prostřednictvím různých stávajících systémů SIEM, Logstash/Elasticsearch, Kibana a dalších určených pro analýzu dat.

Mezi výhody softwaru Suricata patří funkce využití více vláken procesoru a jejich plné ovládání, možnost použít akceleraci výkonu na základě grafických karet, využití vyšší rychlosti zachytávání packetů za pomoci nástroje PF_RING a plná podpora internetového protokolu verze 6 (IPv6). [17], [18], [19], [20]

3.3.2. Snort

Systém snort je jedním z nejstarších bezpečnostních open source nástrojů na světě. Tento systém byl uveden do provozu v roce 1998 panem Martinem Roeschem a umožňuje analyzovat síťové protokoly v reálném čase, zkoumat obsah jednotlivých IP packetů a může být využit pro detekci útoků jako například skenování portů, přeplnění vyrovnávacích pamětí a mnoha dalších bezpečnostních hrozeb.

Nástroj Snort poskytuje tři hlavní režimy provozu. Prvním režimem je takzvaný sniffer, který umožnuje zachytávat packety proudící sítí a analyzovat jejich obsah. Dalším režimem je logování packetů, při kterém jsou packety zaznamenávány do logů na disku počítačové stanice. Posledním režimem nástroje Snort je mód síťově zaměřené detekce odhalení průniků, zkráceně NIDS. NIDS slouží pro analýzu provozu v reálném čase a na základě pravidel definovaných správcem sítě učiní příslušné opatření.

Podpora systému Snort je velice rozšířená, a to především díky rozsáhlé komunitě a open-source řešení. Velká část komerčních produktů SIEM umožňuje zpracování textových nebo binárních výstupů přímo z nástroje Snort. Podporované systémy, na kterých je možné Snort nainstalovat jsou Linux, Windows, MacOS, Solaris a HP-UX. [20], [21]

3.3.3. OSSEC

Mezi další open-source IDS systémy patří systém OSSEC. OSSEC zajišťuje kontrolu integrity souborů, analýzu logů, monitoring systémových politik, detekci rootkitů a odhalování hrozeb v reálném čase. Na základě výstražných protokolů a upozornění prostřednictvím e-mailů je správce systému upozorněn na probíhající útok a může tak provést protiopatření ke konkrétní hrozbě. Výstupní data lze využít v libovolném řešení SIEM, které umožní provést analýzu v reálném čase a pozdější kontrolu událostí v počítačové síti.

Nástroj OSSEC je multiplatformní a je schopen provádět kontrolu podnikových prostředí, která využívají více operačních systémů současně, a to včetně operačního systému Windows. Jako jediný z předešlých IDS systémů umožňuje centralizovanou správu monitoringu podnikového prostředí a tím zjednodušuje údržbu. Pro zajištění bezpečí v konkrétním podnikovém prostředí je možné vytvoření vlastních bezpečnostních pravidel a skriptů, která zcela odpovídají potřebám podniku. Jelikož se jedná o software s otevřeným zdrojovým kódem, lze také vytvořit nové funkce a tím optimalizovat systém pro svou potřebu. [20], [21]

3.3.4. Bro

Nástroj pro síťovou analýzu Bro byl vytvořen panem Vernem Paxsonem a má za sebou více jak 15 let výzkumu, během kterých přešel z akademického prostředí do prostředí reálného světa. Nástroj je využíván na mnohých univerzitách, výzkumných laboratořích a superpočítačových centrech. Software cílí především na vysoce výkonné počítačové sítě, které mohou být velice rozsáhlé.

Software je složen ze dvou vrstev, a to z vrstvy systému pro události a vrstvy skriptů pro podnikové politiky. Vrstva systému pro události má za úkol analyzovat živý či logovaný síťový provoz a upozorňovat na neobvyklou aktivitu v počítačové síti. Druhá vrstva programu Bro na dané upozornění předchozí vrstvy musí reagovat, a to pomocí specifikovaných politik a skriptů vyvolávajících různé akce jako například odeslání e-mailů správci systému, vyvolání upozornění či provedení specifických systémových příkazů na zamezení hrozby. Tento nástroj je využíván pro monitoring sítě a forenzní vyšetřování. Obsahuje funkce pro sofistikovanou analýzu a detekci hrozeb. Pro komplexní detekci hrozeb jsou využívána pravidla napsaná ve skriptovacím jazyku specifickým pro systém Bro. [20], [21]

4. Výběr vhodného řešení IDS/IPS systému

Se stále se zvyšujícím počtem útoků na síťovou infrastrukturu firmy WEDOS Internet, a.s. a služby jejich zákazníků, bylo nutné nalézt vhodné řešení těchto bezpečnostních incidentů. Proto bylo vybráno několik IDS/IPS systémů, které by mohly zmírnit následky útoků či jim úplně zamezit. Konkrétně se jedná o čtyři známé IDS/IPS systémy: Suricata, Snort, OSSEC a Bro, které byly popsány v předešlé kapitole číslo 3.3. Běžně používané IDS/IPS systémy. Pro určení nejvhodnějšího řešení byla firmou stanovena kritéria, jež specifikují hlavní požadavky společnosti. Mezi tato kritéria patří:

- Opensource řešení
- Přehledná dokumentace
- Využití vícevláknových procesorů
- Oba režimy – IDS i IPS
- Vlastní pravidla

Proč právě tato kritéria? Open source řešení bylo pro firmu velice důležité z důvodu možnosti upravovat kód pro vlastní specifické potřeby, a navíc byla firma od 1.1.2018 zcela open-source po ukončení spolupráce s firmou Microsoft. [22] Pro lepší práci se systémem a získání základních znalostí o bezpečnostních pravidlech bylo vhodné využít systém, který disponuje přehlednou dokumentací, jenž bude uživatelsky přívětivá. Možnost využívat systém na více vláknových procesorech bylo z hlediska výkonu nezbytností. Firma na svých serverech používala více vláknové procesory od samého počátku, a proto bylo nutné, aby i IDS/IPS systém mohl pro urychlení běhu zátěž rozložit na jednotlivá vlákna procesorů. Jako dalším kritériem byla možnost systému pracovat v režimu IPS. Režim IPS umožní firmě lepší reakce na aktuálně probíhající útoky. Posledním kritériem byla schopnost systému zpracovávat vlastní pravidla a poskytnout tak flexibilitu pro dané firemní prostředí.

Na základě výše specifikovaných kritérií byla vytvořena tabulka zobrazující hodnoty pro jednotlivé IDS/IPS systémy. Kritériu „přehledná dokumentace“ byla stanovena stupnice tří hodnot: přehledná, relativně přehledná a nepřehledná. Následně došlo společně se zadavatelem k ohodnocení všech čtyř systémů z hlediska jejich dokumentace. Hodnoty více vláknového zpracování a IPS režimu byly získány z diplomové práce „Detekce útoků cílených na odepření služeb“. [23]

Tabulka č. 1 Porovnání IDS/IPS systémů

Systém	Open-source	Přehledná dokumentace	Více vláknové zpracování	IPS	Vlastní pravidla
Suricata	Ano	Přehledná	Ano	Ano	Ano
Snort	Ano	Nepřehledná	Ne	Ano	Ano
Bro	Ano	Relativně přehledná	Ne	Ne	Ano
OSSEC	Ano	Relativně přehledná	Ne	Ne	Ano

K těmto kritériím byly postupně doplněny i informace týkající se detailů kontroly síťového provozu IDS/IPS systémem a také jakých služeb či zařízení se kontrola v prostředí firmy WEDOS Internet, a.s. týkala. Kontrola síťového provozu byla prováděna pouze na službách poskytovaným zákazníkům a nikoli na zařízeních samotné společnosti. Prozkoumávání síťového provozu mělo zajistit lepší kvalitu a dostupnost nabízených služeb, a to především webhostingů. Do provozu virtuálních serverů a dedikovaných serverů nemělo být nijak zasahováno. Navíc u webhostingů se jednalo pouze o kontrolu protokolu HTTP(s). Data uložená na jednotlivých webhostinzích prozkoumávána IDS/IPS systémem nebyla. Z těchto doplňujících informací vyplývalo, že IDS/IPS systém OSSEC, který patří mezi uzlově orientované systémy, není vhodným řešením pro účely této společnosti. Ze zbývajících tří IDS/IPS systémů byl vybrán systém Suricata, na základě hodnot v tabulce č. 1. Jejími hlavními přednostmi jsou více vláknové zpracování na procesoru, schopnost práce v režimu IPS, jedná se o software s otevřeným zdrojovým kódem a disponuje velice přehlednou a uživatelsky přívětivou dokumentací.

5. Úvod do problematiky pravidel systému Suricata

Bezpečnostní pravidla tvoří základ systému Suricata. Ihned po nainstalování systému jsou přítomna některá bezpečnostní pravidla, která jsou hojně využívána začínajícími uživateli. Existují i volně dostupná pravidla, jenž jsou tvořena skupinami zabývajícími se síťovou bezpečností. Mezi tyto skupiny jednoznačně patří VRT, která se specializuje na objevení nových zranitelností, škodlivého softwaru a trendů v oblasti bezpečnostních hrozeb. [24] Nelze se ovšem spoléhat pouze na již vytvořená pravidla a pro specifické případy je nutné vytváření pravidel vlastních.

5.1. Formát pravidel

Pravidla jsou složena z několika základních částí:

1. Akce
2. Hlavička
3. Volby

Jednotlivé části pravidla budou popsány na příkladu níže [25]:

```
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";
reference:url,doc.emergingthreats.net/2008124;
classtype:trojan-activity; sid:2008124; rev:2;)
```

První částí pravidla je akce. Akce udává, co se má stát, když kontrolovaná komunikace odpovídá popisu pravidla. Akce může nabývat jedné ze čtyř hodnot: Pass, Drop, Reject a Alert. Hodnota Pass v případě shody komunikace s pravidlem, zastavuje kontrolu dalšími pravidly a packet je propuštěn do střežené oblasti. Hodnota Drop je použita pouze v režimu IPS a zamezuje dalšímu odeslání packetu do zabezpečené oblasti. Nedochází k upozornění cílového uzlu, že packet pro něj určen byl zahozen a spojení skončí po uplynutí časového limitu. Po této události je vygenerováno pouze upozornění systémem Suricata a je zaznamenáno do logů. Třetí možností je hodnota Reject. Tato hodnota stejně jako v předešlém případě zamezí dalšímu šíření komunikace, ovšem na rozdíl od předešlé možnosti je odesílatel i příjemce upozorněn na zahození packetu. Pokud se jedná o TCP

segment, je vyžádáno opakování přenosu, v ostatních případech dochází k odeslání chybové zprávy prostřednictvím protokolu ICMP. Na závěr opět dochází k zaznamenání bezpečnostní hrozby do logů systému. Poslední akcí je Alert, při které dochází k vygenerování upozornění do logů, ale komunikace není nijak omezena a je propuštěna dále. [25] Jak je patrné z příkladu výše, jedná se o první položku pravidla. V tomto případě jde o hodnotu drop znázorněnou červeně.

Druhou částí pravidla je hlavička udávající protokol, zdroj a cíl, port a směr. V příkladu výše hlavičku znázorňují následující hodnoty:

```
tcp $HOME_NET any -> $EXTERNAL_NET any …
```

Prvním zástupcem hlavičky je protokol. Protokol sděluje systému Suricata, jaký druh komunikace je v daném pravidle kontrolován. Mezi základní protokoly patří tcp, udp, icmp a IP. Nicméně protokolů je velké množství a lze používat i další, kupříkladu HTTP, tls (ssl), ftp, smb, ssh, smtp, imap, dns atd. Použití těchto protokolů závisí na nastavení konkrétního protokolu v konfiguračním souboru systému Suricata. V úryvku bezpečnostního pravidla je protokol znázorněn modrou barvou. Druhým zástupcem hlavičky je zdroj a cíl, v příkladu hlavičky zobrazen zeleně. Hodnoty udávají zdroj a cíl komunikace a mohou být reprezentovány konkrétní IP adresou verze 4 nebo 6, rozsahem IP adres nebo proměnnými. Tyto hodnoty mohou být doplněny o operátory: ../.. – cidr, ! – negace a [.., ..] – vymezení skupiny. Aby mohly být použity proměnné, musí být předem definovány v konfiguračním souboru suricata.yaml. Předposlední část hlavičky představují porty vyznačené oranžově ve výňatku pravidla výše. Pomocí portu lze přesně identifikovat o jaký protokol se jedná a ten poté povolit či zakázat. Typickým příkladem portů jsou čísla 80 a 443, jenž jsou předěleny protokolům HTTP a HTTPS. Cílový port označuje protokol, který je dostupný na serveru, kam se klient hodlá připojit. Jako zdrojový port je operačním systémem vygenerováno náhodné číslo v rozsahu 1 023 – 65535. Z tohoto důvodu je nutné při psaní pravidel myslet na náhodnost portu a je vhodné využít parametr „any“ vyhovující všem číslům portů. Opět je možné použít operátory stejně jako je uvedeno výše u zdroje a cíle, ovšem s výjimkou operátoru ../. který byl nahrazen operátorem : vymezující rozsah. Poslední částí hlavičky je směr. Směr znázorňuje odkud kam komunikace proudí a jakými packety se má kontrola zabývat. Ve většině pravidel je směr orientován od zdroje k cíli (->), ovšem může nastat případ, kdy

je zapotřebí sledovat obousměrný provoz a proto existuje i možnost (< >). Ve výše uvedené části pravidla je směr označen fialovou barvou. [25]

Vše ostatní, co pravidla obsahují, jsou volby. Od samotného pravidla jsou odděleny závorkami a jednotlivé možnosti jsou od sebe separovány středníky. Volby se dělí na základní dva druhy. Jeden druh voleb obsahuje klíčové slovo, za kterým je specifikováno nastavení určující danou možnost a druhým druhem je možnost pouze klíčového slova, jenž plní specifický význam. Obecný předpis je znázorněn na následující ukázce:

```
<keyword>: <settings>;
```

```
<keyword>;
```

Jednotlivé volby mají určené konkrétní pořadí a změnou pořadí může docházet ke změně jejich významu. [25] Druhy volitelných možností budou popsány v následující kapitole.

5.2. Druhy volitelných možností v bezpečnostních pravidlech

Druhy voleb jsou rozděleny podle klíčových slov. Existuje velké množství kategorií, a proto zde budou uvedeny pouze nejznámější. Mezi tyto kategorie patří Meta Keywords, IP Keywords, TCP Keywords a Payload Keywords.

5.2.1. Meta Keywords

Meta Keywords obsahují meta informace, které na samotnou detekci nemají žádný vliv. Příkladem těchto meta informací jsou mgs (zpráva), sid (identifikační číslo), rev (revize), gid (identifikační číslo skupiny), classtype, reference. Další meta keywords lze najít v technické dokumentaci systému Suricata [26].

Mgs (zpráva)

Mgs poskytuje možnost přiřadit bezpečnostnímu pravidlu textový popis o jeho vlastnostech a potenciálním upozornění při shodě komunikace s pravidlem. Pravidlem je zprávu umísťovat na začátek volitelných možností.

Formát pravidla:

```
msg: "textový popis";
```

Příklad:

```
ET TROJAN Likely Bot Nick in IRC (USA +..)
```

Sid (identifikační číslo)

Identifikační číslo sid jednoznačně určuje, o jaké pravidlo se jedná. Jeho umístění je úplně na konci celého bloku, pokud pravidlo neobsahuje dodatečné revizní číslo, které bývá zařazeno ještě za sid.

Formát pravidla:	Příklad:
sid: číslo;	sid:2008124

Rev (revize)

Revizní číslo označuje verzi daného pravidla a téměř vždy doplňuje identifikační číslo. Pokud nastane úprava pravidla, zvýší se hodnota revize. Obě dvě čísla jsou umístěna na závěr voleb.

Formát pravidla:	Příklad:
`rev: číslo;`	`rev:2;`

Gid (identifikační číslo skupiny)

Identifikační číslo skupiny pomáhá rozdělit pravidla s podobným zaměřením do shluků. Původní hodnota je nastavena na číslo 1. Gid lze najít ve výpisu upozornění. Celý výpis vypadá takto:

```
10/15/09-03:30:10.219671 [**] [1:2008124:2] ET TROJAN Likely Bot Nick in IRC (USA +..) [**] [Classification: A Network Trojan was Detected] [Priority: 3] {TCP} 192.168.1.42:1028 -> 72.184.196.31:6667
```

Uzavřená závorka s hodnotou [1:2008124:2] představuje klíčová slova gid (1), sid (2008124) a rev (2). [26]

Classtype

Classtype poskytuje údaj o hodnocení pravidel a upozornění. Například pomocí klasifikace lze pravidlo označit pouze za informativní či naopak za velice nebezpečné narušení bezpečnosti. Je složeno z krátkého a dlouhého jména a priority. Classtype je definován i s prioritou v konfiguračním souboru classification.config.

Příklad definice classtype je zobrazen na následující ukázce:

```
config classification: trojan-activity,A Network Trojan was
detected, 1
```

Po definici classtype v konfiguračním souboru jej lze použít při vytváření pravidla. Pravidlo bude obsahovat pouze klíčové slovo `classtype:trojan-activity;`, které bude umístěno těsně před klíčovým slovem sid a rev. Při výskytu upozornění bude zobrazeno celé jméno klasifikace „A Network Trojan was detected“. [26]

Reference

Reference plní úlohu odkazu na další informace o pravidlu či problému, kterým se pravidlo zabývá. Tyto informace slouží pro tvůrce pravidla a další analytiky, kteří se budou věnovat zjištění, proč daná komunikace spadá pod konkrétní pravidlo. Reference se může v jednom pravidle objevit i vícekrát.

Formát pravidla:

```
reference: type, reference
```

Příklad:

```
reference:url,doc.emergingthreats.net/2008124;
```

5.2.2. Payload Keywords

Payload slouží ke kontrole obsahu dat posílaných v packetu. Pro tyto účely lze do pravidla vkládat klíčová slova content, nocase, depth a offset. Tato klíčová slova nejsou ovšem jediná. Popis a příklady dalších klíčových slov je možné najít v technické dokumentaci [27].

Content

Klíčové slovo content slouží pro určení hledaného obsahu v datech packetu. Kontrola obsahu probíhá na základě shody bytů. V rozsahu 0-255 bytů existuje 256 možných hodnot. Tyto hodnoty mohou nabývat všechny písmena abecedy, malá velká písmena a speciální znaky. Některé speciální znaky, které jsou použity v syntaxi pravidel je nutné napsat pomocí hexadecimálního přepisu, který je uzavřen mezi dvěma | |. Využití přepisu je patrné v následujícím příkladu, kdy je nutné přepsat znak „:“.

Formát pravidla: Příklad:

`content: „........“;` `content: „http|3A|//“;`

Samotné klíčové slovo nijak nespecifikuje, kde by se hledaný obsah měl nacházet, a proto dochází k prohledání všech bytů dat v IP packetu. V počátečním stavu je hledaný obsah citlivý na malá a velká písmena. Pro vyloučení obsahu z množiny hledaných dat je možné použít „!“.

Nocase

Jak bylo výše zmíněno v počátečním stavu je hledaný obsah citlivý na malá a velká písmena. Klíčové slovo content lze rozšířit o speciální možnost nocase, jenž zaručuje nezávislost na velikosti znaků. [27]

Formát pravidla: Příklad:

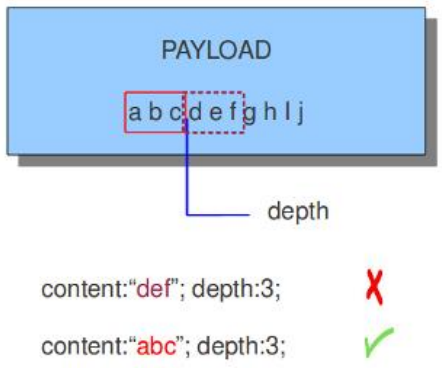
`nocase;` `content: “abc”; nocase;`

Depth

Klíčové slovo depth umožňuje blíže určit oblast dat, kde má dojít k vyhledání požadovaného obsahu. Číslo uvedené v klíčovém slovu depth identifikuje, kolik bytů od počátku dat má být zkontrolováno.

Formát pravidla: `depth: číslo;`

Příklad:



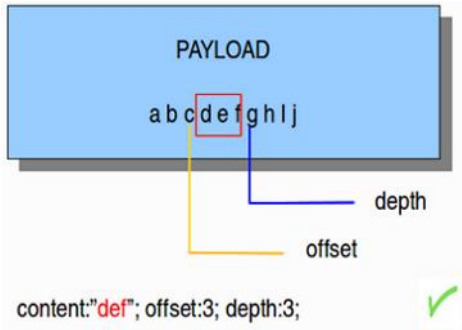
Obrázek č. 7 Příklad klíčového slova depth [27]

Offset

Klíčové slovo offset identifikuje počet bytů, které budou od začátku dat přeskočeny a hledaný obsah bude vyhledáván v následujících bytech. Offset lze kombinovat v klíčovým slovem depth.

Formát pravidla: `offset: číslo;`

Příklad:



Obrázek č. 8 Příklad klíčového slova offset [27]

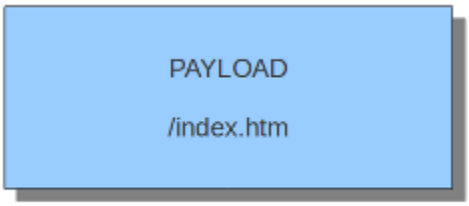
Z příkladu výše je patrné, že došlo k přeskočení prvních tří bytů (offset:3;) a zároveň vymezení kolik následujících bytů je potřeba prohledat (depth:3;). [27]

Pcre

Pcre celým názvem „Perl Compatible Regular Expressions“ přináší možnost kontrolovat protékající síťový provoz pomocí velice mocného nástroje jakým jsou regulární výrazy. Bohužel čím složitější regulární výraz se v bezpečnostním pravidle nachází, tím větší jsou hardwarové nároky stroje, kde je provozován IDS/IPS systém Suricata. Z tohoto důvodu je obvykle klíčové slovo pcre kombinováno s klíčovým slovem content. V prvním kroku proběhla kontrola, zda protékající síťový provoz obsahuje hodnoty z klíčového slova content a pokud ano až poté dochází ke kontrole za pomoci regulárních výrazů z pcre.

Formát pravidla: `pcre:"/<regex>/opts";`

Příklad:



content:"index."; http_uri; pcre:"/html?$/UR";

Obrázek č. 9 Příklad položky pcre [27]

V defaultním nastavení je klíčové slovo pcre citlivé na velikost písmen v regulárních výrazech a také provádí kontrolu dat přicházející síťové komunikace, jako by tato data byla všechna v jednom řádku. Nicméně je možné přidat za regulární výraz modifikátory, jenž toto klíčové slovo nabízí. Jedním z hlavních modifikátorů je „i“, které přináší nezávislost na velikosti písmen. Mezi další patří například „R, U, P a mnoho dalších. Všechny modifikátory lze dohledat v přehledné dokumentaci systému Suricata [27].

6. SPAM prostřednictvím webových formulářů

Pojem spam označuje nevyžádanou elektronickou zprávu rozesílanou velkému počtu uživatelů. Aby se jednalo opravdu o spam nesmí být uživatelem zpráva nijak vyžádána. V prostředí e-mailové komunikace je spam velice dobře redukován prostřednictvím spamových filtrů. Nicméně v prostředí webových formulářů je situace podstatně horší. Pokud se nachází na webové prezentaci nezabezpečený formulář, tak je zpravidla velice rychle zneužit k odesílání spamu. Obvykle se jedná o kontaktní a registrační formuláře či komentáře. Problém je umocňován absencí zabezpečení formulářů ihned po nainstalování většiny dnes používaných redakčních systémů, jenž se využívají na jednoduchou tvorbu webových prezentací. Nevyžádané zprávy obyčejně obsahují reklamní sdělení, nabídku finančních produktů či práce, erotické nabídky a další obtěžující typy zpráv včetně odkazů na stránky, které útočníci mohou použít pro získání osobních údajů, nebo nahrání viru či malware do počítače návštěvníka. [28], [29]

6.1. Webový formulář

Webové aplikace jsou založeny na architektuře klient/server, kde klientem obvykle bývá webový prohlížeč a serverem se obvykle rozumí webový server například Apache či Nginx. Klient odesílá požadavky na server prostřednictvím HTTP protokolu a server mu pomocí stejného protokolu odpovídá. HTTP požadavek obsahuje specifické informace od uživatele webové aplikace, které mohou být odeslány například pomocí webového formuláře. Webový formulář je pole ve webové aplikaci, jenž má různé podoby. Může se jednat o jednořádkové či víceřádkové textové pole, zaškrtávací pole atd. Hodnoty vyplněné ve formuláři by měli specifikovat návštěvníci webové aplikace, kteří chtějí předat určitou informaci. V případě vyplnění pole uživatelem a následné stisknutí tlačítka s názvem například „odeslat“, jsou všechny vyplněné hodnoty odeslány skriptu běžícímu na webovém serveru, kde je webová aplikace uložena. Aby mohlo dojít k odeslání vyplněných informací, je nutné určit, jak mají být data odeslána a jaký skript bude tato data zpracovávat. HTML prvek „form“ a jeho atributy vymezují podobu odesílaných informací v HTTP požadavku od klienta webovému serveru po stisknutí tlačítka pro odeslání informací. V HTML kódu může mít prvek „form“ následující podobu:

```
<form action=... method=...>
```

V HTML kódu je za definicí formuláře uveden atribut „action“. Pomocí tohoto atributu je určeno, jaký skript bude odeslaná data zpracovávat. Konkrétně specifikuje cestu k danému skriptu na webovém serveru nebo je možné uvést pouze název daného souboru. Pokud není atribut uveden, jsou data odeslána na URL adresu, kde se nachází samotný webový formulář. V druhém atributu „method“ je uveden způsob odeslání informací do zpracovávajícího skriptu. Pro odesílání informací z webových formulářů se obvykle používají dvě HTTP metody. Jedná se o metodu GET a POST. HTTP metody jsou součásti HTTP požadavku společně s hlavičkami a v některých případech také tělem, které je od hlaviček odděleno prázdným řádkem. [30], [34], [35]

Metoda GET je používána klientem k odeslání požadavku o vrácení objektu. Tímto objektem může být HTML soubor, obrázek či jakýkoli jiný soubor, který je na webovém serveru uložen. V případě metody GET nemá zpravidla HTTP požadavek tělo. Tělo je u metody GET prázdné, protože nejsou data odesílána na server v těle požadavku, ale jsou součástí URL adresy. Příklad GET požadavku odesílaného z webového formuláře je zobrazen níže:

```
GET /?say=Hi&to=Mom HTTP/2.0

Host: example.com
```

V tomto příkladě jsou data odesílána pomocí formuláře, který obsahuje dvě textová pole. První pole slouží pro sdělení zprávy a druhé pole slouží pro identifikaci příjemce dané zprávy. HTTP požadavek je odeslán pomocí metody GET a verze HTTP/2.0. Odesílaná data jsou přiložena k URL adrese. K oddělení adresy webové aplikace a dat slouží znak „?“. Poté následuje dvojice název pole a hodnota, které byly do pole zaznamenány uživatelem. Pokud je jako v tomto případě přítomno více polí, jsou navzájem od sebe odděleny znakem „&“. Další položkou HTTP požadavku je hlavička „host“ určující název domény, které se bude požadavek týkat. Jelikož jsou data umístěna již v URL adrese, není zde uvedeno tělo požadavku. [34], [35]

Metoda POST se od metody GET liší. Klient využívá metodu POST pro odesílání informací v těle HTTP požadavku na webový server. Tato metoda je hojně využívána při odesílání informací z formulářů webových aplikací. Pokud jsou informace z formuláře odeslány touto metodou, nedochází k předání informací do url adresy.

Příklad HTTP požadavku je znázorněn níže:

```
POST / HTTP/2.0

Host: example.com

Content-Type: application/x-www-form-urlencoded

Content-Length: 13

say=Hi&to=Mom
```

Z příkladového HTTP požadavku je patrné, že použitá metoda je POST a verze HTTP protokolu je 2.0. Následuje hlavička „host“ specifikující název domény. Mezi další zde uvedené hlavičky patří „Content-type“ a „Content-Length“. Hlavička „Content-Length“ udává velikost těla HTTP požadavku. Hlavička „Content-type“ určuje typ zdroje, ze kterého jsou data na server odesílána a umožňuje tak serveru rozpoznat o jaký typ dat se jedná. Pokud není u formuláře uveden atribut „enctype“ je výchozí hodnota v HTTP požadavku nastavena na `application/x-www-form-urlencoded`. Stejně jako v předešlém příkladu pochází odesílaná data ze dvou textových polí. Nicméně nyní jsou data obsažena v těle HTTP požadavku. Aby došlo k jasnému oddělení hlaviček a dat v HTTP požadavku je mezi ně umístěn prázdný řádek, který slouží jako oddělovač. [34], [35], [36]

Nezávisle na tom, jaká HTTP metoda je pro odeslání dat použita, je serveru odeslána dvojice dat, která se skládá z názvu daného formuláře a jeho vyplněné hodnoty. Způsob práce s touto dvojicí informací se liší podle použité vývojové platformy. Skript, který je umístěn na serveru a umožňuje zpracování dat od klienta, obsahuje objekt nebo funkci sloužící k odeslání e-mailů. V jazyce ASP od firmy Microsoft je možno použít pro odeslání zprávy například objekt „CDO“, naproti tomu v jazyce PHP existuje funkce php mail(). Aby bylo možné zprávu odeslat, je zapotřebí pro dané funkce a objekty zajistit následující informace: komu je e-mail zaslán, předmět e-mailu a tělo e-mailu. Velice často je vyžadována informace o e-mailové schránce odesílatele dat. Po vložení těchto informací objektu či funkci dochází k odeslání zprávy. [30], [34]

6.2. Odesílání spamu

Odesílání nevyžádaných zpráv prostřednictvím webových formulářů je realizováno dvěma způsoby. Prvním způsobem je manuální odesílání člověkem a druhým jsou spamboti.

6.2.1. Manuální odesílání nevyžádaných zpráv

Společnosti si najímají lidi, aby ručně vyplňovali všechny webové formuláře, které se na webové aplikaci nachází a odesílali nevyžádanou poštu sloužící jako reklama dané společnosti. Ochrana takovéhoto druhu odesílání nevyžádané pošty je velice náročná, protože člověk projde přes klasické ochranné mechanizmy proti spamu bez jakýchkoli problémů. Nicméně tento druh odesílání spamů je velice časově náročný, a proto se obyčejně vyskytuje pouze na nejznámějších webových aplikacích.[28]

6.2.2. Spamboti

Pro účinné rozesílání nevyžádané pošty útočníci nejdříve použijí aplikace, které prohledají internet a vyhledávají webové formuláře. Pokud aplikace nějaký formulář objeví, provede test zranitelnosti. Tyto aplikace se nazývají spamboti. Běžný uživatel, který chce použít váš webový formulář, jej nejdříve vyplní a poté zmáčkne tlačítko odeslat. To způsobí akci, která všechny vyplněné informace přenese skriptu, jenž se nachází na FTP vašeho webhostingu. Spamboti musí provést téměř totožný postup. Nejdříve prozkoumají celý formulář a specifikují jaká pole obsahuje. Následně tato pole vyplní a provede testovací odeslání, jenž předá vyplněné údaje skriptu běžícímu na pozadí. Pro test využijí existující e-mailovou adresu, aby útočník zjistil, zda odeslání opravdu proběhne. Pokud tento test proběhne úspěšně a útočníkovi byl testovací e-mail doručen, je jasné že se jedná o nezabezpečený formulář. Po tomto prvotním testování a odhalení zranitelného formuláře dochází stejným způsobem k odesílání skutečné nevyžádané pošty. Formát odesílaných dat do skriptu může být upraven tak, aby mohlo dojít k modifikaci hlavičky e-mailu. Tento útok se nazývá „e-mail header injection“. Typickým příkladem modifikace hlavičky e-mailu je přidání pole bbc. Toto pole slouží k přidání skrytého dalšího příjemce. V takovém případě je zpráva poslána původně zamýšlené osobě ale i dalším cílovým osobám, aniž by to původní adresát zjistil. [30]

6.3. Ochrana proti odesílání nevyžádaných zpráv

Aby bylo možné zamezit spambotům využívat webový formulář, je potřeba učinit opatření, které ztíží odeslání informací a bude vyžadovat další akci po uživateli formuláře. Nicméně nesmí se jednat o příliš náročný úkol, s kterým by měli problémy i skuteční uživatelé. Způsobů, jak omezit odesílání spamu, je několik. Mezi tyto způsoby patří například: přidání neviditelného pole pro běžného uživatele, použití nástroje (re)captcha, cookies a systém testovacích otázek. Pokud ovšem nemáte přímý přístup/oprávnění přidat do webové aplikace některou z předchozích ochran, je možno použít filtrování obsahu pomocí bezpečnostních pravidel IDS/IPS systému. Tomuto způsobu se věnuje kapitola číslo 10.

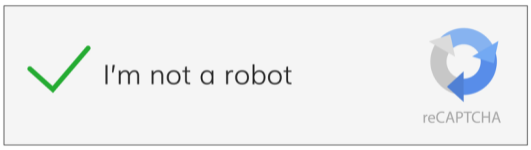
6.3.1. Přidání neviditelného pole pro běžného uživatele

Tento způsob ochrany spoléhá na omezenou funkčnost spambotů skenujících stránky HTML pro vyhledání nezabezpečených formulářů. Metoda přidává k regulárnímu formuláři, který je určen běžným uživatelům, ještě jeden formulář. Druhý formulář je skryt pomocí kaskádových stylů nebo javascriptu. V případě že, spamboti procházejí pouze HTML stránky a nikoli soubory s kaskádovými styly či javascriptem, uvidí formuláře dva a oba dva je vyplní. Pokud dojde k vyplnění obou dvou formulářů může dojít k filtraci takto zaslaných informací, protože regulární uživatelé by neměli nikdy vyplnit oba dva formuláře. Jeden pro ně bude vždy skryt. Toto zabezpečení je ovšem závislé na malé komplexnosti spambotů. Pokud spamboti umí procházet i soubory s kaskádovými styly nebo javascripetem, tak tato metoda nebude funkční. Nicméně zajistí odfiltrování alespoň malého podílu spamu. [31]

6.3.2. Recaptcha

Captcha je mechanizmus zabraňující spambotům odesílání informací přes webový formulář. Mechanizmus funguje na principu opsání určitých znaků či určení specifických obrázků nebo audia které je vhodné především pro nevidomé uživatele. Tímto relativně snadným úkolem by měl projít každý běžný uživatel webového formuláře, ale spambot nikoli. Některé mechanizmy captcha byly již překonány, a i přes jejich přítomnost u webového formuláře dochází k odesílání spamu. Ve společnosti WEDOS Internet, a.s je prozatím doporučováno používat hlavně mechanizmus recaptcha, který poskytuje v tuto chvíli dostatečnou ochranu proti hromadnému odesílání spamu. Ve třetí verzi

mechanizmu recaptcha dochází k odstranění hlavní nevýhody, a to je nutnost vyplňování, opisování či určování znaků či obrázků, což mohlo vést k odrazení běžných uživatelů v použití formuláře. Třetí verze přidává navíc skrytou verzi mechanizmu. Systém sám rozpozná, zda jsou informace odeslány přes webový formulář od regulárního uživatele, či spambota a pokud nejde přesně určit o koho se jedná, je provedena původní verze testu. [31], [32]



Obrázek č. 10 Podoba mechanizmu recaptcha [33]

6.3.3. Cookies

Soubory cookies jsou často používány pro uložení session, která umožňuje navázání spojení mezi uživatelem a webovým serverem. Jelikož běžný uživatel s největší pravděpodobností navštíví nejdříve úvodní stránku webové prezentace a až poté přejde na konkrétní formulář, obsahují soubory cookies relaci o navázaném spojení. Spamboti ovšem tuto relaci uloženou nemají, protože přistupují přímo na webové formuláře. V takovém případě je možné před odesláním informací přes webový formulář zkontrolovat, zda má uživatel uloženy cookies či nikoli. Tento způsob zabezpečení má ale problém s uživateli, kteří mají adresu webového formuláře uloženou a přistupují přímo na něj. V takovém případě jsou považováni za spamboty.

6.3.4. Systém testovacích otázek

Tento systém je založen na přidání otázky před odesláním informací přes webový formulář. Otázky jsou často založeny na jednoduchém matematickém příkladu nebo logické odpovědi. Na tento druh otázek by měl být schopný odpovědět pouze člověk a nikoli spambot. Je důležité určit, jak moc komplikovaná odpověď na danou otázku je po uživatelích vyžadována. Pokud bude požadována odpověď na otázku, kterou bude znát pouze malá skupina lidí, odradí se od použití formuláře i běžní uživatelé. [31]

7. Implementace řešení IDS/IPS systému Suricata

Tato část diplomové práce se zabývá nasazením open-source řešení IDS/IPS systému Suricata do prostředí Docker ve společnosti WEDOS Internet, a.s. Vývoj výsledného řešení je realizován v několika verzích. V prvotních verzích se jedná především o instalaci a počáteční nastavení open-source IDS/IPS systému Suricata v prostředí Docker. V každé další verzi dochází k přidání nějakého prvku, jenž je vyžadován do výsledného řešení. Mezi tyto prvky patří: obnovitelnost výsledného kontejneru se systémem IDS/IPS Suricata, blacklisty, zpřehlednění konfiguračního souboru IDS/IPS systému Suricata, automatická aktualizace pravidel, optimalizace současných pravidel a přidání několika nových pravidel.

7.1. Vývojové prostředí

Před nasazením výsledného řešení na servery společnosti WEDOS Internet, a.s. byl vývoj realizován na jednom z virtuálních serverů (VPS). Na VPS byl při objednávce defaultně nainstalován operační systém Debian 9, který obsahoval téměř všechny potřebné balíčky. Mezi doplňující balíčky patřily htop, lswh, sudo, iptables-persistent a docker, který bude popsán v následující kapitole. VPS se skládalo ze dvou jedno jádrových procesorů, 8 GB RAM a 60 GB pevného disku. Výpis komponentů je zobrazen na následujícím obrázku č. 11.

```
root@vm27366:~# lshw -short
H/W path      Device      Class      Description
=============================================
                          system     KVM
/0                        bus        Motherboard
/0/0                      memory     96KiB BIOS
/0/401                    processor  QEMU Virtual CPU version 1.5.3
/0/402                    processor  QEMU Virtual CPU version 1.5.3
/0/1000                   memory     8GiB System Memory
/0/1000/0                 memory     8GiB DIMM RAM
/0/100                    bridge     440FX - 82441FX PMC [Natoma]
/0/100/1                  bridge     82371SB PIIX3 ISA [Natoma/Triton II]
/0/100/1.1                storage    82371SB PIIX3 IDE [Natoma/Triton II]
/0/100/1.2                bus        82371SB PIIX3 USB [Natoma/Triton II]
/0/100/1.2/1  usb1        bus        UHCI Host Controller
/0/100/1.3                bridge     82371AB/EB/MB PIIX4 ACPI
/0/100/2                  display    GD 5446
/0/100/3                  network    Virtio network device
/0/100/3/0    eth0        network    Ethernet interface
/0/100/4                  generic    Virtio memory balloon
/0/100/4/0                generic    Virtual I/O device
/0/100/5                  storage    Virtio block device
/0/100/5/0    /dev/vda    disk       64GB Virtual I/O device
/0/100/5/0/1  /dev/vda1   volume     285MiB EXT4 volume
/0/100/5/0/2  /dev/vda2   volume     59GiB EXT4 volume
/0/1          scsi1       storage
/0/1/0.0.0    /dev/cdrom  disk       QEMU DVD-ROM
/1            docker0     network    Ethernet interface
```

Obrázek č. 11 Výpis hardwarových komponentů zobrazený pomocí příkazu lswh -short

7.2. Prostředí Docker a open-source IDS/IPS Suricata

Instalace Docker prostředí byla provedena na výše popsaném VPS za pomoci dokumentace, jež tato technologie nabízí. [37] Po instalaci došlo ke zkoušce funkčnosti prostřednictvím příkladu uvedeném taktéž v této dokumentaci. Následně bylo nutné vytvořit dva oddělené kontejnery. První z kontejnerů byl určen pro vytvoření webového serveru, který sloužil k testování funkčnosti sledování síťového provozu. Pro vytvoření kontejneru byly zapotřebí tři soubory: Dockerfile, start.sh a index.html. Z těchto souborů byl vytvořen image s názvem "web" pomocí příkazu:

```
docker build -t web .
```

Jakmile byl image vytvořen, bylo zapotřebí spustit samotný kontejner. K tomuto účelu byl použit příkaz:

```
docker run --name web --privileged=true --network="host" -d
-v
/root/docker/2_prvni_pokusy_suricata_www/centos_httpd/www:/
var/www/html/ web
```

Tento příkaz vytvořil kontejner s názvem „web“ z výše vytvořeného image se stejným názvem. Pomocí parametrů privileged a network byl nastaven přístup k síťovému rozhraní fyzického serveru z vnitřku kontejneru. Parametr -d umožnil běh kontejneru na pozadí a parametr -v sloužil k připojení lokální složky, v tomto případě složky www, přímo do kontejneru. Toto propojení zajišťovalo možnost modifikace souborů v dané složce i přímo z VPS bez nutnosti vstoupit do kontejneru. Po nasměrování subdomény dip.pracovnidomena.eu na IP adresu VPS, mohlo dojít k ověření funkčnosti kontejneru. Výsledek je zobrazen na obrázku č. 12.

Obrázek č. 12 Test funkčnosti Docker kontejneru www

Druhý kontejner obsahoval samotný IDS/IPS systém Suricata. Stejně jako předchozí kontejner byl vytvořen za pomoci souborů Dockerfile a start.sh. Opět došlo k vytvoření image za použití příkazu `docker build` s tím rozdílem, že nyní název (parametr -t)

byl „suricatav1“. Po vytvoření image a před samotným spuštěním kontejneru bylo ještě vytvořeno perzistentní úložiště dat. K vytvoření byly využity následující příkazy:

```
docker volume create --driver local --opt type=none --opt
device=/root/docker/2_prvni_pokusy_suricata_www/suricata/lo
gs --opt o=bind surlogs
```

```
docker volume create --driver local --opt type=none --opt
device=/root/docker/2_prvni_pokusy_suricata_www/suricata/su
ricata --opt o=bind surconf
```

Perzistentní úložiště dat sloužilo pro uložení konfiguračních souborů IDS/IPS systému Suricata, pravidel a logů. Také velice zjednodušilo příkaz pro spuštění kontejneru, jenž již nemusel obsahovat celou cestu k tomuto úložišti. Celý příkaz pro spuštění kontejneru vypadal následovně:

```
docker run --name sur --hostname suricata –privileged=true
--network="host" -ti -d -v surconf:/etc/suricata -v
surlogs:/var/log/suricata suricatav1
```

Jakmile byl kontejner vytvořen, bylo do něj možné vstoupit za pomoci příkazu `docker exec -ti sur /bin/bash` a začít upravovat konfiguraci samotného systému IDS/IPS Suricata. V první verzi došlo jen k drobným úpravám, jako například zakomentování stávajících pravidel v konfiguračním souboru a vložení názvu souboru s testovacími pravidly a také přidání IP adres do proměnné HOME_NET. Testovací soubor s pravidly „vlasni.rules“ obsahoval pouze jednoduché pravidlo. Toto pravidlo zaznamenávalo veškerý tcp provoz z IP adresy 46.28.104.66, jenž směřoval na VPS, do souboru fast.log a eve.json jako alert. Byl proveden i test odmítnutí daného provozu. Bylo zapotřebí pouze v pravidle modifikovat klíčové slovo „alert“ na „drop“. V takovém případě došlo k zaznamenání komunikace do logů a zároveň k jejímu zahození. Pravidlo můžete vidět na následující ukázce.

```
alert tcp 46.28.104.66 any -> 31.31.76.173 80 (msg:"Pokus
pravidla.";sid:1;)
```

Po aktualizaci pravidel a znovunačtení služby IDS/IPS systému Suricata bylo bohužel zjištěno, že řešení není funkční. Podle dokumentace [38] bylo odhaleno, že je nutno

modifikovat pravidla Iptables a to tak, aby veškerý síťový provoz směřoval do fronty, jenž bude IDS/IPS systém Suricata zpracovávat. Nasměrování síťového provozu na fyzickém serveru bylo realizováno pomocí přidání těchto dvou pravidel do Iptables:

```
Iptables -I INPUT -p tcp --dport 80 -j NFQUEUE
iptables -I OUTPUT -p tcp --sport 80 -j NFQUEUE
```

Pravidla zajišťují přesměrování veškerého tcp provozu na portu 80 do fronty, kterou již může analyzovat IDS/IPS systém Suricata. Výsledná zkouška načtení stránek dip.pracovnidomena.eu byla zaznamenána do souborů fast.log a eve.json.

Po úspěšném testu obou kontejnerů, bylo potřeba modifikovat kontejner se systémem IDS/IPS Suricata, tak aby vyhovoval prostředí ve společnosti WEDOS Internet, a.s.. Tato modifikace byla rozdělena na několik verzí, ve kterých byly modifikovány různé parametry. Ve verzi jedna byla modifikace spjata s možností kontejner opětovně načíst bez neočekávaných změn. Proto došlo ke změně instalace IDS/IPS systému Suricata z instalace z balíčku [39], kde není možno kontrolovat verzi systému, na instalaci ruční. Systém byl stažen ve verzi 4.1.0 a v souboru Dockerfile byla instalace upravena podle dokumentace [40]. Další změnou v první verzi souboru Dockerfile byla změna umístění instalace samotného systému. Změna modifikovala původní cestu instalace systému z */etc/suricata* na */config/suricata*. Tuto změnu bylo nutné také zanést do konfiguračního souboru suricata.yaml. Při testování každé úpravy v souboru Dockerfile bylo nutné vždy celý image vytvořit znovu a až poté spustit nový kontejner. To mělo za následek neustálé opakování příkazů `docker buid` a `docker run`, jenž jsou uvedeny výše. Z tohoto důvodu byly vytvořeny skripty ve skriptovacím jazyce bash, obsahující mírně upravené předchozí příkazy na vytvoření image a spuštění kontejneru a jednoduché rozhraní, umožňující velice snadné použití. U každé verze po vytvoření kontejneru byl proveden test funkčnosti analýzy provozu a výsledky uloženy do souborů fast.log a eve.json. Tyto testy se prováděly vně kontejneru a každá změna v pravidlech systému IDS/IPS Suricata se projevila až po znovunačtení pravidel. Pro obnovení pravidel bez nutnosti ukončit probíhající proces Suricaty sloužil příkaz:

```
kill -USR2 $(ps -axu | grep "[s]uricata" | awk '{print $2}')
```

Tento příkaz je složen z příkazu na znovunačtení pravidel IDS/IPS systému Suricata a vnořeného příkazu, který z listu současných procesů vyhledá proces s názvem S/suricata

a vypíše pouze číslo procesu, které má být použito pro znovunačtení pravidel. Příkaz byl používán napříč všemi verzemi.

Druhá verze souboru Dockerfile přinesla pouze několik menších změn. Jednou z hlavních změn byla strukturalizace kódu. Zamezila častému opakování příkazu yum install, což by mohlo vést k pomalejšímu vytvoření image. Také došlo k vytvoření složky */packages* do které byly na počátku vytváření image ukládány soubory s instalací systému IDS/IPS Suricata a Oinkmaster. Systém Oinkmaster sloužil pro aktualizaci pravidel, která v dosavadních verzích nebyla příliš řešena a bude plně využita až v pokročilejších verzích výsledného řešení.

Ve verzi číslo 3. došlo k aktualizaci samotného systému IDS/IPS Suricata z verze 4.1.0 na verzi 4.1.3. Tato změna v souboru Dockerfile nevyžadovala téměř žádné úpravy, kromě změny názvu instalačního souboru. Nicméně při spuštění IDS/IPS systému Suricata v ně kontejneru byla odhalena následující chyba:

```
[ERRCODE: SC_WARN_DEFAULT_WILL_CHANGE(317)] - in 5.0 the
default for decoder event stats will go from
'decoder.<proto>.<event>' to
'decoder.event.<proto>.<event>'. See ticket #2225. To
suppress
this message, set stats.decoder-events-prefix in the yaml.
13/10/2019 -- 12:40:54 - <Warning> - [ERRCODE:
SC_WARN_EVE_MISSING_EVENTS(318)] - eve.stats will not
display all decoder events correctly. See #2225. Set a
prefix in stats.decoder-events-prefix. In 5.0 the prefix
will default to 'd
ecoder.event'.
```

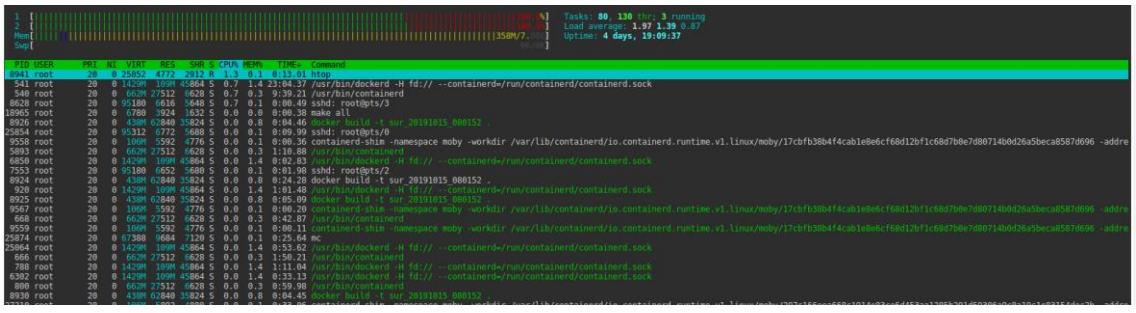
K odhalení chyby byl vytvořen kontejner bez perzistentního úložiště, což způsobilo nové vygenerování konfiguračního souboru suricata.yaml. Po porovnání defaultního konfiguračního souboru s již upraveným konfiguračním souborem z předchozí verze systému bylo odhaleno, že je nutné přidat položku decoder-events-prefix do globálního nastavení statistik. Výsledná podoba globálního nastavení statistik je znázorněna na následující straně.

```
stats:
  enabled: yes
  interval: 8
  decoder-events-prefix: "decoder.event"
```

Po této úpravě konfiguračního souboru suricata.yaml a znovu spuštění systému Suricata, byla chyba eliminována. Analýza provozu a zaznamenávání do souborů s logy opět probíhalo v pořádku.

Čtvrtá verze souboru Dockerfile přidává podporu pro GeoIP a nástroje htop a nload. Tyto nástroje byly vyžadovány ze strany společnosti WEDOS Internet, a.s. Také došlo při instalaci systému IDS/IPS Suricata k zakázání protokolů FTP, ICMP, ICMPV4 a ICMPv6. Zde zmíněné protokoly nesměly být v žádném případě ovlivňovány výsledným řešením. Kdyby docházelo k nechtěnému blokování těchto protokolů mohlo by to ohrozit funkčnost zákaznických služeb.

Předposlední pátá verze souboru Dockerfile byla velice významná. Byla zde pozměněna samotná instalace IDS/IPS systému Suricata z instalace rozdělené na `install` a `install-conf` na `install-full`, což mělo za následek opětovné zjednodušení souboru Dockerfile a tím pádem i rychlejší vytvoření image. Rozdíly mezi instalacemi lze nalézt v dokumentaci [40]. Při kompilaci byl také použit parametr -j, jenž umožňuje definovat kolik bude použito jader procesoru. Testovací VPS disponovalo dvěma jedno jádrovými procesory, a proto byla nastavena hodnota na dvě. Průběh vytváření image pomocí skriptu build.sh byl zachycen pomocí příkazu htop, který umožňuje zobrazit využití jednotlivých jader procesorů. Průběh lze vidět na následujícím obrázku č. 13.



Obrázek č. 13 Příkaz htop

Na obrázku je patrné, že jsou obě jádra každého procesoru testovacího VPS při kompilaci vytížena na 100 %. Aby bylo možné dynamicky měnit parametry verze IDS/IPS systému Suricata, verzi systému Oinkmastru a výše zmíněné parametry příkazu make, došlo k vytvoření několika proměnných, které umožňovaly definovat pevně dané parametry příkazů a cest k daným systémům.

Poslední, šestá verze Dockefile přinesla zásadní změny výsledného řešení. Mezi hlavní změnu patřila aktualizace IDS/IPS systému Suricata z původně nainstalované verze 4.1.3 na 5.0.0. Tato aktualizace ovšem přinesla i několik změn v povinných závislostech, jenž mají být před samotnou instalací v operačním systému zakomponovány. Potřebné balíčky byly zjištěny při vytváření image. Proces vždy skončil chybou požadující instalaci potřebného balíčku. Jednalo se o balíčky libmaxminddb-devel, lz4-devel, rustc a cargo. Velké změny byly provedeny také v konfiguračním souboru systému IDS/IPS Suricata. Aby bylo možné se lépe orientovat v konfiguraci a také umožnit přenositelnost výsledného řešení, byl konfigurační souboru IDS/IPS systému Suricata rozdělen do tří souborů. První soubor suricata.yaml obsahoval pouze příkazy include, pomocí kterých docházelo k vnoření dalších dvou konfiguračních souborů. Jako první byl vnořen konfigurační soubor suricata_default.yaml. Soubor suricata_default.yaml byl konfiguračním souborem, jenž byl vygenerován při čisté prvotní instalaci systému IDS/IPS Suricata. Obsahoval všechny základní nastavení bez jakýchkoliv úprav. Posledním souborem je konfigurační soubor suricata_zmeny.yaml obsahující všechny změny, které mají být nastaveny ve výsledném řešení. Položky v tomto souboru přepisují původní hodnoty z defaultního konfiguračního souboru. Tento soubor musí být vnořen jako poslední, aby došlo k přepsání pouze zde uvedených změn v konfiguraci IDS/IPS systému Suricata. Mezi tyto změny patří například: změna cesty k pravidlům a jaká pravidla mají být použita, nastavení logování a nastavení ip reputace, o které bude více napsáno v kapitole „Blacklisty“. Při aktualizaci systému bylo také zapotřebí provést změnu zápisu do logu zahozených packetů drop.log. Po aktualizaci a spuštění systému se v terminále zobrazovalo varování uvedené níže:

```
18/10/2019 -- 08:06:41 - <Warning> - [ERRCODE:
SC_WARN_DEPRECATED(203)] - The drop log has been deprecated
and will be removed by June 2020. Please use eve-log.
```

```
18/10/2019 -- 08:06:41 - <Info> - drop output device
(regular) initialized: drop.log
```

Toto varování oznamovalo ukončení podpory pro soubor drop.log v červnu roku 2020. Proto byla z preventivních opatření provedena změna zápisu do tohoto logu a všechny zamítnuté packety budou nadále logovány pouze v logu eve.log. Při dalším spuštění systému Suricata již varování nebylo zobrazeno. Poslední změna v šesté verzi souboru Dockerfile se týkala odstranění konfigurace systému Oinkmaster a aktualizace pravidel právě pomocí tohoto systému. Konfigurace systému Oinkmaster a aktualizace pravidel byla řešena již odděleně v samostatném souboru start.sh, který bude popsán v následující kapitole.

7.3. Soubor start.sh

Soubor start.sh je jeden ze základních souborů při vytváření image v prostředí Docker. Obsahuje příkazy, jenž se mají provést po spuštění kontejneru. V počátečním testování obsahoval pouze příkaz na spuštění systému Suricata zajišťující start služby po vytvoření kontejneru. V pozdějších fázích testování do souboru start.sh byl přesunut systém Oinkmaster, který se staral o aktualizace pravidel. Byla zde přidána funkce zajišťující detekci běhu služby a také správa blacklistů.

7.3.1. Blacklisty a pravidla IDS/IPS systému Suricata

Před použitím blacklistů v souboru start.sh bylo pro testování využito odděleného skriptu test.sh, ve kterém probíhal vývoj a testování právě funkce starající se o stažení, úpravu a použití blacklistů. Vývoj souboru test.sh byl rozdělen do sedmi verzí, kde sedmá verze byla přenesena a použita v souboru start.sh.

První verze skriptu sloužila pouze pro otestování, jakým způsobem bude k IP adresám v souborech s koncovkou .list přidána kategorie a reputace dané IP adresy. Test proběhl na jednoduchých souborech obsahujících pouze IP adresy, které byly již ve správném formátování a nemusely se nijak upravovat.

Druhá verze test_v2.sh se zaměřovala na testování a kontrolu IP adres v blacklistech v pravidelných intervalech. Kontrola ověřovala přítomnost kategorie a čísla reputace. Na začátku skriptu se pomocí proměnné CHECK_MINUTES definoval

interval, jak často bude ke kontrole docházet. Pro testovací účely byl interval nastaven na každých 5 minut.

Verze souboru „test“ s číslem 3. byla první verzí, ve které byl umístěn systém Oinkmaster, který zajišťoval stažení a aktualizaci defaultních pravidel. Tato funkcionalita sem byla přesunuta z poslední verze souboru Dockerfile. V konfiguračním souboru systému Oinkmaster byla uvedena URL adresa, která směřovala na stránky rules.emergingthreats.net, ze kterých byly stahovány defaultní pravidla IDS/IPS systému Suricata. Dodatečně byla přidána ještě jedna url adresa směřující na interní servery společnosti WEDOS Internet, a.s. poskytující pravidla používaná v původním řešení a nově vytvořená pravidla. Soubor test_v3.sh byl také první verzí, ve které došlo k testování na skutečných blacklistech stahujících se ze zdrojů uvedených v souboru blacklist_source_list. Po stažení blacklistů bylo zapotřebí zdrojové soubory upravit, aby obsahovaly pouze IP adresy a žádné další znaky. Blacklisty bohužel nebyly totožného formátování, a proto muselo dojít ke dvěma různým úpravám. První skupina blacklistů obsahovala IP adresy ve formátu:

```
"ip adresa" , "země" , "kód země" , "město" , "datum"
```

V takto formátovaném zápisu bylo možné IP adresu oddělit pomocí uvozovek. Zápis IP adres v blacklistech uvedený výše se týkal většiny zdrojových souborů. Pouze dva testované blacklisty obsahovaly jiné formátování zápisu IP adres. Rozdílné formátování zápisu obsahovalo IP adresu a popis, který byl od IP adresy oddělen čárkou. Bohužel formátování neobsahovalo uvozovky, a proto nebylo možné IP adresu pomocí nich oddělit. Muselo dojít k oddělení IP adresy pomocí uvedené čárky. Již upravené soubory byly uloženy s koncovkou .list a patřičným jménem blacklistu. Zároveň došlo k odstranění původních zdrojových souborů. Dále byla přidána metoda umožňující stahovat blacklisty a whitelisty společnosti WEDOS Internet, a.s. a zároveň IP adresy uvedené ve whitelistech mazat ze všech blacklistů, tak aby nedocházelo k jejich blokaci. Poslední změna v souboru „test“ verze číslo 3. byla úprava automatického spouštění stahování základních blacklistů. Stažení probíhalo pouze jednou denně ve stanovený čas. Toto ovšem neplatilo pro blacklisty a whitelisty společnosti WEDOS Internet, a.s., které se stahovaly při každé změně v uvedených souborech.

Ve čtvrté verzi souboru test_v4.sh došlo k úpravě metody update_our_list. Tato metoda dříve stahovala blacklisty a whitelisty pomocí jednoho cyklu „for“. Bylo ovšem nutné whitelisty stahovat odděleně, kvůli umožnění další práce právě s těmito soubory. Byla přidána nová metoda pro aktualizaci pravidel společnosti WEDOS Internet, a.s. Nejednalo se již o defaultně stažená pravidla IDS/IPS systému Suricata ale o pravidla, jenž byla vytvořena za provozu původní verze systému Suricata. Tato původní verze byla nahrazena právě výsledným řešením. Stažení se provádělo vždy při jakékoli změně v souboru s pravidly. Po aktualizaci pravidel bylo také automaticky provedeno znovunačtení procesu Suricata. To umožnilo promítnutí změn do běžícího procesu, aniž by bylo nutné celý systém zastavit a znovu spustit s novými pravidly.

Pátá verze přinesla pouze několik menších změn. Došlo k úplnému rozdělení stahování blacklistů a whitelistů společnosti WEDOS Internet, a.s. do dvou oddělených metod. Tato změna konečně umožnila implementaci mazání IP adres z blacklistů pouze při aktualizaci whitelist souborů a nebylo tak nutné tento náročný úkon dělat v případě, že k žádné změně nedošlo.

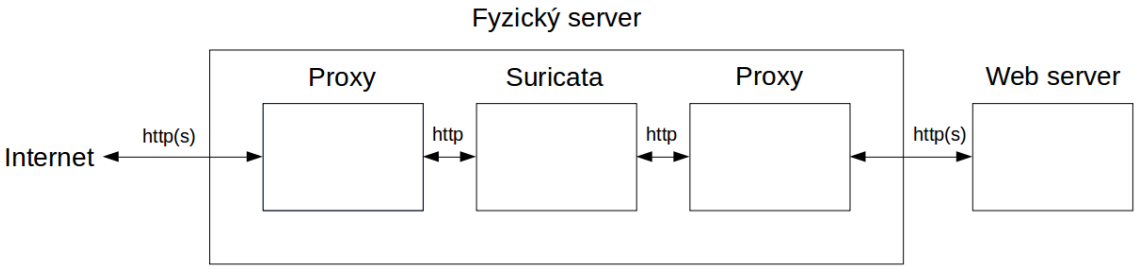
Předposlední verze test_v6.sh byla velice významnou verzí. Jednou z prvních změn bylo oddělení příkazu na znovunačtení IDS/IPS systému Suricata do samostatné metody. To umožnilo tuto metodu volat na více místech bez nutnosti opakovat daný kód. Další změna odstranila jednu z chyb předchozí verze. Jednalo se o chybu, kdy nedocházelo k odstranění IP adres, které byly v souborech whitelistů, z defaultních blacklistů. Mazání těchto IP adres bylo do této verze možno pouze u blacklistů společnosti WEDOS Internet, a.s. Z tohoto důvodu bylo mazání přidáno i do metody pro stažení defaultních blacklistů. Bylo ovšem nutné ošetřit, aby k mazání došlo pouze tehdy, když jsou již whitelisty staženy. To znamenalo, že prvotní stažení defaltních blacklistů proběhlo ještě bez smazání whitelistovaných IP adres. Poslední velkou změnou bylo přidání příkazu pro spuštění IDS/IPS systému Suricata a hlídání běhu daného procesu. Pokud došlo ke zjištění, že daný proces není spuštěný, bylo provedeno ukončení celého skriptu. Toto ve výsledku zajišťuje i běh celého kontejneru. V případě, že byl proces z jakéhokoli důvodu zrušen, dochází i k restartu kontejneru s IDS/IPS systémem Suricata. Kontejner je spuštěn znovu s původními hodnotami a běžícím procesem systému Suricata.

Poslední verze test_v7.sh přidala znovunačtení procesu Suricata do metody pro aktualizaci defaultních pravidel. K tomuto však dochází pouze pokud je proces IDS/IPS systému Suricata již spuštěn. Jestliže není, je provedena samostatná aktualizace defaultních pravidel. Velká změna byla provedena v procesu mazání IP adres ze všech blacklistů. Proces mazání IP adres byl přesunut do samostatné metody a byl kompletně přepracován. V předešlých verzích byly IP adresy z whitelistů vyhledávány ve všech souborech blacklistů jednotlivě. Takovéto vyhledávání bylo značně pomalé a zpomalovalo celý proces aktualizace blacklistů a whitelististů. Proto došlo ke změně a IP adresy z whitelistů byly nejdříve předpřipraveny a uloženy do pomocného souboru, ze kterého poté byly pomocí příkazu `sed` načteny a vyhledávány v blacklistech. Na závěr byl pomocný soubor odstraněn. Tato verze po otestování byla použita pro skript start.sh pouze s malými úpravami. Došlo k odstranění všech kontrolních výpisů, který byly realizovány pomocí příkazu `echo` a byly vytvořeny složky pro logování výstupu ze systémů IDS/IPS Suricata a Oinkmaster.

Po úspěšném vytvoření a otestování souboru start.sh bylo pro funkčnost blacklistů nutné ještě upravit konfigurační soubor suricata_zmeny.yaml, kde se nacházela konfigurace funkce Iprep. Funkce Iprep zajišťovala podporu blacklistů. V konfiguračním souboru byla upravena cesta ke kategoriím a složce s blacklisty. Jednotlivé kategorie bylo nutno definovat v souboru categories.txt ve složce s konfiguračními soubory IDS/IPS systému Suricata. Dále byl v konfiguračním souboru uveden seznam s jednotlivými blacklisty, které měly být použity. Je nutné zde ovšem vysvětlit, že se nejedná o klasické blacklisty. U klasických blacklistů se předpokládá, že každá IP adresa uvedena v souboru blacklist je automaticky blokována. V tomto případě tomu tak ovšem není. Blokování konkrétních IP adres je ve společnosti WEDOS Internet, a.s. řešeno mimo IDS/IPS systém Suricata. Blacklisty v systému IDS/IPS Suricata jsou přítomny z důvodu použití IP adres v bezpečnostních pravidlech. V takovýchto pravidlech je možné IP adresám z blacklistů nastavovat určitou kategorii, reputaci a na základě toho poté IP adresy omezovat. Například počet spojení z konkrétní kategorie IP adres omezit na určitý časový interval.

8. Nasazení a počáteční testování výsledného řešení ve firemním prostředí WEDOS Internet, a.s.

Po fázi vývoje na virtuálním serveru VPS, bylo nutné řešení nasadit na fyzické servery společnosti WEDOS Internet, a.s. Původní řešení IDS/IPS systému Suricata bylo umístěno na samostatném serveru přímo před webové servery. Nicméně toto řešení nebylo úplně vhodné. Server byl nadmíru zatížen příchozí komunikací a tím pádem byl velice náročný na hardwarové požadavky. Navíc toto řešení nebylo funkční při útocích prostřednictvím šifrovaného spojení HTTPS. IDS/IPS systém Suricata neumí prozatím komunikaci dešifrovat a provádět nad ní kontrolu v reálném čase. Z tohoto důvodu bylo nové řešení nasazeno na fyzický server společně s proxy servery. Schéma zapojení je znázorněno na obrázku č. 14.



Obrázek č. 14 Schéma zapojení nového řešení systému IDS/IPS Suricata ve společnosti WEDOS Internet, a.s.

Na obrázku výše je patrné, že příchozí komunikace je nejdříve poslána na proxy server. Zde je dešifrována, pokud se jedná o komunikaci HTTPS, pomocí privátního klíče a dále je distribuována nešifrovaně prostřednictvím protokolu HTTP. Tato komunikace prochází kontrolou v reálném čase systémem Suricata v IPS módu. Již zkontrolovaná komunikace je poslána nešifrovaně na další proxy server, který tuto zkontrolovanou komunikaci vezme a znovu zašifruje (pokud se původně jednalo o šifrovanou komunikaci HTTPS). Následně je odeslána interní sítí společnosti WEDOS Internet, a.s. formou HTTP(s), kterou byla původně přijata až k cílovému web serveru. Stejným procesem musí komunikace projít i v případě zpáteční cesty ze serveru ke klientovi s tím rozdílem, že pokud byla příchozí komunikace zkontrolována a propuštěna dále do interní sítě, je již označena jako bezpečná a není nutné znovu provádět kontrolu.

Toto řešení však přináší jednu velkou nevýhodu. V případě, že komunikace při kontrole odpovídala nějakému bezpečnostnímu pravidlu a byl tak vytvořen záznam

v logu, zdrojová a cílová IP adresa v tomto záznamu obsahovala IP adresy obou proxy serverů a nikoli skutečné IP adresy klienta a webového serveru. Tuto nevýhodu bylo nutné eliminovat, a proto byla využita X-Forwarded-For hlavička protokolu HTTP. X-Forwarded-For hlavičku zkráceně xff bylo nutné nastavit v konfiguračním souboru IDS/IPS systému Suricata. Zde bylo možné nastavit dva módy této hlavičky, konkrétně mód extra-data a overwrite. Mód extra-data přidává v HTTP hlavičce pole navíc obsahující právě původní IP adresu. Naproti tomu mód overwrite přepisuje zdrojovou či cílovou IP adresu proxy IP adresou skutečného cíle či zdroje. Druhou položkou v konfiguračním souboru IDS/IPS systému Suricata byl druh proxy serveru. Na výběr bylo ze dvou variant. První variantou byl druh reverse. Druhou variantou byl druh proxy forward. Pro použití v prostředí firmy WEDOS Internet, a.s. byl použit mód overwrite a druh proxy serveru reverse.

Testování funkčnosti řešení na vývojovém prostředí VPS, probíhalo nejdříve na uměle vytvořeném kontejneru s jednoduchou stránkou. Doména směřující na tuto stránku byla dip.pracovnidomena.eu a pro počáteční testování byla zcela dostačující. Nicméně pro testování funkčnosti již ve skutečném prostředí firmy WEDOS Internet, a.s. nebyl tento testovací způsob dostačující. Z tohoto důvodu byl objednán nový reálný webhosting, na kterém mohla být vytvořena testovací stránka. Pro zjednodušení simulace reálného prostředí, byl nainstalován redakční systém wordpress a v něm vytvořena jednoduchá firemní stránka obsahující úvodní stránku a kontakty. V souboru s pravidly tohoto systému bylo vytvořeno několik testovacích pravidel:

```
alert http any any -> any any (msg:"pokus";
flow:established,to_server;
content:"proxy.pracovnidomena.eu"; http_host; sid:1;)

alert http any any -> any any (msg:"Zneuziti formulare";
flow:established,to_server; content:"POST";
content:"Funguje"; nocase; http_client_body; rev:1; sid:2;)
```

První z pravidel testuje příchozí HTTP komunikaci z jakékoli zdrojové IP adresy a zdrojového portu směřující na jakoukoli cílovou IP adresu a port. Pomocí klíčového slova flow a jeho parametrů „established“ a „to_server“ je zajištěno, že se bude týkat pouze o komunikaci již navázanou a směřující od klienta k serveru. Navíc musí

komunikace směřovat na doménu proxy.pracovnidomena.eu. V takovém případě je vygenerován alert do souboru s logy. Druhé pravidlo je velice podobné, ale na rozdíl od prvního kontroluje komunikaci, která obsahuje HTTP metodu POST a v jejím těle kontroluje obsah jenž je na server odesílán. V tomto případě musí obsahovat slovo „funguje“ a to bez ohledu na velikost písmen. Toto konkrétní pravidlo bylo testováno právě pomocí kontaktního formuláře na stránkách proxy.pracovnidomena.eu.

9. Optimalizace původních pravidel

Před vytvořením nového řešení byl již systém IDS/IPS Suricata ve firemním prostředí WEDOS Internet, a.s. používán. Toto staré řešení ovšem nebylo dostačující kvůli stále rostoucímu počtu hrozeb přicházejících prostřednictvím protokolu HTTPS a také bylo vzhledem ke změně infrastruktury ve společnosti nutné IDS/IPS systém Suricata převést do prostředí Docker. Po této změně a vytvoření nového řešení IDS/IPS systému Suricata v prostředí Docker bylo však zapotřebí použít některá stará bezpečnostní pravidla, která již provoz ve firemním prostředí filtrovala. Bylo důležité určit jaká pravidla mají být přenesena do nového řešení a jaká naopak nikoli. Tento proces byl rozdělen na dvě části. Na část, kdy byl kladen důraz na pravidla stažená po prvotním spuštění IDS/IPS systému Suricata a pravidla vytvořená přímo společností WEDOS Internet, a.s.

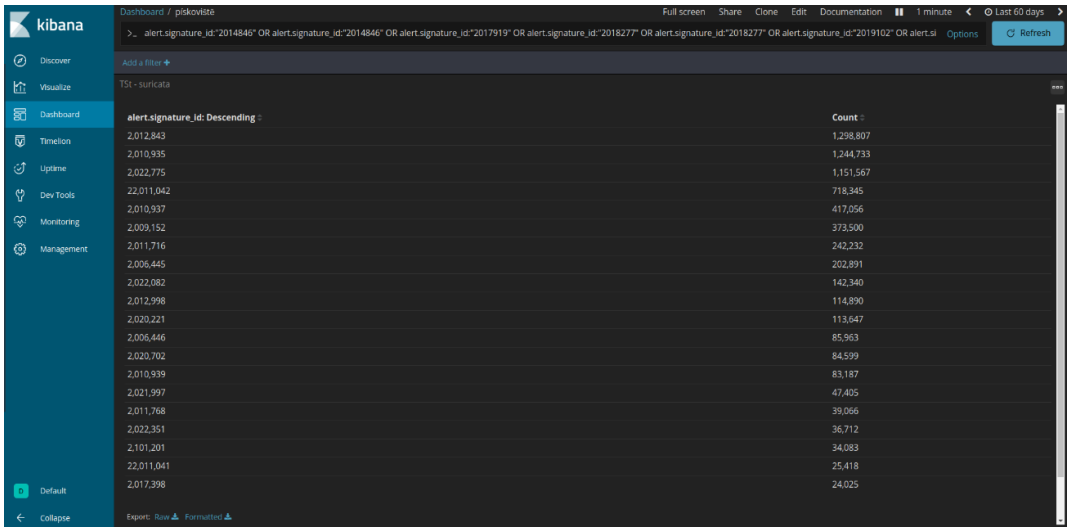
9.1. Použití defaultních pravidel

Po prvotním spuštění IDS/IPS systému Suricata byla stažena defaultní pravidla ze zdroje „rules.emergingthreats.net“. Tato všechna pravidla obsahují klíčové slovo „alert“, jenž označuje, že po prvotním spuštění systému nedochází k žádné blokaci ale pouze k upozornění na podezřelý provoz, který těmto pravidlům odpovídá. Velká část pravidel je také zakomentována a není vůbec do filtrace zapojena. Jaká konkrétní defaultní pravidla mají být skutečně použita, bylo zjištěno již před touto prací při tvorbě prvního řešení systému IDS/IPS Suricata ve společnosti WEDOS Internet, a.s. „Tento test byl proveden za pomoci odkomentování všech zakomentovaných pravidel a podle logů bylo zjištěno jaká pravidla mají být použita.“ [41] Modifikace těchto pravidel poté byla zaznamenána do souboru oinkmaster.conf, který umožňuje modifikaci provést. Nicméně od doby, kdy bylo vytvořeno prvotní řešení, uběhla již dlouhá doba a některá pravidla již nebyla používána. Aby bylo možné zjistit, jaká pravidla jsou používána a jaká již používána nejsou, byl soubor oinkmaster.conf prozkoumán za pomoci vytvořeného skriptu vypis_ID.sh v jazyce bash. Skript byl vytvořen tak, aby vyhledával v souboru oinkmaster.conf klíčové slovo „modifysid“, které sloužilo právě pro modifikaci konkrétního pravidla s určitým ID. Ve skriptu také probíhala kontrola, na jaké pozici se modifikované ID pravidla nachází, a to z toho důvodu, že bylo možno modifikovat samotné ID pravidla, což způsobovalo problém při následném vyhledávání pravidel podle těchto identifikátorů. Všechna identifikační čísla pravidel byla ze souboru

oinkmaster.conf zaznamenána do pomocného souboru ID_from_oinkmaster.txt. Tento soubor tak obsahoval všechny identifikátory z prvotně stažených defaultních pravidel, která byla použita ve starém řešení. Následně byla pomocí druhého skriptu vypis_pravidel.sh jednotlivá pravidla na základě identifikátorů nalezena a vypsána do souboru wedos_default_ALL.rules. Pro zjištění, jaká pravidla již použita v současné době nebyla, došlo k převedení jednotlivých ID do podoby, kterou dokázal zpracovat firemní systém pro vizualizaci souborů s logy. Konkrétně se jednalo o open-source systém Kibana. Podoba jednotlivých příkazů pro systém Kibana byla následující:

```
alert.signature_id:"ID"
```

Místo slova ID byl doplněn identifikátor konkrétního pravidla. Pro řetězení těchto příkazů bylo mezi ně vloženo klíčové slovo OR. Všechny příkazy jsou k dispozici v souboru ID_for_kibana.txt. Po vložení do systému Kibana a vyhledání všech dostupných logů za 60 dnů, došlo k zobrazení počtu použití jednotlivých bezpečnostních pravidel. Rozhraní systému Kibana je znázorněno na následujícím obrázku č. 15.



Obrázek č. 15 Open source systém Kibana a znázornění počtu použití jednotlivých bezpečnostních pravidel

Z tohoto rozhraní byl vyexportován textový soubor, obsahující identifikátory pravidel, které byly alespoň jednou za určitou dobu použity. Soubor byl pojmenován ID_used_from_kibana.txt. Pomocí souboru s identifikátory použitých pravidel a souboru ID_from_oinkmaster.txt, ve kterém byly všechny identifikátory, bylo možné vypsat všechny pravidla do dvou různých souborů. První soubor obsahoval pouze používaná

bezpečnostní pravidla a druhý naopak pravidla nepoužívaná. Tento soubor byl důležitý z pohledu budoucí možnosti, že některá pravidla, která nyní nejsou používána, použita opět budou. Oba soubory byly naformátovány, tak aby se v nich dalo lépe orientovat a zároveň došlo k odstranění duplicitních pravidel, která byla přidána ze starých souborů s pravidly, jenž již dávno nebyly uvedeny v konfiguračním souboru suricata.yaml.

Na závěr bylo nutné změnit v bezpečnostních pravidlech klíčové slovo „drop“, sloužící pro zahození celé komunikace odpovídající tomuto pravidlu, na reject. Tato změna byla důležitá pro proxy server, který při násilném odmítnutí komunikace nedokázal identifikovat, proč k odmítnutí došlo. Z tohoto důvodu bylo nastaveno klíčové slovo „reject“, pomocí kterého dochází při zamítnutí komunikace k odeslání zprávy o přerušení spojení obou stranám. Tato změna byla provedena ve všech pravidlech obsahující klíčové slovo „drop“ za pomoci vytvořeného jednoduchého bash skriptu drop_to_reject.sh. Poslední úprava ve výsledném souboru s bezpečnostními pravidly wedos_default_USED.rules proběhla po optimalizaci pravidel společnosti WEDOS Internet, a.s. v následující kapitole.

9.2. Optimalizace pravidel společnosti WEDOS Internet, a.s.

Společnost WEDOS Internet, a.s. za dobu používání starého řešení IDS/IPS systému Suricata vytvořila několik svých vlastních pravidel. Tato pravidla korespondovala s útoky, které v danou dobu probíhaly, a to jak na samotné servery společnosti WEDOS Internet, a.s. tak i na klientské služby. Nicméně mnoho z těchto vytvořených bezpečnostních pravidel již nebylo potřeba používat, ať už z pohledu ukončení konkrétní služby zákazníka, na které bylo pravidlo psáno, či na neaktuálnost daného útoku. Tato pravidla byla vytvořena ve dvou odlišných souborech. Pro optimalizaci pravidel došlo ke sloučení obou souborů do jednoho. Tento soubor byl posléze prohledán bash skriptem sloužícím k výpisu jednotlivých ID všech zde umístěných bezpečnostních pravidel. Skript ukládal všechny identifikátory do souboru wedos_our_ID_ALL.txt a zároveň vytvářel soubor (ID_for_kibana.txt) obsahující příkazy, které dokáže zpracovat firemní systém pro správu logů Kibana. Pomocí systému Kibana bylo opět určeno, jaká pravidla byla použita v předešlých 60 dnech a ta poté byla uložena do souboru ID_from_kibana_USED.txt. Bohužel vzhledem k obtížnosti formátování zápisu pravidel společnosti WEDOS Internet, a.s. nebylo jednoduše možné vypsat použitá pravidla do samostatného souboru.

Vzhledem k této skutečnosti byla používaná pravidla ručně přesouvána z původního souboru se všemi pravidly do nového souboru wedos.rules. Nepoužitá pravidla, která nebyla přesunuta do souboru wedos.rules, byla uložena do souboru wedos_rules_NOT_USED.rules. Tento soubor byl následně sloučen se souborem s nepoužívanými pravidly z předešlé podkapitoly a byl přejmenován na backup.rules. Opět byla provedena změna klíčového slova „drop“ na „reject“ pomocí již dříve použitého bash skriptu drop_to_reject.sh. Na úplný závěr byl soubor wedos.rules a výsledný soubor s defaultními pravidly z předešlé kapitoly upraven tak, aby identifikátory odpovídaly rozsahu pro lokální pravidla [42] a nezasahovaly nijak do identifikátorů prvotně stažených defaulních pravidel. V opačném případě by mohlo dojít k duplikacím ID u úplně odlišných pravidel. Tato změna byla provedena za pomoci nově napsaného skriptu v jazyce bash change_ID.sh. Skript byl určen pro změnu ID u obou výsledných souborů. Změna se netýkala pouze souboru s nepoužívanými pravidly backup.rules, který sloužil pouze jako záloha pro možné budoucí použití a nebyl uveden ani v konfiguračním souboru suricata_zmeny.yaml. Po uvedení výsledných souborů v konfiguračním souboru systému Suricata a následném restartu kontejneru bylo v pravidlech odhaleno ještě několik chyb spojených s formátováním, duplikací pravidel a nedostatkem paměti. Po ruční úpravě výsledných souborů a opětovném znovu načtení pravidel v IDS/IPS systému Suricata, byly chyby spjaté s duplicitami a formátováním eliminovány. Nicméně problém s nedostatkem paměti přetrvával. V souboru se záznamem spouštění systému Suricata se vyskytovala tato chyba:

```
[ERRCODE: SC_ERR_NO_REPUTATION(224)] - failed to get a host, increase host.memcap
```

Tento problém byl odstraněn až po úpravě specifických parametrů v konfiguračním souboru suricata_zmeny.yaml. Jednalo se o nastavení paměti u položek defrag, flow, steam a host, které v počátečním nastavení alokovali paměť pouze v řádu desítek maximálně stovek megabitů. Tyto hodnoty byly mnohonásobně navýšeny.

10. Detekce hrozeb a tvorba nových bezpečnostních pravidel

Po optimalizaci původních pravidel bylo možné k těmto pravidlům přidat pravidla zcela nová, která měla za úkol zamezit aktuálním hrozbám směřujícím na služby klientů společnosti WEDOS Internet, a.s. Velkou aktuální hrozbou, která postihovala obrovské množství klientů společnosti WEDOS Internet, a.s. byly nezabezpečené kontaktní, registrační či komentářové formuláře. Tento problém byl spojen s jednoduchou instalací redakčních systémů, jenž společnost podporuje. Nicméně většina volně dostupných redakčních systémů nemá ihned po instalaci implementovanou jakoukoli ochranu proti odesílání nevyžádaných zpráv prostřednictvím běžně používaných formulářů. Nejpoužívanější redakční systémy ve společnosti WEDOS Internet, a.s. byly Wordress, Prestashop a Joomla. Četnost použití jednotlivých redakčních systémů byla zjištěna na základě výsledků programu této společnosti, který pomáhá zákazníkům automaticky nainstalovat redakční systém, bez nutnosti jej stahovat, vytvářet databázi a nastavovat přístupové údaje do příslušné databáze.

Hrozby postihující běžně používané formuláře byly odhaleny prostřednictvím překročených denních limitů pro odeslání zpráv přes formuláře využívající php funkci mail(), případně pomocí počtu nedoručitelných e-mailů využívajících totožnou php funkci. Na webhostinzích společnosti WEDOS Internet, a.s. jsou tyto denní limity nastaveny na 500 zpráv prostřednictvím php funkce mail(). Kvůli odhalení, že se skutečně jedná o nevyžádané zprávy, došlo k nahlédnutí do jejich obsahu. Toto je povoleno pouze pokud dochází k porušování výše zmíněného limitu nebo nemožnosti zprávy doručit příjemci. Obecný vzorec pro vyhodnocení spamových zpráv u jednotlivých webhostingů byl následující:

```
SPAM = (překročen limit odchozích zpráv u webhostingu |
adresát neexistuje) & zpráva obsahuje podezřelá slova, fráze
či odkazy & zpráva se opakuje ve stejné nebo mírně upravené
podobě vícekrát
```

Po kontrole velkého počtu problémových zpráv, které byly vybrány na základě předešlého vzorce, došlo k výběru konkrétních slov, frází, e-mailových adres,

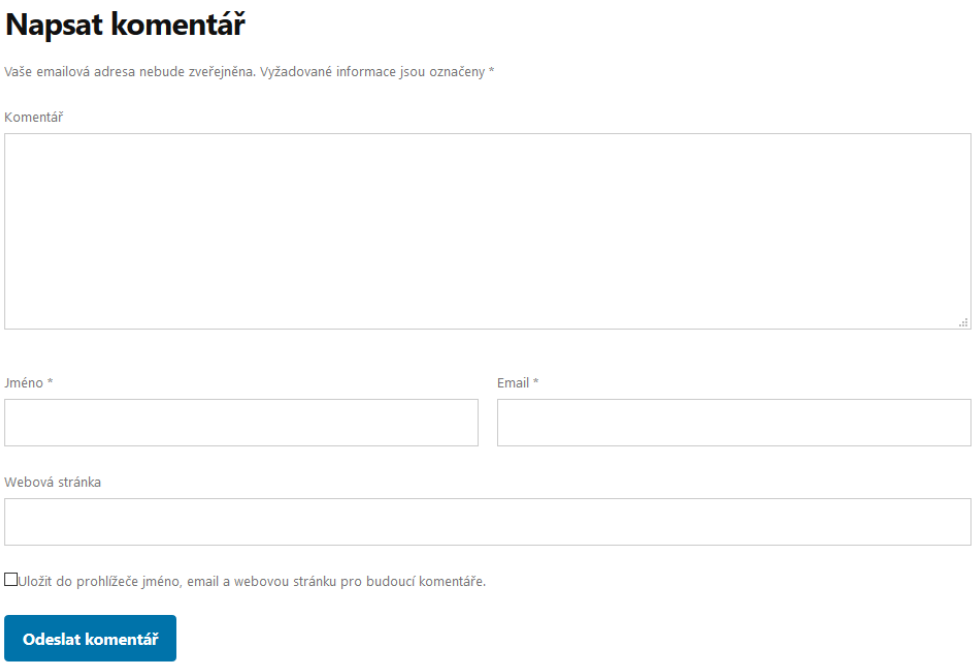
problémových domén a dalších parametrů, které mohly být následně použity v jednotlivých bezpečnostních pravidlech.

10.1. Nová bezpečnostní pravidla

První nové bezpečnostní pravidlo s identifikačním číslem sid:1 bylo použito již pro otestování funkčnosti výsledného řešení v kapitole číslo 8, kde ovšem mělo identifikační číslo sid:2. Toto pravidlo testuje také zneužití nezabezpečeného formuláře, ale pouze pro testovací účely a neobsahuje téměř žádné parametry ze spamových zpráv. Jediné, co je u pravidla již použito, je parametr flow s hodnotami „established“ a „to_server“. Tyto hodnoty umožňují kontrolovat již navázané spojení, a to ve směru od klienta k serveru. V testovacím pravidle je dále klíčové slovo content s hodnotou „Funguje“ zaručující aktivaci pravidla při odeslání formuláře právě s tímto konkrétním slovem. Toto testovací pravidlo se s mírnými úpravami stalo základem pro pravidla následující. Tvorba nových bezpečnostních pravidel byla rozdělena do tří skupin. První skupinou byla pravidla pro redakční systém Wordpress. Druhou skupinou byla pravidla pro redakční systém Prestashop a třetí skupinou byla pravidla pro redakční systém Joomla. Pro testování nových pravidel byly jednotlivé redakční systémy nainstalovány na pokusné subdomény. Testování systému Wordpress verze 5.2.2, Prestashop verze 1.6.1.13 a Joomla verze 1.5.0 a 3.9.9 probíhalo na subdoméně proxy se začátečním písmenem každého redakčního systému. Celý tvar subdomén byl následující: proxy-w.pracovnídomena.eu, proxy-p.pracovnídomena.eu a proxy-j.pracovnídomena.eu. Pro testování pravidel byly využity také dva online nástroje. Prvním byl nástroj pro testování správnosti regulárních výrazů [43] a druhým nástrojem byl převodník textu kódovaném v ASCII na hexadecimální podobu a opačně [44].

10.1.1. Bezpečnostní pravidla pro redakční systém Wordpress

Redakční systém Wordpress byl nejpočetnějším redakčním systémem, jenž byl u společnosti WEDOS Internet, a.s. instalován prostřednictvím instalátoru aplikací. Po provedení čisté instalace sice neobsahoval běžný kontaktní formulář jako další redakční systémy, ale obsahoval nezabezpečený formulář pro komentáře. Formulář je znázorněn na obrázku č. 16.



Obrázek č. 16 Formulář pro komentáře v redakčním systému Wordpress

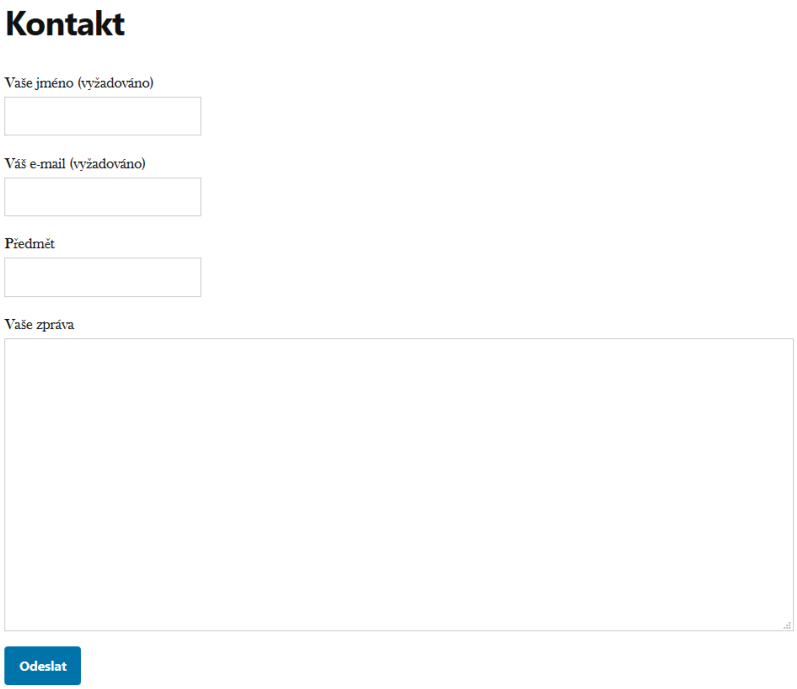
Jak je patrné na obrázku výše, komentářový formulář se skládal z několika polí. Prvním bylo pole pro samotné vložení komentáře, druhým pro jméno uživatele, následováno polem pro e-mailovou adresu a webovou stránku. Za všemi poli se nacházelo tlačítko sloužící pro odeslání vložených informací na server bez jakéhokoli bezpečnostního prvku. Aby bylo možné bezpečnostní pravidla vytvořit, bylo nezbytné určit přesnou podobu odeslaných dat. K tomuto účelu byl využit nástroj pro monitoring sítě v nástrojích pro vývojáře v prohlížeči Chrome. Tělo požadavku bylo následující:

```
comment=test&author=test&email=test%40test.cz&url=http%3A%2F%2Ftest.cz&submit=Odeslat+koment%C3%A1%C5%99&comment_post_ID=1&comment_parent=0
```

Na základě celého HTTP požadavku bylo možné určit přesnou podobu bezpečnostních pravidel zabraňujících zneužití formuláře. Jednalo se o dvě bezpečnostní pravidla pro komentářový formulář. Obě pravidla jsou dostupná v přílohách této práce na straně 81. Pravidlem s řadovou číslovkou 2. bylo docíleno odfiltrování HTTP POST požadavků, které obsahovaly ve svém těle data z pole „author“. Toto pole je součástí komentářového formuláře a bylo často zneužíváno k odesílání informací přímo v názvu autora komentáře. Proto došlo k omezení počtu znaků, jenž do tohoto pole je možno zadat. Pokud byla do pole vložena celá věta došlo k překročení 20 a více znaků a HTTP požadavek byl

pravidlem zachycen a odfiltrován. Velké množství spamu mělo počet znaků v rozmezí od 20 do 30 znaků. Druhé z bezpečnostních pravidel pro systém Wordpress s identifikačním číslem 3 obsahovalo kombinaci polí e-mail a comment. V poli e-mail byly uvedeny podezřelé domény, které se často vyskytovaly ve spamových e-mailech. Jednalo se o domény .ru, .xyz, .pl případně domény druhého řádu yahoo.com, qq.com, ttmail.pro, yandex.com. V poli comment se naproti tomu nacházela slova, která musel komentář obsahovat: sex, porn, viagra, casino, nudism, pussy, dating-sites, hot teen, naked, teens a url odkazy s doménami .ru, .com, .hk, .net .site a .org. Tato kombinace tvořila dostatečně bezpečné pravidlo pro filtrování požadavků odeslaných prostřednictvím HTTP metody POST vytvořených pomocí komentářového formuláře.

Jelikož redakční systém Wordpress v základním nastavení nezahrnoval kontaktní formulář, byl často doinstalován pomocí pluginu. Z důvodu existence nepřeberného množství pluginů, které sloužily právě pro doplnění možnosti kontaktu prostřednictvím kontaktního formuláře, byl vybrán jeden konkrétní plugin, který se často nacházel na zákaznických webhostinzích odesílajících spam. Jednalo se přesně o plugin „Contact Form 7“.



Obrázek č. 17 Formulář kontaktů pluginu Contact Form 7

Na doméně proxy-w.pracovnidomena.eu, kde se nacházela testovací verze redakčního systému Wordpress, byl tento plugin doinstalován a jeho podobu je možno vidět na obrázku č. 17. Po nainstalování a odeslání několika požadavků bylo zjištěno, že bezpečnostní pravidla, která byla v procesu testování, nebyla funkční. I přes zobrazení požadavku v prohlížeči nebylo možno přesně specifikovat podobu celého požadavku. Tato metoda nebyla dostačující a odeslaná data mohla obsahovat znaky, které nebyly tímto nástrojem zachyceny. Proto došlo k přesměrování pokusných webových stránek na testovací verzi IDS/IPS systému Suricata, kde bylo možné zapnout v konfiguračním souboru v sekci logů položku „payload-printable“, určenou pro zaznamenávání celého HTTP požadavku i s daty. Tato položka nemohla být zapnuta na produkční verzi systému Suricata z důvodu velkého zatížení serveru, který by musel ukládat obrovské množství dat. Podoba HTTP požadavku byla následující:

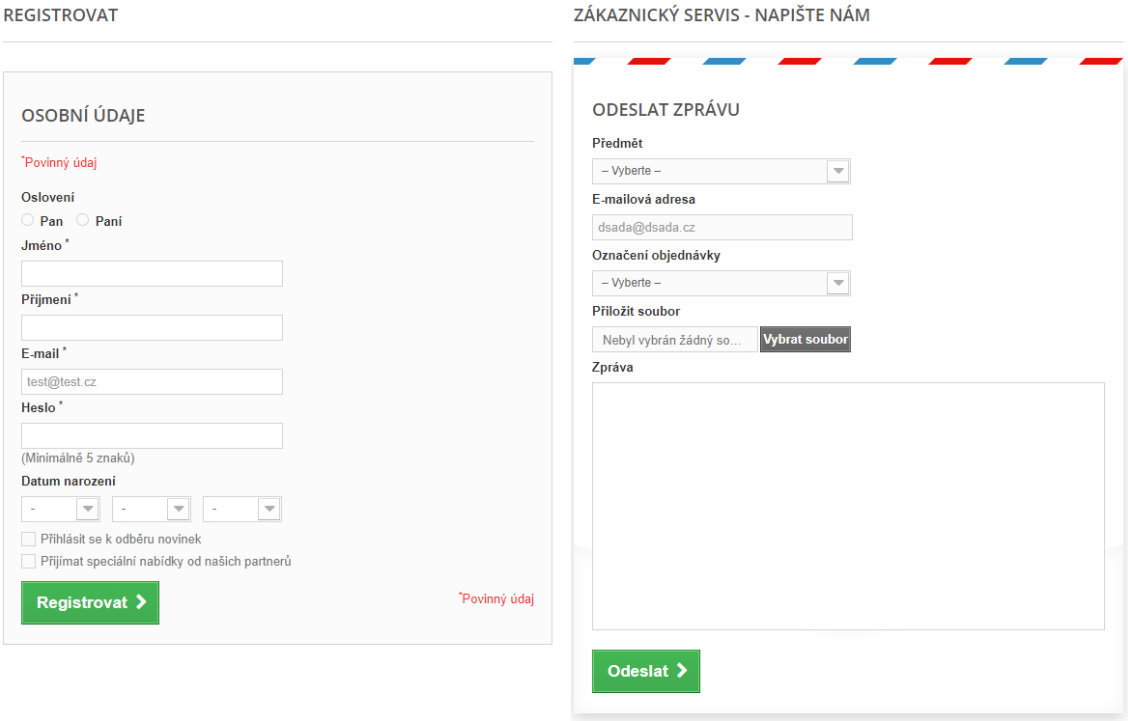
```
Content-Disposition: form-data; name=\"your-
name\"\r\n\r\nTESTOVACI JMENO\r\n------
WebKitFormBoundaryjA71lqPkKNzMUVj1\r\nContent-Disposition:
form-data; name=\"your-email\"\r\n\r\ntest@test.ru\r\n-----
-WebKitFormBoundaryjA71lqPkKNzMUVj1\r\nContent-Disposition:
form-data; name=\"your-subject\"\r\n\r\nPOKUS\r\n------
WebKitFormBoundaryjA71lqPkKNzMUVj1\r\nContent-Disposition:
form-data; name=\"your-message\"\r\n\r\nTEST\r\n------
WebKitFormBoundaryjA71lqPkKNzMUVj1--\r\n","stream":1}
```

Jedná se pouze o zkrácenou podobu z důvodu velikosti celého HTTP požadavku. Nicméně je zde patrné, že jednotlivá pole obsahovala neviditelné znaky `\r\n`. Po úpravě prvotně testovaných pravidel byla vytvořena dvě pravidla, která byla určena k omezení spamu odesílaného přes kontaktní formulář pluginu „Contact Form 7“. Obě pravidla byla přidána do příloh pod čísly 4 a 5 na straně 81. Pravidlo číslo 4 bylo podobné pravidlu s identifikačním číslem 3, které testovalo POST požadavky na přítomnost e-mailové adresy s doménami .ru, .xyz a doménami druhého řádu yahoo.com, qq.com. E-mailová adresa byla v případě kontaktního formuláře pluginu „Contact Form 7“ skryta pod polem „your-email“. Druhé pole, které bylo testováno, bylo pole „your-message“, u kterého byl prozkoumáván obsah zahrnující podezřelá slova a odkazy. Aby bylo jasné, že se jedná o požadavky pocházející právě z tohoto pluginu, bylo do pravidla přidáno klíčové slovo

content s hodnotou 5f7770636637. Hodnota klíčového slova content byla uvedena v šestnáctkové soustavě, aby byl text správně interpretován. V kódování ASCII byla podoba textu „_wpcf7“ bez uvozovek. Pravidlo číslo 5 mělo za úkol filtrovat HTTP požadavky obsahující pole „your-name“ s hodnotou obsahující odkaz s doménami .ru, .com, .hk, .ly, .mp, .tech, .nl, .us a doménou druhého řádu bitly.com. V názvu autora odesílaných informací byl hypertextový odkaz použit pouze v případě, pokud se jednalo o nevyžádanou zprávu. Legitimní zprávy obsahovaly jméno osoby, která použila kontaktní formulář.

10.1.2. Bezpečnostní pravidla pro redakční systém Prestashop

Redakční systém Prestashop byl druhým nejvíce instalovaným redakčním systémem skrze aplikaci na instalaci cms ve společnosti WEDOS Internet, a.s. Jednalo se však o redakční systém, který byl nejvíce zneužíván k odesílání nevyžádaných zpráv. Nejčastější výskyt spamových zpráv byl spojen s kontaktním formulářem a v některých případech i s formulářem registračním. Oba se nacházely v redakčním systému ihned po nainstalování a nebyla u nich přidána žádná ochrana proti zneužití. Podoba těchto formulářů je zachycena na následujícím obrázku č. 18.



Obrázek č. 18 Registrační a kontaktní formulář redakčního systému Prestashop

První výše uvedený formulář byl registrační. Sloužil pro vytvoření nového uživatele a obsahoval povinné pole jméno, příjmení, e-mail a heslo. Nepovinná pole pro vytvoření bezpečnostních pravidel nebyla důležitá, protože ve většině případů nebyla odesílatelem nevyžádané pošty vyplněna. Při tvorbě bezpečnostních pravidel bylo důležité, stejně jako v případě pravidel pro redakční systém Wordpress, specifikovat přesnou podobu odeslaného HTTP požadavku. Pro tento druh formuláře bylo dostačující použití monitoringu sítě v nástrojích pro vývojáře v prohlížeči Chrome. Pro kontrolu registračního formuláře byla vytvořena dvě bezpečnostní pravidla. Obě pravidla kontrolovala přítomnost hypertextového odkazu ve jméně či příjmení uživatele. Obě jsou zaznamenána v přílohách na straně 83 pod čísly 6 a 7. Pravidlo číslo 6 filtrovalo síťový provoz odeslaný prostřednictvím HTTP metody POST a obsahující hypertextový odkaz s doménami .ru, .com, .net, .info, .pl, .org, .cf, .ga, .me, a .gg v křestním jméně uživatele. Konkrétní pole bylo v pravidle uvedeno v hexadecimální soustavě pod hodnotou „637573746f6d65725f66697273746e616d653d“. V kódování ASCII byla hodnota rovna výrazu „customer_firstname=“. Dále byl kontrolován typ HTTP obsahu, který musel být roven "application/x-www-form-urlencoded". Pravidlo číslo 7 je velice podobné, ale kontroluje hodnotu v poli s příjmením. Hexadecimální hodnota pro pole „customer_lastname=“ byla „637573746f6d65725f6c6173746e616d653d“.

Druhým výše znázorněním formulářem byl formulář kontaktní. Jednalo se o nejčastěji zneužívaný formulář v tomto redakčním systému. Obsahoval povinná pole předmět, e-mail, označení objednávky a pole pro zanechání výsledné zprávy. HTTP požadavek prostřednictvím metody POST nebylo možno kompletně zaznamenat prostřednictvím prohlížeče Chrom. Z tohoto důvodu byl požadavek opět získán za pomoci testovacího prostředí systému Suricata, kde bylo v konfiguračním souboru povoleno zaznamenávat do souborů s logy i celé tělo požadavku. Na základě tohoto požadavku došlo k vytvoření dvou bezpečnostních pravidel s pořadovými čísly 8 a 9 v přílohách práce na straně 84. Osmé pravidlo sloužilo k zamítnutí síťové komunikace v případě, že komunikace obsahovala POST požadavek přenášející hodnoty v poli e-mail (from) rovné e-mailu s doménou .ru, .xyz, yahoo.com, qq.com, nebo doménou druhého řádu složenou pouze z čísel a koncovky .com. Společně s těmito hodnotami muselo být vyplněno také pole zprávy (message) obsahující podezřelá slova a hypertextové odkazy.

Z pravidla bylo v průběhu testování odstraněno několik podmínek. Tyto podmínky byly následující:

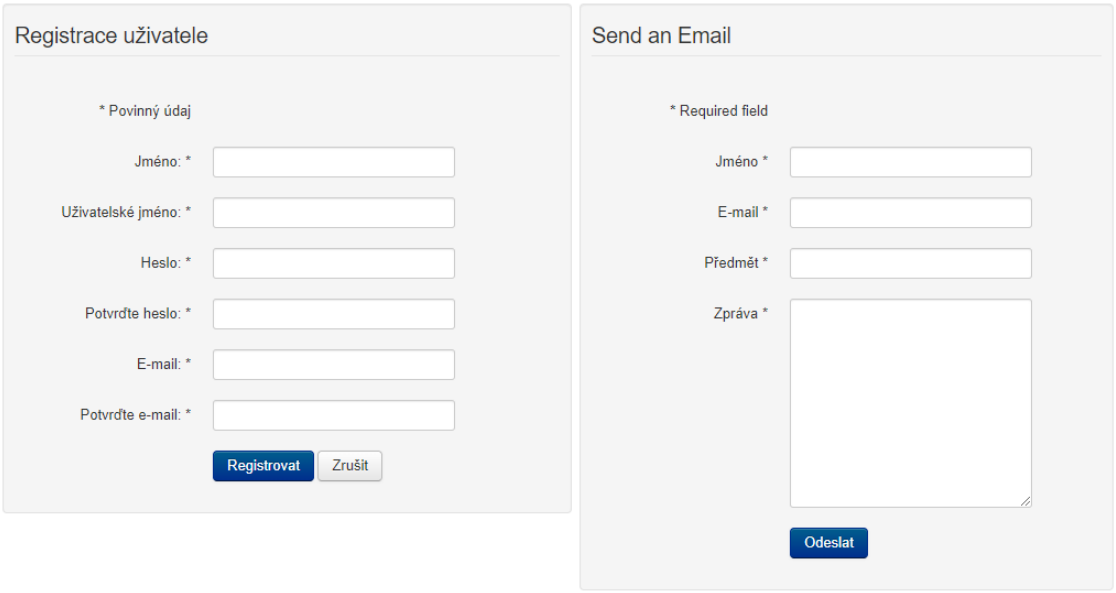
- `content:"prestashop"; nocase; http_cookie; \`
- `content:"|2f696e6465782e7068703f636f6e74726f6c6c65723d636f6e74616374|"; http_uri; \`
- `http_content_type; content:"application/x-www-form-urlencoded"; \`

První podmínka sloužila k zachycení síťové komunikace pouze, pokud bylo v hlavičce cookie přítomno slovo prestashop. Nicméně bylo zjištěno, že ne všechny webhostingy využívající redakční systém Prestashop a odesílající nevyžádané zprávy obsahovaly v HTTP požadavku v hlavičce cookie toto konkrétní slovo. Druhou podmínkou byl obsah url adresy, který musel obsahovat řetězec „index.php?controller=contact“ v ASCII kódování. Tato podmínka nebyla nastavena správně, protože řetězec bylo možno měnit uživatelem přímo v nastavení redakčního systému, a proto nebylo jeho použití pro větší množství webhostingů vhodné. Poslední podmínkou, která byla z pravidla vyloučena, byl typ obsahu HTTP požadavku. Původní hodnota byla nastavena na „application/x-www-form-urlencoded“, ale některé webhostingy s tímto redakčním systémem obsahovaly jinou hodnotu typu obsahu HTTP požadavku. Po dokončení předchozího pravidla bylo vytvořeno pravidlo s číslem 9. To sloužilo k zachycení komunikace obsahující také e-mail s hodnotami jako předchozí pravidlo, ale již nekontrolovalo hodnoty v poli message. Toto pravidlo bylo vytvořeno v důsledku objevení nového typu spamu, který obsahoval v poli message určitý typ čínských znaků, které narušily strukturu požadavku a nebylo možno je pomocí regulárního výrazu účinně blokovat. Proto byly požadavky zachyceny přímo na webovém serveru, kde se nacházel webhosting postižený tímto problémem a na základě celých požadavků bylo odhaleno, že nevyžádané zprávy obsahují vždy stejné hlavičky Accept-Language a User-Agent. U hlavičky preferovaného jazyka se hodnota rovnala „zh-CN“ a u hlavičky identifikující prohlížeč začínala hodnota na „Mozilla/5.0“ Pomocí této kombinace byl spam úspěšně odfiltrován.

10.1.3. Bezpečnostní pravidla pro redakční systém Joomla

Posledním redakčním systémem, pro který byla tvořena bezpečnostní pravidla, byl systém Joomla. Počet instalací ve společnosti WEDOS Internet, a.s. byl téměř stejně

početný jako u předchozího redakčního systému. Po nainstalování systému Joomla na testovací subdodoménu proxy-j.pracovnidomena.eu došlo k povolení registrace nových uživatelů. Tento úkon je nutný pouze u novějších verzích redakčního systému Joomla. Starší verze mají registraci nových uživatelů povolenou ihned po instalaci. Komponenta kontaktního formuláře byla přítomna ihned po nainstalování. Bylo ji ovšem nutné přiřadit na konkrétní stránku, aby byla běžným uživatelům dostupná. Oba formuláře jsou zachyceny na obrázku č. 19.



Obrázek č. 19 Registrační a kontaktní formulář redakčního systému Joomla

Registrační formulář obsahoval šest povinných polí. Mezi tyto pole patřilo jméno, uživatelské jméno, heslo a potvrzení hesla, e-mail a na závěr potvrzení e-mailu. Nevyžádané zprávy často obsahovaly hypertextový odkaz v poli jméno nebo uživatelské jméno. Proto bylo vytvořeno pravidlo s číslem 10 právě pro tento případ. Pokud se v polích jform[name] nebo jform[username] vyskytl odkaz s doménou .ru, .com, .pl, .tw, .org, .us, .fr, .net, .tk, .la, .gd, .do, .ht, .uk, .link, .jp, .cf, .de nebo .me byl požadavek zachycen a odmítnut. Pravidlo obsahovalo hexadecimální hodnotu „6a666f726d5b6e616 d655d“, která určovala, o jaké konkrétní pole se jedná. V tomto případě šlo o řetězec „jform[name]“ v ASCII kódování. Bohužel bylo zjištěno, že ve starší verzi redakčního systému Joomla, konkrétně ve verzi 1.5, jsou názvy polí jiné a původní pravidlo pro kontrolu registračního formuláře nebylo funkční. Proto byla vytvořena dvě nová pravidla, která měla za úkol bránit odchozímu spamu z registračních formulářů starší verze

systému. Všechna tři pravidla jsou umístěna v přílohách na straně 85. Pravidlo s číslem 11 bylo téměř totožné jako pravidlo předchozí, ale upravovalo názvy a strukturu polí z HTTP požadavku tak, aby odpovídalo starší verzi systému Joomla. Další pravidlo omezující spam prostřednictvím registračního formuláře je pravidlo s číslem 12. Toto pravidlo omezuje počet znaků pro pole jméno (name). Velké množství nevyžádané pošty obsahovalo pole „jméno“, zahrnující nahodilé znaky a číslice, které tvořily dvanácti a více znakový řetězec. Jelikož se ovšem jednalo o poměrně malé číslo a mohlo by docházet k blokaci i legitimních registrací, byla do pravidla přidána ještě jedna podmínka. Jednalo se o podmínku zahrnující pole „email“ a jejím obsahem musel být e-mail s doménou .ru, .xyz, .pl, .au, případně doménou druhého řádu yahoo.com, qq.com, course-fitness.com, live.com, yandex.com nebo hotmails.com. Tato kombinace podmínek tvořila dostatečně účinné bezpečnostní pravidlo pro filtraci spamu z registračního formuláře.

Posledním nezabezpečeným formulářem, pro který bylo vytvořeno bezpečnostní pravidlo, byl formulář kontaktní, který obsahoval položky jméno, e-mail, předmět a zpráva. Přesná podoba HTTP požadavku byla určena opět pomocí nástroje pro monitoring sítě v nástrojích pro vývojáře v prohlížeči Chrome. Tělo s daty z HTTP požadavku mělo následující podobu:

```
jform%5Bcontact_name%5D=test&jform%5Bcontact_email%5D=test%
40test.cz&jform%5Bcontact_subject%5D=test&jform%5Bcontact_m
essage%5D=test&option=com_contact&task=contact.submit&retur
n=&id=1%3Acontact-us&4ae4c287b05a092ee7c34d81ed480b87=1
```

Pravidlo určené pro omezení nevyžádané zprávy skrze tento typ formuláře bylo pravidlo číslo 13, které obsahovalo kombinaci kontroly polí contact_email a contact_message. HTTP požadavek musel pro odmítnutí obsahovat hodnoty polí contact_email rovné e-mailu s doménami .ru a .xyz nebo doménami druhého řádu yahoo.com, qq.com, course-fitness.com a live.com. Pole contact_message muselo oproti tomu obsahovat slova porn či sex, případně odkazy obsahující domény .ru, .com, .pl a .vip. Všechny tyto hodnoty se nacházely především v nevyžádané komunikaci, která byla skrze formulář odesílána správci webové prezentace.

11. Porovnání stavu před a po nasazení bezpečnostních pravidel

Aby bylo možné otestovat účinnost bezpečnostních pravidel bylo nutné pravidla nasadit na reálné webhostingy zákazníků společnosti WEDOS Internet, a.s. Bohužel z důvodu ochrany osobních údajů nemůže být specifikováno o jaké konkrétní domény se jednalo, ale pravidla pro běžné použití nemusí mít přesně specifikovanou doménu a budou zamezovat hrozbě se spamem také. V přílohách této práce jsou pravidla přiřazena testovacím doménám, u kterých probíhalo testování a tvorba výsledných pravidel ještě před nasazením na zákaznické webhostingy. Pro zjednodušení byla vždy skupina pravidel nasazena na množinu 4 zákaznických webhostingů s příslušným redakčním systémem. U každé množiny webhostingů byl naměřen počet odeslaných e-mailů, prostřednictvím php funkce mail(), před a po nasazení skupiny pravidel. Měření trvalo vždy čtyři dny. Dva první dny množině webhostingů nebyly přiřazeny žádná bezpečnostní pravidla a bylo jim umožněno odesílání nevyžádaných zpráv. Bohužel nebylo možné nechat webhostingy odesílat nevyžádané zprávy delší dobu, aby nedošlo ke snížení reputace webových serverů, případně přidání jejich IP adres na veřejné blacklisty. Během dalších dvou dnů byl měřen počet odeslaných zpráv na stejné množině webhostingů, která již ovšem byla chráněna bezpečnostními pravidly. Testování a měření probíhalo v termínu od 11.11.2019 až do 2.12.2019. Výsledky měření byly zaznamenány do tabulky č. 2.

Tabulka č. 2 Počet odeslaných e-mailů prostřednictvím php funkce mail()

Množiny 4 webhostingů pro konkrétní CMS	1. den	2. den	3. den	4. den
Wordpress	1 642	1 186	2	50
Prestashop	2 259	1 978	148	227
Joomla	5 805	6 510	340	193

Jak je patrné z tabulky výše, tak bezpečnostní pravidla splnila svůj účel a opravdu omezila počet nevyžádaných zpráv, které webhostingy postihovaly. Snížení odeslaných zpráv prostřednictvím php funkce mail() bylo v řádu tisíců. Aby byla určena i účinnost jednotlivých bezpečnostních pravidel, byl změřen počet výskytů jednotlivých pravidel napříč množinou webhostingů. Výsledky byly zaneseny do tabulky č. 3. Z tabulky vyplývá, že některá bezpečnostní pravidla byla mnohem účinnější než ostatní. I když se

jednalo o malé množství výskytů daného pravidla, neznamená to, že bylo pravidlo zbytečné. U jednoho testovaného webhostingu, který byl z měření vyloučen, byl pozorován jev, kdy po pár hodinách od nasazení pravidel se přestaly nevyžádané zprávy odesílat a nenavyšoval se ani počet výskytů dané skupiny pravidel. Z toho vyplývá, že někteří spamboti si nejdříve ověří, zda lze bezproblémově formulář zneužít a pokud narazí na překážku, tak s odesíláním přestanou. Proto i pravidla s malým výskytem použití mohou být velmi užitečná.

Tabulka č. 3 Počet výskytů jednotlivých bezpečnostních pravidel

Jméno skupiny pravidel	ID pravidla	Počet výskytů
Wordpress	2	4185
	3	88
	4	194
	5	405
Prestashop	6	12
	7	12
	8	2358
	9	1196
Joomla	10	280
	11	866
	12	68
	13	681

Závěr

Prvním z cílů diplomové práce bylo nasazení open-source řešení IDS/IPS systému prostřednictvím platformy Docker v prostředí velkého poskytovatele internetových služeb. Pro splnění cíle bylo nutné se seznámit s platformou Docker a jejími vlastnostmi. Druhou teoretickou částí, důležitou pro splnění prvotního cíle, bylo prozkoumání IDS/IPS systémů a určení vhodného systému pro společnost WEDOS Internet, a.s. Na základě stanovených kritérií a požadavků společnosti, došlo k vybrání nejvhodnějšího IDS/IPS systému, kterým byl systém Suricata. Pomocí nově nabytých znalostí bylo vytvořeno výsledné řešení IDS/IPS systému Suricata v prostředí Docker, které bylo přesunuto z vývojového virtuálního serveru na proxy servery společnosti.

Druhým cílem diplomové práce bylo modifikovat stávající pravidla a vytvořit pravidla nová odpovídající na aktuální hrozby. Modifikace pravidel byla rozdělena na dvě části. První částí byla úprava defaultních pravidel systému Suricata a druhou částí byla modifikace původních pravidel společnosti WEDOS Internet, a.s. U obou částí bylo nutné zjistit, jaká pravidla byla reálně používána a jaká nikoli a používaná pravidla poté zanést do nového řešení IDS/IPS systému. U pravidel byla modifikována akce, kterou měla jednotlivá pravidla vykonávat. Tato změna se týkala akce drop na reject, u které neměly proxy servery problémy s nedostupností webového serveru. Na závěr došlo ke kompletnímu přečíslování identifikačních čísel, tak aby v pravidlech nedocházelo k duplicitnímu výskytu ID. Po úpravě počátečních pravidel byla vytvořena nová bezpečnostní pravidla, která odpovídala na aktuální problém postihující velké množství zákazníků společnosti WEDOS Internet, a.s. Tímto problémem byly nezabezpečené formuláře u nejčastěji instalovaných redakčních systémů Wordpress, Prestashop a Joomla.

Posledním cílem diplomové práce bylo experimentálně ověřit funkčnost výsledného řešení a nově vytvořených bezpečnostních pravidel. Funkčnost nově vytvořených bezpečnostních pravidel byla ověřena na základě nasazení skupiny pravidel pro jednotlivé redakční systémy na množiny složené ze čtyř zákaznických webhostingů postižené zranitelností nezabezpečeného formuláře pro konkrétní redakční systém. Měření probíhalo po dobu čtyř dnů, kdy první dva dny nebyly webhostingy chráněny a v druhých dvou dnech probíhalo měření již s nasazenými bezpečnostními pravidly.

Následně byl porovnán počet odeslaných e-mailů prostřednictvím php funkce mail() u jednotlivých množin webhostingů před a po nasazení bezpečnostních pravidel. Výsledky měření prokázaly účinnost nově vytvořených pravidel, kdy byl pokles výskytu nevyžádaných zpráv v řádech tisíců. Nově vytvořená bezpečnostní pravidla byla účinná na hrozby přicházející v období od 11.11.2019 až do 2.12.2019, kdy docházelo k filtrování nevyžádaných zpráv. I když jsou pravidla z velké části tvořena regulárními výrazy, které umožňují velkou flexibilitu podezřelých řetězců a hypertextových odkazů, tak je možné, že bez jejich aktualizace budou pravidla postupně ztrácet na účinnosti. Nicméně struktura bezpečnostních pravidel může být nadále zachována a nové podezřelé řetězce mohou být v budoucnu získávány například pomocí Bayesovo klasifikace a databáze podezřelých řetězců, které budou získávány automaticky z aktuálních nevyžádaných zpráv.

Seznam použité literatury

[1] MATTHIAS, Karl a Sean P. KANE. *Docker: up and running*. Sebastopol, CA: O'Reilly Media, [2015]. ISBN 978-1-491-91757-2.

[2] VOHRA, Deepak. Pro Docker. Berkeley, California: Apress, [2016]. ISBN 978-1-4842-1829-7.

[3] Docker facebook share. In: *Docker* [online]. [cit. 2019-12-01]. Dostupné z: https://www.docker.com/sites/default/files/social/docker_facebook_share.png

[4] About images, containers, and storage drivers. *Docker Documentation* [online]. [cit. 2019-12-01]. Dostupné z: https://docs.docker.com/v17.09/engine/userguide/storagedriver/imagesandcontainers/

[5] Docker overview. *Docker Documentation* [online]. [cit. 2019-12-01]. Dostupné z: https://docs.docker.com/engine/docker-overview/

[6] What is a Container? *Docker* [online]. [cit. 2019-12-01]. Dostupné z: https://www.docker.com/resources/what-container

[7] Comparing Virtual Machines vs Docker Containers. *Tips, Tricks and Tutorials* [online]. 2017 [cit. 2019-12-01]. Dostupné z: https://nickjanetakis.com/blog/comparing-virtual-machines-vs-docker-containers

[8] What is Hypervisor and what types of hypervisors are there. *Private cloud* [online]. 2016 [cit. 2019-12-01]. Dostupné z: https://vapour-apps.com/what-is-hypervisor/

[9] NOVOSELTSEVA, Ekaterina. TOP 10 BENEFITS YOU WILL GET BY USING DOCKER. *Software Development Company in Barcelona* [online]. 2018 [cit. 2019-12-01]. Dostupné z: https://apiumhub.com/tech-blog-barcelona/top-benefits-using-docker/

[10] TOZZI, Christopher. Docker Downsides. Channel Futures [online]. 2017 [cit. 2019-12-01]. Dostupné z: https://www.channelfutures.com/open-source/docker-downsides-container-cons-to-consider-before-adopting-docker

[11] Advantages and Disadvantages of Docker. *Live instructor* [online]. 2018 [cit. 2019-12-01]. Dostupné z: https://data-flair.training/blogs/advantages-and-disadvantages-of-docker/

[12] Serverless vs Docker Containers. *Medium* [online]. 2019 [cit. 2019-12-01]. Dostupné z: https://medium.com/@TechMagic/serverless-vs-docker-what-to-choose-in-2019-80cb80f4b680

[13] ENDORF, Carl F., Eugene SCHULTZ a Jim MELLANDER. Detekce a prevence počítačového útoku. Praha: Grada, 2005. ISBN 80-247-1035-8.

[14] HIDS. *Medium* [online]. 2016 [cit. 2019-12-01]. Dostupné z: https://en.bmstu.wiki/HIDS_(Host-Based_Intrusion_Detection_System)

[15] SZEP, David. Detekce a analýza průniků systémy IDS a IPS. Č. Bud., 2013. bakalářská práce (Bc.). JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH. Přírodovědecká fakulta

[16] PŘIBYL, Tomáš. IDS: základní informace. *Computerworld.cz* [online]. 2007 [cit. 2019-12-01]. Dostupné z: http://computerworld.cz/securityworld/ids-zakladni-informace-46154

[17] Suricata. *Open Source IDS / IPS / NSM engine* [online]. [cit. 2019-12-01]. Dostupné z: https://suricata-ids.org

[18] All features. *Open Source IDS / IPS / NSM engine* [online]. [cit. 2019-12-01]. Dostupné z: https://suricata-ids.org/features/all-features/

[19] KHALIL, George. SANS Institute Information Security Reading Room: Open Source IDS High Performance Shootout [online]. 2.2.2015, , 23 [cit. 2019-12-02]. Dostupné z: https://www.sans.org/reading-room/whitepapers/intrusion/open-source-ids-high-performance-shootout-35772

[20] TYRSINA, Radu. Best intrusion detection software for Windows. *Windows Report* [online]. 2019 [cit. 2019-12-01]. Dostupné z: https://windowsreport.com/5-best-intrusion-detection-software-pc/

[21] NAYYAR, Anand. The Best Open Source Network Intrusion Detection Tools. *News - Open Source For You* [online]. 2017 [cit. 2019-12-01]. Dostupné z: http://opensourceforu.com/2017/04/best-open-source-network-intrusion-detection-tools/

[22] Od 1. 1. 2018 budeme 100% opensource. *Datacentrum WEDOS* [online]. 2017 [cit. 2019-12-01]. Dostupné z: https://datacentrum.wedos.com/a/383/od-1-1-2018-budeme-100-opensource.html

[23] GERLICH, Tomáš. Detekce útoků cílených na odepření služeb [online]. Brno, 2017 [cit. 2019-11-24]. Dostupné z: http://hdl.handle.net/11012/65650. Diplomová práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací. Vedoucí práce Zdeněk Martinásek.

[24] Talos - Author of the Official Snort Rule Sets. *Snort - Network Intrusion Detection* [online]. 2019 [cit. 2019-12-01]. Dostupné z: https://www.snort.org/talos

[25] Rules Format. *Suricata User Guide* [online]. [cit. 2019-12-01]. Dostupné z: https://suricata.readthedocs.io/en/latest/rules/intro.html

[26] Meta Keywords. *Suricata User Guide* [online]. [cit. 2019-12-01]. Dostupné z: https://suricata.readthedocs.io/en/latest/rules/meta.html

[27] Payload Keywords. *Suricata User Guide* [online]. [cit. 2019-12-01]. Dostupné z: https://suricata.readthedocs.io/en/latest/rules/payload-keywords.html

[28] LIEDKE, Lindsay. What Is Form Spam and How Can You Stop It? *Kali Forms* [online]. 2019 [cit. 2019-12-01]. Dostupné z: https://kaliforms.com/blog/form-spam/

[29] ČÍPKOVÁ, Lenka. Spam. *WikiKnihovna* [online]. 2012 [cit. 2019-12-01]. Dostupné z: http://wiki.knihovna.cz/index.php/Spam

[30] Spam relay via web forms. *TwoLogs* [online]. 2019 [cit. 2019-12-01]. Dostupné z: https://www.twologs.com/en/resources/spamrelay.asp

[31] KYRNIN, Jennifer. 6 Modern Solutions to Protect Web Forms From Spam. *Lifewire - Tech Untangled* [online]. 2019 [cit. 2019-12-01]. Dostupné z: https://www.lifewire.com/solutions-to-protect-web-forms-from-spam-3467469

[32] Introducing reCAPTCHA v3: the new way to stop bots. *Official Google Webmaster Central Blog* [online]. 2018 [cit. 2019-12-01]. Dostupné z: https://webmasters.googleblog.com/2018/10/introducing-recaptcha-v3-new-way-to.html

[33] Now anyone can fool reCAPTCHA. *Naked Security* [online]. 2017 [cit. 2019-12-01]. Dostupné z: https://nakedsecurity.sophos.com/2017/11/01/now-anyone-can-fool-recaptcha/

[34] Where does the data go? *MDN Web Docs* [online]. 2019 [cit. 2019-12-01]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Sending_and_retrieving_form_data

[35] ZAPLETAL, Lukáš. Protokol HTTP 1.1 pod lupou. *Root.cz - informace nejen ze světa Linuxu* [online]. 2001 [cit. 2019-12-01]. Dostupné z: https://www.root.cz/clanky/protokol-http-1-1-pod-lupou/

[36] Content-Type. *MDN Web Docs* [online]. 2019 [cit. 2019-12-01]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type

[37] Get Docker Engine - Community for Debian. *Docker Documentation* [online]. [cit. 2019-12-01]. Dostupné z: https://docs.docker.com/install/linux/docker-ce/debian/

[38] Setting up IPS/inline for Linux. *Suricata User Guide* [online]. [cit. 2019-12-01]. Dostupné z: https://suricata.readthedocs.io/en/suricata-4.1.5/setting-up-ipsinline-for-linux.html

[39] Installation. *Suricata User Guide* [online]. [cit. 2019-12-01]. Dostupné z: https://suricata.readthedocs.io/en/suricata-4.1.5/install.html

[40] CentOS Installation. *Open Information Security Foundation* [online]. [cit. 2019-12-01]. Dostupné z:
https://redmine.openinfosecfoundation.org/projects/suricata/wiki/CentOS_Installation

[41] podle ústního sdělení Mgr. Josefa Grilla (majitele firmy, Masarykova 1230, 373 41 Hluboká nad Vltavou) dne 20. června 2019

[42] TRUDEAU, Francis. Emerging Threats Threats SID Allocation. Emerging Threats Rule *Documentation Wiki* [online]. 2017 [cit. 2019-12-01]. Dostupné z: https://doc.emergingthreats.net/bin/view/Main/SidAllocation

[43] Online regex tester and debugger. *Regular expression* [online]. [cit. 2019-12-01]. Dostupné z: https://regex101.com/

[44] Text to Hex Converter. *Online Toolz* [online]. [cit. 2019-12-01]. Dostupné z: https://www.online-toolz.com/tools/text-hex-convertor.php

Seznam tabulek

Seznam obrázků

Seznam použitých zkratek

1) **IDS -** Intrusion Detection system
2) **IPS -** Intrusion Prevention Systems
3) **IP –** Internet Protocol
4) **TCP -** Transmission Control Protocol/Internet Protocol
5) **HTTP -** Hypertext Transfer Protocol
6) **WWW -** World Wide Web
7) **AWS -** Amazon web services
8) **GCP -** Google compute platform
9) **X11 -** X Window System
10) **HIDS -** Host-based intrusion detection system
11) **NIDS -** Network intrusion detection system
12) **RFC -** Request for Change
13) **UDP -** User Datagram Protocol
14) **FTP -** File Transfer Protocol
15) **OISF -** Open Information Security Foundation
16) **SIEM -** Security Information and Event Management
17) **HP-UX -** Hewlett Packard UniX
18) **ICMP -** Internet Control Message Protocol
19) **TLS -** Transport Layer Security
20) **SSL -** Secure Sockets Layer
21) **SMB -** Server Message Block
22) **SSH -** Secure Shell
23) **SMTP -** Simple Mail Transfer Protocol
24) **IMAP -** Internet Message Access Protocol
25) **DNS -** Domain Name System
26) **MGS -** Message
27) **SID –** Signature identification
28) **REV -** Revision
29) **GID -** Group identification
30) **PCRE -** Perl Compatible Regular Expressions
31) **HTML -** Hypertext Markup Language

32) **URL -** Uniform Resource Locator
33) **ASP -** Active Server Pages
34) **PHP -** Hypertext Preprocessor
35) **BBC -** Blind Carbon Copy
36) **VPS -** Virtual private server
37) **DVD -** Digital versatile disc
38) **ID -** Identification
39) **CMS -** Content management system
40) **NSM -** Network Security Monitoring

Přílohy

A) Zdrojové kódy

Přílohy na flash disku obsahující zdrojové kódy všech verzí řešení IDS/IPS systému Suricata v prostředí Docker, jednotlivé verze souboru start.sh a soubory, které byly použity při optimalizaci pravidel. Adresářová struktura flask disku je následující:

- 1_prvni_pokusy_suricata_www
    - centos_httpd
    - suricata
- 2_suricata_docker
    - Dockerfile_v1
    - Dockerfile_v2
    - Dockerfile_v3_aktualizace_suricaty
    - Dockerfile_v4
    - Dockerfile_v5
    - Dockerfile_v6_final
- 3_start_sh
    - blacklist_v1_v2
    - iprep
- 4_optimalizace
    - wedos_default_rules
    - wedos_our_rules
    - vysledna_pravidla

Elektronické přílohy na flash disku nejsou veřejně dostupné z bezpečnostních důvodů. Výsledné řešení je majetkem společnosti WEDOS Internet, a.s., která vypracované řešení využívá pro filtraci nelegitimního síťového provozu.

B) Nová bezpečnostní pravidla

Testovací bezpečnostní pravidlo:

1)

```
alert http any any -> any any (msg:"Zneuziti formulare";
flow:established,to_server; content:"POST"; \
content:"Funguje"; nocase; http_client_body; \
sid:1; rev:1;)
```

Bezpečnostní pravidla pro redakční systém Wordpress:

2)

```
reject http any any -> any any (msg:"proxy-
w.pracovnidomena.eu comment"; flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-w.pracovnidomena.eu"; http_host; \
content:"author"; pcre:"/author\=.{20,}\&email/i";
http_client_body; \
http_content_type; content:"application/x-www-form-
urlencoded"; \
sid:2; rev:1;)
```

3)

```
reject http any any -> any any (msg:"proxy-
w.pracovnidomena.eu comment"; flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-w.pracovnidomena.eu "; http_host; \
content:"email"; pcre:"/email\=[a-z0-9.-]+(@|\%40)([a-z0-9.
-]+(\.ru|\.xyz|\.pl|\.nl)|yahoo\.com|qq\.com|ttmail\.pro|
yandex\.com).*\&/i"; http_client_body; \
content:"comment";
pcre:"/comment\=.*(sex|porn|viagra|casino|(http\:\/\/|https
\:\/\/)?[a-z0-9.-]+(\.ru|\.com|\.hk|\.net|\.site|\.org)|
```

```
nudism|pussy|dating-sites|hot
teen|naked|teens).*\&author/i"; http_client_body; \
http_content_type; content:"application/x-www-form-
urlencoded"; \
sid:3; rev:1;)
```

4)

```
reject http any any -> any any (msg:"proxy-
w.pracovnidomena.eu wordpress Contact Form 7";
flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-w.pracovnidomena.eu"; http_host; \
content:"|5f7770636637|"; http_client_body; \
content:"your-email"; pcre:"/your\-email\"\r\n\r\n
[a-z0-9.-]+(@|\%40)([a-z0-9.-]+(\.ru|\.xyz)|
yahoo\.com|qq\.com)\r\n/i";  http_client_body; \
content:"your-message"; pcre:"/your\-message\"\r\n\r\n.*
(sex|porn|viagra|casino|(http\:\/\/|https\:\/\/)?[a-z0-9.-]
+(\.ru|\.com|\.hk|\.ly|\.mp|\.co|\.nl)|nudism|pussy|dating-
sites|hot teen|naked|teens).*\r\n\-\-\-\-/i";
http_client_body; \
sid:4; rev:1;)
```

5)

```
reject http any any -> any any (msg:"proxy-
w.pracovnidomena.eu wordpress Contact Form 7 jmeno-odkaz";
flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-w.pracovnidomena.eu"; http_host; \
content:"|5f7770636637|";  http_client_body; \
content:"your-name"; pcre:"/your\-name\"\r\n\r\n.*
(http\:\/\/|https\:\/\/)?[a-z0-9.-]+((\.ru|\.com|\.hk|
```

```
\.ly|\.mp|\.tech|\.nl|\.us)|bitly\.com).*\r\n\-\-\-\-/i";
http_client_body; \
sid:5; rev:1;)
```

Bezpečnostní pravidla pro redakční systém Prestashop:

6)

```
reject http any any -> any any (msg:"proxy-
p.pracovnidomena.eu registration";
flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-p.pracovnidomena.eu"; http_host; \
content:"|637573746f6d65725f66697273746e616d653d|";
pcre:"/customer\_firstname\=.*(http\:\/\/|https\:\/\/)?[a-
z0-9.-]+(\.ru|\.com|\.net|\.info|\.pl|\.org|\.cf|\.ga|
\.me|\.gg).*\&customer/i"; http_client_body; \
http_content_type; content:"application/x-www-form-
urlencoded"; \
sid:6; rev:1;)
```

7)

```
reject http any any -> any any (msg:"proxy-
p.pracovnidomena.eu registration";
flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-p.pracovnidomena.eu"; http_host; \
content:"|637573746f6d65725f6c6173746e616d653d|";
pcre:"/customer\_lastname\=.*(http\:\/\/|https\:\/\/)?[a-
z0-9.-]+(\.ru|\.com|\.net|\.info|\.pl|\.org|\.cf|\.ga|
\.me|\.gg).*\&email/i"; http_client_body; \
http_content_type; content:"application/x-www-form-
urlencoded"; \
sid:7; rev:1;)
```

8)

```
reject http any any -> any any (msg:"proxy-
p.pracovnidomena.eu spam z kontaktniho formulare";
flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-p.pracovnidomena.eu"; http_host; \
content:"from"; pcre:"/from\"\r\n\r\n[a-z0-9.-]+(@|\%40)
([a-z0-9.-]+(\.ru|\.xyz)|yahoo\.com|qq\.com|[0-9]+\.com).
*\r\n\-\-\-\-/"; http_client_body; \
content:"message"; pcre:"/message\"\r\n\r\n.*(sex|porn|
viagra|casino|(http\:\/\/|https\:\/\/)?[a-z0-9.-]+
(\.ru|\.com|\.hk)|nudism|pussy|dating-sites|hot
teen|naked|teens).*\r\n\-\-\-\-/i"; http_client_body; \
sid:8; rev:1;)
```

9)

```
reject http any any -> any any (msg:"proxy-
p.pracovnidomena.eu spam z kontaktniho formulare";
flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-p.pracovnidomena.eu"; http_host; \
content:"Mozilla/5.0"; http_user_agent; \
content:"from"; pcre:"/from\"\r\n\r\n[a-z0-9.-]+(@|\%40)
([a-z0-9.-]+(\.ru|\.xyz)|yahoo\.com|qq\.com|[0-9]+\.com).
*\r\n\-\-\-\-/"; http_client_body; \
http_accept_lang; content:"zh-CN"; nocase; \
sid:9; rev:1;)
```

Bezpečnostní pravidla pro redakční systém Joomla:

10)

```
reject http any any -> any any (msg:"proxy-
j.pracovnidomena.eu registration";
flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-j.pracovnidomena.eu"; http_host; \
content:"|6a666f726d5b6e616d655d|";
pcre:"/(jform\[name\]|jform\[username\])\"\r\n\r\n.*(http\:
\/\/|https\:\/\/)?[a-z0-9.-]+(\.ru|\.com|\.pl|\.tw|\.org|
\.us|\.fr|\.net|\.tk|\.la|\.gd|\.do|\.ht|\.uk|\.link|\.jp|\
.cf|\.de|\.me).*\r\n/i"; http_client_body; \
sid:10; rev:1;)
```

11)

```
reject http any any -> any any (msg:"proxy-
j.pracovnidomena.eu registration";
flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-j.pracovnidomena.eu"; http_host; \
content:"name";
pcre:"/(name|username)\=.*(http\:\/\/|https\:\/\/)?[a-z0-9
.-]+(\.ru|\.com|\.pl|\.tw|\.org|\.us|\.fr|\.net|\.tk|\.la|
\.gd|\.do|\.ht|\.uk|\.link|\.jp|\.cf|\.de|\.me).*\&/i";
http_client_body; \
sid:11; rev:1;)
```

12)

```
reject http any any -> any any (msg:"proxy-
j.pracovnidomena.eu "; flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-j.pracovnidomena.eu"; http_host; \
content:"name"; pcre:"/name\=.{12,}\&username/i";
http_client_body; \
content:"email";
pcre:"/email\=.+(@|%40)(.+(\.ru|\.xyz|\.pl|\.au)|yahoo\.com
|qq\.com|course\-fitness\.com|live\.com|yandex\.com|
hotmails\.com).*\&/i"; http_client_body; \
sid:12; rev:6;)
```

13)

```
reject http any any -> any any (msg:"proxy-
j.pracovnidomena.eu contact us";
flow:established,to_server; \
content:"POST"; http_method; \
content:"proxy-j.pracovnidomena.eu"; http_host; \
content:"contact_email"; pcre:"/contact\_email\%5D\=[a-z0-
9.-]+(@|\%40)([a-z0-9.-]+(\.ru|\.xyz)|yahoo\.com|qq\.com|
course\-fitness\.com|live\.com)\&/i"; http_client_body; \
content:"contact_message"; pcre:"/contact\_message\%5D\=.*
(sex|porn|(http\:\/\/|https\:\/\/)?[a-z0-9.-]+(\.ru|\.com|
\.pl|\.vip)).*\&/i"; http_client_body; \
sid:13; rev:1;)
```