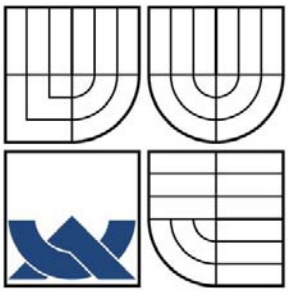
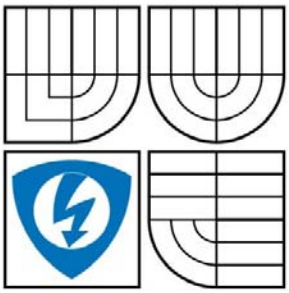


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

PROXY FIREWALL

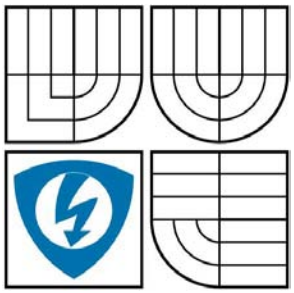
PROXY FIREWALL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. ZDENĚK KUGLER
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADIM PUST
SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Zdeněk Kugler	***ID:***	85327
Ročník:	2	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Proxy firewall

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce je popsat princip činnosti proxy firewallu a porovnat jej s ostatními typy firewallů. Dále navrhnout řešení proxy firewallu postavené na operačním systému Linux. Součástí proxy firewallu bude antivirová ochrana a content filtering.

DOPORUČENÁ LITERATURA:

[1] STREBE, Matthew, PERKINS, Charles. Firewally a proxy-servery. Brno : Computer Press, 2003. 472 s. ISBN 80-722-6983-6.
[2] TOXEN, Bob. Bezpečnost v Linuxu. Brno : Computer Press, 2003. 876 s. ISBN 80-7226-716-7.

Termín zadání:	9.2.2009	***Termín odevzdání:***	26.5.2009

Vedoucí práce: Ing. Radim Pust

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ANOTACE

Diplomová práce souhrnně pojednává o tematice proxy serverů a firewallů, s přihlédnutím k dalším souvisejícím technologiím a síťovým technikám. Systematicky popisuje obecnou problematiku firewallů se zaměřením na proxy firewally a jejich bezpečnost. Dalšími systémy, které jsou v rámci práce zmíněny, a které určitým způsobem souvisí s bezpečností sítí, serverů nebo klientských stanic, popřípadě s omezováním zdrojů na Internetu, jsou systémy k detekci průniků (IDS), antivirové systémy a filtery pro regulaci internetového obsahu. IDS systémy lze typicky doplnit různými přídavnými aplikacemi a nástroji, které obohacují a zvyšují jejich potenciál včetně grafických nástaveb. I na tuto druhotnou část je v práci pamatováno. Některé provozované systémy spolu navíc dokáží vzájemně spolupracovat, čehož se s úspěchem využívá (např. součinnost FW & IDS).

První velká kapitola má za cíl vystínit technologie firewallů, přehled jejich typů, základní funkcionalitu a závěrečné srovnání. Okrajově se zmiňuje o nasazení firewallech v praxi. Druhá kapitola teoreticky objasňuje problematiku překladu síťových adres (NAT) ve spojitosti s funkčností, bezpečností a omezením mechanismu NAT. Třetí kapitola zahrnuje ucelený výklad o proxy serverech, především vysvětlení principu z funkčního hlediska a specifikace oblasti nasazení, kapitola je zakončena přehledným výčtem a popisem druhů proxy serverů. Zbývající kapitola, pojmenovaná Linuxový proxy firewall, je věnována stěžejní části práce. Obecně pojednává o platformě Linux, distribuci Debian GNU/Linux, základech bezpečnostní politiky, konfiguraci sítě, bezpečnosti serveru v síti, firewallech v Linuxu (framework Netfilter, nástroj Iptables) a proxy serveru Squid. Následující části v podobě dalších podkapitol strukturou navazují na předchozí a rámcově popisují oblasti teorie systémů k detekci průniků, antivirové kontroly a filtrování obsahu na základě různých metod. Vše obdobně jako v předešlém.

V praktické části práce bylo navrženo řešení proxy firewallu postavené na operačním systému Linux, konkrétně byla zvolena distribuce Debian GNU/Linux, která je svými charakteristickými vlastnostmi velice vhodná pro serverové nasazení. Na tomto prostředí staví další bezpečnostní software, který je součástí proxy firewallu v podobě antivirové ochrany, content filteringu a systému detekce průniků. Prioritou je tedy co nejkomplexnější zabezpečení počítačové sítě, z čehož plyne pokrytí detekčními schopnostmi co možná nejširšího spektra působnosti na poli síťové bezpečnosti.

Cílem této diplomové práce je nejen popis principu činnosti proxy firewallů a vzájemné porovnání s ostatními typy, včetně uplatnění dalších, již zmíněných systémů, ale i vlastní navržené řešení zdarma zvyšující zabezpečení v síti se snahou přiblížení se k čistě komerčním produktům dostupným na trhu.

Klíčová slova: proxy server, firewall, antivirus, IDS, content filtering, blacklisting, linux, debian, NAT, síťový útok

ABSTRACT

This diploma thesis deals with the topic of proxy servers and firewalls and considers other associated technologies and network techniques. It systematically describes the general issues of firewalls, with a special focus on proxy firewalls and their safety. Additional systems mentioned in this document are intrusion detection systems (IDS), antivirus systems and content control filters – as these are also connected with safety of networks, servers and workstations or with limiting various Internet sources. IDS systems can be typically supplemented with various additional applications or tools that enrich them and increase their potential – including graphic additions. This part is remembered too. Some systems can communicate with each other, which is successfully utilised (FW & IDS co-operation, for example).

The purpose of the first large chapter is to present firewall technologies, to list firewall types, their basic functionality and to present the final comparison. It marginally mentions firewall applications in practice. Chapter two explains the theory of network address translation (NAT), deals with its functionality, safety and with limiting the NAT mechanism. Chapter three brings a comprehensive presentation of proxy servers. It explains their principle from the point of view of functionality and the specification of application areas. The chapter is complete with a clear list of proxy server types and their descriptions. The last chapter named Linux Proxy Firewall is the key part of the work. It deals generally with the Linux platform, the Debian GNU/Linux distribution, principles of safety policy, network configuration, network server safety, Linux firewalls (Netfilter framework, Iptables tool) and with the Squid proxy server. The following subchapters respect the previous structure: they describe the theories of intrusion detection systems, antivirus checks and content filtering based on different methods. All this is presented similarly to the previous chapters.

A proxy firewall solution built on the Linux operating system has been proposed in the practical part. The Debian GNU/Linux distribution has been chosen, being very suitable for server use due to its features. This environment is also used for additional safety software contained in the proxy firewall: antivirus protection, content filtering and an intrusion detection system. The priority is the most comprehensive computer network security, which requires detection abilities with the broadest possible coverage in the area of network safety.

The purpose of this diploma thesis is not only to describe the principle of operation of proxy servers and to compare them with other types and other systems, but it also brings my own proposed free solution, which increases network safety and has the ambition of comparing it with clearly commercial products available on the market.

Keywords: proxy server, firewall, antivirus, IDS, content filtering, blacklisting, linux, debian, NAT, network attack

Bibliografická citace práce:

KUGLER, Z. *Proxy firewall.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 127 s. Vedoucí diplomové práce Ing. Radim Pust.

Prohlášení o původnosti práce

Prohlašuji, že svou diplomovou práci na téma „Proxy firewall“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení §152 trestního zákona č. 140/1961 Sb.

V Brně dne ……. ……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Radimu Pustovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování práce.

V Brně dne ……. ……

podpis autora

Obsah

Seznam obrázků a tabulek

1 Úvod

Počítačová bezpečnost patří k ožehavým tématům dneška. V době stále se rozšiřujícího Internetu číhá nebezpečí na každém rohu. Profesionálové o něm dobře vědí a málokdo z nich bere tento problém na lehkou váhu, avšak spousta běžných uživatelů není dostatečně informována, a ti, kteří jsou, si myslí, že se jich nebezpečí napadení cizím útočníkem netýká. To je jeden z největších omylů v této oblasti. Všichni máme na pevném disku informace, které bychom neradi zveřejnili nebo ukázali někomu cizímu. Na jedné straně se oháníme zákony na ochranu osobnosti, na straně druhé necháváme často až příliš nápadně „otevřené dveře“ do našeho systému. Problém počítačové kriminality je čím dál hrozivější. Počítače dnes využívá kdekdo, staly se součástí našich životů a svou práci si bez nich nedokážeme představit.

Největší hrozbou je jistě připojení k počítačové síti. A platí přímá úměra mezi její velikostí a nebezpečím, které se v ní ukrývá. Z toho plyne, že největší síť znamená také největší riziko. Touto je bezesporu celosvětová síť Internet. Díky své rozlehlosti, anonymitě jednotlivých uživatelů, kterých jsou miliony a nedůvěryhodné povaze znamená pro naše data a soukromé informace obrovské riziko, jemuž je potřeba rázně čelit. Vznikly tedy síťové programy a zařízení nazývané firewally, jejichž úlohou je oddělit sítě s různou úrovní důvěryhodnosti a kontrolovat datový tok mezi těmito sítěmi. Kontrola dat probíhá na základě pravidel, která určují podmínky a akce. Podmínky se stanovují pro údaje, které je možné získat z datového toku. Úloha firewallu spočívá ve vyhodnocení podmínky a provedení akce, pokud je podmínka splněna. Základní akcí je povolit či blokovat datový paket. Existují i takové akce, které slouží pouze na zaznamenání nebo změnu hlaviček paketu. Pro snadnější integraci do sítě dnešní firewally podporují i směrování.

Mezi základní způsoby ochrany sítě firewallem patří filtrování paketů na základě zdrojové či cílové IP adresy nebo podle TCP/UDP portů, dále stavová inspekce, která umí přiřadit pakety k příslušnému spojení (díky této vlastnosti firewall rozpozná, že se jedná o pakety vracející se do sítě v rámci spojení, které bylo navázáno zevnitř) a proxy firewall/aplikační brána pracující na aplikační úrovni a vyhodnocující průběh komunikace na této nejvyšší vrstvě. Proxy jednoduše řečeno ukončují spojení směrem od klienta a směrem k serveru navazují spojení nové. Míra bezpečnosti je proto pro konkrétní aplikace velmi vysoká. Spolu s firewally, chránícími síť jako celek, jsou často zmiňovány i tzv. proxy servery, umožňující oddělit klienty (hostitelské stanice) od přímého styku s počítači (resp. servery) „na druhé straně“ informačního řetězce a poskytnout tak anonymitu a bezpečnost uživateli.

Tato práce se bude z velké části zaobírat především tematikou principu činnosti proxy firewallů, ovšem budou zde zmíněny i ostatní typy firewallů, spolu se vzájemným porovnáním jejich vlastností, výhod a nevýhod. Další dvě velké kapitoly budou tvořit překlady síťových adres (NAT) a proxy servery, kterým bude poskytnuto více prostoru. Praktickou část práce bude představovat návrh řešení proxy firewallu postavené na platformě GNU/Linux. Této části spolu s popisem principu činnosti proxy firewallů bude věnována největší pozornost. V samotné příloze na závěr přiložím komentované konfigurační soubory a skripty, které budou na linuxovém serveru otestovány a funkčně využity.

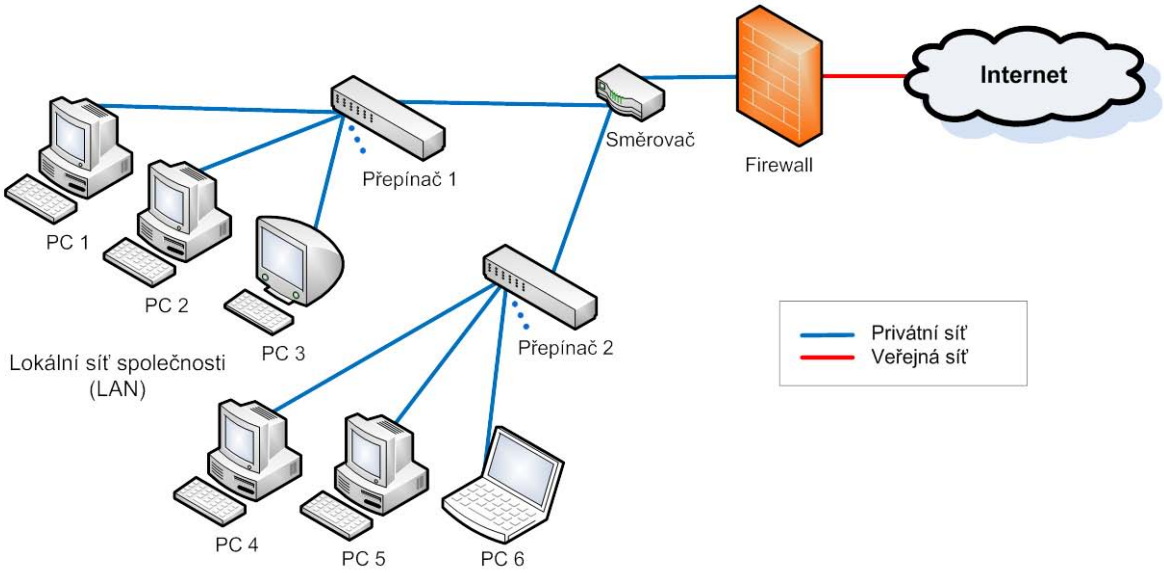
Nezanedbatelná část internetových serverů dnes běží na strojích, které jsou spravovány některým z operačních systémů patřících do rodiny Unix. Proto veškeré, později jmenované programy či softwarové aplikace, budou přizpůsobeny operačním systémům založeným na svobodném jádru Linux a spadající pod licenci GNU General Public License.

2 Firewally

2.1 Technologie firewallů

Firewall (česky by se dalo říct „bezpečnostní brána“ či „ochranný počítač“, pojem se však nepřekládá) je jednoduše řečeno brána oddělující sítě. Může být realizován jako program spuštěný na počítači nebo jako samostatné zařízení. Primárním účelem firewallu je zabránit nechtěné komunikaci z jedné sítě (zóny) do jiné sítě (zóny) a jeden firewall může oddělovat i více než dvě různé sítě (zóny).

Rozlišení, která komunikace bude povolena či zakázána, je řízeno bezpečnostní politikou. Tato politika je vložena do konfigurace firewallu a pro každý požadavek na průchod firewallem jsou aplikována pravidla bezpečnostní politiky, podle nichž firewall rozhodne, zda komunikaci povolit či zakázat. Tato činnost je často nazývána filtrování komunikace a to je příčina, proč jsou firewally někdy označovány jako síťové filtry [5].



Obr. 1: Postavení firewallu v síťovém komunikačním řetězci

2.2 Přehled typů firewallů

Rozdělení firewallů je možné podle úrovně, na které firewall filtruje komunikaci. Buď může fungovat na vrstvě síťové (nejrychlejší a zpravidla také nejméně nákladná varianta, ale filtruje velmi povrchně - rozhoduje se pouze dle informací dostupných na této vrstvě), tyto firewally jsou zpravidla označovány jako **paketové**. Pro jejich rychlost je vhodné je umístit na místa s hustým provozem, např. vstupní brána do sítě apod.

Firewall může být i **stavový**, když dokáže rozlišit již navázaná spojení a k nim příbuznou komunikaci (například pro FTP protokol). V tomto případě již ale musí pracovat na vrstvě transportní.

Na aplikační vrstvě pracující firewall je často označován jako **proxy firewall**. Výhodou těchto firewallů je, že zpravidla plně „rozumí“ fungování aplikací a protokolů a jsou schopny detekovat např. případy jejich nestandardního chování apod., tedy filtrují velmi důkladně. Je tedy nejvhodnější je umístit až na hostitelské stanice či v jejich bezprostředních blízkostech (předřazený jednoúčelový server - firewall).

Poslední dvě možnosti rozdělení tvoří **analyzátory paketů** a **NAT (Network Address Translation)** [18].

2.3 SW/HW firewally

2.3.1 Hardwarové firewally

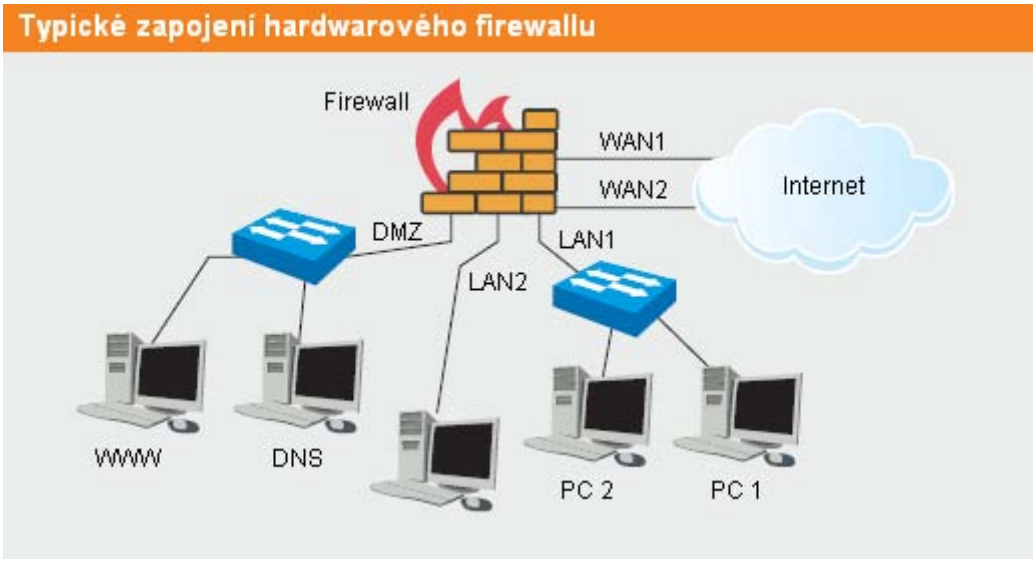
HW firewall je aktivní síťové zařízení, které slouží zejména ke kontrole a filtrování datového toku. Obvykle má alespoň jedno vstupní a jedno výstupní síťové rozhraní. Firewall kontroluje procházející data podle definovaných pravidel. Pravidla se aplikují buď na jednotlivé pakety (tzv. paketový filtr) na základě směru toku (vstupní/výstupní), nebo IP adresy a portu (zdroje/cíle). Pokud se zaznamenává stav pro rychlejší zpracování paketů patřících do jednoho toku dat, jedná se o stavový filtr. V případě zpracování aplikačních protokolů jde o tzv. aplikační brány.

D-Link DFL-800 [31]

Cisco PIX 501 firewall [25]

Obr. 2: Ukázka hardwarových firewallů

Hardwarové firewally jsou specializovaná síťová zařízení, která na první pohled připomínají menší přepínače. Umějí však mnohem více. Kromě přepínání a filtrace paketů provádějí např. analýzu dat na aplikační úrovni (obsah WWW stránek, e-mailů, atp.). Kombinují funkce stavového i paketového filtru a přidávají kontrolu obsahu dat pomocí vyhledávání řetězců. Oproti softwarovému řešení jsou rychlejší, poměrně jednodušeji se konfigurují a spravují (velmi relativní). Poskytují komplexní ochranu a monitorování bezpečnosti sítě. Konfigurovat je můžeme pomocí grafického rozhraní (přes WWW) nebo textového CLI (Command Line Interface) [35].



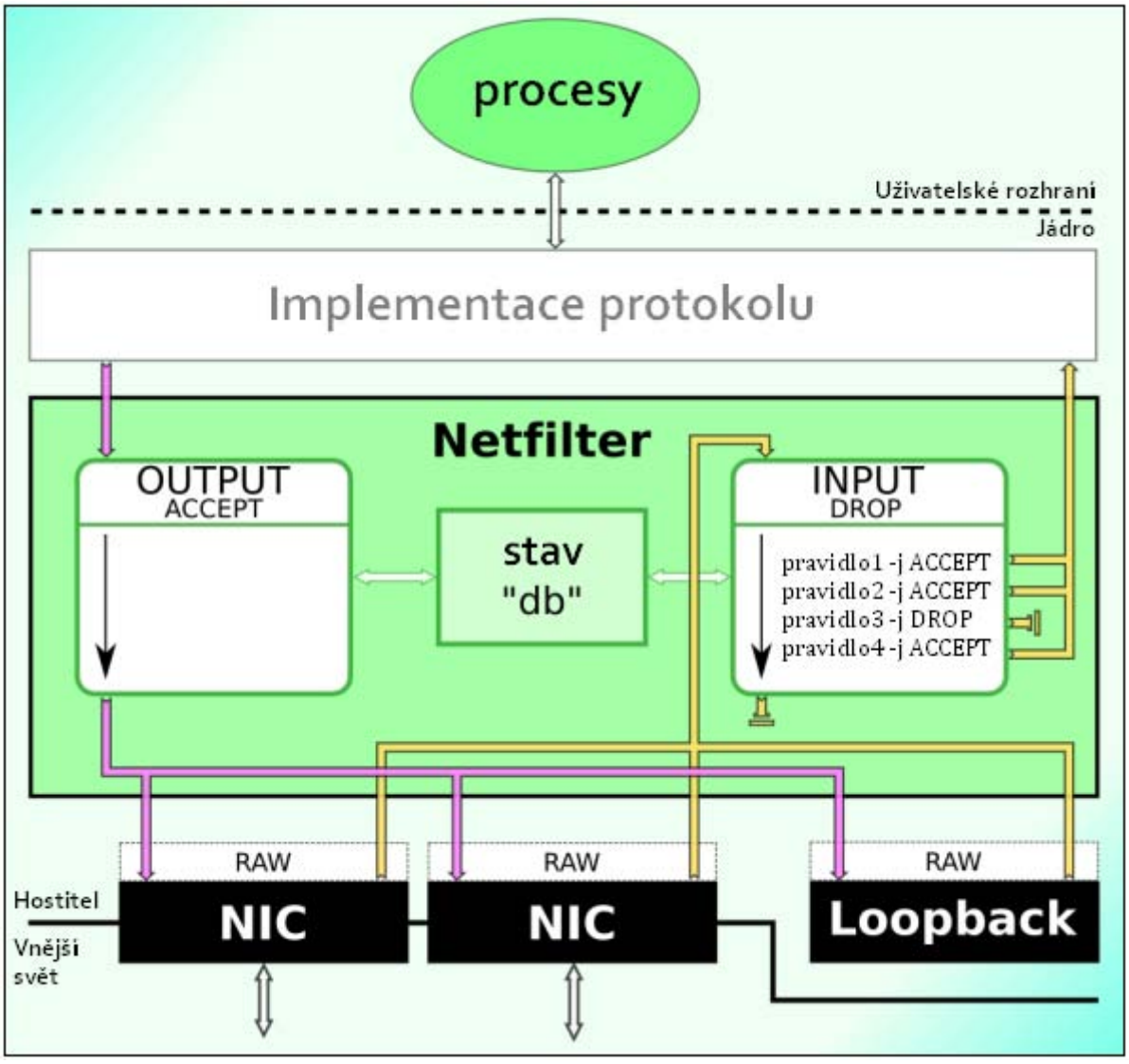
Obr. 3: Obvyklé zapojení zařízení – dvě interní LAN sítě s omezeným přístupem do Internetu (WWW, ping, ssh), demilitarizovaná zóna (DMZ) s kontrolovaným přístupem z Internetu a blokováním přístupu do vnitřní sítě LAN a Internet (WAN síť) [35]

Z technického pohledu je hardwarový firewall vlastně specializovaný počítač, který obsahuje procesor, paměť a síťová rozhraní.

Součástí mohou být i hardwarové akcelerátory postavené na technologiích ASIC nebo FPGA pro rychlý výpočet šifrovacích algoritmů u VPN či vyhledávání řetězců u systémů IDS (Intrusion Detection System). Např. u firewallu ZyWall 35 umožňuje přídavná PCMCIA karta prohledávání obsahu dat a antivirovou kontrolu. Operačním systémem těchto zařízení může být (kromě proprietárních řešení) například Linux.

2.3.2 Softwarové firewally

Dobře známé jsou softwarové firewally, které mohou běžet na osobním počítači, který může být pro tento účel vyhrazen nebo také ne. V prvním jmenovaném případě bychom se bavili o serverech s více síťovými kartami, které kromě filtrování provádějí i překlad adres NAT. Obvykle je na serveru s firewallem také služba DHCP pro dynamické přidělování IP adres, případně DNS služba pro překlad doménových jmen na IP adresy. Druhý případ značil (koncové) počítače uživatelů. SW řešení je levné a oblíbené – můžeme využít volně dostupný software a přizpůsobit si ho podle svých požadavků. Nastavení však vyžaduje zkušeného administrátora, a také analýza přenosů nebývá příliš jednoduchá. Pod OS Linux je k dispozici již vestavěný firewall v jádře operačního systému s názvem Netfilter [35].



Obr. 4: Ukázka softwarového firewallu Netfilter (nástroj Iptables)

2.4 Kde vytvořit firewall

Podle chráněného subjektu můžeme firewally rozdělit na dva druhy - firewally na ochranu (jednoho) počítače (tzv. „host-based“ firewally) a firewally na ochranu celé sítě (tzv. „network-based“ firewally).

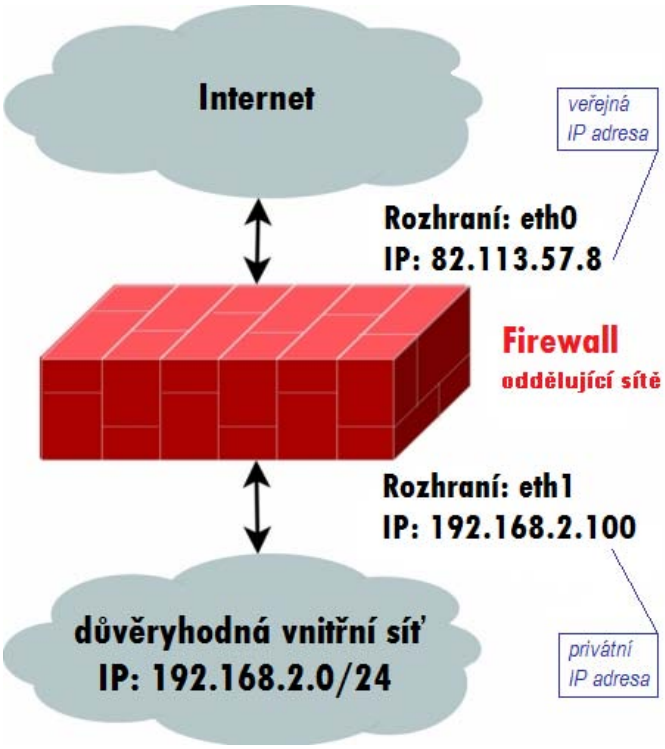
2.4.1 Firewally na ochranu (jednoho) počítače

Firewally určené k ochraně jediného počítače jsou relativně „jednoduché“. Jejich úlohou je chránit jeden jediný počítač (typicky s jednou síťovou kartou), nepotřebují se zaobírat přesměrováním paketů ani překladem adres. Jsou nakonfigurované na počítači, který mají chránit, a tak mohou být pravidla těchto firewallů velmi specifická a umožňovat jen ten síťový provoz, který je potřebný pro zabezpečení služby (služeb) či aplikací na daném počítači.

Takovéto firewally se často používají na doplňkovou ochranu serveru, který je umístěný za nějakým již existujícím („front-line“) firewallem, například (ovšem ne výlučně) ve webhostingovém centru a dále k ochraně jednotlivých osobních počítačů (pracovních stanic) připojených k síti. Proto jsou také často nazývány osobními firewally [19].

2.4.2 Firewally na ochranu sítě

Ve většině případů je úlohou firewallu oddělit dvě či vícero sítí s různými přístupovými právy nebo obsahem. Dále se budu držet příkladu, ve kterém firewall odděluje Internet (vnější síť) a Intranet (vnitřní síť). Z předešlého logicky vyplívá, že přes firewall musí procházet všechny pakety pohybující se mezi sítěmi. V opačném případě je firewall není schopen kontrolovat. Firewall funguje pro sítě jako bezpečnostní brána (ve skutečnosti vykonává i směrování paketů, takže tato analogie není od věci).



Obr. 5: Firewall, který má chránit síť, musí být umístěný mezi sítěmi, mezi kterými má kontrolovat a analyzovat síťový provoz

2.4.3 Politika firewallu

V kapitole o bezpečnosti serveru v síti budu hovořit o tom, jak důležité je stanovit si dobrou bezpečnostní politiku. V případě firewallu je důležité zejména rozhodnutí, jak firewall naloží s pakety, jejichž osud explicitně neurčíme [19].

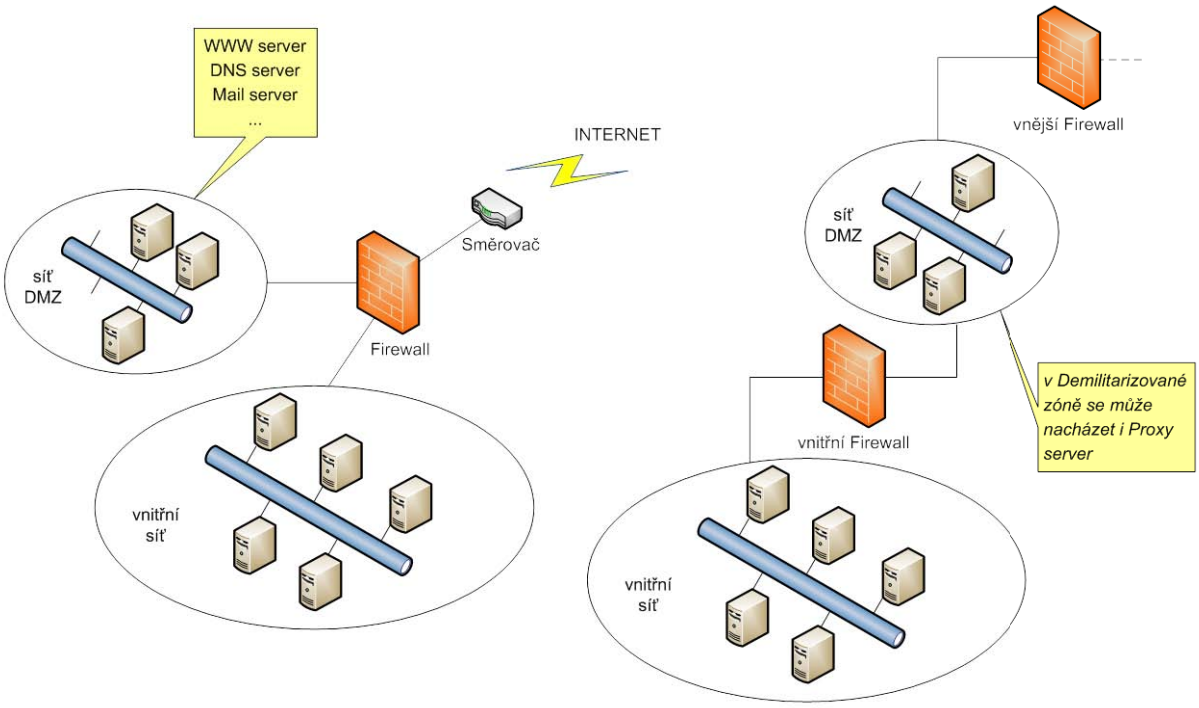
Ze zkušeností odborníků na počítačovou bezpečnost se obecně uplatňuje postup, při kterém „co není výslovně povoleno, je zakázáno“. Tato zvolená strategie se z hlediska zabezpečení jeví jako ucelenější.

2.4.4 Firewall & server versus samostatný firewall

Existují v podstatě dvě řešení, jak implementovat firewall do existující sítě. První řešení předpokládá, že firewall bude na serveru, který bude poskytovat určité služby pro některou ze sítí (nebo obě), které vzájemně propojuje. Bude tedy bránou, firewallem a serverem současně. Toto řešení je méně nákladné, protože nepotřebujeme samostatný počítač na firewall, ale (relativně) není příliš bezpečné. V případě, že na firewallu běží nějaké služby, je možno je za určitých okolností zneužít (ať už jsou chráněné jakýmkoliv způsobem), získat práva administrátora a upravit/smazat pravidla firewallu.

Naproti tomu, když se firewall umístí na úplně samostatný počítač, na kterém neběží žádné služby, je řešení mnohem bezpečnější. V takovém případě je vhodné, jestliže se na firewall přihlašuje administrátor pouze z konzole, aby nemusely na počítači běžet ani služby jako SSH. Všechny služby, které mají být dostupné z Internetu, běží fyzicky na jiném počítači (serveru) a firewall zabezpečí správné směrování paketů pomocí routovací tabulky či překladu adres NAT [19].

V případě, že síť bude poskytovat služby do Internetu, nejbezpečnějším řešením se stává takové, při kterém firewall propojuje ne dvě, ale tři sítě (Internet, Intranet a DMZ – demilitarizovanou zónu). DMZ je síť, která se používá pro servery s veřejnými službami přístupnými z Internetu. Firewall umožňuje chránit Intranet (do něho proniknou maximálně odpovědi na spojení již iniciované z Intranetu), veřejně dostupné služby jsou v odděleném segmentu sítě a firewall nad všemi drží „stráž“.



Obr. 6: Ukázky demilitarizované zóny (DMZ)

2.5 Paketové filtry

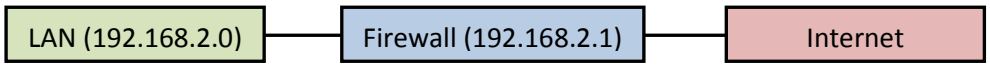
Paketový filtr umožňuje sledovat síťový provoz na třetí vrstvě ISO/OSI modelu (tj. logické IP adresy). Je mnohem rychlejší, ale jeho správa je komplikovanější a nemá tolik možností jako aplikační proxy, protože se například nedostane ke jménu uživatele, kterému patří zachycené pakety.

Nevýhodou paketového filtru je i to, že někdy musí na serveru fungovat služby, které při komunikaci využívají náhodně vybrané porty (např. při přenosu souborů pomocí služby FTP). Paketový filter „nevidí“ (nepozná) souvislosti mezi pakety a každý analyzuje samostatně. Je tedy třeba povolit buď celý rozsah portů (příliš benevolentní) nebo jej zakázat a takovéto služby nepoužívat (někdy nerealizovatelné) [19].

Postup paketového filtru:

1. Zjištění zdrojové i cílové IP adresy a portů.
2. Průchod tabulkou po jednotlivých řádcích odshora dolů, dokud není nalezen řádek s pravidlem, který odpovídá danému paketu. Většinou definována i poslední výchozí „policy“.
3. Provede se akce uvedená v příslušném řádku pravidla. Zpravidla se paket propustí („allow“) nebo zahodí („deny“; „drop“), popřípadě i zaloguje, čili se zaznamená přísl. akce do souboru.

Příklad filtračních pravidel pro lokální síť s adresou 192.168.2.0:



pořadí pravidla	zdrojová IP adresa	zdrojový port	cílová IP adresa	cílový port	akce	popis
1	any	any	192.168.2.0	>1023	allow	příjem reakcí na výzvy zevnitř
2	192.168.2.1	any	any	any	deny	nelze zneužít adresu firewallu
3	any	any	192.168.2.1	any	deny	blokování spojení s firewallem
4	192.168.2.0	any	any	any	allow	povolení komunikace zevnitř
5	any	any	192.168.2.2	80	allow	přístup na www server zvenku
6	any	any	any	any	deny	vše ostatní nepustit (zakázat)

První pravidlo zajišťuje navázání TCP spojení iniciovaných z vlastní sítě. Druhé pravidlo zabraňuje útočníkovi vystupovat v síti jako firewall. Třetí pravidlo zabraňuje vnějšímu útočníkovi komunikovat s firewallem. Čtvrté pravidlo umožňuje všem prvkům sítě komunikovat s kýmkoliv z vnější sítě a použít přitom jakýkoliv protokol. Páté pravidlo umožňuje projít všem paketům z vnější sítě, pokud nesou data protokolu HTTP. Poslední šesté pravidlo je velmi důležité - vše co nespadá pod výše uvedená pravidla nepustit do (ze) sítě. Jak je názorně vidět z tabulky, paketový filtr je založen na pravidlech stanovených pro dvojici zdrojová IP adresa a cílová IP adresa. Tyto údaje jsou často ještě upřesněny pro konkrétní TCP nebo UDP porty.

2.6 Stavové firewally

Speciálním případem paketového filtru je stavový paketový filtr. Ten si dokáže uvést v souvislost procházející pakety a tak si uchovat stav spojení. Jedno navázané spojení a všechny pakety, které k němu patří, se nazývá relace (session). Stavový firewall se oproti paketovému filtru liší také tím, že není obecně univerzální, ale je využitelný pouze pro TCP/IP protokoly.

Stav spojení je možné uplatnit v pravidlech, a tak například automaticky povolit odpovědi na všechny odeslané pakety, povolit spojení, která souvisí s daným, již navázaným spojením po dobu jeho trvání, atd. Toho se s výhodou prakticky využívá např. u řešení již zmíněného problému s FTP.

Tento typ firewallu rozezná paket, který otevírá nové spojení, od paketů, které tuto komunikaci realizují, a díky tomu může precizněji filtrovat datové toky. Pokusy o spoofing, podvržení paketů, které se tváří, jako by se "vracely do sítě" v rámci fiktivního spojení, jsou blokovány.

Firewally se stavovou inspekcí využívají k filtraci stejně jako paketové filtry IP adresy i čísla portů. Navíc však využívají i informaci o stavu spojení, takže umožňují hlídat souvislosti mezi pakety.

Příklad: Klient vnitřní sítě s adresou 192.168.2.10 si chce prohlédnout webové stránky umístěné na internetové adrese 77.75.72.3 (*http://www.seznam.cz*). Klient tedy vyšle na danou adresu paket s TCP segmentem typu SYN (Synchronization) pro navázání spojení:

- **zdroj:** IP = 192.168.2.10, port = 1030 (port alokovaný prohlížečem),
- **cíl:** IP = 77.75.72.3, port = 80 (protokol HTTP).

Firewall si tento paket zapíše do své stavové tabulky a propustí jej do vnější sítě (ovšem pouze tehdy, splnil-li příslušné pravidlo pro přístup do vnější sítě - Internetu). Očekává, že potvrzující paket (od protistrany) by měl být TCP segmentem typu SYN+ACK:

- **zdroj:** IP = 77.75.72.3, port = 80,
- **cíl:** IP = 192.168.2.10, port = 1030.

Paket je propuštěn do vnitřní sítě, pouze pokud splňuje očekávané parametry. Do vnitřní sítě se tak zvenčí dostanou pouze pakety z komunikace iniciované vnitřním prvkem sítě => zvýšení bezpečnosti.

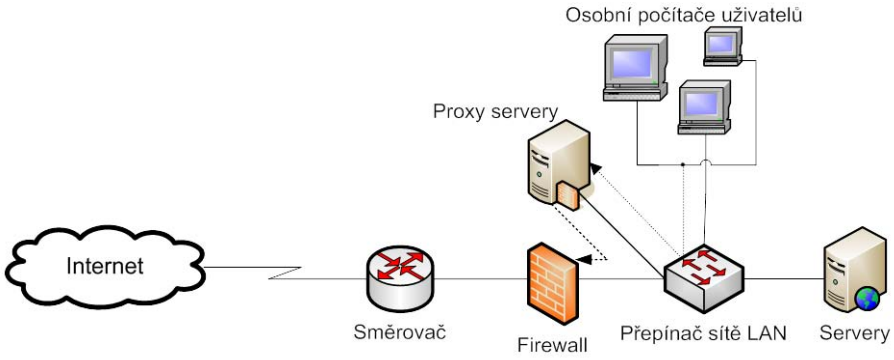
Softwarový firewall je implementovaný přímo v jádře Linuxu. Ve verzích jádra 2.4.X a 2.6.X je to bezpečnostní nástroj s názvem "`iptables`", který kromě správy filtrovacích pravidel umožňuje i správu NATu. Samotný firewall (framework pro manipulaci s pakety) se nazývá "`netfilter`".

Protože je Iptables z technologického hlediska nejdokonalejší (nejpokročilejší), a jelikož je doporučeno používat aktuální stabilní verze jádra Linuxu, budu se v dalších částech textu zaobírat pouze tímto mocným firewallovým nástrojem, který umožňuje linuxovému systému plně pracovat se síťovou komunikací [19].

2.7 Proxy firewally

Firewally se stavovou inspekcí paketů jsou v podstatě rozšířenou verzí běžného paketového filtru. Zde popisovaná zařízení jdou ještě dále a provádějí analýzu paketů na aplikační vrstvě (ochrana na úrovni aplikací). Tuto úroveň ochrany implementuje několik různých technologií, které se označují různými názvy; každá z nich pracuje trochu jinak, ale jejich cíl je stejný - posílit bezpečnost sítě.

Firewally na aplikační úrovni zajišťují nejbezpečnější typ datových spojení, protože dokáží v komunikačním procesu zkoumat úplně všechny vrstvy modelu TCP/IP. Pro zajištění této úrovně ochrany musí uvedené firewally neboli proxy (proxy servery) fakticky vstoupit do probíhající komunikace (do role „prostředníka“, což je v podstatě český význam slova „proxy“) a detailně kontrolovat každé spojení. Jestliže proxy označí dané spojení za povolené, otevře směrem k serveru druhé spojení od sebe sama, jménem původního hostitele, jak vidíme na obrázku 7 [4].



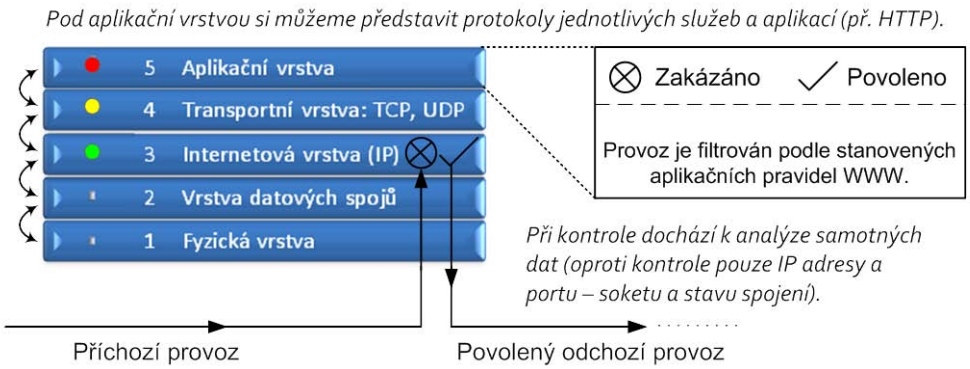
Obr. 7: Činnost proxy serverů a jejich pozice

Při této inspekci je nutné odříznout datovou část každého procházejícího paketu, zkontrolovat jej, znovu sestavit a odeslat po druhém spojení. Uvedené funkce lze zajistit v různých typech firewallů[4]:

- **Standardní proxy firewally.** Běžný proxy firewall neprovádí směrování paketů a pouze je přeposílá; pracuje v aplikační vrstvě modelu TCP/IP. Z funkčního hlediska přijímá nad jedním síťovým rozhraním pakety, kontroluje je podle definované množiny pravidel, a pokud se rozhodne pro jejich povolení, odešle je přes jiné rozhraní. Mezi vnějším a vnitřním počítačem nikdy neexistuje přímé spojení; z pohledu počítače ve vnitřní síti tak veškeré informace zdánlivě pocházejí od proxy firewallu.
- **Dynamické proxy firewally.** Tento typ proxy firewallů se vyvinul ze standardních proxy firewallů, oproti kterým je navíc rozšířen o filtrování paketů. Dynamický proxy firewall tak provádí úplnou inspekci paketů; po prvotním vytvoření spojení a zejména po jeho schválení již stačí ostatní pakety kontrolovat v rychlejším, i když slabším mechanismu filtrování paketů. Podtrženo a sečteno, spojení se nejprve zkontroluje na aplikační vrstvě, a poté již další kontrola probíhá jen na vrstvě síťové.

Tyto proxy firewally tak „vidí“ veškeré informace z aplikační vrstvy modelu TCP/IP. To znamená, že mohou vyhledávat přesněji definované údaje, než jakékoli jiné typy dosud probíraných technologií. Dokáží například rozlišit mezi paketem s e-mailovou zprávou a paketem s javovým appletem či nebezpečným prvkem ActiveX, jak vidíme na obrázku 8.

Speciálním případem je SSL proxy, což je varianta proxy firewallu založená na protokolu SSL. S pomocí tohoto protokolu je umožněn uživatelům z vnější sítě (př. Internetu) bezpečný přístup do vnitřní sítě.



Obr. 8: Inspekce paketů v proxy serveru (HTTP)

Při vstupu do proxy serveru podle obrázku 8 se z paketu odstraní veškeré parametry hlavičky TCP/IP a samotné inspekci dále podléhají vlastní přenášená data. Informace zjištěné při této inspekci se poté předloží firewallovým pravidlům a podle výsledků bude průchod paketu povolen nebo zamítnut. Je-li paket shledán „nezávadným“, tedy povoleným, uloží si proxy firewall informace o daném spojení z hlavičky, přepíše novou hlavičku a upravený paket odešle dále. Zamítnutý paket se jednoduše odstraní [4].

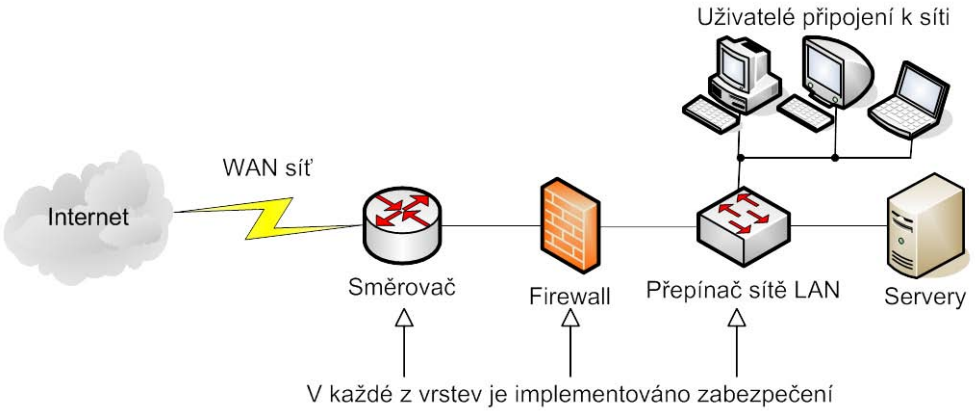
2.7.1 Omezení možností proxy

Každá technologie má svá omezení nebo určité svoje nevýhody. Zde jsou některá obecná omezení proxy firewallů [4], [16]:

- **Pomalejší činnost.** Vzhledem k důkladnému zkoumání a pečlivému zpracování paketů jsou proxy firewally velice bezpečné, ale zároveň také dosti pomalé. Protože se na této úrovni zabezpečení kontrolují v podstatě všechny části všech paketů, bývá činnost proxy firewallů opravdu pomalejší oproti ostatním.
- **Nejsou vždy aktuální.** S vývojem nových protokolů a aplikací je nutné odpovídajícím způsobem doplnit či rozšířit i proxy servery, které musí daný provoz označit za přípustný či nikoliv. To znamená, že je k nové aplikaci nutné také vyvinout a otestovat nové proxy servery. To ovšem nějaký čas trvá a do té doby je příslušné bezpečnostní zařízení neaktuální.

Z bezpečnostního hlediska se za nejbezpečnější firewall dá považovat standardní proxy firewall, který provádí inspekci veškerého provozu na aplikační vrstvě. V některých dnešních sítích není takové řešení vždy zrovna praktické a patřičné. Pro návrh správného a dostatečně silného zabezpečení je proto důležité se pečlivě připravit a seznámit se s charakterem síťového provozu i s požadovaným zabezpečením. Firma pro údržbu parků a zahrad bude mít například jiné potřeby zabezpečení než společnost, která vyvíjí elektronické komponenty pro armádní zbraňové systémy.

V každé navrhované síti je nutné v rámci vrstveného zabezpečení provozovat alespoň jeden ze dvou probíraných typů firewallů - stavový nebo proxy firewall. Pokud kromě některé z těchto technologií spustíme také na hraničním směrovači paketový filtr a ve firewallovém zařízení zapneme překlady adres NAT, dostaneme solidní základ vrstvené bezpečnosti sítě [4].



Obr. 9: Místa vrstveného zabezpečení sítě

Jelikož primárním cílem diplomové práce je detailně popsat princip činnosti proxy firewallů, věnuji této bezpečnostní problematice samostatnou kapitolu, která podrobně rozvede všechny možnosti využití proxy, objasní jak proxy fungují a nakonec vyzdvihne přednosti tohoto řešení.

2.8 Omezení firewallů

Firewall je důležitou komponentou zabezpečení každé sítě a jeho úkolem je řešit problémy spojené s integritou dat a s autentizací síťového provozu (prostřednictvím stavové inspekce paketů), a dále zajišťovat důvěrnost vnitřní sítě (pomocí překladového mechanismu NAT). Pokud bude síť přijímat veškerý provoz jen přes firewall, dostane se jí plnohodnotné ochrany. O důležitosti firewallu ve strategii zabezpečení nelze pochybovat, přesto je ale důležité si uvědomit, že i firewall se potýká s některými omezeními [4]:

- Firewall nedokáže zabránit uživatelům a útočníkům s modemy v přímém přihlášení do vnitřní sítě (nebo naopak ze sítě ven); tím ovšem tento uživatel úplně obchází jak firewall, tak i jeho ochranu.
- Firewally nedokáží uskutečňovat přijaté zásady práce s hesly a nezabrání ani v možném zneužití hesel. Stanovit zásady pro práci s hesly je proto velice důležité, a to včetně sankcí za případné porušování.
- Naprosto neúčinné jsou firewally také proti netechnickým bezpečnostním rizikům, jako je například známé „sociální inženýrství“.
- Každý firewall tvoří úzké hrdlo síťového provozu, protože se v něm veškerá komunikace a zabezpečení koncentrují do jediného místa; tím vzniká jediné kriticky zranitelné místo sítě.

2.9 Porovnání jednotlivých typů firewallů

Výsledné stručné srovnání jednotlivých typů firewallů z hlediska jejich výhod a nevýhod. U každé vzpomenuté technologie se krátce zmíním o vlastnostech a připomenu základní princip.

Paketové filtry (Packet filter)

Nejjednodušší a nejstarší typ firewallu, kde jsou pakety řízeny pravidly, která uvádějí, z jaké adresy a portu na jakou adresu a port může být doručen procházející paket. Tyto informace jsou získány z hlaviček paketů. Paketový filtr pracuje na síťové vrstvě ISO/OSI modelu.

Toto řešení se vyznačuje jednoduchostí a transparentností, což se projevuje vysokou rychlostí. Z toho důvodu je řešení hojně využíváno v místech, kde není potřeba důkladnější analýza procházejících dat. Další nespornou výhodou proti aplikačním proxy je přizpůsobivost na libovolný typ protokolu.

Nevýhodou je nízká úroveň kontroly procházejících spojení. To je způsobeno schopností sledovat pouze jednotlivé pakety bez možnosti hledání závislostí mezi nimi. Další nevýhody jsou absence autentizace vůči firewallu, relativně omezené možnosti logování, sledování pouze hlaviček paketů, zneužití nedokonalosti TCP/IP protokolu a konfigurace filtrů.

Typickým představitelem paketových filtrů je ipchains, který byl součástí jádra. Od jádra verze 2.4 je nahrazen nástrojem iptables (frameworkem netfilter).

Stavové paketové filtry (Stateful packet inspection)

Stavové paketové filtry poskytují to samé jako paketové filtry a navíc umožňují ukládání informací o povolených spojeních, které lze používat při rozhodování o budoucnosti paketů. Dochází ke zvýšení bezpečnosti, protože lze nastavit, která komunikující strana může otevřít spojení a firewall bude povolovat i pakety jdoucí z druhé strany jako odpovědi na požadavky (request <-> response).

Mezi výhody patří stejně jako u paketových filtrů rychlost zpracování požadavků a současně lepší možnost zabezpečení, než u paketových filtrů. Další výhodou je poměrně jednoduchá konfigurace minimalizující škody způsobené například překrýváním některých pravidel a kontrola stavu spojení.

Nevýhodou je nižší úroveň zabezpečení, než poskytují aplikační brány a o něco náročnější režie oproti paketovým filtrům (musí se udržovat informace o spojeních a kontrolovat souvislosti).

Typickým představitelem je iptables v linuxovém jádře.

Stavové paketové filtry s kontrolou protokolů a IDS

Moderní stavové paketové filtry kromě informací o stavu spojení a schopnosti dynamicky otevírat porty pro různá řídící a datová spojení složitějších známých protokolů implementují něco, co se v marketingové terminologii různých společností nazývá nejčastěji *Deep Inspection* nebo *Application Intelligence*. Znamená to, že firewally jsou schopny kontrolovat procházející spojení až na úroveň korektnosti procházejících dat známých protokolů i aplikací.

Nejnověji se do firewallů integrují tzv. *in-line IDS* (*Intrusion Detection Systems* – systémy pro detekci útoků). Tyto systémy pracují podobně jako antiviry a pomocí databáze signatur a heuristické analýzy jsou schopny odhalit vzorce útoků i ve zdánlivě nesouvisejících pokusech o spojení, např. skenování adresního rozsahu, rozsahu portů, známé signatury útoků uvnitř povolených spojení apod.

Výhodou těchto systémů je vysoká úroveň bezpečnosti kontroly procházejících protokolů při zachování relativně snadné konfigurace, vyšší rychlost kontroly ve srovnání s aplikačními branami, nicméně je znát významné zpomalení proti stavovým paketovým filtrům.

Nevýhodou je zejména to, že z hlediska bezpečnosti designu je základním pravidlem bezpečnosti udržovat bezpečnostní systémy co nejjednodušší a nejmenší (i když aktuálním trendem je opak). Tyto typy firewallů integrují obrovské množství funkcionality a zvyšují tak pravděpodobnost, že v některé části jejich kódu bude zneužitelná chyba, která povede ke kompromitaci celého robustního systému.

Podobná funkce je k dispozici ve formě experimentálních modulů pro iptables v linuxovém jádře.

Aplikační brány (Application proxy firewall gateway)

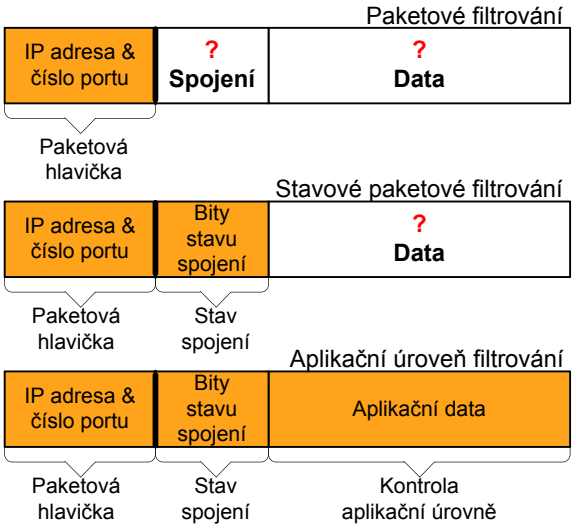
Aplikační brána (proxy firewall, proxy server, aplikační proxy) je ochrana na aplikační vrstvě. Dochází k úplnému oddělení sítí. Spojení probíhá tak, že klient pošle proxy severu požadavek na otevření spojení s nějakou službou v jiné síti a aplikační brána toto spojení otevře. Všechna data prochází vždy přes proxy server, který rozhodne o jejich osudu. Jinými slovy, proxy server slouží jako prostředník mezi klientem v jedné síti a službou v druhé síti. Vedlejším efektem tohoto způsobu komunikace je skrytí zdrojové adresy klienta, protože jako klient vždy vystupuje aplikační brána.

Výhodou tohoto řešení je možnost kontroly obsahu přenášených paketů (např. antivirová kontrola, filtrování nevhodného obsahu, systém detekce průniků, apod.), autentizace uživatelů, možnost cachování dat, skrytí zdrojové adresy klienta (anonymizace) a četné logovací možnosti.

Proti tomuto řešení hovoří především vyšší hardwarové nároky (výkonová a paměťová náročnost) a netransparentnost (nutná úprava aplikací), kdy musí každá aplikace podporovat připojení pomocí proxy a tyto aplikace musí být správně nastaveny. Pro každou aplikaci musí být zvláštní proxy.

Tj. nejsou univerzální jako paketové filtry - HTTP proxy, FTP proxy, atp.

Pro Linux existuje mnoho proxy serverů, jedněmi z nich jsou např. Squid, FWTK nebo SOCKS.



Pro úplnost dále uvádím přehled vlastností a charakter. atributů ostatních síťových technologií.

Network Address Translation (NAT)

Překlad síťových adres slouží ke změně hlaviček paketů, kde jsou přepsány zdrojové - DNAT (Destination NAT), nebo cílové adresy - SNAT (Source NAT). Toto řešení lze použít, pokud nemáme mnoho IP adres, aby všechny počítače v síti mohly mít svou vlastní veřejnou IP adresu nebo ke zvýšení zabezpečení. Za výhodu i nevýhodu lze považovat „neviditelnost“ počítačů z vnější sítě, což vede k již zmíněnému zvýšení zabezpečení a také k omezení, protože nelze používat některé služby vyžadující inicializaci spojení serverem (řeší Port Forwarding, česky přesměrování/mapování portů). Na překladu adres je založena aplikační brána.

Analyzátory paketů (Network Packet Analyzer)

Analyzátory paketů fungují podobně jako aplikační brána, ale jsou pasivní a transparentní. Umožňují neměnnou analýzu paketů a případné reakce na podněty. Příkladem může být IDS balík Snort. Mezi výhody tedy patří pasivní kontrola přenášených dat s možností definice reakce na potenciální útok (např. zablokování příjmu paketů z podezřelé IP adresy). Nevýhodou je náročnost na systémové prostředky a fakt, že nejsou univerzální.

HW versus SW firewally

Pravděpodobně by bylo vhodné uvést i přednosti obecně fyzických řešení firewallů. Hardwarové firewally mají vůči softwarovým firewallům výhodu především v rychlosti, nezávislosti na operačním systému, jednoduchosti nasazení a správy a v méně bezpečnostních „trhlinách“. Zřejmou výhodou může být i integrace firewallu s jinými zařízeními (např. směrovači). Dražší a výkonnější HW firewally mohou rovněž podporovat i externí moduly přidávající komplementární funkce antivirové kontroly či systému detekce průniku IDS. Toto zvolené řešení má ale i své nedostatky v podobě vyšší ceny (některé SW firewally jsou poskytovány zdarma) a pružnosti/adaptace konfigurace. Pouze jako poznámku doplním, že HW firewally jsou spíše firemní záležitostí [6].

3 Překlad síťových adres (NAT)

Když se poprvé objevilo adresování protokolu IPv4, zdálo se, že je množství dostupných adres prakticky neomezené a že musí stačit „na věčné časy“. Teoreticky je v tomto adresovém prostoru k dispozici 2^{32} neboli 4 294 967 296 jedinečných veřejných adres; skutečný počet dostupných adres se pohybuje někde mezi 3,2 a 3,3 miliardami, a to z důvodu rozdělení adresového prostoru do tříd (A, B, C) a vyhrazení určitých adres pro vícesměrné vysílání, testování a další speciální účely (třída D). Uvedené dělení nadefinovalo sdružení IETF (Internet Engineering Task Force), ale později se od něj upustilo, protože nebylo příliš hospodárné. Efektivním se ukázalo být tzv. podsíťování (členění na subsítě), kdy se několik prvních bitů původně určených k adresaci stanice použije na definici podsítě, zbývající bity pak adresují stanice v této podsíti. Subsíť je definována adresou třídy a síť. maskou [4].

Vzhledem k obrovskému rozmachu sítě Internet a neustále rostoucí potřebě přidělování IP adres v domácích sítích a v malých firmách je zřejmé, že počet dostupných adres IPv4 zkrátka nemůže stačit. Jednoduchou úvahou dojdeme k řešení, kterým je změna schématu adresování a rozšíření adresového prostoru. Toto nové adresování je skutečně ve vývoji a nasazení, a to pod označením IPv6, ale jeho implementace potrvá ještě řadu let, protože je u ní nutné změnit činnost infrastruktury celého Internetu. Proces přechodu z dosavadní verze IPv4 na novou verzi IPv6 je proto velmi pomalý a nejspíše bude i nadále postupovat zdlouhavým tempem, protože život dosavadních adres IPv4 dále prodlužuje překladový mechanismus NAT.

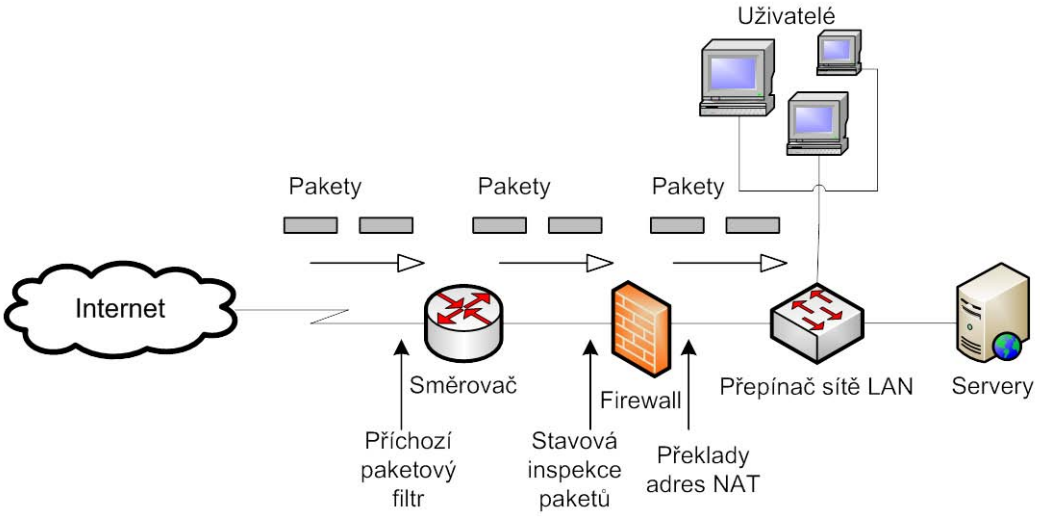
Pomocí překladů síťových adres NAT mohou organizace vyřešit nedostatek veřejných IP adres ve své síti, připojené do Internetu. Sítě, které dosud nemají veřejné IP adresy, registrované u centra NIC, si je musí vyžádat od sdružení IANA (Internet Assigned Numbers Authority) a registru ARIN (American Registry for Internet Numbers), což znamená zdlouhavý úřední postup. Mnohá pracoviště dokonce těmito nepříjemnými úředními procedurami neprojdou; pro většinu sítí je tudíž řešením právě NAT.

Sdružení IANA vyhradilo pro potřeby privátních sítí následující tři bloky adresového prostoru IP:

- 10.0.0.0 – 10.255.255.255 (prefix 10/8)
- 172.16.0.0 – 172.31.255.255 (prefix 172.16/12)
- 192.168.0.0 – 192.168.255.255 (prefix 192.168/16)

Při zapojení překladů NAT může firma používat **veřejné IP adresy** na vnější straně sítě (tedy u zařízení, přímo připojených k veřejnému Internetu). Jak jsem ale naznačil, těžko bude mít stejná firma dostatek veřejných IP adres, aby je přiřadila každému serveru, osobnímu počítači, síťové tiskárně, směrovači, bezdrátovému zařízení, atd. Všechna uvedená zařízení potřebují ale pro komunikaci v protokolu TCP/IP nějakou IP adresu, a proto budeme ve vnitřní síti přiřazovat **privátní IP adresy**. Díky tomu mohou všechna zařízení vnitřní sítě komunikovat v protokolu TCP/IP – a to je také naším cílem. Směrem do veřejného Internetu, se již ale privátní adresy dostat nesmí, a proto musíme uvést do provozu mechanismus NAT.

Překlady síťových adres NAT (Network Address Translation) zavádíme a provozujeme na vhodném zařízení (firewallu, směrovači nebo počítači), umístěném mezi vnitřní sítí s privátními IP adresami a vnějším Internetem s veřejnými IP adresami. Zmíněné zařízení pak provádí takzvané převody neboli překlady adres z privátních na veřejné. Jedna jeho strana bývá připojena k vnitřní síti, druhá pak k Internetu nebo jiné vnější síti. Umístění mechanismu NAT v rámci vrstvené obrany dokresluje obrázek 10 [4].



Obr. 10: Pozice mechanismu NAT v síti

Mechanismus překladů adres NAT znamená také další úroveň zabezpečení sítě. Samotný NAT má několik různých podob a může pracovat ve třech základních režimech činností [4] [11]:

- *Statický NAT* – definuje jednoznačné mapování neboli zobrazení privátních IP adres na veřejné (tedy jedna k jedné). To je užitečné zejména u zařízení, která musí být dostupná z veřejného Internetu (z vnější sítě). Pokud má například webový server takovouto vnitřní IP adresu (10.0.0.1) a má být dostupný z Internetu (je to veřejný webový server), musíme definovat statický překlad NAT, který zajistí trvalý a jednoznačný převod uživatelských požadavků s veřejnou adresou webového serveru na jeho vnitřní adresu 10.0.0.1. Právě pro zařízení dostupná z vnější sítě, jako jsou webové servery apod., jsou statické překlady NAT velice běžné.
- *Dynamický NAT* – tento režim zajišťuje mapování privátních IP adres na veřejnou IP adresu, vybranou ze skupiny registrovaných adres. I při tomto typu překladů NAT je mezi privátními a veřejnými IP adresami jednoznačné zobrazení (jedna k jedné); má-li například náš osobní počítač privátní adresu 10.0.0.2 a kolegův počítač 10.0.0.3, dostaneme při komunikaci s veřejným Internetem přiděleny od firewallu dvě různé veřejné IP adresy (ty mohou být ovšem při každé komunikaci jiné). Dynamický NAT je bezesporu užitečný, ale na některé stanice je krátký; může se například stát, že již firewall všechny veřejné IP adresy vyčerpal a naši komunikaci tak zamítne. To může být v určitých situacích vážný problém, proto byl vyvinut takzvaný přetížený NAT.
- *Přetížený NAT* – jedná se o speciální typ dynamického NAT, který převádí větší skupinu privátních IP adres na jedinou veřejnou IP adresu, přičemž je rozlišuje pomocí různých portů TCP. Uvedený mechanismus se označuje také jako překlady portů PAT (Port Address Translation), případně jednoadresový NAT. Název ale není tak podstatný jako mechanismus činnosti: pro jedinou IP adresu je totiž k dispozici více než 64 000 portů (celkem 65 535) TCP, a proto PAT umožňuje efektivní přístup k Internetu velkému množství uživatelů s privátními IP adresami. Tento poslední typ překladu NAT se používá nejčastěji, protože dokáže najednou obsloužit největší množství uživatelů.

3.1 Zvýšení bezpečnosti sítě

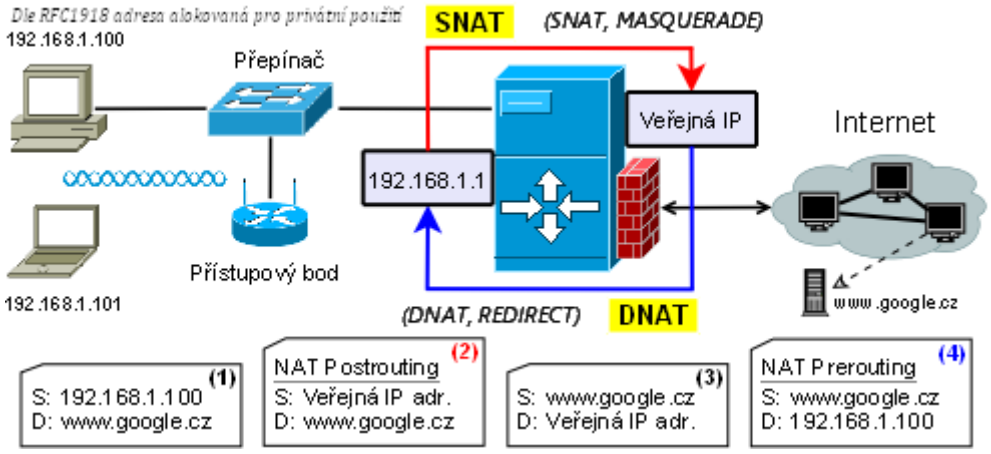
Vývoj překladového mechanismu NAT byl veden především snahou o vyřešení problému s nedostatkem veřejných adres protokolu IPv4. NAT poskytuje ale další vrstvu zabezpečení a ochrany sítě. Obecně se totiž dá říci, že NAT útočníkovi velmi výrazně ztěžuje [4]:

- Mapování topologie cílové sítě a zjišťování informací o konektivitě
- Zjištění počtu provozovaných systémů v síti
- Zjištění typu provozovaných počítačů a jejich operačních systémů
- Vedení různých útoků s odepřením služeb (DoS), jako jsou například záplavy synchronizačních paketů SYN, prohledávání portů a vnášení paketů (injekce)

Zařízení uvnitř sítě LAN nejsou díky technologii překladu adres NAT adresovatelná z Internetu, což taktéž posiluje bezpečnost (útočník nezná strukturu sítě a nemůže se spojit přímo s konkrétním počítačem, lépe řečeno zařízením v síti, vždy se dostane „pouze“ na hraniční směrovač nebo počítač provádějící službu překladu adres). NAT ovšem firewall nenahrazuje a existují způsoby, jak počítače za NATem napadnout.

3.2 Překlad adres a OS Linux

Zde je uvedena příkladová situace. Uživatel vnitřní lokální sítě s IP 192.168.1.100 má přístup k Internetu a požaduje po internetovém prohlížeči webovou stránku http://www.google.cz. Internetový prohlížeč naváže TCP spojení pomocí HTTP protokolu s webovým serverem (na portu 80) a vrátí zpět uživateli požadovaný dokument, www stránku, soubor apod. Mezitím se událo (z pohledu překladu síťových adres) několik skutečností zachycených na obrázku 11 [10].



Obr. 11: SNAT (přepis zdrojových adres), DNAT (přepis cílových adres)

Situace, která je symbolizována tabulkou s pořadovým číslem (1), zachycuje požadavek klienta v síti LAN na webový vyhledávací portál s doménovým názvem www.google.cz. S pořadovým číslem (2) následuje přepis zdrojové IP adresy (+ portu) SNAT tohoto požadavku, dále ve (3) kroku příchozí odpověď od serveru na požadavek klienta, a ve (4) finální fázi přepis cílové IP adresy (+ portu) DNAT.

SNAT v *POSTROUTING* – mění se zdrojové adresy a zdrojové porty.
DNAT v *PREROUTING* – mění se cílové adresy a cílové porty.
Využití je zřejmé: připojení sítí LAN k Internetu pomocí jedné veřejné IP adresy.

Pozn. Ne všechny pakety prošlé PRE/POSTROUTING projdou, musí mít pravidlo v řetězci FORWARD!

3.3 Omezení mechanismu NAT

Je zřejmé, že příchod technologie NAT do světa počítačových sítí a Internetu znamenal alespoň částečné vyřešení problémů s nedostatkem IP adres. Mnozí lidé se tudíž ptají, jestli sítě vůbec někdy přejdou na novou verzi protokolu IPv6, když překlady NAT tak výborně fungují. Otázkou ovšem fakticky není zdali, ale kdy. Asijský region dnes například v implementaci protokolu IPv6 jasně vede a mnohé sítě jej již skutečně používají [4].

S ohledem na stále rostoucí konektivitu a konvergenci sítí budou potřeba stále další a další IP adresy. Nakonec se tedy budeme muset k protokolu IPv6 uchýlit. Mechanismus NAT tyto nevyhnutelné změny pouze oddálil. NAT má svoje nepopiratelné výhody, ale na druhé straně má také jistá omezení:

- **Problémy s protokolem UDP** – překladový mechanismus NAT sleduje a kontroluje stav spojení. V protokolu UDP ovšem nelze stav spojení nijak určit (protokol UDP je nespojovaný – spojení se v něm vůbec nevytvářejí). NAT nedokáže tudíž žádným způsobem určit, jestli určitý paket spadá do nějaké probíhající konverzace, nebo jestli tvoří izolovaný přenos dat. Zařízení s překlady NAT tak musí odhadovat, jak dlouho může konverzace UDP trvat a jak dlouho ji tedy po posledním paketu ponechat otevřenou; hovoříme o tzv. době nečinnosti. Například ve firewallech od firmy Cisco je možné nastavením doby nečinnosti tyto případy omezit.
- **Citlivé protokoly** – některé protokoly skrývají, pozměňují nebo jinak zastiňují atributy paketů, které NAT potřebuje ke správnému překladu adres. Příkladem jsou protokoly Kerberos, X Window, vzdálený shell (rsh) nebo protokol SIP (Session Initiation Protocol), které mohou mít při průchodu zařízením s NAT jisté problémy. Příčina problémů se skrývá v aplikacích, jež vkládají IP adresu do paketů. Firewally Cisco mají pro různé protokoly opravné funkce (fixup), například Skinny pro telefonii (SCCP).
- **Vzájemné vlivy systémů šifrování a autentizace** – mnohé systémy šifrování dat se pokoušejí zajistit integritu paketů a kontrolují tak jejich neporušenost při přenosu. Překladový mechanismus NAT ale z principu pakety pozměňuje, a proto šifrovací a autentizační technologie s ním často nedokáží spolupracovat.
- **Komplikovaný záznam do systémových protokolů** – pokud zařízení odesílá přes překladové zařízení určité informace do systémových protokolů, musí cílové zařízení znát překlady prováděné mechanismem NAT. Dosažení souladu systémových protokolů s překlady NAT pak může být velice složité a často se těžko zajišťuje, který z interních systémů vlastně dané události zaznamenal.

Závěrem ale poznamenejme, že překladový mechanismus NAT je skutečně velmi užitečný – celé firemní síti nabízí přístup k Internetu a zároveň tvoří další úroveň zabezpečení. Technika NAT umožňuje komunikaci zařízení v privátní síti se zařízeními jiných sítí, sdílet jednu veřejnou IP adresu více zařízeními privátní sítě, vyrovnávat zatížení serverů poskytujících stejnou službu, v případě výpadku zajistit adekvátní náhradu serveru záložním serverem a hlavně (z hlediska bezpečnosti) skrýt před vnějším útočníkem strukturu vnitřní sítě. Tímto disponují i proxy servery, probírané v další kapitole, jež pracují na nejvyšší aplikační vrstvě. Dokáží navíc zabránit prozrazení některých důležitých informací o hostitelské stanici, jako např. typ OS, www prohlížeč, interní IP adresa a další. Vzdálený server se tak o klientu dozví minimum informací (je odkázán pouze na informace zpřístupněné proxy serverem), což umožňuje určitý (pokročilejší) stupeň anonymizace.

4 Proxy servery

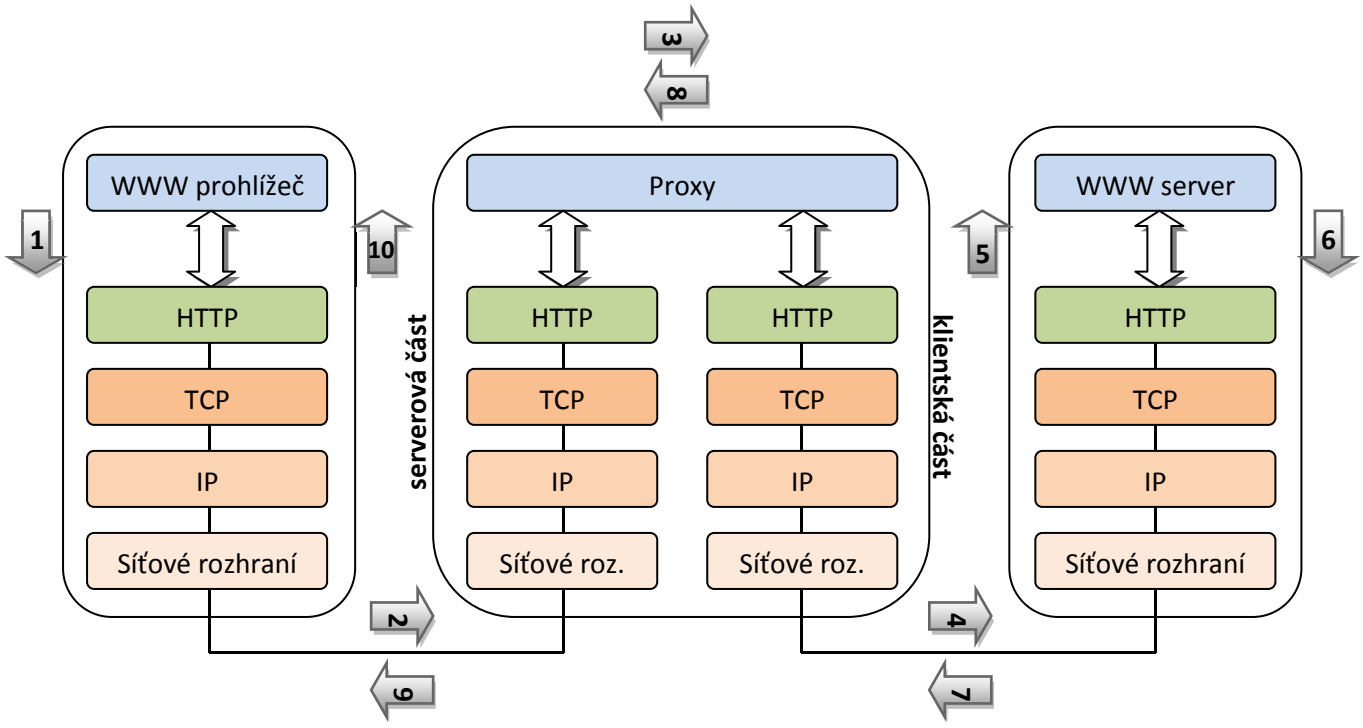
4.1 Jak proxy servery fungují

Přímá komunikace mezi klientem a WWW serverem pomocí HTTP protokolu je nejběžnější způsob získávání informací z WWW. V některých případech je však tato forma komunikace zprostředkována pomocí prostředníka – proxy serveru. Proxy server je zpravidla program (běžící na serveru organizace), který pracuje současně jako klient i server, schematické znázornění funkce je možné nalézt na obrázku 12. V obrázku je záměrně opomenuto DNS zjišťování IP adresy hostitele, které probíhá jak na straně klienta (hledání IP proxy serveru), tak na straně proxy serveru (hledání cílové IP webového serveru).

Podstatnou vlastností je, že proxy server tím, že vystupuje za klienta, poskytuje anonymitu uživatele vzhledem k www serveru.

Jak již bylo nastíněno, proxy server je server počítačové sítě, který umožňuje klientům nepřímé připojení k jinému serveru. Proxy server funguje jako prostředník/zprostředkovatel mezi klientem a cílovým serverem, překládá klientské požadavky a vůči cílovému serveru vystupuje jako klient. Přijatý požadavek následně odesílá zpět na klienta. Může se jednat jak o specializovaný hardware, tak o software běžící například na počítači klienta.

Aplikační proxy server je server speciálně určený pro určitý protokol resp. aplikaci. Za pomocí něj lze analyzovat obsah komunikace, případně ji pozměňovat (např. odstraňování reklamních bannerů z HTTP požadavků, blokování webových stránek podle obsahu, zákaz přístupu na nepovolené webové servery apod.) [18].



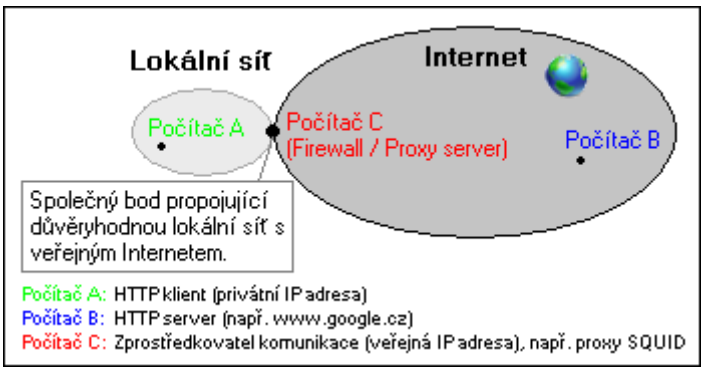
Obr. 12: Fungování komunikace zprostředkované proxy serverem

Stručný popis jednotlivých kroků [18]:

(1) Uživatel chce zobrazit určitou stránku na internetu, např.: http://www.server.cz/stranka.htm, což se předá nižším vrstvám k přenosu.
(2) Nižší vrstvy navážou spojení se serverovou částí proxy serveru a předají požadavek prohlížeče (HTTP metoda GET).
(3) Proxy server rozumí struktuře HTTP protokolu a proto je schopen detailně vyhodnotit požadavek. Na základě vlastní logiky může proxy tento požadavek přepsat a de facto tak např. změnit cílový server, z kterého se bude stránka získávat. Dále může ověřovat, zda je vůbec uživatel oprávněn takový požadavek provést. Další netriviální činností, kterou může proxy provádět, je ukládání odpovědí do své cache paměti a při opakování dotazu na stejnou stránku tak zajistit rychlejší odezvu. Tato činnost však mírně ztrácí význam v souvislosti s rozvojem dynamicky generovaných stránek. Komplexnější proxy dokáže na základě spolupráce s firewallem případně antivirovým programem účinně filtrovat provoz a chránit tak uživatele.
Pokud proxy nenajde ve své paměti požadovanou stránku a uživatel je oprávněn, předá se požadavek klientské části proxy.
(4) Klientská část zajistí navázání spojení s www serverem a předání metody (GET /stranka.htm).
(5) WWW server přijme požadavek a vyhodnotí ho.
(6) Výsledkem je odpověď WWW serveru, zpravidla požadovaná stránka.
(7) Na základě nižších vrstev je odpověď předána zdroji metody GET, což je v tomto případě klientská část proxy serveru.
(8) Stejné operace jako v bodě 3), jen zpětně. Např. přepsání odpovědi do formy v jaké ji očekává uživatel, případně zápis stránky do cache paměti.
(9) Odpověď se pomocí nižších vrstev předá uživateli (původní tazatel).
(10) Prohlížeč odpověď zpracuje a stránku zobrazí.

Jak je patrné z předcházejícího textu, proxy rozumí aplikačnímu protokolu (HTTP) a je schopno samo navazovat spojení, což jsou nejmarkantnější rozdíly např. od techniky překladu síťových adres (NAT).

Umístění proxy serveru v rámci sítě není pevně dáno. Nejčastějším použitím je oddělení vnitřní sítě a Internetu, takže proxy server je umístěn na pomezí těchto dvou oblastí (viz obrázek 13). K proxy serveru se však lze připojovat i za hranice vnitřní sítě (do Internetu). V takovém případě se jedná o veřejné (public) proxy servery, kterých existuje velké množství, avšak jejich důvěryhodnost je často pochybná, protože do souhrnu (ne)žádoucích činností, které provádí, nemáme možnost nahlédnout.



Obr. 13: Ukázka umístění proxy serveru v rámci sítě (LAN ⇔ Internet)

V textu je pojednáno pouze o variantě proxy pracující s HTTP protokolem (nejčastější), proxy server však zpravidla podporuje i další typy, jako např. FTP protokol atp.

4.1.1 Oddělení sítí – bezpečnost

V lokálních sítích se často používají privátní adresy (např. 10.0.0.0/8, 192.168.0.0/16), takže počítače nemohou komunikovat (resp. přistupovat) přímo do Internetu. Používá se buď technika překladu adres na firewallu (NAT), nebo řešení pomocí proxy serveru. Klient používá proxy server na své vlastní privátní síti (má privátní IP adresu) a ten komunikuje (nejčastěji pomocí jiného síťového připojení) s Internetem (tedy vnější sítí). Nutno poznamenat, že pro některé sítě je použití přístupu pomocí proxy serveru jedinou možností přístupu na Internet.

Výhodou použití proxy serveru z hlediska bezpečnosti je:

- Možno ho použít pro přístup k (určitým službám) Internetu i v případě, že klienti používají privátní adresy (netřeba použít NAT).
- Možno ho použít jen pro přístup k těm službám, které podporuje proxy server.
- Klient se připojuje na (lokální) proxy server a ne přímo na vzdálený server – nekomunikuje přímo s vnější sítí, čímž se dosahuje vyšší bezpečnosti, neboť je nemožné zneužít probíhající spojení na útok směřující ze serveru na klienta.

Existují nejméně tyto tři zmíněné důvody, proč nasadit proxy server v praxi. Přirozeně, řešení má i své nevýhody či nedostatky:

- Možno ho použít jen na přístup k těm síťovým službám, které podporuje samotný proxy server (toto je výhoda i nevýhoda zároveň, záleží na úhlu pohledu).
- V případě použití aplikačního proxy serveru, zpracování aplikační vrstvy modelu OSI (běžné síťové protokoly založené na TCP) zanáší do komunikace jisté zpoždění – proxy server musí analyzovat nejvyšší vrstvu modelu OSI.

Obecné doporučení:

V případě užití proxy serveru, vzdálený server nekomunikuje přímo s klientem, takže nemůže proti němu podniknout útok, může však zneužít spojení se samotným proxy serverem. Tento proxy server je třeba proto co nejlépe zabezpečit (pomocí firewallu):

- Proxy server nepotřebuje přijímat spojení z vnější sítě, pouze odpovědi na už vytvořené spojení (stavový filtr v IPTABLES: ESTABLISHED).
- Proxy server se nepotřebuje připojovat do vnitřní sítě, posílá pouze odpovědi na už vytvořené spojení (stavový filtr v IPTABLES: ESTABLISHED).
- Proxy server by měl „poslouchat“ jen na portech, které jsou nevyhnutelné pro správný provoz dané služby (služeb).
- Proxy server by v ideálním případě neměl poskytovat žádné jiné služby (WWW, FTP, MAIL, aj.); důvodů existuje několik, mezi hlavní patří bezpečnostní hledisko (možné zneužití poskytovaných služeb a následné získání vyšších práv - *root*) a rozprostření zátěže mezi více serverů (přičemž každý server plní svou funkci => při pádu jednoho stále fungují ostatní).

4.1.2 Řízení přístupu

Jestliže zabezpečíme, že všechny klientské počítače v síti budou používat proxy server, stane se jediným „úzkým hrdlem“ komunikace mezi klientem a serverem. Vznikne tak místo v síti, kde je možné pomocí stanovených pravidel regulovat přístup k Internetu.

Výhody použití proxy serveru:

- Možnost nasazení a vynucování bezpečnostní politiky z hlediska přístupu k Internetu – např. k určitým nevhodným či nebezpečným webovým stránkám.
- Jednoduché nastavení bez nutnosti konfigurovat politiky na všech klientech.

Nevýhody použití proxy serveru:

- Výkon proxy serveru má vliv na rychlost a efektivitu přenosu (čas odezvy na požadavek)
- Vzniká rizikové místo v síti („single point of failure“), v případě výpadku proxy serveru je nemožné se dostat na Internet (či obecně do vnější/WAN sítě). Toto lze řešit pomocí vícenásobného množství proxy serverů v clusteru, ale jen v případě, že je možné si to dovolit.

4.2 Druhy proxy serverů

Jak už jsem v předešlém poznamenal, proxy je síťová brána, obecně nejen pro HTTP protokol. Většinou umožňuje cachování dat, kontrolu přístupu, filtrování obsahu a některé další služby.

V zásadě rozlišujeme dva druhy proxy serverů [19]:

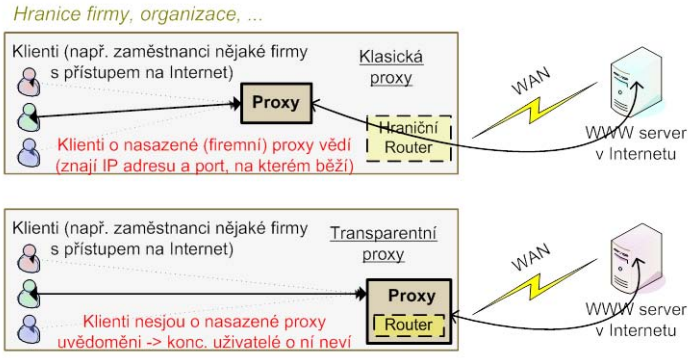
1. **Aplikační proxy servery** – jsou navrhnuté jen pro vybrané síťové služby, jako např. HTTP, HTTPS, FTP, apod. Výhodou je, že nastavování a konfigurace klienta nejsou složité, stačí uvést adresu proxy serveru a jeho port.
2. **SOCKS proxy servery** – fungují pro libovolné služby za předpokladu, že aplikace podporuje tento typ proxy serveru (musí být speciálně upravená).

Další eventuální rozdělení je taktéž na **forward proxy** a **reverse proxy**. Záleží, zda jde o *caching* či *balancing* – umístění před klientem nebo serverem.

Forward (dopředná) proxy – klasická síťová brána s možností cachování, umístěna blíže ke klientovi. Typická je speciální konfigurace klienta (nemusí být vždy pravda).

Vlastnosti: urychlení síťového provozu, snížení datového toku, bezpečnost (lze doplnit o antivir aj.).

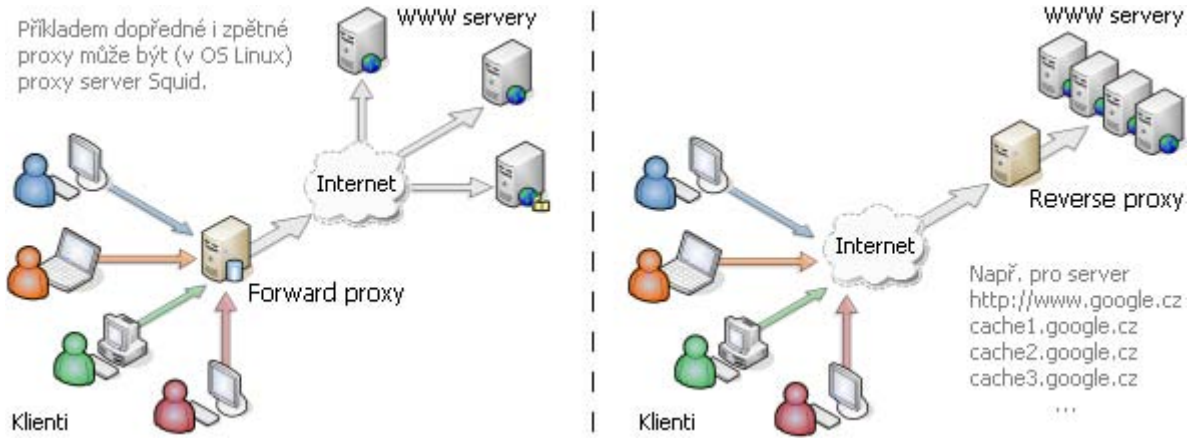
Podrobnější rozčlenění dopředné proxy může být na klasickou proxy a transparentní proxy.



Reverse (zpětná) proxy – je transparentní z pohledu klienta, umístěna před serverem. Výhodou je možnost balancingu (rozložení požadavků na více strojů). Klient nevyžaduje speciální konfiguraci.

Vlastnosti: urychlení síťového provozu (některé požadavky lze vyřídit z cache paměti, eliminace prodlevy ve spojení mezi klientem a serverem, mezi klientem a proxy může probíhat SSL spojení, vlastní server se tak může věnovat vyřizování skutečných požadavků), rozložení zátěže na více serverů, zvýšená bezpečnost (klienti se připojují k proxy, ne k samotnému serveru).

Reverzní proxy server někdy také slouží k šifrování/dešifrování spojení. Nedochází tím k degradaci potřebného výpočetního výkonu serverů. Zajištěno je šifrování směrem ke klientovi, spojení mezi serverem a proxy šifrované není, ale tato oblast se považuje za bezpečnou (součást např. DMZ).



Obr. 14: Aplikované rozdělení proxy na Forward (dopřednou) a Reverse (zpětnou)

4.2.1 Klasická proxy

Klasická proxy se používá zejména pro protokoly Telnet, FTP, HTTP a HTTPS. Například při použití Telnetu se uživatel nejdříve připojí na proxy, kde zadá příkaz connect, ve kterém určí název nebo IP adresu skutečného serveru. Klientská část proxy naváže spojení s cílovým serverem a začne předávat mezi klientem a tímto serverem data. Klientovi se pak jeví, jako by mezi ním a cílovým serverem žádná proxy nebyla [34], [16].

4.2.2 Generická proxy

Někteří klienti nemohou nebo nechtějí vést úvodní dialog s proxy, aby jí sdělili cílový server. Generická proxy je program, který může spouštět a konfigurovat správce sítě podle momentálních požadavků. Dokáže pracovat s TCP i UDP.

Serverová část proxy je spuštěná na konkrétním portu, který určil správce, a je připravena přijímat požadavky od klientů ve vnitřní síti. Klientská část proxy je pevně nastavena na jeden cílový server.

Jedna spuštěná generická proxy dokáže obsloužit jeden cílový server. Naráz jích pochopitelně může být spuštěných více. Generické proxy se používají hlavně pro protokoly POP3, IMAP4, SSH, LDAP, SMTP a NNTP [34], [16].

4.2.3 Transparentní proxy

Klient naváže spojení s transparentní proxy a má pocit, že navázal spojení se skutečným serverem. Klientská část transparentní proxy hned navazuje spojení s cílovým serverem. Transparentní proxy nemusí vést s klientem žádný dialog.
Předpoklady pro práci transparentní proxy jsou:

- Klient ve vnitřní síti má možnost přeložit jméno cílového serveru nebo počítače v Internetu na IP adresu. Cílová adresa IP datagramu je adresa cílového serveru.
- IP datagram odeslaný do Internetu musí projít přes transparentní proxy.
- IP datagramy jednoho spojení musí procházet přes jednu transparentní proxy. Může existovat více transparentních proxy najednou.

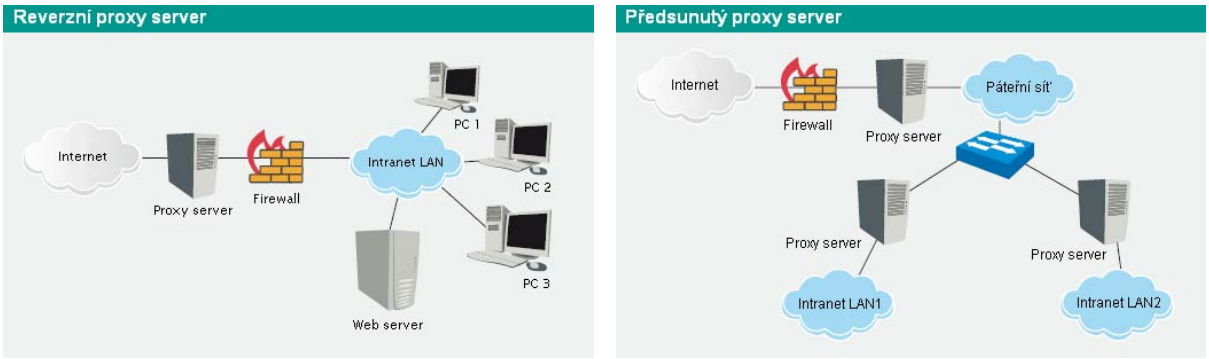
Používá se především pro protokoly HTTP, HTTPS, FTP, NNTP, IMAP, POP3 a Telnet [34], [16].

4.2.4 Reverzní proxy

Reverzní proxy využívají jinou technologii proxy serverů. Můžeme je využít vně firewallu, kde reprezentují bezpečný obsahový server určený pro vnější klienty a zabraňují přímému, nesledovanému přístupu k datům na vnitřních serverech z vnější sítě. Reverzní proxy také urychlují činnost sítě, protože před silně vytížený server můžeme umístit i několik proxy serverů a zátěž mezi nimi vyrovnávat. Dopřednou a reverzní aplikaci může zajišťovat stejný proxy server.

Je patrné, že složitý obvod sítě s různými proxy servery, tunely a paketovými filtry, které pracují v obou směrech, se může velmi rychle zkomplikovat a znepřehlednit. V diagramech propojení sítě je proto vhodné zavést si jednoduchou konvenci a každý z typů serverů označit např. následovně [5],[9]:

- *C Client (Klient)* – systém, který vyžaduje určitou službu
- *S Server* – systém s požadovanou internetovou službou
- *L Listener (Posluchač)* – vnitřní strana proxy serveru, která očekává spojení od klientů
- *I Iniciátor* – vnější strana proxy serveru, která navazuje spojení s vlastním serverem



Obr. 15: Ukázka reverzního a předsunutého proxy serveru

4.2.5 SOCKS

SOCKS je souprava proxy nástrojů, která umožňuje zprostředkovanou proxy komunikaci aplikací bez nutnosti vytváření zvláštních proxy modulů. Hostitel na jedné straně serveru SOCKS může přistupovat k hostitelům na druhé straně serveru bez potřeby přímé konektivity IP. Server SOCKS přesměruje spojení i požadavky služeb mezi hostiteli; zajišťuje přitom autentizaci a autorizaci požadavků, zavádí zprostředkované proxy spojení a přeposílá data mezi oběma propojenými hostiteli.

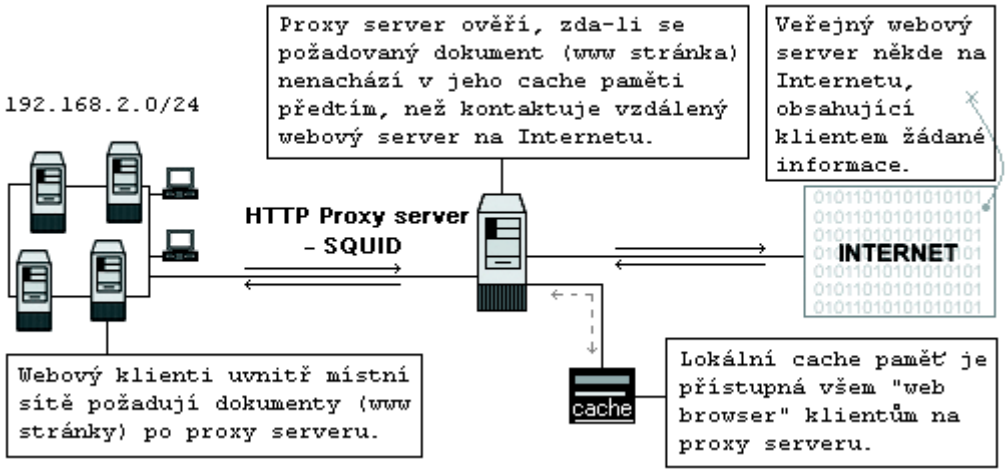
Program SOCKS má dva samostatné konfigurační soubory. První definuje povolené typy přístupu a druhý směruje požadavky služeb na odpovídající proxy server. Dále je zde možné identifikovat IP adresy a uživatele pro filtrování.

Aplikace, které mají s proxy serverem SOCKS spolupracovat, musí být zvlášť upravené. Pro každou z podporovaných služeb jsou potřeba hned dvě aplikace – například dva různé programy Telnet, z nichž jeden zajišťuje přímou komunikaci a druhý komunikaci přes proxy server SOCKS. Instrukce pro nezbytné úpravy najdeme přímo u serveru SOCKS. Určité úpravy jsou potřebné v každé aplikaci; klientský software pak komunikuje přes server SOCKS. Celý SOCKS se skládá ze dvou součástí, a sice ze serveru SOCKS a klienta SOCKS. Server SOCKS je implementován na aplikační úrovni, zatímco klient SOCKS pracuje mezi aplikační a transportní vrstvou [34], [16].

4.3 Cache pro obsah (WWW)

Některé typy aplikačních proxy serverů (např. HTTP proxy) mohou disponovat cache pamětí, ve které se určitý čas uchovávají odpovědi vzdálených serverů a obsah této paměti je k dispozici pro další požadavky. Jestliže proxy server najde požadovanou odpověď v cache paměti, může odpovědět klientovi rychleji, než kdyby musel odpověď znovu získat. To je výhodné zejména v případě, že připojení na Internet není velmi rychlé, zatímco přenosová rychlost lokálních sítí dnes běžně dosahuje 100 Mbit/s i více.

Mohlo by se zdát, že hlavní výhodou je tedy kratší čas odezvy na požadavky klienta, což je i pravda, ale pouze v případě, že se hledaná odpověď nachází v cache paměti proxy serveru. Její velikost je totiž vždy omezená (kromě fyzické paměti se používá i pevný disk). Aby se do cache mohly ukládat stále nové údaje, musí se z ní něco odstranit. Na to existuje několik algoritmů, např. odstraní se nejméně používané údaje. Nejrychlejší odezvy na klientský požadavek lze dosáhnout pomocí proxy serveru tehdy, jestliže bude v případě HTTP proxy přistupováno na nejčastěji využívané stránky.



Obr. 16: Proxy cache

Výhoda použití proxy serveru je tedy, kromě již zmíněné bezpečnosti, zmenšení času odezvy (subjektivně „rychlejší načítání“) pro nejčastější požadavky nacházející se v cache paměti. Nevýhody lze zmínit dvě. První spočívá v omezené velikosti cache paměti a druhá v limitu na minimální a maximální velikost cachovaného objektu. Může se stát, že stránky, které je potřeba uchovávat v paměti, nevyhoví zmíněným limitním požadavkům, a bude tedy nutností je vždy získat z webového serveru. Pro lepší představu, jak funguje cachující proxy server pro WWW stránky, vyzdvihnu následující orientační model. Požadavkem je URL adresa, kterou klient požaduje od HTTP proxy.

Nachází se webová stránka v cache paměti?

- **ANO – Je odpověď stále ještě platná? (Pozn. zjišťuje se například čas poslední modifikace stránky)**
  - Ano – Proxy server vrátí stránku ze svojí cache paměti.
  - Ne – Proxy server aktualizuje kopii stránky v cache paměti a vrátí jí klientovi**.**
- **NE – Proxy server se pokusí stáhnout webovou stránku z cílového serveru. Podařilo se stránku získat?**
  - Ano – Vyhovuje stránka kritériím pro uložení (velikost, typ, direktivy pro proxy server, apod.)?
    - *Ano – Proxy server uloží stránku do cache paměti a vrátí jí klientovi.*
    - *Ne – Proxy server vrátí stránku klientovi bez uložení v cache paměti.*
  - Ne – Proxy server vrátí klientovi hlášení o chybě.

Proxy servery s cache pamětí se používají téměř výhradně na uchovávání obsahu WWW stránek.

5 Linuxový proxy firewall

5.1 Platforma Linux

5.1.1 Proč Linux?

Linux od počátku patřil mezi často nasazované operační systémy na servery. Toto řešení v sobě přinášelo (kromě jiného) spolehlivost, stabilitu, bezpečnost, konfigurovatelnost, efektivnost a ve většině případů může být i zdarma. Pro náš proxy server spolu s firewallem je operační systém GNU/Linux ideálním východiskem. Veškeré aplikace, které zde budou později uváděny a které budou vhodně zvoleny pro projekt (s ohledem na dostupnost, bezpečnost řešení, konfigurovatelnost, apod.), budou tzv. open source [21], [23].

Open source nebo také open-source software (OSS) je počítačový software s otevřeným zdrojovým kódem. Otevřenost zde znamená jak technickou dostupnost kódu, tak legální dostupnost – licenci software, která umožňuje, při dodržení jistých podmínek, uživatelům zdrojový kód využívat, například prohlížet a upravovat. V užším smyslu se OSS míní software s licencí vyhovující definici prosazované Open Source Initiative (OSI). Pro odlišení se někdy open source software vyhovující požadavkům OSI označuje Open Source.

V nepřesném, ale poměrně běžném vyjadřování se označení open source používá i pro mnoho vlastností, které s otevřeností zdrojového kódu nesouvisí, ale vyskytují se u mnoha open source programů. Například může jít o bezplatnou dostupnost software, vývoj zajišťovaný úplně nebo z podstatné části dobrovolnickou komunitou nebo „nekomerčnost“ [32], [37].

5.1.1.1 Bezpečnost open source

Z hlediska bezpečnostních děr v software je otevřenost kódu dvojsečná zbraň. Chyby v programech může hledat mnohem širší skupina lidí (nebo i automatických pomůcek) a je proto naděje, že se snáze opraví. Na druhou stranu zranitelnosti mohou snáze najít i útočníci. V současném paradigmatu informační bezpečnosti *full disclosure* se ovšem považuje za obecně výhodnější, když jsou informace dostupné všem, i za tu cenu, že jsou dostupné útočníkům. Alespoň u populárních programů s velkou základnou uživatelů a vývojářů lze předpokládat, že „uživatelská“ strana má výrazně větší prostředky (především více času kvalifikovaných lidí) než potenciální útočník.

Nespornou výhodou otevřeného zdrojového kódu je ohromné ztížení možnosti propašování zadních vrátek a trojských koní [32].

5.1.2 Kterou distribuci Linuxu zvolit?

Protože praktickým cílem této diplomové práce je navrhnout, nakonfigurovat a zprovoznit funkční proxy firewall na platformě GNU/Linux, je třeba příslušnou konkrétní distribuci tohoto OS zvolit. S ohledem na bezpečnost, stabilitu a spolehlivost (v neposlední řadě také dlouhodobou prověřenost administrátory serverů) byla zvolena distribuce s plným označením Debian GNU/Linux verze 4.0r5 (později upgradováno na stabilní verzi 5.0, čili z Etch na Lenny), která je schopna plnohodnotného serverového nasazení, a která velice dobře splňuje požadavky serverového řešení proxy firewallu.

5.1.3 Debian GNU/Linux

Na úvod stručně charakterizuji vybranou distribuci operačního systému Linux, která je v tomto případě Debian (s verzí jádra 2.6.26).

5.1.3.1 Co je Debian?

Debian je svobodný operační systém určený k provozu na mnoha různých typech počítačů. Operační systém se skládá ze základního programového vybavení a dalších nástrojů, kterých je k provozu počítače třeba. Vlastním základem systému je jádro. Jelikož Debian používá jádro Linux a většina základních systémových programů byla vytvořena v rámci projektu GNU, nese systém plné označení GNU/Linux.

Debian nezávisí na konkrétním jádře. Momentálně používá jádro Linux, které je svobodným softwarem, jehož vývoj byl zahájen Linusem Torvaldsem a je nyní podporováno více než tisíci programátory z celého světa (dobrovolníky i profesionály). Pracuje se na úpravě Debianu i pro jiná jádra, obzvláště Hurd. Hurd je kolekce serverů běžících nad mikrojádrem (např. Mach nebo L4).

Tato distribuce je známa především svou konzervativností. Přesto je to jedna z nejrozšířenějších linuxových distribucí na světě. Oproti klasickým „komerčním distribucím“, jako jsou Red Hat nebo SUSE, nepoužívá balíčkovací systém postavený na RPM (RPM Package Manager, dříve Red Hat Package Manager), ale má vlastní balíčkovací systém tzv. deb-balíčků, který je velmi propracovaný a umožňuje velmi jednoduše provádět správu balíčků z různých zdrojů. Nazývá se Advanced Packaging Tool (APT).

Debian GNU/Linux 4.0 je zatím poslední stabilní verzí distribuce. Tato verze vyšla v dubnu roku 2007 a obsahuje zhruba 18040 balíků (předkompilovaného softwaru zabaleného do formátu umožňujícího snadnou instalaci na každém počítači) svobodného softwaru. Tento software je k dispozici pro všechny podporované architektury.

Debian se ve své společenské smlouvě zavázal, že bude založen na svobodném softwaru a sám vždy svobodným softwarem zůstane. To je velmi podstatný rys této distribuce. Znamená to významnou záruku pro její uživatele – mohou se pro Debian rozhodnout a začít na něm stavět s jistotou, že základní komponenta bude vždy uživateli poskytovat jeho svobody [22], [24].

5.1.3.2 Debian a serverové nasazení

Debian je pro svou stabilitu a poměrně jednoduchou údržbu velmi oblíbený zejména pro serverové instalace, naopak jeho podíl na desktopech v posledních letech poněkud poklesl, zejména po příchodu distribuce Ubuntu, která je ovšem na Debianu založena [20].

Obr. 17: Úvodní okno (plocha) operačního systému Debian GNU/Linux a tzv. Splash screen

Poznámka:

Už při samotné instalaci Debianu se instalační proces zabýval informacemi týkajícími se zadáním proxy ve standardním tvaru http://[[uživatel][:heslo]@]počítač[:port]/ *(Pokud pro přístup k okolnímu světu používáte HTTP proxy, zadejte zde potřebné informace. V opačném případě nic nevyplňujte.)*.

5.1.3.3 Konfigurace sítě

Jak je uvedeno v tabulce 1, nastavení sítě se u distribuce Debian provádí zpravidla v souborech */etc/hostname* a */etc/network/interfaces*, nápomocen je i soubor *options* v adresáři */etc/network*.

Soubor	Co se zde nastavuje
/etc/hostname	Název hostitele
/etc/network/interfaces	Adresa IP, síťová maska, implicitní brána
/etc/network/options	Nízkoúrovňová síťová nastavení (předávání datagramů IP atd.)

Tab. 1: Síťové konfigurační soubory pro Debian

Jméno hostitele se nastavuje v souboru */etc/hostname*. Jméno v tomto souboru by mělo být plně kvalifikované, protože jeho hodnota se používá i v jiných kontextech. Standardní instalace Debianu zde zanechává nekvalifikované jméno. Adresa IP, síťová maska a implicitní brána se nastavují v souboru */etc/network/interfaces* [1].

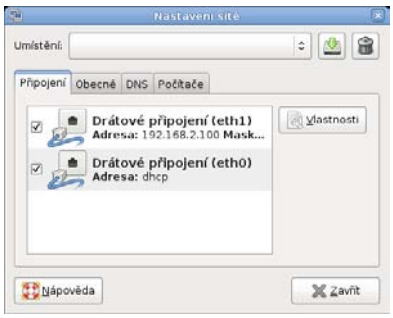
Například:

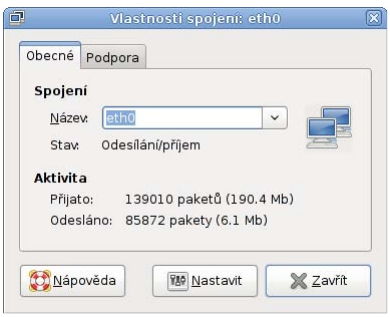
```
iface lo inet loopback
iface eth0 inet static
            address 192.168.1.100
            netmask 255.255.255.0
            gateway 192.168.1.1
```

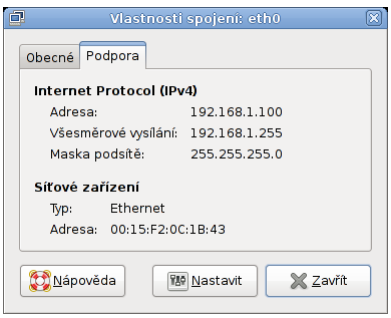
Příkazy *ifup* a *ifdown* čtou tento soubor a zapínají nebo vypínají rozhraní voláním nízkoúrovňových příkazů, jako je např. příkaz *ifconfig*, se správnými parametry. Základní formát souboru *interfaces* obsahuje řádek začínající slovem `iface` pro každé rozhraní, a pak následují odsazené řádky specifikující další parametry.

Klíčové slovo `inet` na řádku `iface` je rodina adres. Klíčovým slovem `static` se nazývá metoda, která udává, že pro rozhraní eth0 bude adresa IP a síťová maska přiřazena přímo. Řádky `address` a `netmask` jsou pro statické konfigurace povinné; starší verze Linuxu vyžadovaly i zadání síťové adresy (dnes už běžně probíhá výpočet síťové adresy z adresy IP a síťové masky). Řádek `gateway` udává adresu implicitní brány a používá se pro určení výchozího směru.

Soubor *options* dovoluje nastavení některých síťových proměnných v době spuštění systému. Debian nastavuje defaultně předávání adres na vypnuto, ochranu před předstíráním cizí identity na zapnuto a syn cookies na vypnuto [1].







5.1.3.4 DHCP server

Protokol DHCP (Dynamic Host Configuration Protocol) se používá ke konfiguraci klíčových síťových parametrů jednotlivých klientů. DHCP protokol umožňuje prostřednictvím jediného DHCP serveru nastavit všem stanicím sadu parametrů nutných pro komunikaci v sítích používajících rodinu protokolů TCP/IP včetně parametrů doplňujících a uživatelsky definovaných.

Významným způsobem tak zjednodušuje a centralizuje správu počítačové sítě (například při přidávání nových stanic, hromadné změně parametrů nebo pro skrytí technických detailů před uživateli). DHCP servery mohou být sdruženy do skupin, aby bylo přidělování adres odolné vůči výpadkům. Pokud klient některým parametrům nerozumí, ignoruje je.

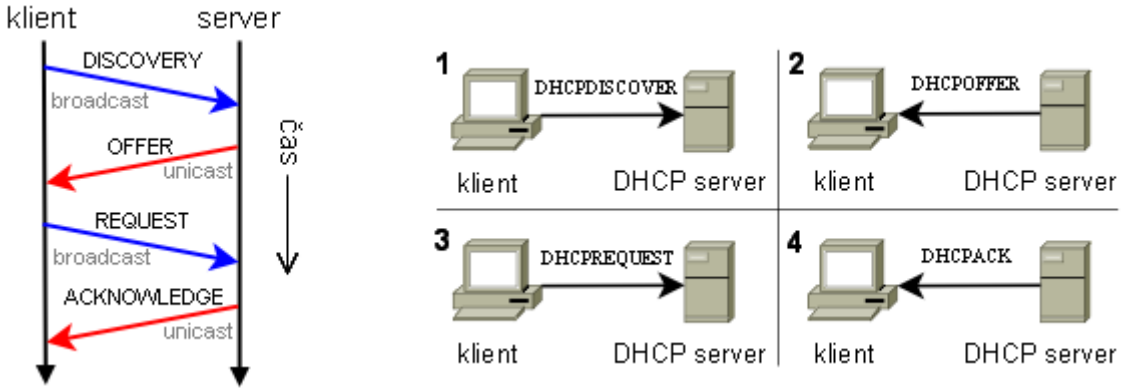
Protože se DHCP využívá hojně i v jiných systémech, pokusím se nakonfigurovat takový server pro spolupráci s okolím. Klient přihlášený do sítě tak automaticky získá potřebné parametry, jako je např. IP adresa zařízení, maska sítě, brána (případně DNS servery), z linuxového DHCP serveru stroje Debianu. Oproti „pevnému“ manuálnímu přidělování adres nabízí DHCP server několik výhod spočívajících ve výrazně nižší náročnosti administrace (vše stačí nastavit pouze na serveru, není třeba konfigurovat jednotlivé stanice) a eliminace mnoha potenciálních chyb (např. přidělení téže IP adresy dvěma různým stanicím, chybné nastavení výchozí brány na některé stanici apod.) [14], [28].

Princip činnosti

Klienti žádají DHCP server o IP adresu. DHCP server eviduje u každého klienta půjčenou IP adresu a čas, do kdy ji klient smí používat *(doba zapůjčení, ang. lease time)*. Poté co vyprší, smí server adresu přidělovat jiným klientům. Klient komunikuje na UDP portu 68, server naslouchá na UDP portu 67.

Po připojení do sítě klient vyšle broadcastem (všesměrově) DHCPDISCOVER paket. Na ten odpoví DHCP server paketem DHCPOFFER s nabídkou IP adresy. Klient si z (teoreticky několika) nabídek vybere jednu IP adresu a o tu požádá paketem DHCPREQUEST. Server mu ji vzápětí potvrdí odpovědí DHCPACK. Jakmile klient obdrží DHCPACK, může již IP adresu a zbylá nastavení používat.

Klient musí před uplynutím doby zapůjčení z DHCPACK obnovit svou IP adresu. Pokud lhůta uplyne, aniž by dostal nové potvrzení, musí klient IP adresu přestat používat [14].



Zobrazení zaznamenaného DHCP logu:

```
------------------------------------------------------------------------------------------------------------
Jan 16 11:05:09 debian syslogd 1.5.0#5: restart.
Jan 16 11:07:22 debian kernel: [796.646371] eth1: link up, 100Mbps, full-duplex, lpa 0xC1E1
Jan 16 11:07:22 debian kernel: [796.646870] ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
Jan 16 11:07:22 debian dhcpd: DHCPDISCOVER from 00:1b:24:0e:73:40 via eth1
Jan 16 11:07:23 debian dhcpd: DHCPOFFER on 192.168.2.1 to 00:1b:24:0e:73:40 (Zdenek-PC) via eth1
Jan 16 11:07:23 debian dhcpd: DHCPREQUEST for 192.168.2.1 (192.168.2.100) from 00:1b:24:0e:73:40 (Zdenek-PC)
via eth1
Jan 16 11:07:23 debian dhcpd: DHCPACK on 192.168.2.1 to 00:1b:24:0e:73:40 (Zdenek-PC) via eth1
------------------------------------------------------------------------------------------------------------
```

Server

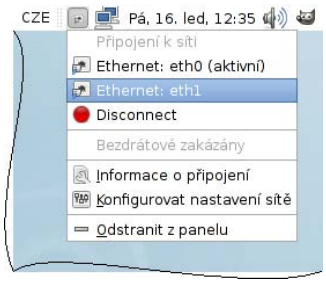
Nastavení konfiguračního souboru */etc/dhcp3/dhcpd.conf*:

```
subnet 192.168.2.0 netmask 255.255.255.0 {
    range 192.168.2.1 192.168.2.50;  //definuje rozsah přidělitelných adres
    default-lease-time 3600;   //implicitní doba platnosti IP [s]
    max-lease-time 7200;   //max. doba platnosti IP, 86400s = 24h
    option subnet-mask 255.255.255.0;   //definuje masku podsítě
    option broadcast-address 192.168.2.255;   //všesměrová adresa
    option routers 192.168.2.100;   //adresa směrovače (eth1 rozhraní)
    option domain-name-servers 212.24.128.8, 82.113.57.2;  //DNS servery
}
```

Ukázka konfigurace DHCP na Linuxu. Máme nastavený rozsah IP adres 192.168.2.1 až 192.168.2.50, kde bude náš DHCP server operovat. Mezi hlavní nastavení můžeme přidat jméno časového serveru, jméno DNS serveru či WINS serveru, atp.

Podrobnější informace o serverem DHCP přidělených IP adresách klientů lze zjistit v souboru */var/lib/dhcpd3/dhcpd.leases*.

```
lease 192.168.2.1 {
starts 3 2008/12/10 09:07:23;
ends 4 2008/12/11 09:07:23;
tstp 4 2008/12/11 09:07:23;
cltt 3 2008/12/10 09:07:23;
binding state active;
next binding state free;
hardware ethernet 00:1b:24:0e:73:40;
uid "\001\000\033$\016s@";
client-hostname "Zdenek-PC";
}
```

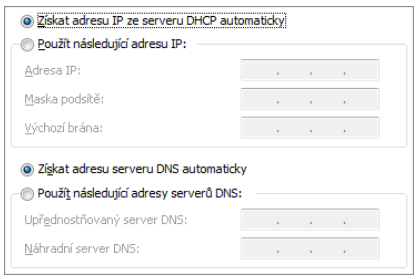


Nebo také pomocí příkazu *ARP*, který zobrazí aktuální ARP tabulku (IP a MAC adresy, síťová rozhraní). V našem případě 192.168.1.1 = IP adresa hraničního předřazeného směrovače (brány), 192.168.2.1 = IP adresa klientského počítače, jemuž byla tato adresa přidělena, resp. propůjčena DHCP serverem.

```
debian:/home/zdenek# arp -an
? (192.168.1.1) na 00:22:15:0b:66:db [ether] na eth0   //router
? (192.168.2.1) na 00:1b:24:0e:73:40 [ether] na eth1   //PC
```

Klient

Instalace DHCP klientů je na každém systému odlišná. V Linuxu stačí nainstalovat příslušný balíček DHCP Client Daemon a nakonfigurovat síť na DHCP. V operačním systému Windows je třeba otevřít okno Síťová připojení a v Konfigurace sítě („Protokol sítě Internet - TCP/IP“) taktéž nastavit získávání IP adresy z DHCP serveru automaticky. Po dalším restartu už bude síť aktivní [28].



Obr. 18: Nastavení klientských stanic na OS Windows (DHCP klient)

DNS

DHCP klient může (ale nemusí - tak se chová implicitně i `dhcpcd`) se svou zprávou zaslat svoje jméno a doménu. DHCP si tuto informaci uloží do *dhcpd.leases*. Tohoto se využívá ve spojení s DNS serverem. Není totiž problém získat z *dhcpd.leases* tyto informace a poslat DNS serveru zprávu, aby si upravil svoje tabulky. Pokud se tak bude dít pravidelně (například z cronu), každý počítač, jehož DHCP klient pošle serveru svoje jméno, dostane relevantní DNS záznam (včetně reverzního) [28].

5.1.3.5 Startovací skripty

Do startovacích skriptů se v rámci praktické části práce umístí skript firewall-iptables.sh, což zajistí automatickou aplikaci pravidel navrženého firewallu při (re)startu systému. Proto se zde krátce zmíním o informacích týkajících se startovních skriptů Debianu (spuštění daemonů a síťových služeb).

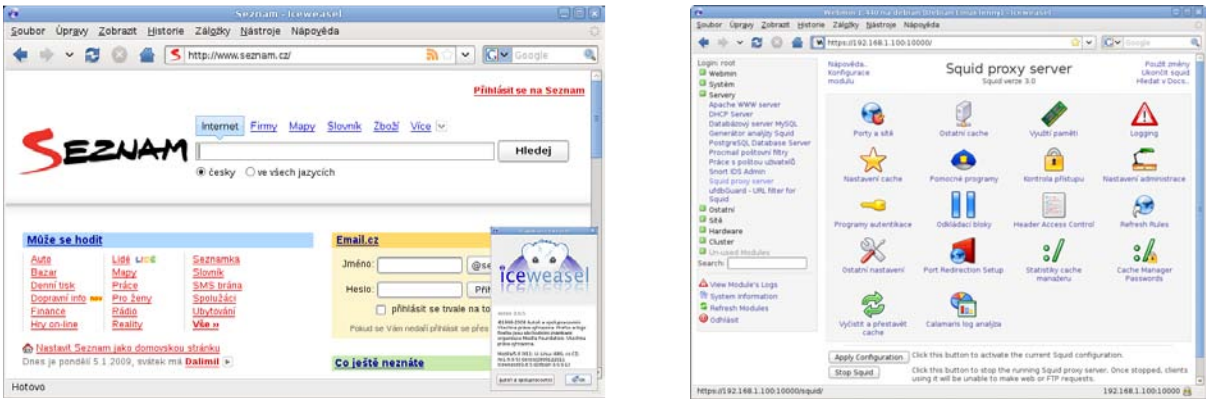
V každé úrovni běhu *init* spouští skript */etc/init.d/rc*, a jako argument předává novou úroveň. Každý skript odpovídá za nalezení svých vlastních konfiguračních informací, které mohou být ve formě jiných souborů v adresáři */etc*, v podadresáři uvnitř adresáře */etc* nebo někde ve skriptu samotném.

Když hledáme název hostitele, je uložen v souboru */etc/hostname*, který je načítán do skriptu */etc/init.d/hostname.sh*. Síťové rozhraní a parametry implicitní brány jsou uloženy v souboru */etc/network/interfaces*, které čte příkaz *ifup* volaný ze skriptu */etc/init.d/networking*. Některá nastavení sítě mohou být uložena i v souboru */etc/network/options* [1].

Více se této problematice věnovat nebudu, protože není hlavní náplní diplomové práce.

5.1.3.6 Webový prohlížeč a používané programy (aplikace, služby) Debianu

Jelikož se v práci často pracuje s webovým prohlížečem (např. v souvislosti s omezováním přístupu na vymezené internetové stránky – regulace zdrojů Internetu, nebo virová kontrola webového provozu tzv. „on-fly“), stručně představím implicitní prohlížeč Debianu, kterým je IceWeasel. Tento prohlížeč je odnoží webového prohlížeče Mozilla Firefox. Je součástí stabilní verze Debianu od verze 4.0 a snahou je synchronizace kódu s Firefoxem. Veškeré testování, ladění a zkoušení proxy serveru na stroji s Debianem probíhalo právě v tomto svobodném prohlížeči.



Obr. 19: Vlevo výchozí internetový prohlížeč Debianu GNU/Linux - IceWeasel, vpravo webové rozhraní Webmin se Squid modulem, který umožňuje jeho přehlednou konfiguraci a správu

O HTTP proxy Squid bude detailně pojednáváno v dalších kapitolách práce. Obrázky zachycující práci s webovým rozhraním Webmin jsou součástí přílohy CD diplomové práce. Vstup do tohoto rozhraní je umožněn prostřednictvím internetového prohlížeče zadáním adresy https://stroj-s-proxy:10000/.

5.1.4 Operační systém jako firewall

5.1.4.1 Linux a Iptables

Důležitým doplňkem jádra operačního systému Linux je funkce filtrování paketů a překládání adres (NAT) v samotném operačním systému. Tato funkce se původně nazývala IP Masquerade kvůli své schopnosti překládání adres. Nyní se označuje jako Ipchains nebo Iptables (v závislosti na použité verzi), protože umožňuje správci vytvořit řetězce nebo tabulky pravidel, které musí splňovat paket doručený do systému Linux, směrovaný v rámci počítače na jiný adaptér nebo odesílaný z počítače do jiné sítě [5]. Dnes se již v distribucích Linuxu setkáváme výhradně s Iptables.

Služba Iptables zajišťuje překládání adres (NAT) a filtrování paketů. Inspekci protokolu musí poskytovat služba na vyšší vrstvě. Squid je vynikajícím balíčkem serveru proxy (protokolu HTTP), který velmi dobře spolupracuje se službou Iptables systému Linux [37].

Hlavní funkce

Linux se službou (resp. nástrojem) Iptables poskytuje následující hlavní funkce [5]:

- Pro každý příchozí paket, paket procházející zásobníkem směrování systému Linux a odchozí paket jsou použita pravidla filtrování. Služba Ipchains je bezstavová, zatímco služba Iptables je stavová. Jedná se o hlavní funkční rozdíl mezi nimi.
- Lze vytvořit servery proxy pomocí filtrů obsahu specifických pro určité protokoly, které jsou poskytovány službami na vyšších vrstvách, jako např. Squid.
- Dynamické nebo statické překládání adres (NAT) zpracovává pakety procházející zásobníkem směrování do skrytých interních sítí.
- Zóny DMZ lze vytvořit filtrováním obsahu do externě viditelné chráněné podsítě, nebo přesměrováním virtuálních veřejných adres na chráněné hostitele s přeloženými adresami.
- Možnosti připojení v síti VPN z firewallu na firewall a z firewallu na vzdáleného klienta jsou k dispozici jako další komponenty systému Linux, které lze bezplatně získat z Internetu.
- Přesměrování portů je nativně poskytováno službou Iptables.
- Promyšleným použitím služby Iptables spolu s linuxovým balíčkem Squid můžeme získat transparentní servery proxy, což je výhodné především z hlediska velkého počtu stanic v síti.
- Linux s balíčkem FWTK také snadno zajišťuje služby reverzních serverů proxy.
- Další balíčky upravují běžný systém protokolování syslog v systému Linux, aby mohl ukládat informace protokolování do databází a poskytoval varování např. e-mailem.

Doplňkové funkce

Linux se službou (resp. nástrojem) Iptables poskytuje následující doplňkové funkce [5]:

- Filtrování paketů firewallem v systému Linux je rychlé, protože běžné počítačové procesory jsou mnohem rychlejší, než procesory používané ve většině vyhrazených firewallů. Dalším důvodem je, že systém Linux nemá zdaleka takovou režii sítě, jako běžnější operační systémy pro všeobecné použití. Díky integraci do zásobníku protokolu IP systému Linux není filtr paketů zatížen režií jiných firewallů, které jsou implementovány jako programy na uživatelské úrovni. Toto řešení dokáže snadno zvládat zatížené připojení sítě LAN k Internetu, i když je povolena možnost překládání adres (NAT).

- Konfigurace pomocí příkazového řádku vyžaduje více zkušeností se správou, ale umožňuje ukládat zásady do textových souborů a použít nástroje skriptování při dynamické správě zásad.
- Vzdálená správa (např. protokolem SSH) umožňuje spravovat firewall z jiných počítačů v síti LAN.
- Díky pravidlům filtrování paketů lze použít překládání adres (NAT) a přesměrování soketů, abychom přesměrovali provoz určený jistým službám (jako např. HTTP) na chráněné interní servery.

Zabezpečení

Systém Linux filtruje pakety před jejich doručením do zásobníku protokolu IP ke zpracování, což umožňuje chránit počítač před chybně formátovanými pakety a dalšími útoky na úrovni protokolu IP.

Protože systém Linux neanalyzuje datové části zpracovávaných paketů, je nutné pomocí serveru proxy zajistit, že provoz procházející určitým portem odpovídá příslušnému protokolu (například, že portem 80, prochází pouze požadavky a odpovědi protokolu HTTP) [5], [37].

Rozhraní

Filtrování paketů systému Linux lze spravovat příkazem `iptables`. Jako argumenty slouží pravidla filtru paketů, které chceme vytvořit nebo upravit. Ukázkový příklad syntaxe příkazu `iptables` pro příkazový řádek je uveden níže:

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

Uvedené dva příkazy zajistí, aby všechny pakety, které přijdou na síťové rozhraní eth0 z jiné zdrojové IP adresy než 192.168.0.1, byly zahozeny. Obdobně se sestavuje celá bezpečnostní politika firewallu.

5.1.5 Linux jako server v síti

5.1.5.1 Síťový server

Většinu operačních systémů (a Linux v tomto ohledu není výjimkou) lze použít mnoha různými způsoby. Linux jakožto univerzální operační systém lze použít na pracovní stanici, jako firewall a proxy server, jako router a mnoha dalšími způsoby. Linuxové systémy dnes patří mezi nejrozšířenější serverová řešení. Nabízí širokou škálu využití, a to jak pro vnitřní síť (Intranet), tak i pro vnější síť (Internet) [2].

5.1.5.2 Linux v roli serveru

Jako server je Linux již tradičně velmi silný - řada firem na Linux plně spoléhá při zpracování elektronické pošty, sdílení souborů a tiskáren, přiřazování IP adres a spousty dalších činnostech. Pro každý ze zmíněných úkolů (a nejen pro ně) Linux nabízí nepřeberné množství open-source programů. Než se ale rozhodneme Linux v úloze serveru nasadit, měli bychom mít jasno v jeho výhodách, nevýhodách, hardwarových a softwarových nárocích [17].

5.1.5.3 Vlastnosti Linuxu jako serveru

Linux lze jako server nasadit v mnoha rozličných konfiguracích. Hodí se výborně jako firewall, WWW server nebo též databázový server. Výhody Linuxu jako serveru v síti jsou následující - spolehlivost, cena, licencování, bezpečnost, poměrně rozsáhlá nabídka softwaru, správa prostředků, uživatelské úpravy a nároky na hardware. Většina zmíněných výhod se vztahuje na všechny operační systémy typu UNIX, tedy např. i na Solaris nebo FreeBSD. Ve srovnání s těmito systémy je největší výhodou Linuxu hardwarová přizpůsobivost, open source licence a nízké náklady (existují ovšem i další levné open source unixové systémy). Linux má přirozeně i své nevýhody, v úloze serveru jde naštěstí pouze o nevýhody drobného rázu - vyšší nároky na správu a potenciální bezpečnostní problémy (Linux je sice imunní vůči běžným červům a virům napadajícím Windows, má však svá vlastní bezpečnostní rizika. Do linuxových serverů se často snaží dostat útočníci a úspěšnost jejich průniků není nulová. Čistě teoreticky by mělo být prolomení ochrany dobře zabezpečeného systému obtížné, někdy ale stačí zapomenout na aktualizaci jednoho balíku a systém může být najednou zranitelný).

Sečteno a podtrženo výhody Linuxu značně převažují jeho nevýhody. Vysoká stabilita a spolehlivost systému, množství dostupných programů, které jsou pro roli serveru k dispozici a nízké pořizovací náklady jsou obrovské plus. Na druhou stranu lze konstatovat, že není v mnoha ohledech schopen kvalitativně nahradit některé funkce a vlastnosti ostatních produktů. Alespoň prozatím ne [17], [2].

5.1.6 Bezpečnost serveru v síti

5.1.6.1 Proč nás zajímá bezpečnost?

Ještě než začneme, musíme si uvědomit jednu věc. Od okamžiku, kdy připojíme počítač do sítě, je tento počítač vystaven eventuálním útokům zvenčí. Poněvadž počítač může být nejen bránou do naší sítě, ale i bránou do našeho soukromí, jistě budeme souhlasit s tím, že útokům všeobecně na počítače je třeba předcházet a jestliže je to možné, tak jim úplně zabránit.

Bohužel, účinná stoprocentní ochrana proti útočníkům neexistuje. Jelikož nechceme server úplně odpojit od Internetu, musíme se spokojit s reálnějšími metodami ochrany. Tyto metody si kladou za cíl co pokud možno nejvíce minimalizovat riziko útoku na server, jeho rychlou detekci v případě, že byl úspěšný a zejména rychlé a bezpečné zotavení se z jeho následků.

Klíčové slovo této kapitoly je bezpečnost. Bezpečnost jako taková se stanoví a řídí pomocí bezpečnostní politiky (*„secure policy“*) [19].

5.1.6.2 Bezpečnostní politika

Bezpečnostní politika je soubor pravidel, jejichž dodržování má zaručit počítačovou bezpečnost. Poněvadž bezpečnost je příliš důležitá na to, aby byla dobrovolná, musí být vynucována (*„enforced“*) operačním systémem nebo speciálními aplikacemi a zařízeními, které musí být navrhnuté tak, aby se nedali obejít (proxy servery, firewally, atd.).

K tomuto si dovolím poznamenat několik věcí. Každý informační systém by měl mít stanovenu jasnou a kompletní bezpečnostní politiku, která určuje, kdo má jaká práva. Počítače na Internetu (obecně v jakékoliv jiné/cizí síti) jsou též informačními systémy, jejichž přínosná užitková hodnota je různorodá. V případě připojení na Internet je vypracování vhodné bezpečnostní politiky téměř nutností, nedojde k možnému ohrožení privátních dat a citlivých informací [19].

Základní poučky pro tvorbu bezpečnostní politiky [19]:

1. To, co není explicitně povolené, musí být zakázané.
2. Nedávat uživatelům a aplikacím vyšší práva, než je nezbytně nutné pro jejich činnost.
3. Nikdy se nespoléhat na jediný ochranný prostředek. Například na omezení přístupu používat nejen TCP wrapper, ale zkombinovat ho i s firewallem.
4. Pravidelně aktualizovat software a zálohovat data.

Při tvorbě bezpečnostní politiky nezapomenout zohlednit i to, že nemalá většina útoků pochází z vnitřní sítě, tedy ze sítě, kterou obyčejně chráníme proti útokům zvenku. Jestliže je prevence založená pouze na ochraně „hranice“ mezi sítěmi, kterou obyčejně zabezpečuje firewall, ve skutečnosti jsou servery stejně zranitelné - z vnitřní strany lokální sítě.

Pravděpodobně každého napadlo nejjednodušší řešení vypnout všechny služby z venkovní i vnitřní sítě, s výjimkou DNS, WWW a elektronické pošty a zakázat či přiměřeně omezit jednotlivým uživatelům provádět nepovolené operace. Přesto nastává problém s potenciálními útoky na tyto běžící služby (útoky na DNS, přesměrování WWW) při nedodržení dalších bezpečnostních opatření. Navíc, proti některým útokům se brání velice obtížně právě proto, že jde o služby, které mají být (musí) přístupné všem, např. už zmíněné WWW, DNS a elektronická pošta [19].

Základní princip, který v administraci serverů vyznává určité nemalé procento správců je takový, že se snažíme omezit destruktivní potenciál uživatelů bez současného drastického omezení možností využívání služeb. Pokusíme se omezit to, co je (resp. eventuálně může být) opravdu nebezpečné. Protože se nikdy nedá dosáhnout stoprocentní bezpečnosti, při zabezpečení systému se snažíme o to, abychom případnému útočníkovi co nejvíce znesnadnili práci.

Bezpečnostní slabiny

Rozlišujeme následující pojmy [19]:

- ***zranitelné místo („vulnerability“)***: slabina v bezpečnostním systému (aplikaci, operačním systému, apod.), která může být využitá
- ***hrozba („threat“)***: okolnost, která má potenciál způsobit ztrátu nebo škodu
- ***útok („attack“)***: aktivita, která využívá zranitelné místo na průnik do systému
- ***ochrana („protection“)***: ochranné opatření, jehož úlohou je redukce slabin

Bezpečnost operačního systému

Do této kategorie bezpečnosti můžeme zahrnout ochranu na úrovni jádra operačního systému a kontrolu s použitím systémových aplikací.

Na začátek je třeba si uvědomit, že Linux, jako každý jiný operační systém, obsahuje chyby, které se dříve nebo později objeví. Linux se vyvíjí velmi rychlým tempem, takže téměř zároveň se zveřejněním chyby je již k dispozici i potřebná bezpečnostní záplata.

Každá chyba v jádře může mít katastrofální následky. Jádro totiž běží v privilegovaném režimu a nic na počítači nemá tak neomezená práva, jako právě jádro. V případě kritické chyby se lokální (a někdy i vzdálený) uživatel může stát rootem (superuživatelem). Naštěstí, tyto z bezpečnostního hlediska nebezpečné chyby se už dnes nevyskytují tak často [19].

Klíčové otázky:

- Je sledován vývoj verzí jádra a jeho aktualizace v případě, že byla odhalena závažná chyba?
- Je používán mechanismus zabezpečující vyšší ochranu na úrovni jádra operačního systému (většinou záplaty/„patche“ do jádra, např. Grsecurity, Medusa NG)?

Bezpečnost lokální sítě

V případě, že server funguje jako firewall/brána mezi Internetem a místní LAN sítí, nebo v případě plánu nasadit samostatný počítač jako firewall (doporučované), rozrůstá se bezpečnostní politika o další rozměr – neochraňujeme pouze jeden počítač, ale určitou skupinu počítačů (nacházejících se v této síti) s různými operačními systémy. Naší úlohou je zaručit relevantní bezpečnost pro lokální síť.

Klíčové otázky:

- Omezení přístupu – je kontrolován přístup dovnitř sítě? Je používán firewall? Ideální je využít samostatný segment sítě pro veřejně přístupné služby, tzv. Demilitarizovaná zóna (DMZ).
- Je používáno šifrování pro přenos důležitých a citlivých údajů a hesel po síti? (TLS/SSL, šifrování protokolů, které přenášejí přihlašovací jméno a heslo, např. POP3 a IMAP).
- Jakým způsobem přistupují uživatelé z vnitřní sítě k Internetu? Je aplikován proxy server?
- Je lokální síť chráněna před viry, spyware, spamem, apod. Je používán systém IDS?

5.2 Firewall

5.2.1 Paketové filtry na Linuxu

Paketový filtr je součástí Linuxu již od verze jádra 1.1 (1994). V těchto nejstarších verzích byl založen na *ipfw* převzatém ze systému BSD. Ve verzi 2.0 přibyl uživatelský nástroj *ipfwadm* pro nastavení filtrovacích pravidel. Ve verzi 2.2 byl rozšířen nástrojem *ipchains*. V dalších verzích jádra byl kód firewallu zcela přepsán a proto se ve verzích jádra 2.4 a novějších (včetně aktuální řady 2.6.X) pro nastavení filtrovacích pravidel používá nástroj *iptables*, který je v řadě dalším rozšířením *ipfwadm* a *ipchains* [30]. V současné době jsou známa první fakta o firewallu *nftables*, jakožto nástupci *iptables*.

5.2.2 Netfilter

Základem nové implementace pro jádra 2.4 a novější je firewallový subsystém netfilter. Je součástí jádra (možno i v modulech). Jedná se o firewall, který kromě základního paketového filtru podporuje také stavové filtrování a NAT. Čili s jeho pomocí můžeme provádět paketovou a stavovou inspekci, překlad adres, využít jej k implementaci transparentního proxy, TOS, aj. Pro jeho provoz potřebujeme mít v konfiguraci jádra zapnuto alespoň (v originálních jádrech distribucí toto již zapnuto je) [7], [8]:

- CONFIG_PACKET
- CONFIG_NETFILTER
- CONFIG_IP_NF_IPTABLES
- CONFIG_IP_NF_FILTER

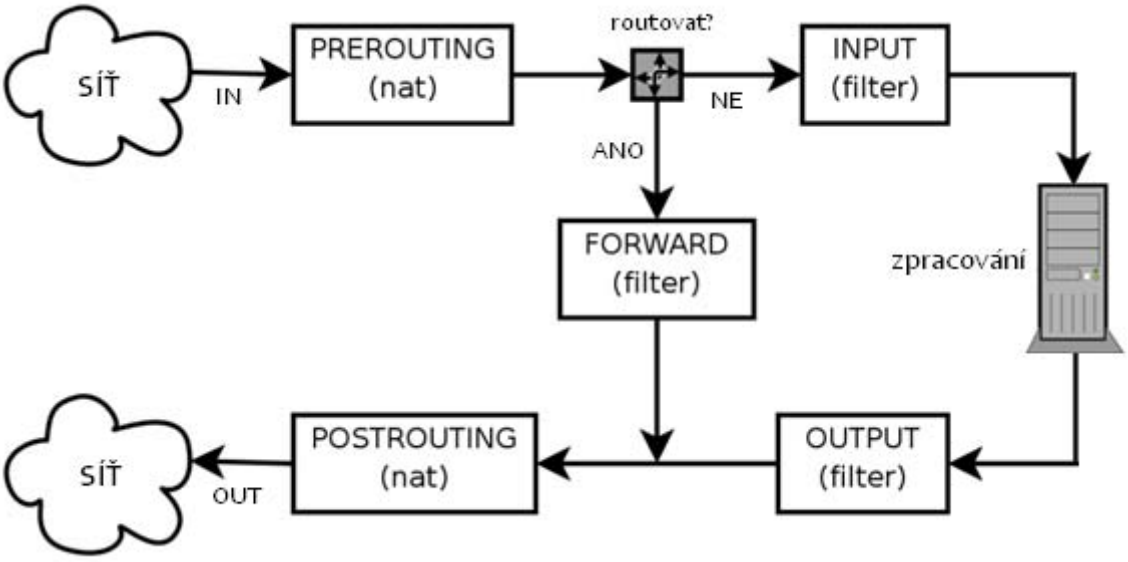
Pro využití všech možností netfilteru je nutné zapnout i další volby v Networking Options -> IP: Netfilter Configuration. Za zmínku stojí také přídavné moduly, např. pro FTP → ip_conntrack_ftp a ip_nat_ftp, které řeší problém pasivního spojení zajištěním tzv. port forwardingu pro každou relaci.

Důležité je povolit volbu IP Forward, která umožňuje směrovat pakety, které nejsou určeny pro místní procesy (čili mezi dvěma síťovými rozhraními). Nastavuje se pomocí parametru jádra (viz dále).

K vytvoření linuxového firewallu typu netfilter v operačním systému Linux je potřeba [19]:

1. Podpora routování v jádře: Networking options -> TCP/IP networking -> IP: advanced router
2. Při startu systému je třeba routování paketů povolit příkazem: `echo "1" > /proc/sys/net/ipv4/ip_forward`
3. Podpora firewallu v jádře: Networking options -> IP: Netfilter Configuration -> vybrat všechny potřebné možnosti, základem jsou:
   - Connection tracking *(požadováno pro masq/NAT)*
   - IP tables support *(požadováno pro filtering/masq/NAT)*
   - Connection state match support
   - Packet filtering
   - REJECT target support
   - Full NAT *(v případě, že chceme překládat adresy)*
   - MASQUERADE target support *(jestliže chceme překlad adres pomocí MASQUERADE - pro servery s pevnou IP adresou postačí NAT)*
   - REDIRECT target support *(jestliže chceme přesměrování služeb, např. pro účely transparentního proxy serveru)*
   - LOG target support
4. Nástroj "`iptables`" (vytváří rozhraní pro práci s firewallem)
5. Čas na návrh, implementaci a otestování firewallu. To souvisí s vytvořením bezpečnostní politiky.

Pokud si při kompilaci jádra nejsme jistí, co budeme potřebovat, nemusíme zbytečně kompilovat vše (včetně experimentálních modulů), typicky postačí běžné jádro. Pokud budeme potřebovat něco navíc, tak využijeme moduly, nebo přikompilujeme to, co nám prozatím chybí. Jestliže vytváříme modulární jádro a komponenty netfilteru zkompilujeme jako moduly, zavedou se do paměti, jen když to bude potřebné. V případě monolitického jádra se jeho velikost mírně zvětší.



Obr. 20: Struktura netfilteru - tabulky a řetězce

Řetězce

Netfilter má 5 základních řetězců pro filtrování a případnou další úpravu paketů (NAT). Lze nadefinovat i další vlastní a do nich přesměrovat pakety (vhodné pro přehlednost a jednoduchost pravidel, resp. zpřehlednění funkce firewallu). Základní řetězce není možné smazat.

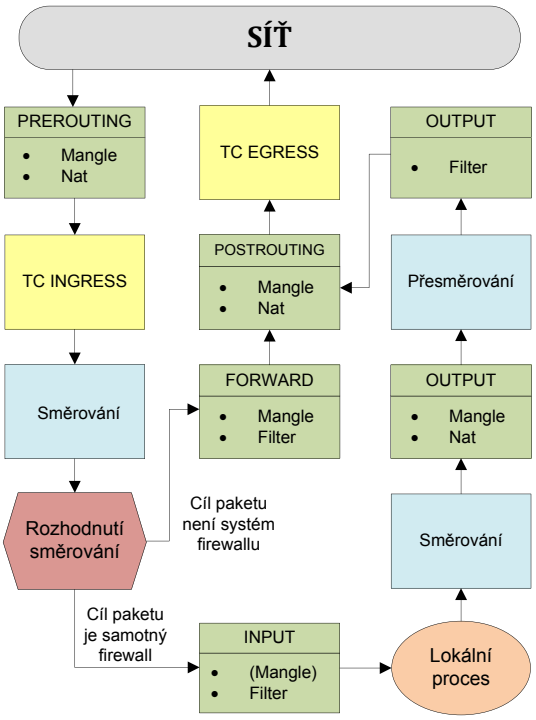
5 základních řetězců:

- PREROUTING
- INPUT
- FORWARD
- OUTPUT
- POSTROUTING

Systém netfilteru se skládá z několika tabulek („*tables*“) obsahujících tyto řetězce pravidel („*chains*“).

Tabulky a řetězce

- ***filter*** - používá se na filtrování všeobecně, tedy na akce typu ACCEPT, REJECT, DROP apod. V této tabulce by nemělo docházet k úpravě paketů, provádí se zde pouze filtrování paketů v řetězcích.
  - *INPUT* (pakety určené pro lokální aplikace),
  - *FORWARD* (pakety negenerované lokálně a neurčené pro localhost),
  - *OUTPUT* (lokálně vygenerované paketu).
- ***nat*** - v této tabulce se provádí překládání adres (NAT); prochází jí pouze první paket každého spojení, ve kterém se provádí překlad adres. Výsledek průchodu paketu tabulkou je potom aplikován na všechny ostatní pakety spojení.
  NAT (Network Address Translation) lze rozdělit na *Source NAT* (překlad zdrojových adres) a *Destination NAT* (překlad cílových adres); maškaráda (*masquerading*) je speciální případ SNAT; překládání portů (*port forwarding*) a transparentní proxy jsou speciální případy DNAT. NAT se provádí v řetězcích:
  - *PREROUTING* (pakety přicházející ze zařízení),
  - *POSTROUTING* (pakety odcházející na zařízení).
- ***mangle*** - tabulka opět pro úpravu paketů - změny v hlavičkách (*TTL, TOS a další*), ale nikoliv NAT a „maškarádování“. Tyká se zejména řetězců *PREROUTING* a *OUTPUT* [13].



Obr. 21: Netfilter - „tok“ paketů

5.2.3 Iptables

Jak funguje firewall Iptables

Řízení toku paketů probíhá na síťové vrstvě na úrovni jádra. V jádru jsou uložené tabulky *filter, nat, mangle* a *raw*. Každá tabulka obsahuje řetězce pravidel, které jsou buď předdefinované, anebo uživatelsky definované. Předdefinované řetězce má každá tabulka svoje. Kromě nich si může uživatel definovat svoje vlastní řetězce pravidel, které může volat z předdefinovaných, podobně jako subrutinu v programu. Každý řetězec obsahuje seznam pravidel, která určují, jaké akce budou s pakety prováděny.

Konfigurace Iptables

Netfilter/Iptables nemají žádný konfigurační soubor v tradičním smyslu. Na konfiguraci slouží program iptables, pomocí kterého se manipuluje s pravidly jednotlivých tabulek. Firewall je tedy ovládán prostřednictvím programu iptables. Při používání programu iptables se jako parametr uvádí tabulka a řetězec pravidel, se kterými chceme pracovat, typ operace a specifikace pravidla [8].

Základní parametry pro manipulaci s řetězci	
-L	vypíše pravidla definovaná na řetězci
-F	smaže pravidla definovaná na řetězci
-Z	reset počítadel v řetězci
-N	vytvoří nový řetězec
-X	zruší řetězec (kromě tří standardních)
-P	nastaví policy pravidlo (jen u standardních řetězců)
-E	přejmenování uživatelského řetězce

Základní parametry pro manipulaci s pravidly v řetězcích	
-A	vytvoří nové pravidlo (za stávající)
-D	odstraní pravidlo (podle pořadí)
-C	změní stávající pravidlo
-R	nahradí pravidlo novým
-I	vloží nové pravidlo (mezi stávající)
-D	odstraní pravidlo (první, které odpovídá zadání)

Základní parametry pravidel, které určují podmínky pro pakety	
-s	zdrojová adresa paketu
-d	cílová adresa paketu
-p	protokol (následuje název protokolu, např. TCP, UDP, ICMP)
-i	vstupní zařízení (např. eth0)
-o	výstupní zařízení
--sport	zdrojový port (číslo nebo název)
--dport	cílový port
--state	rozdělení dle stavové informace
-f	pravidlo se týká pouze fragmentů paketů
-j	určuje, co je provedeno při splnění pravidla

Poslední, ale velmi důležitou částí pravidla je určení cílového kroku, který následuje po konečném rozhodnutí. Tento krok je určen již zmíněným parametrem -j a může být vybrán například z následujících možností:

- *DROP* … paket je tiše zahozen
- *REJECT* … paket je zahozen a jeho odesilatel je o tom informován chybovým hlášením
- *ACCEPT* … paket volně prochází/pokračuje přes *iptables*
- *LOG* … existence paketu je zaznamenána daemonem *syslogd*, toto pravidlo není definitivní a paket pokračuje v cestě řetězcem

Akcí je samozřejmě více, tyto jsou však zdaleka nejpoužívanější (podobně platí i u výše uvedených tabulek). Více informací - viz manuálové stránky Iptables.

Stavové filtrování

Jak již bylo řečeno, při filtrování dokáže netfilter zohledňovat také stavové informace. Rozlišujeme tyto stavy spojení (nutno informace udržovat):

- *ESTABLISHED* - paket je součástí již existujícího spojení (ve kterém proběhly pakety oběma směry)
- *NEW* - jde o paket, který otvírá nové spojení (nebo je součástí dosud jednostranného spojení)
- *RELATED* - paket, který se vztahuje k nějakému existujícímu spojení, ale není přímo jeho součástí (např. ICMP chybová zpráva)
- *INVALID* - paket který nelze přiřadit žádnému existujícímu spojení (neplatné spojení)

Konfigurace firewallu

Zápis probíhá ve formě:

```
iptables tabulka jméno_řetězce pravidlo_firewallu
```

Logování

Logování se provádí prostřednictvím daemona `syslogd` do souboru /var/log/messages (v některých případech je důležitý také hlavní systémový log /var/log/syslog).

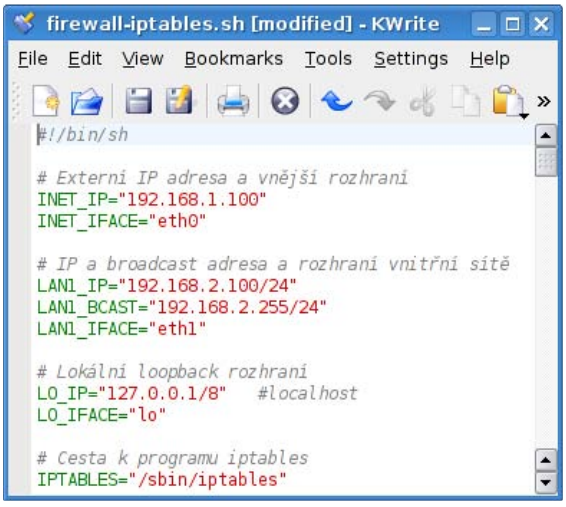
Rozšíření iptables

Target Extensions - rozšířené cíle
- *pomocí přepínače -j*

- LOG
- DNAT, SNAT
- REDIRECT
- TTL

Match Extensions – rozlišení paketů jinak, než dle IP
- *pomocí přepínače -m*

- state
- iprange, multiport
- mac, physdev
- limit
- time, account



Záplavovým DoS zprávám se lze většinou bránit na firewallu, v kombinaci se syncookies.

Ochrana firewallem *//pozn. DoS (Denial of Service) = odmítnutí, odepření síťových služeb*
iptables -N syn flood
iptables -A INPUT -i eth0 -p tcp --syn -j syn flood
iptables -A syn flood -m limit --limit 1/s --limit-burst 5 -j RETURN
iptables -A syn flood -m limit --limit 1/s --limit-burst 5 -j LOG --log-prefix="SYN flood"
iptables -A syn flood -j DROP

5.2.3.1 Analýza a statistika záznamových souborů síťového firewallu Iptables

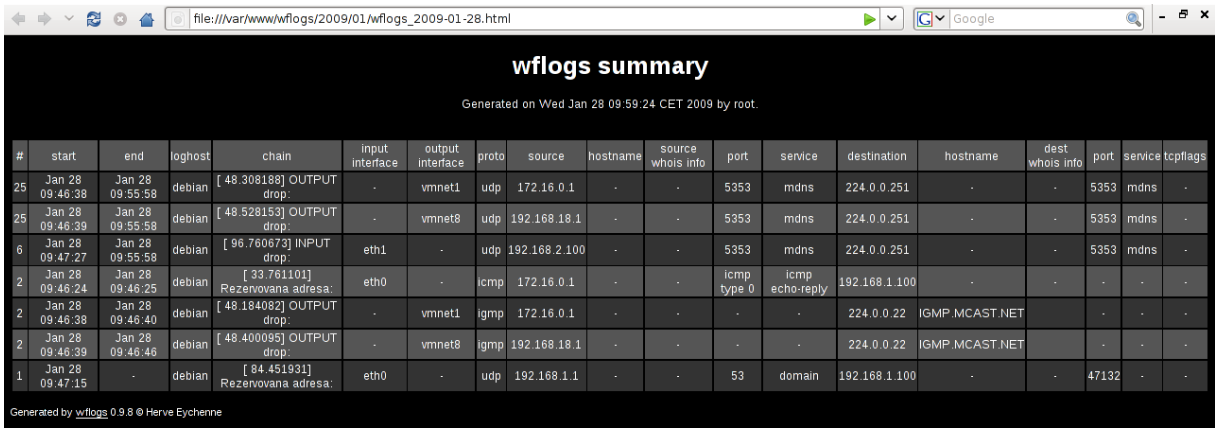
Existují tucty různých řešení síťových firewallů a pro každé řešení existuje unikátní formát záznamových souborů. Některá taková zařízení zaznamenávají pouze málo informací o přenosech - přibližně ty samé informace jako většina směrovačů. Jiné firewally jsou schopné zaznamenat i spoustu detailů o přenosech, které monitorují. Hloubka naší analýzy incidentů a podezřelé aktivity bude velice záležet na rozmanitosti v množství zaznamenaných informací [3].

Iptables zaznamenává události mnohem komplexněji, než většina ostatních typů firewallů.

```
Dec  1 14:04:12 debian kernel: [6111.337472] INPUT drop: IN=eth1 OUT= MAC=
SRC=192.168.2.100 DST=224.0.0.251 LEN=260 TOS=0x00 PREC=0x00 TTL=255 ID=0
DF PROTO=UDP SPT=5353 DPT=5353 LEN=240
```

Zde je stručné vysvětlení, jaký význam mají jednotlivá políčka této položky záznamového souboru: datum, čas, jméno hostitele firewallu Iptables, úroveň syslog, příchozí a odchozí rozhraní, MAC adresy, zdrojová IP adresa, cílová IP adresa, délka paketu (LEN), typ služby (TOS), priorita typu služby (PREC), doba života (time-to-live, TTL), IP identifikační číslo (ID), IP protokol, zdrojový port, cílový port a délka užitečného obsahu. Příznak DF (Don't fragment) značí zákaz fragmentace (DF = 1 nebo 0).

Iptables zaznamenává také několik dalších informací, které jsou užitečné při provádění analýzy záznamových souborů. Tyto informace mohou být hodnotné v případech, když určujeme povahu útoku stejně tak jako vlastnosti a záměry potenciálního útočníka (ovšem v případě velmi dobrého obeznámení s detekcí síťového vniknutí).



#	start	end	loghost	chain	input interface	output interface	proto	source	hostname	source whois info	port	service	destination	hostname	dest whois info	port	service	tcpflags
25	Jan 28 09:46:38	Jan 28 09:55:58	debian	[48.308188] OUTPUT drop:	-	vmnet1	udp	172.16.0.1	-	-	5353	mdns	224.0.0.251	-	-	5353	mdns	-
25	Jan 28 09:46:39	Jan 28 09:55:58	debian	[48.528153] OUTPUT drop:	-	vmnet8	udp	192.168.18.1	-	-	5353	mdns	224.0.0.251	-	-	5353	mdns	-
6	Jan 28 09:47:27	Jan 28 09:55:58	debian	[96.760673] INPUT drop:	eth1	-	udp	192.168.2.100	-	-	5353	mdns	224.0.0.251	-	-	5353	mdns	-
2	Jan 28 09:46:24	Jan 28 09:46:25	debian	[33.761101] Rezervovana adresa:	eth0	-	icmp	172.16.0.1	-	-	icmp type 0	icmp echo-reply	192.168.1.100	-	-	-	-	-
2	Jan 28 09:46:38	Jan 28 09:46:40	debian	[48.184082] OUTPUT drop:	-	vmnet1	igmp	172.16.0.1	-	-	-	-	224.0.0.22	IGMP.MCAST.NET		-	-	-
2	Jan 28 09:46:39	Jan 28 09:46:46	debian	[48.400095] OUTPUT drop:	-	vmnet8	igmp	192.168.18.1	-	-	-	-	224.0.0.22	IGMP.MCAST.NET		-	-	-
1	Jan 28 09:47:15	-	debian	[84.451931] Rezervovana adresa:	eth0	-	udp	192.168.1.1	-	-	53	domain	192.168.1.100	-	-	47132	-	-

Wflogs je nástroj k analýze logů firewallů. Může být použit k tvorbě souhrnných reportů, a to v otevřeném textu, HTML a XML, nebo k monitorování firewall-logů v reálném čase. Tento nástroj je součástí tzv. WallFire projektu, ale může být využíván nezávisle. WallFire je určený k práci na reálných systémech jako jsou Unix, Linux a *BSD, od toho se samozřejmě odvíjí podporované systémy. Současnými vstupními moduly wflogs jsou netfilter, ipchains, ipfilter, cisco_pix, cisco_ios a snort. Příklad použití, kdy se překonvertuje patřičný logovací soubor netfilteru do HTML výstupu, je dále.

```
# wflogs -i netfilter -o html netfilter.log > /var/www/wflogs/logs.html
```

5.2.3.2 Automatický aplikace pravidel firewallu iptables

Jestliže si manuálně anebo nějakým skriptem nastavíme potřebná pravidla firewallu, je třeba zabezpečit automatické spuštění firewallu při startu/restartu operačního systému a přímo tak vynutit aplikaci pravidel filtrování. Na Debianu je podporovaný následující způsob [12], [27].

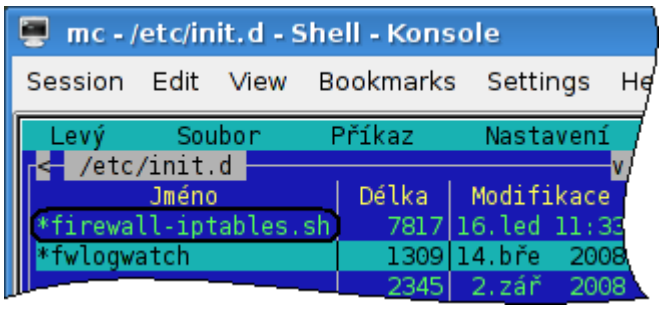
Jako první provedeme linuxový skript spustitelným (vlastník může soubor spouštět, zapisovat a číst):
#chmod 700 firewall-iptables.sh Pozn. *skript firewallu je vhodné pouštět ihned po spuštění síť. rozhraní*

Pro takzvané „Debian-based“ distribuce (distribuce založené na Debianu) platí:

- při startu systému používáme vlastní skript
- zkopírujeme jej tedy do adresáře /etc/init.d/
- spustíme následující příkaz

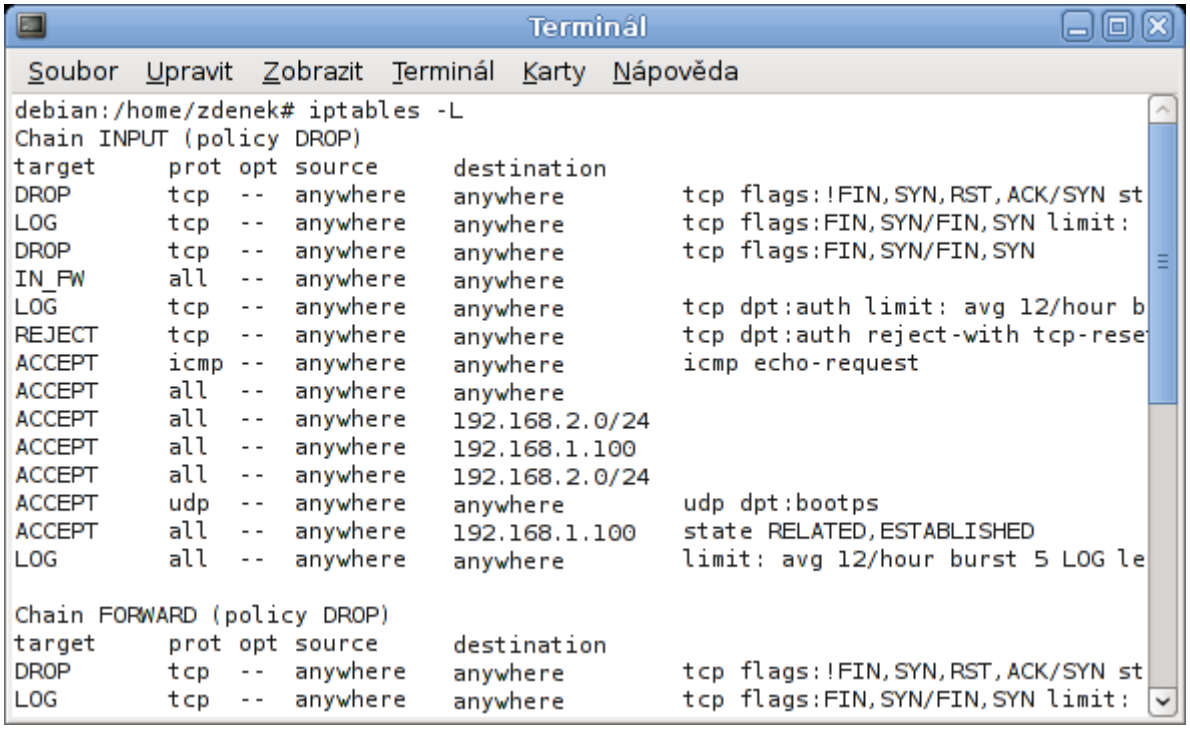
#update-rc.d firewall-iptables.sh defaults 19

nebo v libovolném souborovém manažeru (mc) vytvoříme symlink (symbolický odkaz) na skript

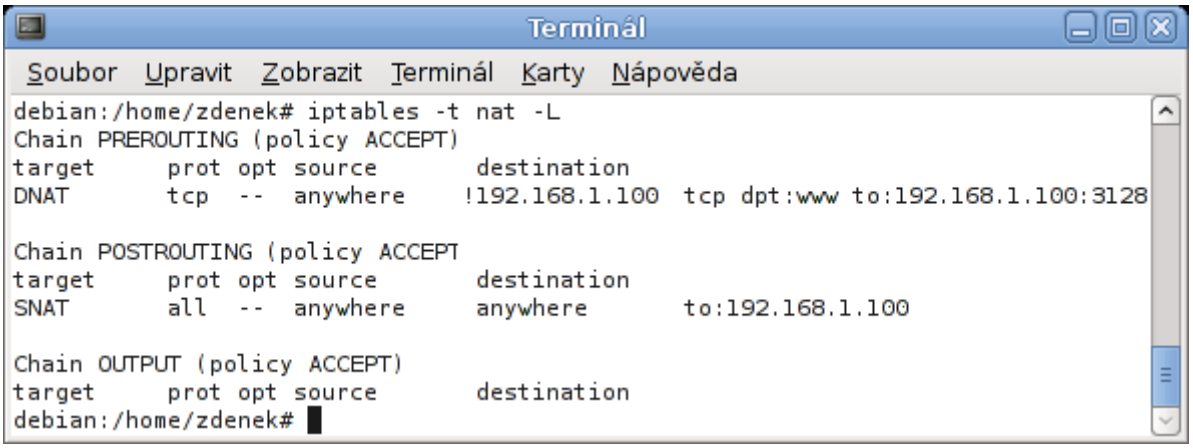



```
Adding system startup for /etc/init.d/firewall-iptables.sh ...
/etc/rc0.d/K19firewall-iptables.sh -> ../init.d/firewall-iptables.sh
/etc/rc1.d/K19firewall-iptables.sh -> ../init.d/firewall-iptables.sh
/etc/rc6.d/K19firewall-iptables.sh -> ../init.d/firewall-iptables.sh
/etc/rc2.d/S19firewall-iptables.sh -> ../init.d/firewall-iptables.sh
/etc/rc3.d/S19firewall-iptables.sh -> ../init.d/firewall-iptables.sh
/etc/rc4.d/S19firewall-iptables.sh -> ../init.d/firewall-iptables.sh
/etc/rc5.d/S19firewall-iptables.sh -> ../init.d/firewall-iptables.sh
```

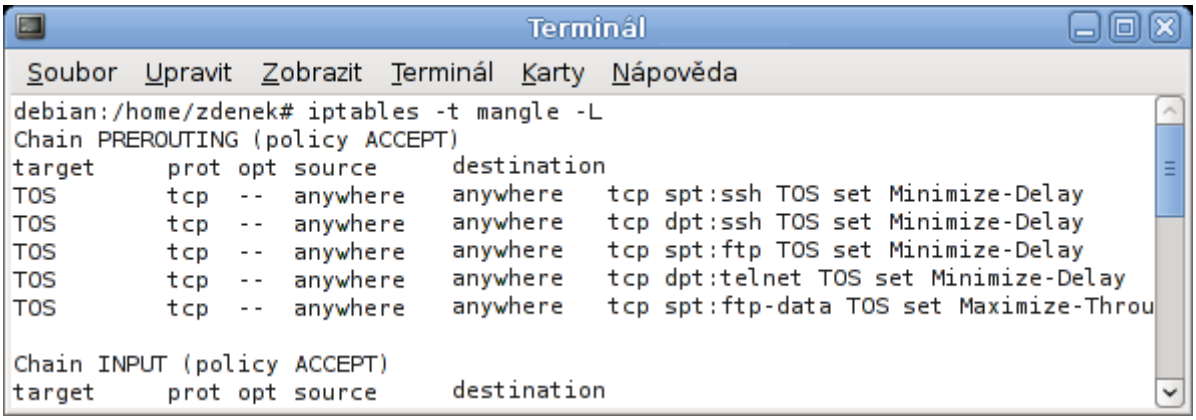
Tímto příkazem jsme učinili skript firewall-iptables.sh startovacím skriptem (tj. je zajištěno jeho automatické spuštění). Kontrolu nastavených pravidel v jednotlivých řetězcích výchozích tabulek filter, nat a mangle provedeme v konzole příkazového řádku následovně -viz obrázky (tato aplikovaná pravidla je nutné vyzkoušet a otestovat také v praxi, aby se ověřilo, zda vše funguje, jak má!):



Obr. 22: Zobrazení tabulky ***filter*** *(filtrování paketů)* s vloženými řetězci pravidel



Obr. 23: Zobrazení tabulky ***nat*** *(přepis zdroj./cíl. adresy)* s vloženými řetězci pravidel



Obr. 24: Zobrazení tabulky ***mangle*** *(modifikace paketů)* s vloženými řetězci pravidel

Z výše uvedeného je zřejmé, že tabulky nezůstaly prázdné ani po restartu OS. Jsou naplněny příslušnými pravidly definovanými při tvorbě skriptu pro automatické spouštění. Ruční spuštění skriptu, tzn. manuální uplatnění pravidel firewallu je možné příkazem #sh /etc/init.d/firewall-iptables.sh. Skript neběží jako daemon → nelze restartovat, úprava za běhu je možná pomocí příkazů.

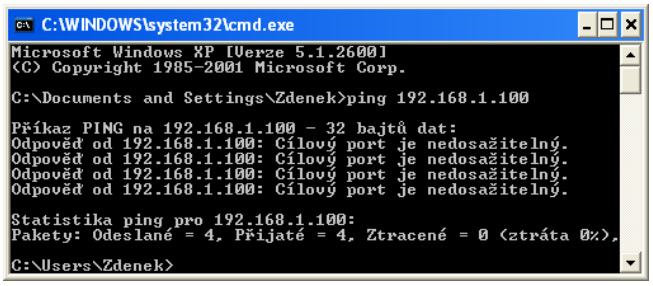
5.2.3.3 TCP Wrappers - omezení přístupu k síťovým službám

Nejjednodušší způsob zabezpečení počítače je nespouštět na něm žádné služby, které by reagovaly na síťový provoz. To však není u serveru možné, protože potřebujeme obsluhovat nejrůznější služby. Kompromisním řešením je omezit spolupráci serveru ke komunikaci s klienty na nejnižší možnou míru. Zřejmě nejsnadnější metoda omezení příjmu síťových spojení je TCP wrapper (balíček *tcp_wrappers*). Je to knihovna, kterou lze snadno přilinkovat k prakticky jakémukoliv programu. Její funkce spočívá v tom, že veškerá příchozí TCP spojení jsou nejprve podrobena kontrole a teprve pak předána k obsloužení samotnému programu. Kontroly, které povolí nebo odmítnou příchozí spojení, jsou pro všechny programy, které TCP wrapper používají, soustředěny ve dvou souborech: `/etc/hosts.allow` a `/etc/hosts.deny`. Kupříkladu lze zakázat přístup všem strojům, jejichž jméno neodpovídá jejich IP adrese (tzv. paranoidní mód). V současné době jsou s touto knihovnou slinkovány téměř všechny programy, které přijímají TCP spojení (dokonce i NFS démoni pracující s UDP). Přesto existují výjimky - např. WWW server Apache, u něhož omezení přístupu nastavujeme v jeho vlastním konfiguračním souboru.

Tcp_wrapper pracuje se dvěma konfiguračními soubory:
/etc/hosts.deny – zakázaný přístup (obsahuje seznamy strojů, které k daným službám přístup nemají)
/etc/hosts.allow – povolený přístup (definuje seznamy strojů, ze kterých je přístup k uvedeným službám povolen)

Firewall spoluprácí s TCP Wrappers umožňuje vhodnou obranu např. proti útokům hrubou silou, kdy se útočník snaží získat přístup do systému (prolomit heslo) generováním náhodných sekvencí - slov, slovních spojení, číselných kombinací, atp., a vzniká přitom velké množství neúspěšných spojení směrovaných na určitou službu daného systému. Tato podezřelá síťová aktivita je firewallem detekována (postup stále opakujeme, jen zkoušíme jiná hesla => náhlé zvýšení zátěže, alokace prostředků systému; ovšem požadavky přicházejí z jedné IP adresy stroje, popřípadě několika málo IP adres, pokud se jedná o distribuovaný útok). Firewall dynamicky upraví soubor /etc/hosts.deny přidáním zdrojové IP adresy potenciálního útočníka (zjištěné z hromadně přicházejících paketů) a následně „odstřihne“ vzdáleného uživatele, který se již nikdy nepřipojí. Stanovením vhodného limitu počtu spojení za sekundu (popř. minutu, hodinu) se lze takovému útoku efektivně bránit, ovšem tak, aby nedošlo k omezení právoplatných uživatelů odepřením služby (v případě nedostačujících limitních hodnot). V linuxovém prostředí je osvědčena spolupráce firewallu Iptables s TCP Wrappers.

Port může být otevřený, což znamená, že na něm běží nějaká služba, nebo zavřený, pokud na něm nic neběží. Je možno rozpoznat ještě jeden stav, a to je filtrovaný, kdy je port chráněn firewallem. Při pokusu o připojení na port TCP je zpět odeslán paket s příznaky RST a ACK. V případě, že se jedná o UDP porty, je zpět poslán ICMP paket (zpráva) typu 3, z čehož vyplývá, že port je nedosažitelný. ICMP zprávy jsou vkládány do datové části IP paketů (za 20B hlavičku), číslo protokolu vyšší vrstvy je 1.



Obr. 25: Ukázka portu chráněného firewallem Iptables

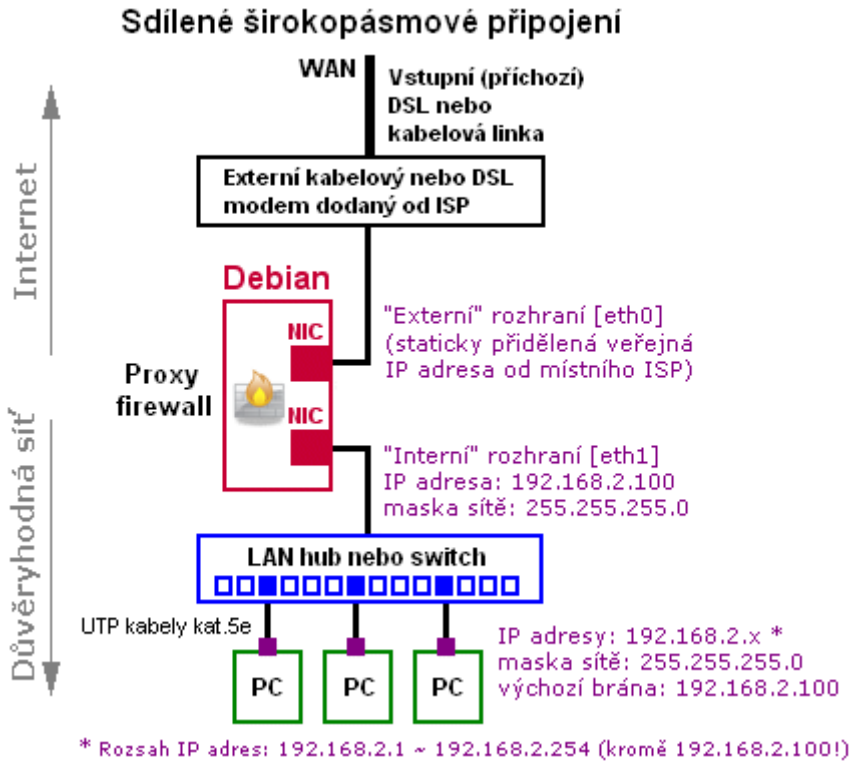
5.3 Proxy server

5.3.1 Squid HTTP Proxy 3.0

Dříve jmenované tři možnosti využití proxy serveru si představíme na kvalitním open-source proxy serveru pro cachování obsahu WWW stránek. Jeho jméno je Squid a pro své možnosti využití je velmi dobře znám nemalé linuxové komunitě.

Squid umožňuje klientům pracovat s protokoly HTTP, HTTPS, FTP a podporuje i několik dalších, méně používaných aplikačních protokolů.

Před instalací Squidu je třeba si uvědomit, že převážná část činnosti proxy serveru spočívá v uchovávání objektů v paměti a na disku. Čím více budeme mít paměti (doporučeno nejméně 128 - 256 MB), tím rychlejší bude proxy server. Jestliže se údaje budou uchovávat na pevném disku, jeho rychlost rapidně ovlivní výkon samotného proxy serveru. Z toho plyne použití velmi rychlých disků se sběrnicí SAS (serverové), případně SATA disků nejlépe zapojených do tzv. RAID pole 0 (stripping) [19].



Obr. 26: Schematické (reálné) zapojení proxy serveru Squid na stroji Debianu v síti

5.3.1.1 Kompilace a instalace Squidu

Pro kompilaci ze zdrojových souborů vystačíme s:

./configure --enable-linux-netfilter
make all
make install

Kompilace Squidu s parametrem „--enable-linux-netfilter“ => kernel je kompilován s podporou transparentního proxy.

5.3.1.2 Instalace adresářové struktury cache

Po nainstalování Squidu je třeba vytvořit adresářovou strukturu, ve které bude uchovávat obsah svojí cache paměti (cachované soubory). Některé linuxové distribuce vytvoří tuto strukturu při instalaci nebo prvním spuštění. V některých případech se musí tato struktura vytvořit ručně (např. při kompilaci Squidu ze zdrojových souborů):

```
/usr/sbin/squid3 -z
```

Výsledkem je vytvoření adresářové struktury v adresáři uvedeném v konfiguračním souboru "`/etc/squid3/squid.conf`", standardně je to "`/var/spool/squid3`" [26].

5.3.1.3 Spouštění a zastavování Squidu

Program Squid se spouští automaticky při startu systému v runleveli 3 a 5, protože nemá význam jej pouštět manuálně. Co však význam určitě má, je jeho ovládání z příkazového řádku v případě údržby, upgrade, změny konfigurace a podobně [26].

V případě použití „init“ skriptů jsou všechny operace velmi jednoduché:

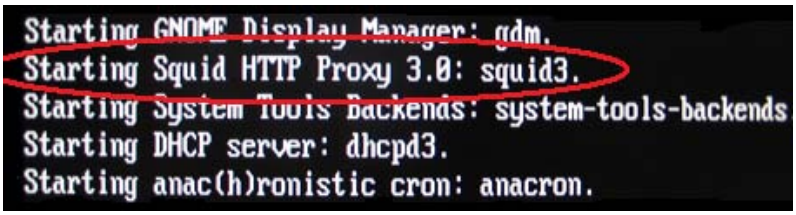
- Spuštění Squidu: `/etc/init.d/squid3 start`
- Zastavení Squidu: `/etc/init.d/squid3 stop`
- Restartování Squidu: `/etc/init.d/squid3 restart`
- Opětovné načtení konfiguračních souborů: `/etc/init.d/squid3 reload`

Poznámka: „init“ skripty jsou jen rozhraním k ovládání Squidu pomocí příkazového řádku konzole.

```
debian:/home/zdenek# /etc/init.d/squid3 restart
Restarting Squid HTTP Proxy 3.0: squid3 Waiting.....................done.
```

Jestliže „init“ skripty použity nejsou, operace jsou částečně odlišné (cesta závisí na instalaci Squidu):

- Spuštění Squidu: `/usr/sbin/squid3`
- Zastavení Squidu: `/usr/sbin/squid3 -k shutdown`
- Opětovné načtení konfiguračních souborů: `/usr/sbin/squid3 -k reconfigure`



Obr. 27: Spuštění služby Squid HTTP proxy 3.0 při startu systému

5.3.1.4 Konfigurační soubor "/etc/squid3/squid.conf"

Konfigurační soubor obsahuje direktivy pro proxy server Squid. Každý řádek má tvar:

```
direktiva hodnota1 [hodnota2 hodnota3 … hodnotaN]
```

Mnoho direktiv má nastavené implicitní hodnoty, takže je není nutno ani odkomentovat (o tom vždy informuje komentář v příslušné části konfiguračního souboru `"squid.conf"`).

Soubor je velmi dobře komentovaný, většina parametrů je už defaultně nastavena správně, kromě parametrů týkajících se zabezpečení, nastavení vlastností cache či vlastní adresy sítě. Implicitně nastavený proxy server zabraňuje připojení uživatelům z jakékoliv síťové adresy kromě 127.0.0.1.

Nastavení sítě (sekce NETWORK OPTIONS)

Následující nastavení umožňuje určit porty, na kterých Squid „naslouchá“. Čím méně portů je otevřených, tím nižší je riziko jejich zneužití. Některé z těchto portů ne vždy potřebujeme, takže je můžeme zakázat [19].

- **http_port 3128**: port, na kterém Squid defaultně přijímá požadavky klientů. Tento port je nevyhnutelný pro činnost Squidu.
- **icp_port 3130**: port, pomocí kterého Squid spolupracuje s rovnocennými proxy servery. V případě nevyužití je možné tuto funkci zablokovat hodnotou čísla portu „0“.
- **htcp_port 4827**: port, pomocí kterého Squid spolupracuje s rovnocennými proxy servery. V případě nevyužití je možné tuto funkci zablokovat hodnotou čísla portu „0“.

Nastavení cache (sekce OPTIONS WHICH AFFECT THE CACHE SIZE)

Pomocí těchto nastavení můžeme ovlivnit velikost diskové cache paměti, objekty, které se nemají uchovávat v cache paměti a algoritmus odstraňování objektů z cache paměti v případě, že dochází potřebný paměťový prostor a je třeba uložit objekty nové [19].

- **cache_mem 8 MB**: velikost paměti používaná na nejvíce užívané objekty. Hodnotu zvětšíme dle našich možností a nároků. Pozor, nejde o paměť, kterou bude Squid zabírat. Ve skutečnosti bude proces „squid“ zabírat v paměti více!
- **maximum_object_size 4096 KB**: maximální velikost objektu uchovávaného v cache paměti. Větší objekty se uchovávat nebudou.
- **minimum_object_size 0 KB**: minimální velikost objektu uchovávaného v cache paměti. Menší objekty se uchovávat nebudou. Hodnota „0“ znamená ignoraci tohoto požadavku.
- **maximum_object_size_in_memory 8 KB**: maximální velikost objektu, který má být uchovávaný ve fyzické paměti.
- **cache_replacement_policy lru**: algoritmus odstraňování objektů z cache paměti (určuje, které objekty budou odstraněny a které zůstanou v paměti). Výchozím nastavením je „lru“ (least recently used – nahradí nejdéle nevyužívanou stránku). Pomocí tohoto parametru lze radikálně ovlivnit výkon proxy serveru, pravděpodobně to však vyžaduje rekompilaci kódu (tzv. balíčkové verze nepodporují všechny algoritmy). Některé z dalších užívaných algoritmů jsou „lfu“ (least frequently used – nahradí nejméně využívanou stránku), „nfu“ (not frequently used – stránka nepoužívaná často), dále pak aging (stárnutí), apod.
- **memory_replacement_policy lru**: algoritmus odstraňování objektů z fyzické paměti (stejný princip jako u parametru „cache_replacement_policy“).

Cesty k „log“ souborům a cache (sekce LOGFILE PATHNAMES AND CACHE DIRECTORIES)

- **cache_dir ufs /var/spool/squid3 100 16 256**: nastavení adresáře s cachovanými objekty. První parametr („ufs“) určuje způsob uchovávání souborů. Druhý parametr („/var/spool/squid3“) určuje adresář, ve kterém se uchovávají objekty. Třetí parametr („100“) určuje velikost cache paměti na disku (v jednotkách MB). Poslední dva parametry určují počet vytvořených adresářů nižší úrovně v adresáři s cache. Při defaultním nastavení se vytvoří 16 podadresářů, každý z nich má dalších 256 podadresářů, v kterých se uchovávají objekty. Tyto adresáře se vytvářejí pro zrychlení přístupu k objektům – nemělo by do nich být zasahováno. Je nutné pamatovat také na to, že po změně hodnoty této direktivy je třeba znovu vytvořit/upravit adresářovou strukturu cache paměti (pozn. spustit Squid s parametrem „-z“).
- **cache_access_log /var/log/squid3/access.log**: logovací soubor se záznamy každého požadavku.
- **cache_log /var/log/squid3/cache.log**: logovací soubor se všeobecnými informacemi o správě cache paměti. Můžeme ho sledovat například po spuštění proxy serveru a dozvědět se, zda je vše v pořádku.
- **cache_store_log none**: logovací soubor obsahující záznamy o objektech, čase vložení do cache paměti, čase odstranění z cache paměti a podobně. V konfiguračním souboru je přímo zmínka o neexistenci seriózní utility pro analýzu tohoto souboru, takže ho můžeme zakázat (hodnota „none“).
- **emulate_httpd_log off**: umožňuje zapnout a vypnout emulaci formátu HTTP serveru (Apache) v logu „cache_access_log“. Emulovaný formát lze jednodušeji číst (správcem), ale pro přirozený formát existují speciální nástroje na analýzu přístupů a podobně.

Vyladění cache paměti (sekce OPTIONS FOR TUNING THE CACHE)

- **request_body_max_size 1 MB**: maximální velikost HTTP požadavku.
- **reply_body_max_size 0**: maximální velikost HTTP odpovědi. Umožňuje zablokovat stahování objemných souborů (MP3, AVI, apod.) přes proxy server. Výchozí nastavení nelimituje příchozí odpověď.

Řízení přístupu pomocí ACL (Access Control Lists)

Ke kontrole přístupu slouží v nastavení Squidu přístupové záznamy (ACL – Access Control Lists).

Každý ACL se skládá nejméně ze čtyř částí:

```
acl jméno typ hodnota . . .
acl jméno typ "soubor" . . .
```

1. direktiva **acl**
2. **jméno** ACL (bude se na něho odvolávat při samotném řízení přístupu)
3. **typ** ACL (určuje, co se bude kontrolovat, např. zdrojová adresa, cílová adresa, atd.)
4. **hodnota, případně vícero hodnot** (hodnota je uvedená jako řetězec, popřípadě je možné uvést **jméno souboru** v uvozovkách – soubor musí být textový, ve kterém každý řádek znamená jednu hodnotu)

Následuje záznam nejpoužívanějších typů ACL s uvedením jejich syntaxe a krátkým popisem. Všechny jsou uvedené v konfiguračním souboru "`/etc/squid3/squid.conf`". Připomínám, že cílová adresa znamená adresu (HTTP) serveru, zdrojová adresa je adresa klientského počítače, který si žádá od proxy serveru zpřístupnění www stránky na dané cílové adrese [19].

- **acl aclname src 10.0.0.0/255.255.255.0 ...**: definuje ACL založené na zdrojové IP adrese a masce
- **acl aclname src 10.0.0.1-10.0.63/255.255.255.255 ...**: definuje ACL pro zadaný rozsah zdrojových IP adres
- **acl aclname dst 192.168.0.0/255.255.255.0 ...**: definuje ACL pro zadanou cílovou IP adresu s maskou
- **acl aclname srcdomain .foo.com ...**: definuje ACL pro zadanou část domény ve zdrojové adrese (vykoná se reverzní překlad zdrojové adresy)
- **acl aclname dstdomain .foo.com ...**: definuje ACL pro zadanou část domény v cílové adrese (z URL adresy)
- **acl aclname srcdom_regex [-i] xxx ...**: definuje ACL pro zdrojovou adresu (adresu klienta), která vyhovuje regulérnímu výrazu. Parametr "-i" znamená, že se nebude brát v úvahu velikost písmen.
- **acl aclname dstdom_regex [-i] xxx ...**: definuje ACL pro cílovou adresu (adresu serveru), která vyhovuje regulérnímu výrazu. Parametr "-i" znamená, že se nebude brát v úvahu velikost písmen.
- **acl aclname time [day] [h1:m1-h2:m2]**: definuje ACL pro určitou časovou relaci – jsme schopni určit konkrétní dny (S - *Sunday*, M - *Monday*, T - *Tuesday*, W - *Wednesday*, H - *Thursday*, F - *Friday*, A - *Saturday*) a časy (h1:m1 musí být méně jako h2:m2).
- **acl aclname url_regex [-i] ^http://www.foo.com ...**: definuje ACL pro URL, která vyhovuje regulérnímu výrazu. V tomto případě musí URL začínat textem http://www.foo.com.
- **acl aclname urlpath_regex [-i] \.gif$ ...**: definuje ACL pro část URL adresy (cesty), která vyhovuje regulérnímu výrazu. V tomto případě musí cesta končit řetězcem ".gif".
- **acl aclname port 80 70 21 ...**: definuje ACL pro cílový port nebo více cílových portů
- **acl aclname port 0-1024 ...**: definuje ACL pro rozsah cílových portů
- **acl aclname proto HTTP FTP ...**: definuje ACL pro typ protokolu
- **acl aclname method GET POST ...**: definuje ACL pro metodu protokolu HTTP
- **acl aclname browser [-i] regexp**: definuje ACL pro hlavičku "User-Agent" v protokolu HTTP (umožňuje detekovat typ prohlížeče), která vyhovuje regulérnímu výrazu

Další důležitá nastavení

Tyto direktivy nastavují rozličné administrativní parametry [19]:

- **cache_mgr root**: e-mailová adresa správce cache.
- **cache_effective_user squid**: uživatel, s jehož právy běží procesy Squidu, jestliže byl spuštěný pod tzv. rootem. Doporučuje se vytvořit speciálního uživatele (např. „squid“).
- **cache_effective_group squid**: skupina, s jejímiž právy běží procesy Squidu, jestliže byl spuštěný pod tzv. rootem. Doporučuje se vytvořit speciální skupinu (např. „squid“).
- **memory_pools on**: zapnutí této volby způsobí, že si Squid udržuje alokovanou (i když nevyužitou) paměť pro další použití. V případě nedostatečného množství paměti lze volbu zakázat pomocí hodnoty „off“.
- **memory_pools_limit 50 MB**: velikost paměti, kterou si Squid udržuje alokovánu, jestliže používáme „memory_pools“.
- **error_directory /usr/share/squid3/errors/Czech**: adresář s chybovými hlášeními – defaultně anglický jazyk. Zde uvedená direktiva zabezpečí zobrazování českých chybových hlášení. Chybová hlášení jsou uložená jako samostatné HTML soubory, které jsme schopni libovolně modifikovat a vhodně doplňovat.
- **snmp_port 0**: v případě nevyužití SNMP služby (resp. protokolu), můžeme zakázat další port pomocí hodnoty „0“.
- **chroot /chroot_jail**: umožní uzavřít program Squid ve vyhrazeném adresáři (systémové volání chroot ()). Všechny cesty, které Squid používá, se stanou relativními cestami od tohoto adresáře, tedy např. "`/var/spool/squid3`" znamená "`/chroot_jail/var/spool/squid3`". Ve výchozím nastavení je volba vypnuta.
- **logfile_rotate**: počet udržovaných logovacích souborů při používání "`squid -k rotate`" (přípony předcházejících log. souborů budou čísla). Tento příkaz se volá automaticky pomocí programu „logrotate“. Jestliže je hodnota parametru "`logfile_rotate`" rovna „0“, nastavení řídí program „logrotate“.
- **append_domain**: doména, která se přidá za nekompletní adresy v URL.

Redirector

Jakmile Squid u nového požadavku zkontroluje přístupová práva (ACLs), může předat URL ke zpracování externímu redirectoru a pokračovat až se zpracovanou URL. Redirector je jednoduchý filtr, co na standardním vstupu přečte URL, zpracuje ji a na standardní výstup vypíše novou verzi. K zapnutí redirectoru slouží klauzule `redirect_program`, kam se zadá cesta k programu. Pomocí ACL operátoru `redirector_access` můžeme nastavit, pro které požadavky bude redirector uplatněn. Součástí Squidu žádný redirector není, je tedy třeba využít software třetí strany nebo vytvořit vlastní.

Refresh algorithm

Pokud není cachování úplně zakázáno direktivami HTTP jako Pragma: `no-cache` nebo Cache-control: `{Private,No-Cache,No-Store}`, může být objekt při průchodu proxy serverem uložen do paměti cache. Refresh algorithm je algoritmus, podle kterého Squid určuje, jestli objekt může ještě klientovi vrátit z cache (objekt je *FRESH*), nebo zda už je příliš zastaralý a neaktuální (*STALE*) a musí se tudíž obnovit z původního serveru a následně poté předat klientovi, který o něj žádal.

Hierarchické cachování (pro komunikaci vyvinut speciální protokol ICP)

Pokud Squid potřebuje pro klienta získat nějaký dokument, obrací se přímo na vzdálený server. Mnohem efektivnější by ale bylo, kdyby se obrátil nejdříve na nějakou jinou cache poblíž, kde latence spojení bude mnohem menší. Vztahy mezi cache jsou dvou druhů: "parent" (rodič) a "sibling" (sourozenec). Sibling oproti parent serveru nebude nikdy obstarávat objekt mimo svou cache.

Více informací o projektu Squid lze nalézt na domovské internetové adrese http://www.squid-cache.org/Doc/.

5.3.1.5 Konfigurace Squidu

Transparentní režim Squidu

Nastavení transparentního režimu Squidu v předchozích verzích [26]:

- *httpd_accel_host* *virtual*
- *httpd_accel_uses_host_header* *on*
- *httpd_accel_port* *80*
- *httpd_accel_with_proxy* *on*

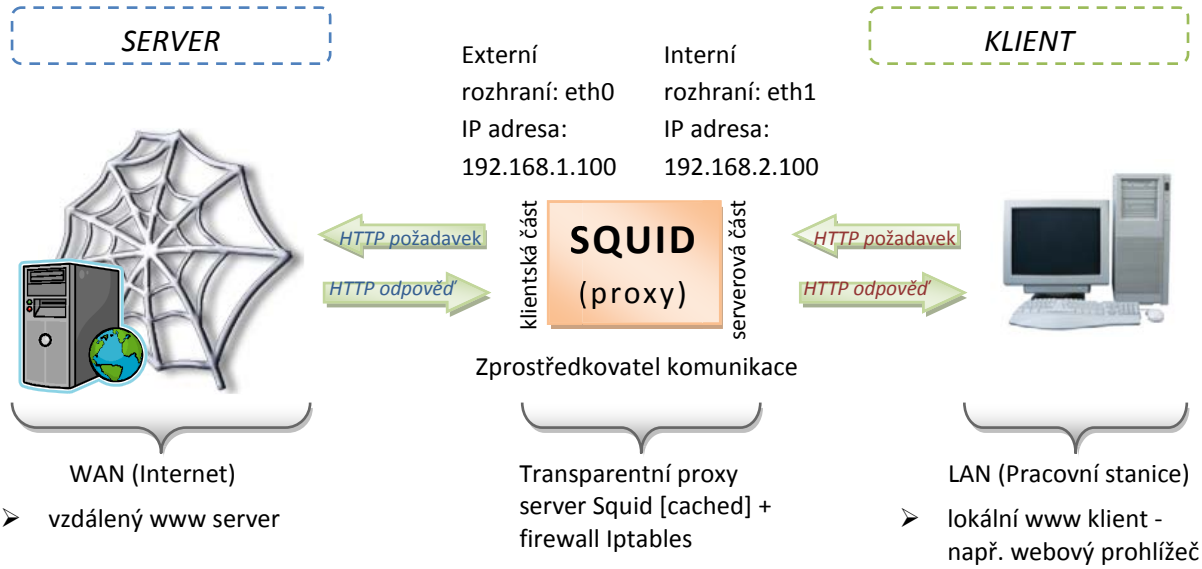
Od verze 2.6 stačí v konfiguračním souboru *squid.conf* nastavit pouze:

- *http_port 3128 transparent*
- *always_direct allow all*

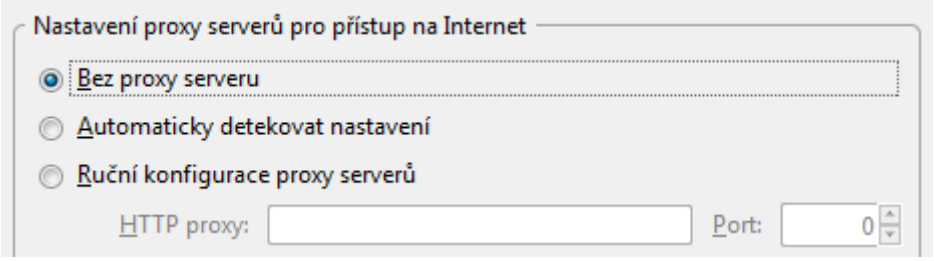
Plus samozřejmě zapnout přesměrování ve firewallu Iptables [20], [21]:

 # Protože používáme proxy server Squid jako transparentní proxy, je nutné nastavit firewall Iptables tak, aby veškerý provoz směřující na port 80 (WWW) byl automaticky přeposílán tomuto proxy serveru.
iptables -t nat -A PREROUTING -i $LAN_IN -p tcp --dport 80 -j DNAT --to $SQUID_SERVER:$SQUID_PORT
 # Jestliže se jedná o stejný systém, pak platí následující.
iptables -t nat -A PREROUTING -i $INTERNET -p tcp --dport 80 -j REDIRECT --to-port $SQUID_PORT

V našem případě - $LAN_IN: **eth1 [192.168.2.100/24]**, $INTERNET: **eth0 [192.168.1.100/24]**, $SQUID_SERVER: **192.168.1.100**, $SQUID_PORT: **3128**



Obr. 28: Názorný klient-server model komunikace procházející přes transparentní proxy



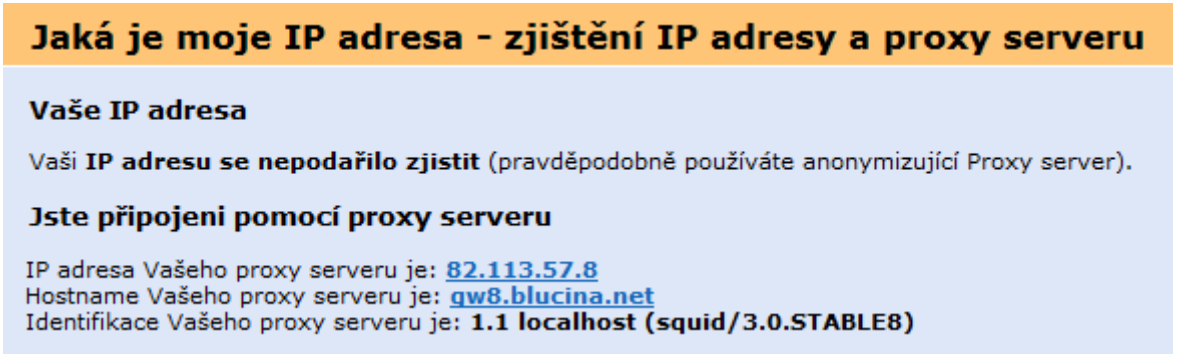
Obr. 29: U transparentního režimu není třeba upravovat prohlížeče klientských počítačů

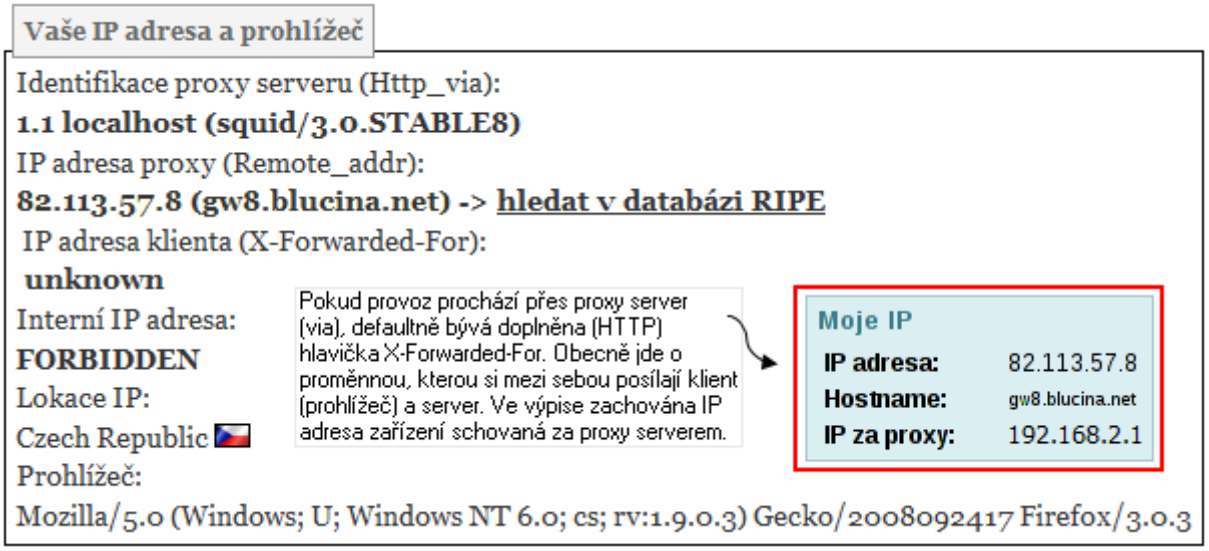
Vyjmutí X-Forwarded-For hlavičky z HTTP požadavků

Tento úkon se provádí z důvodu anonymity klientských PC. Nedojde tak k nechtěnému prozrazení IP adres počítačů v interní síti „schovaných“ za proxy firewallem. Výsledek je zobrazen na 2 příkladech.

- *forwarded_for off* *[možnosti proměnné jsou: “**on**” … zapnuto, ”**off**” … vypnuto]*

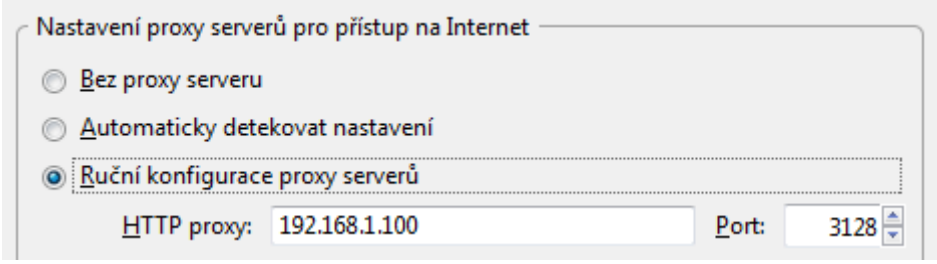
Kromě toho se pomocí hlavičky zjišťuje unikátnost návštěvníka, např. při hlasování v anketě apod. Hlavičku lze snadno podvrhnout, proto na ní nelze spoléhat. Squid disponuje dalšími možnostmi anonymizace klienta prostřednictvím konfiguračních direktiv anonymize_headers a fake_user_agent.





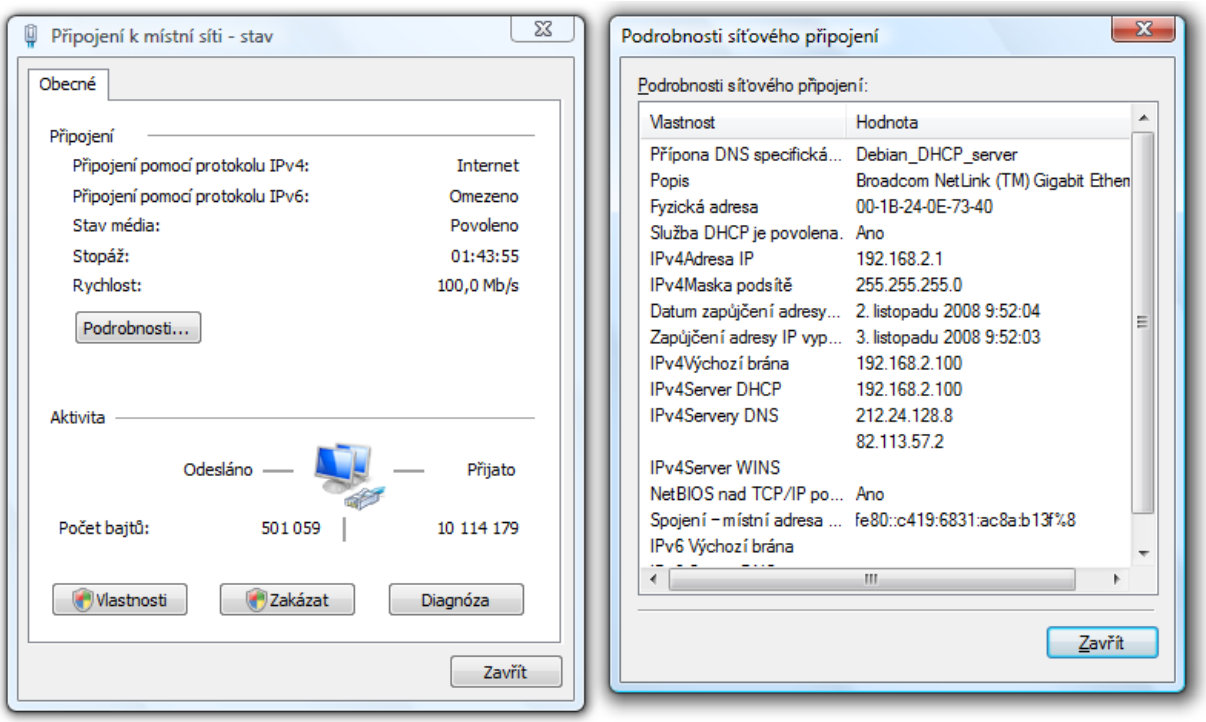
Zde bylo poukázáno pouze na některé z obrovských možností konfigurace Squidu. Další prováděná nastavení - viz Příloha diplomové práce, komentovaný konfigurační soubor */etc/squid3/squid.conf*.

5.3.1.6 Konfigurace klientských webových prohlížečů

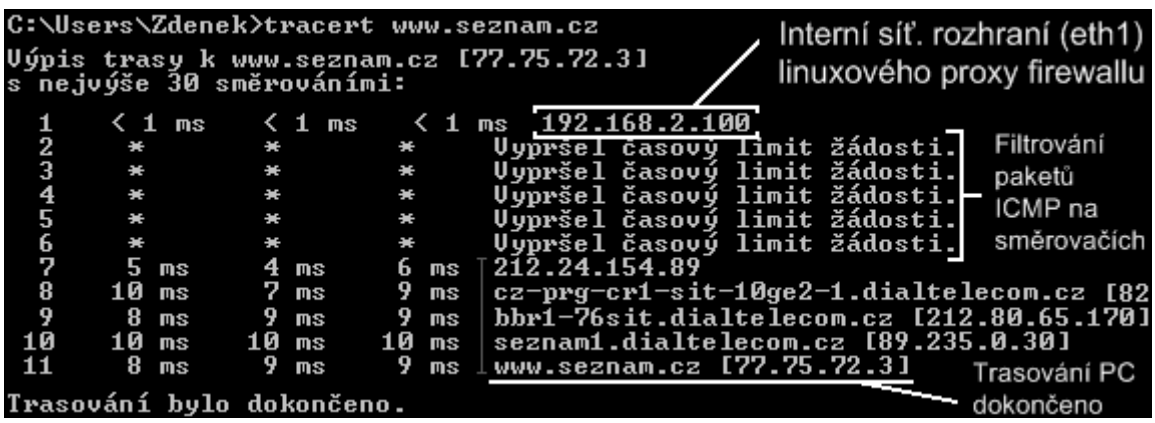


Obr. 30: Webové prohlížeče (*„browsery“*) je ke spolupráci s proxy nutno nastavit

Obr. 31: Nastavení sítě v Centru sítí a sdílení - hostitelský systém Windows

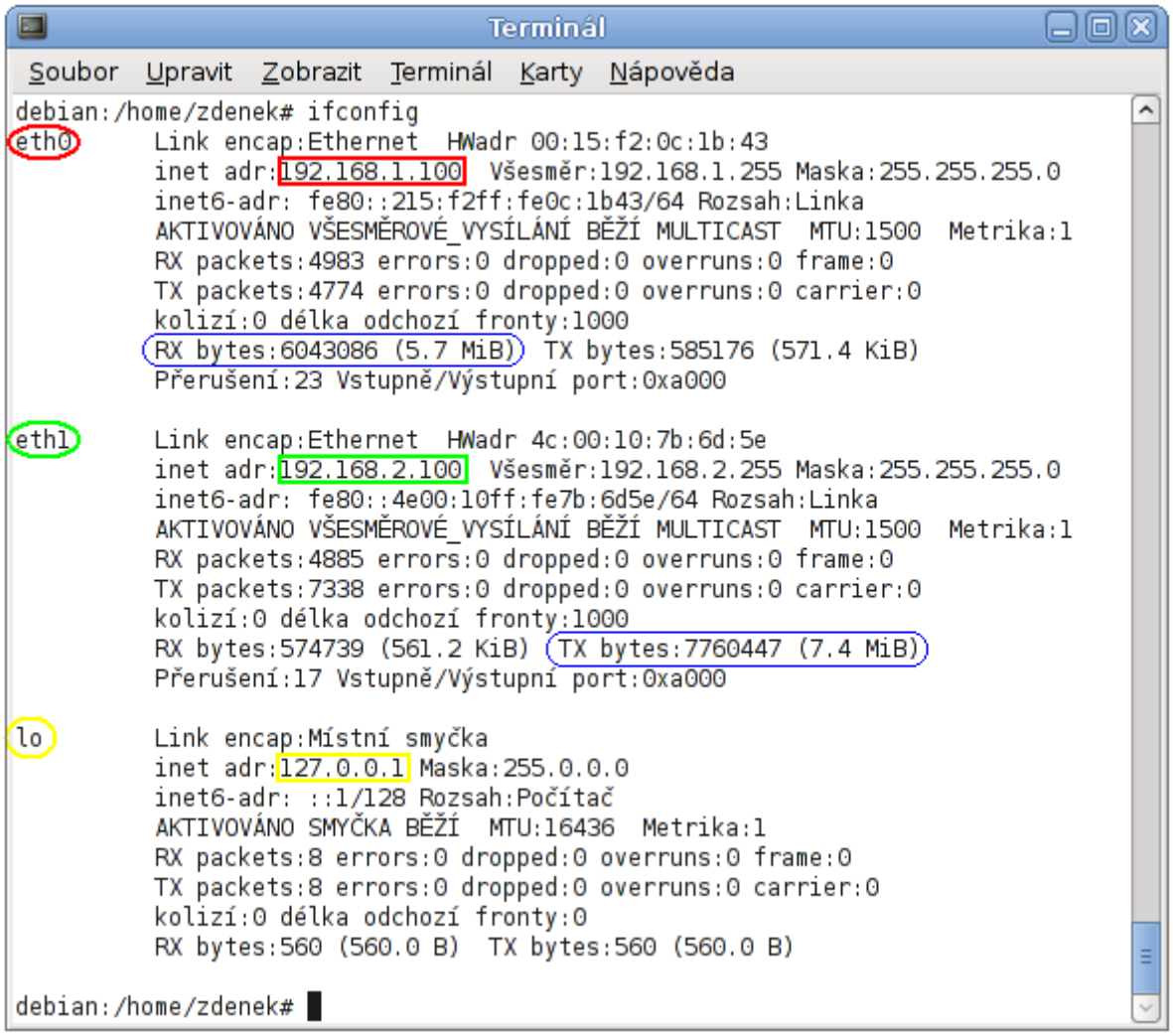


Obr. 32: Stav připojení k místní síti a podrobnosti síťového připojení na klientské stanici



Obr. 33: Příkazem tracert vrátíme seznam směrovačů na cestě k cílové adrese IP

5.3.1.7 Testování sítě & proxy serveru



Obr. 34: Linuxový příkaz *ifconfig* a dostupná síťová rozhraní

Popis předchozího obrázku 34:

eth0 - externí síťové rozhraní [WAN]; nedůvěryhodná nebo neznámá síť (např. Internet)
eth1 - interní síťové rozhraní [LAN]; důvěryhodná místní síť (např. Intranet)
lo - lokální loopback, místní smyčka (IP adresa: 127.0.0.1, maska: 255.0.0.0); pro účely testování

Vnějším rozhraním eth0 byla data přijímána z Internetu a vnitřním rozhraním eth1 poskytována místním hostitelským stanicím. Z obrázku je patrné, že modře vyhraničená pole značí u síťového rozhraní eth0 download (RX bytes) a u síťového rozhraní eth1 upload (TX bytes). Hodnoty uvedené v těchto polích (v jednotkách megabajtů) názorně dokazují funkčnost a efektivnost využití proxy serveru. V našem případě proxy serveru Squid.

Zobrazené hodnoty množství dat je nutno brát pouze v orientačním měřítku, jelikož byly v domácích omezených podmínkách testování k dispozici pouze tři dostupné osobní počítače a jeden notebook, všichni připojení k proxy serveru přes 100 Mbit/s switch (přepínač), neprojevila se zdaleka tolik efektivnost využití proxy cache, jako například v případě menší či střední firmy o několika desítkách či dokonce stovkách zaměstnanců.

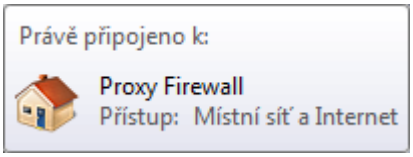
Při takovém počtu připojených počítačů se často opakující se požadavky načítají „úsporně“ vzhledem k přenosové lince pouze z lokální cache proxy (za předpokladu splnění určitých kritérií) a šetří tak potenciál (výkon, někdy i cenu) síťových prostředků komunikační linky.

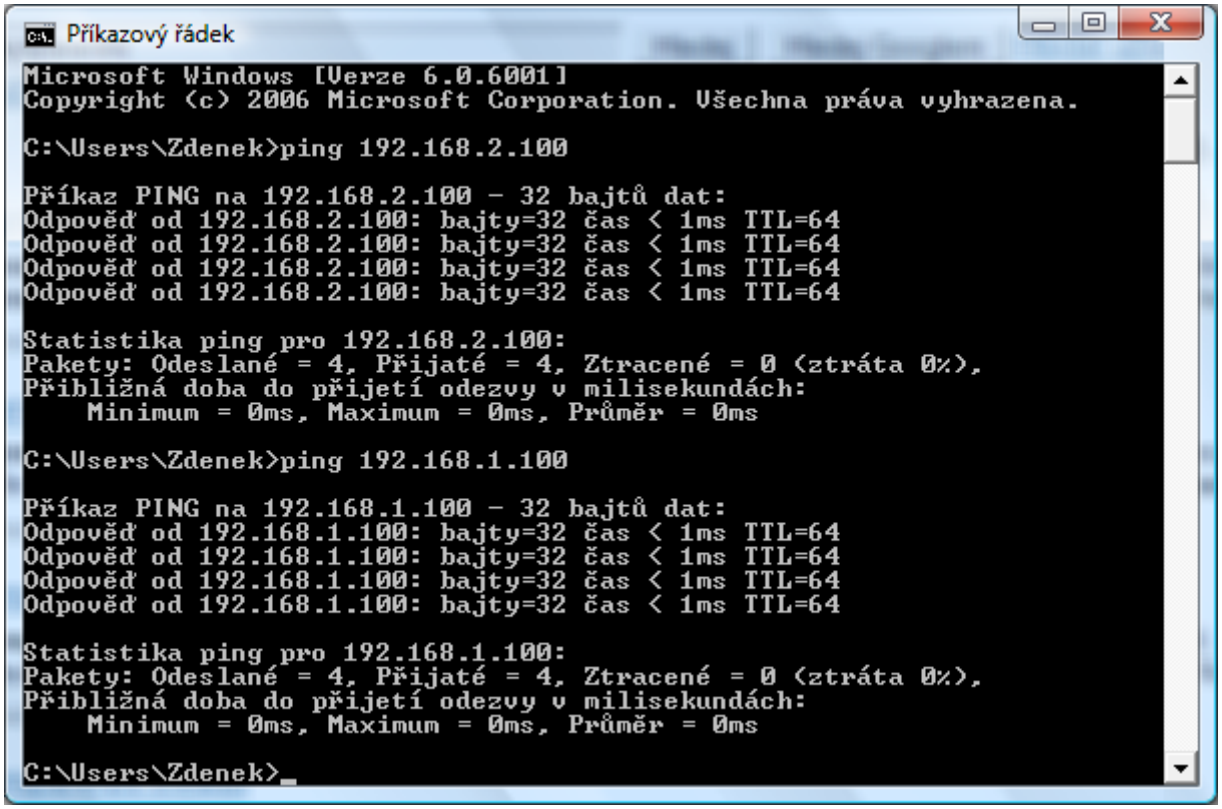
Vzniklý rozdíl mezi přijatými daty rozhraním eth0 a odeslanými daty rozhraním eth1 tvoří právě nacachované objekty v proxy, které byly rozhraním eth1 odeslány klientským stanicím jako odpovědi na požadavky, aniž by proxy server s těmito hostitelskými žádostmi kontaktoval vzdálený server na Internetu a vyžádal si novou/nenacachovanou, případně aktualizovanou stránku (tzn., požadavky od klientů vůbec neprošly vnějším síťovým rozhraním eth0). Diference mezi RX (pakety přijatými eth0) a TX (pakety odeslanými eth1) může být ovlivněna také lokálními broadcast pakety (tj. lokální oběžník).

δ = TX bytes$_{eth1}$ - RX bytes$_{eth0}$ = 7,4 MB - 5,7 MB = 1,7 MB - rozdílová hodnota značící velikost objektů uložených v cache proxy (lokálně k dispozici klientským stanicím); [TX bytes$_{eth1}$ ≥ RX bytes$_{eth0}$]

Otestování konektivity a zjištění dostupnosti obou rozhraní linuxového proxy zobrazuje obrázek 35. Program ping testuje funkčnost cesty ke zvolenému cíli paketem ICMP a poskytuje statistiku zpoždění. Ze zobrazených výsledků (výsledných časů) lze soudit, že síťová komunikace je v pořádku.

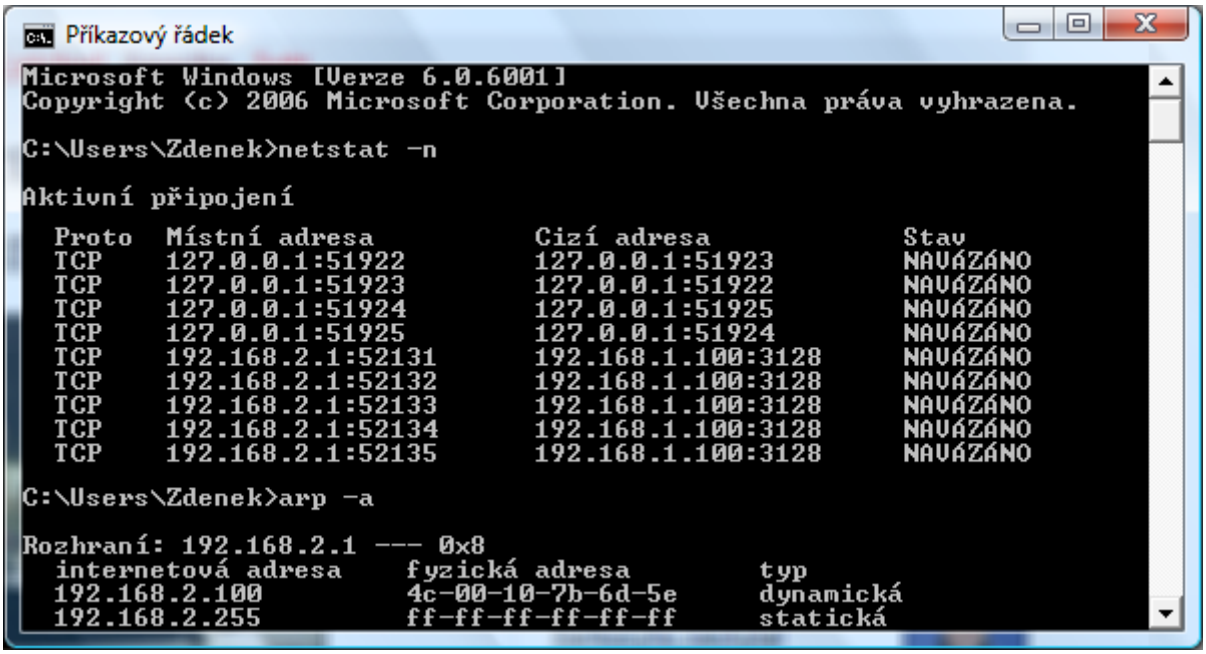
IP adresa 192.168.2.100 - vnitřní rozhraní [eth1].
IP adresa 192.168.1.100 - vnější rozhraní [eth0].
Testováno z počítače uvnitř sítě (PC1), IP adresa 192.168.2.1.
Počítač má přes proxy server zajištěnou konektivitu do Internetu.





Obr. 35: Příkaz ping provedený z hostitelské stanice v interní síti (OS Windows)

Zadáním příkazu netstat -n zobrazíme aktivní připojení TCP a provedeme kontrolu otevřených síťových portů. Program netstat podává informace o funkci celého komunikačního systému a jeho jednotlivých komponent.

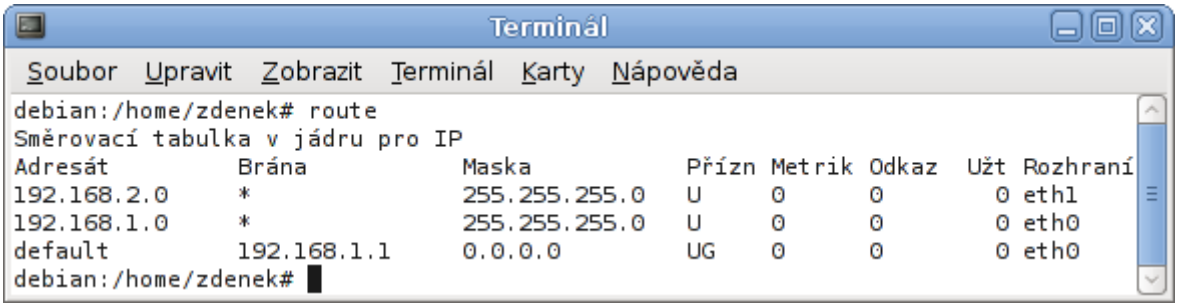


Obr. 36: Příkaz netstat provedený z hostitelské stanice v interní síti (OS Windows)

Popis předchozího obrázku 36:

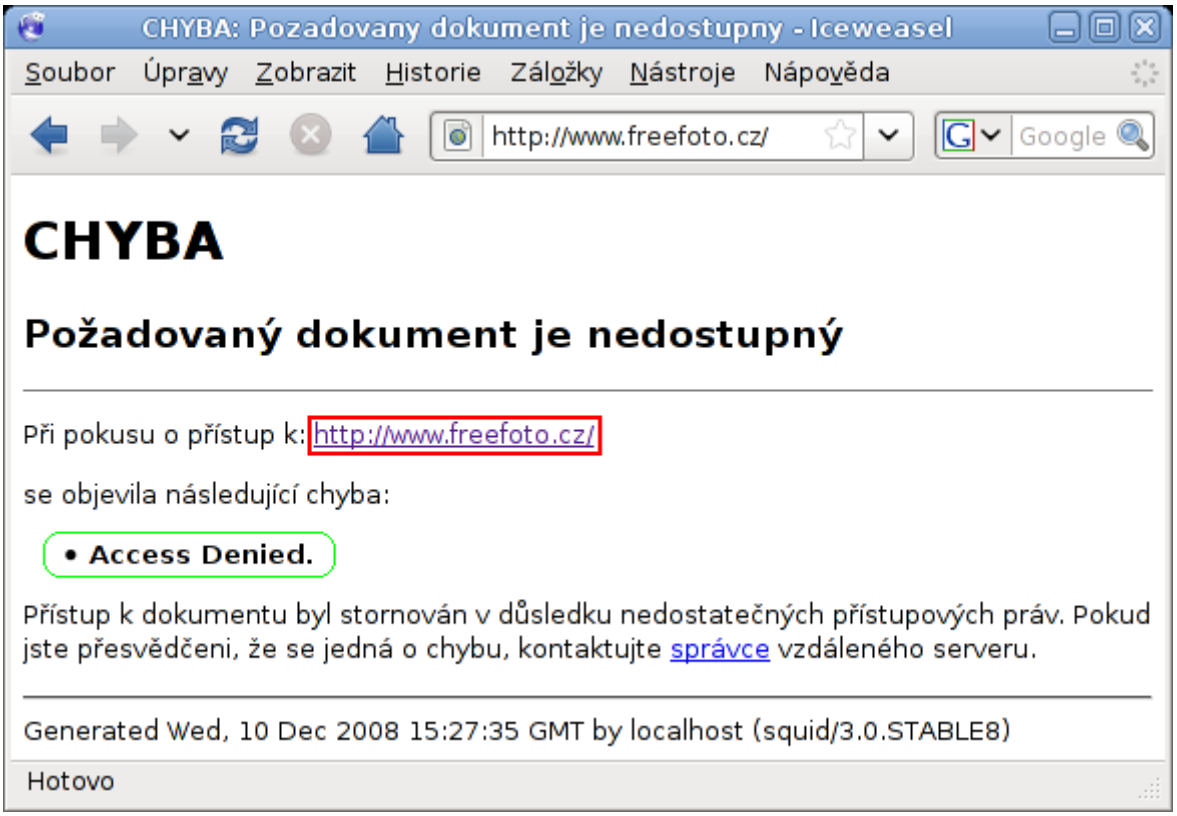
Proxy server Squid běží na síťové IP adrese 192.168.1.100 a naslouchá příchozím požadavkům klientů na portu 3128. Hostitelský počítač (klient) navazující spojení má IP adresu 192.168.2.1, privátní/dynamický port 5213X (porty 49152 - 65535 jsou používány pro komunikaci klienta se serverem). Z obrázku lze dále vyčíst, že pro komunikaci byl použit spojově orientovaný transportní protokol TCP a také, že se jedná o navázaný stav. IP adresa 127.0.0.1 (localhost) je rezervována pro tzv. loopback, logickou smyčku umožňující posílat pakety sám sobě.

Současný stav směrovací tabulky můžeme získat příkazem route (bez parametrů) na stroji Debianu. Zkontrolujeme v ní obsažené údaje. Příkazem route konfigurujeme statickou směrovací tabulku (nastavení routování a výchozí brány).



Obr. 37: Příkaz route a výpis směrovací tabulky v konzole příkazového řádku

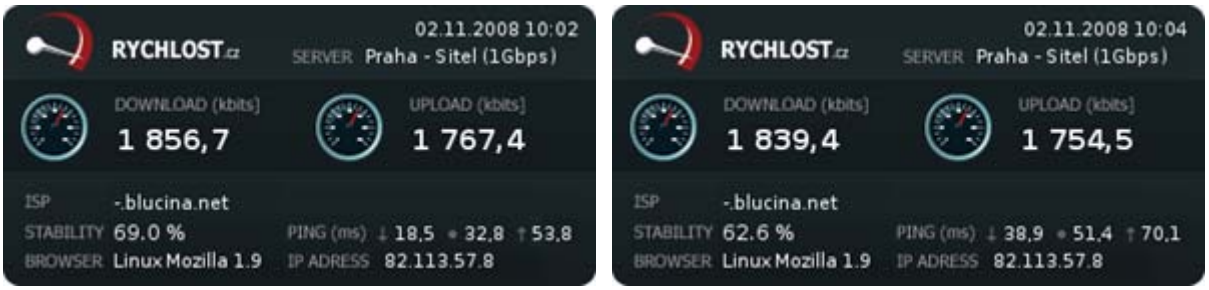
IP 192.168.2.0 je adresa vnitřní sítě (dostupná přes rozhraní eth1), IP 192.168.1.0 je adresa vnější sítě (dostupná přes rozhraní eth0). U obou maska třídy C (prefix /24). Defaultní adresát s IP 192.168.1.1 je brána (gateway) s maskou sítě 0.0.0.0 pro výchozí trasu. Uvedené příznaky znamenají: U (*up*) je funkční směrování a G (*gateway*) je označení místa, kde se paketů ujímá směrovač. Rozhraní je označení síťového adaptéru.



Obr. 38: Zakázaná URL adresa blokovaná Squidem

5.3.1.8 *Vliv proxy serveru na síťovou komunikaci*

Jaký vliv na síťovou komunikaci bude mít praktické nasazení proxy zkoumá následující měření přenosových rychlostí a dalších kvalitativních parametrů pro zjištění reálných hodnot při využití proxy. V druhém případě (zapojení s proxy serverem) byly všechny nacachované [nakešované] objekty (stránky) vymazány - z důvodu objektivnosti testování. Měření proběhlo celkem třikrát.



Obr. 39: Porovnání dvou měření přenosových rychlostí a ostatních parametrů pro zapojení bez proxy serveru („přímo“ - obrázek vlevo) a zapojení s proxy serverem („přes prostředníka“ - obrázek vpravo)

Z předchozího vyplývá, že při reálném nasazení proxy nedochází téměř k žádnému výraznému zpomalení či prodlevě vlivem jeho použití (je tím myšlen především samotný průchod paketů skrze proxy a bez něj, tj. přímou cestou od klienta k veřejnému serveru a zpět). Samozřejmě také záleží na ostatních souvisejících faktorech, jako například vytíženost serveru a přenosové linky, výkonnostní a paměťové parametry serveru, apod. Měření proběhlo na serveru s následující konfigurací: CPU - AMD Athlon 64 3400+ (2,21 GHz), RAM - 1 GB DDR (400 MHz), 2x síťová karta 32-bit PCI 100Mbps Fast Ethernet Adapter, HDD - 500 GB SATA. Měřeno prostřednictvím portálu www.rychlost.cz.

5.4 Antivirové systémy

Antivirový systém se skládá z několika podstatných částí. Jednou z výkonných částí je nepřetržitý dohled antivirové kontroly nad daty, se kterými uživatel aktuálně pracuje. Další část umožňuje provádět antivirové testy na vybraných oblastech. Test je zpravidla vyvolán na základě požadavku uživatele (tzv. „on-demand“, na přání). Následuje část udržující antivirový systém v aktuální podobě, zejména co se týče pravidelného stahování aktualizací z Internetu. Důležitá je taktéž část vykonávající automatickou antivirovou kontrolu příchozí a odchozí elektronické pošty (e-mail). Mezi další části, které nemusí být zdaleka tak obvyklé patří například plánovač událostí (scheduler), kontrola integrity, karanténa, monitorovací programy a různorodé AV plug-iny [42].

Počítačovými viry se rozumí malé softwarové programy určené k šíření z jednoho počítače do jiného a narušování fungování počítače (popřípadě jeho bezpečnostních funkcí). Počítačový virus může poškodit nebo znehodnotit data v počítači, využít e-mailovou aplikaci k vlastnímu rozšíření do jiných počítačů, nebo dokonce vymazat všechna data na pevném disku. Viry se nejsnadněji šíří v přílohách e-mailových nebo rychlých zpráv (IM – Instant Messaging). Mohou být skryty jako přílohy v podobě obrázků, fotografií, audio a video souborů (obecně multimediálního obsahu). Šíří se také stahováním z Internetu, mohou být skryty např. v nelegálním softwaru (warez – zejména co se týče různých pofiderních cracků apod.) nebo umístěny na webových stránkách s pornografickou tematikou [33].

Ochrana proti malware & obecné zásady ochrany proti malware

Na ochranu proti škodlivému software se používají specializované programy, které jej cíleně vyhledávají – tzv. antivirové programy. Antivirový program sleduje všechny nejpodstatnější vstupní a/nebo výstupní místa, kterými by viry mohly do počítačového systému proniknout. Pokud jde o viry samotné, můžeme říci, že se jedná o nežádoucí a ve většině případů škodlivý kód, který se cíleně šíří a reprodukuje. Vyhledávání se většinou děje na základě charakteristických znaků strojového nebo zdrojového kódu škodlivého software. Ke ztížení své detekce proto škodlivý software používá různé techniky maskování, jako např. polymorfismus, kdy se jeho kód na každém počítači modifikuje nebo šifrování, kdy se škodlivý software na každém počítači šifruje jiným klíčem [38].

Škodlivý kód by se do počítače uživatele neměl vůbec dostat. Pokud do něj pronikne, tak by neměl být spuštěn. Pokud se tak stalo, jeho možnosti narušit bezpečnost, by měly být minimální. Přenosové protokoly na všech vnějších rozhraních počítače (síťové rozhraní, USB, bluetooth, DVD mechaniky aj.) by neměly dovolit jakýkoliv přenos dat z neautorizovaného zdroje a neměly by umožnit skrytý přenos v rámci autorizovaného přístupu. Využívají se různé autentizační a kryptografické mechanismy. Kontrola každého souboru před jeho spuštěním zamezí spuštění neautorizovaného programu. To lze realizovat certifikací nebo autentizačními kódy souboru v souborovém systému. V neposlední řadě také minimalizace práv programu, kdy program obdrží pouze nejnutnější práva k využívání HW prostředků počítače, k využívání služeb OS a k interakci s jinými programy [38].

Antivirové systémy („antiviry“) se často kombinují s dalšími bezpečnostními nástroji jako antispyware a antispam. Základní, a dá se říci, že i prvotní ochranou před škodlivým softwarem, zůstává průběžně aktualizovaný operační systém pomocí tzv. bezpečnostních záplat, které znemožní využití známých kritických chyb či nedostatků OS. Pro škodlivý kód lze stanovit několik termínů a rozlišit mezi sebou trojský kůň, virus, makrovirus, červ, hoax aj. Společným rysem všech bezpečnostních programů je pravidelná aktualizace jak programové, tak zejména virové báze, z důvodu schopnosti čelit aktuálním hrozbám. Jedním takovým nástrojem, který slouží k vyhledávání a odstranění malware je ClamAV.

5.4.1 Antivirový program ClamAV

Operační systém Linux se často používá pro poštovní, souborové, proxy a další servery, sloužící (převážně) uživatelům Windows. Proto se v Linuxu můžeme se škodlivým softwarem setkat poměrně často, byť systému samotnému nebezpečí nehrozí. Ze stejného důvodu mají na Linuxu svůj smysl i antivirové programy. Většina jejich tvůrců poskytuje nějakou linuxovou verzi svého produktu, i když třeba ne v celé škále funkcí. Jedním z takových programů je multiplatformní antivirový software ClamAV - pravděpodobně jediný aktivně vyvíjený a dlouhodobě udržovaný svobodný antivirový program, který byl zvolen ke kontrole a ochraně přenášených dat (souborů, dokumentů, www stránek, apod.) na proxy (stroj s Debianem) [36].

Některé ze základních funkcí, které ClamAV nabízí:

- Scanner z příkazové řádky
- Multi-threadový (vícevláknový) agent s podporou kontroly při přístupu
- Vestavěnou podporu prakticky všech e-mailových formátů
- Podporu pro kontrolu HTML, RTF, PDF souborů, dále přípon známých kancelářských balíků

5.4.1.1 Představení ClamAV

ClamAV je kvalitní antivirový nástroj pro vyhledání a odstranění virů, červů, trojských koní a dalších škodlivých programů a malware. Lze ho ovládat jak pomocí příkazového řádku (defaultně), tak i přes grafické uživatelské rozhraní (přídavné GUI front-end). Nabízí širokou škálu použití při ochraně před škodlivým (pro uživatele nebezpečným) softwarem. Aktualizace virové báze jsou vydávány i vícekrát denně. Program je šířen jako Open Source a je tedy poskytován zdarma (licence GNU GPL verze 2).

ClamAV se skládá z několika částí. Především je to knihovna *LibClamAV*, která zajišťuje vlastní kontrolní činnost a kterou lze použít v libovolném programu. Dalšími komponentami jsou skenovací program `clamscan`, démon `clamd`, klientská aplikace `clamdscan`, "on-access" skener `clamuko`, aktualizátor `freshclam` a poštovní filtr `clamav-milter` (pro poštovní server Sendmail) [36].

5.4.1.2 Detekční schopnosti

ClamAV není jen antivirovým programem, ve své databázi má kromě škodlivého softwaru také vzorky phishingových e-mailových zpráv. To naznačuje, že jednou z hlavních oblastí použití je právě kontrola pošty. Databáze aktuálně obsahuje cca 500 tisíc vzorků, proběhlo již více než 8900 malých ("denních") a téměř 50 velkých aktualizací. Důležité samozřejmě není, kolik virů a dalšího malware program zná, nýbrž jak se osvědčuje při jejich detekci v reálné praxi - včetně toho, jak brzy je schopen detekovat nový škodlivý kód, jak často se vyskytuje planý poplach, atd.

Co se týká schopnosti detekce v souborech, ClamAV je schopen pracovat s různými druhy spustitelných souborů a dokumentů, s různými formáty e-mailových úložišť a dokáže kontrolovat i archivy (pro některé - např. ZIP nebo RAR - ovšem potřebuje externí podporu), a to i rekurzivně [36].

5.4.1.3 Kontrola elektronické pošty

Jedním z hlavních úkolů pro ClamAV bývá kontrola e-mailového provozu na serveru. Většinou to funguje tak, že SMTP server přijme zprávu, tu nechá zkontrolovat programem ClamAV na případnou přítomnost virů, a pak s ní nějak dál naloží. Pokud je oním SMTP serverem Sendmail, lze využít filtr *clamav-milter*. Obecně se ale většinou používá řešení založené na programu AMaViS (typicky amavisd-new), který rozebere každou zprávu na jednotlivé součásti a ty pak předkládá k virové (zde ClamAV) a spamové (např. SpamAssassin) kontrole.

Poštu lze kontrolovat v zásadě dvojím způsobem. Buď běží démon `clamd` a tomu se data předkládají ke kontrole, anebo lze spouštět skenovací program (`clamscan`) vždy na jednotlivé zprávy. Kromě výjimečných případů je vhodnější metoda první, protože není potřeba pokaždé znovu načítat obsáhlou virovou databázi.

V linuxových distribucích to bývá běžně tak, že není potřeba se starat o kooperaci mezi programy AMaViS a ClamAV, konfigurační soubory AMaViSu jsou již připravené pro různé antivirové programy, stačí jen případně změnit parametry, je-li třeba [36].

5.4.1.4 Ruční kontrola souborů

Ruční kontrola je nejsnazším způsobem použití programu ClamAV. Jednou z možností je použít grafické rozhraní, ale i z příkazové řádky se program ovládá poměrně intuitivně. Zde je několik příkladů:

```
clamscan
clamscan podezrely_program
clamscan /home/zdenek
generator | clamscan -
```

První příkaz zkontroluje všechny soubory v aktuálním adresáři. Druhý provede kontrolu určeného souboru a třetí kontrolu souborů v zadaném adresáři (pouze v něm, ne v podadresářích). Konečně čtvrtý příkaz znamená, že `clamscan` bude přijímat data ze standardního vstupu, kam půjdou přes rouru (pipe) z programu *generator*. Všechny tyto příkazy provedou standardní kontrolu, tedy s výchozím nastavením.

Pokud potřebujeme ovlivnit chování skeneru, je k dispozici řada parametrů, kterými toho lze dosáhnout - například takto:

```
clamscan -r /home/zdenek
clamscan --move=/tmp/viry
clamscan -i -bell
```

Na prvním řádku je mírná modifikace příkazu pro kontrolu určeného adresáře - v tomto případě bude kontrola rekurzivní, tedy včetně podadresářů. Druhý příkaz přesune infikované soubory do adresáře */tmp/viry*, třetí pak hlásí jen infikované soubory (jinak se hlásí výsledek kontroly každého souboru) a navíc ještě při nálezu infekce zvukově upozorní.

Kromě použití skeneru `clamscan` máme také možnost předávat soubory ke kontrole běžícímu démonu `scand`, což značně urychluje skenování. Démon ale samozřejmě musí běžet, navíc nelze používat externí programy pro rozebírání archivů a některé parametry nelze nastavovat (démon pracuje podle své konfigurace, uložené obvykle v souboru /etc/clamav/clamd.conf). Možné příklady:

```
clamdscan
clamdscan podezrely_program
clamdscan /home/zdenek
clamdscan -m /home/zdenek
```

Jde o obdobu prvních tří příkazů ukázaných u skeneru `clamscan`. Opět se kontroluje aktuální adresář, resp. zadaný soubor a zadaný adresář. Ovšem pozor - v tomto případě se adresář skenuje vždy rekurzivně. Poslední příkaz ukazuje drobnou obměnu prováděné kontroly. Démon totiž rozlišuje několik různých režimů. V tomto případě se použije tzv. *multiscan*, což znamená, že se kontrola paralelizuje do více vláken (vhodné na systémech s více procesory či procesorovými jádry) [36].

5.4.1.5 Automatická (plánovaná) kontrola

Automatická kontrola podle plánu se prakticky nijak neliší od kontroly ruční. Lze opět použít buď `clamscan` nebo démon `clamdscan`. Je každopádně žádoucí provádět kontrolu (ať už kterýmkoli způsobem) se sníženou prioritou procesu, vzhledem k procesorové náročnosti, a případně (pokud je to možné) též s nižší I/O prioritou [36].

Samotné spouštění může probíhat pomocí daemonu `cron (anacron)`, případně z plánovacího nástroje grafického prostředí. Lze si představit také automatickou kontrolu všech nově vznikajících souborů v určitém adresáři (třeba při FTP uploadu), spouštěnou například programem `incron`.

5.4.1.6 Kontrola při přístupu k souboru (On-access skener)

Antivirové programy se mnohdy používají tak, že kontrolují soubor v okamžiku, kdy ho někdo otevírá (tzv. "on-access" kontrola) - tak mohou zachytit nebezpečný kód dříve, než se stihne spustit. Toto umí i ClamAV, byť to není úplně triviální.

Problém je totiž v tom, že přístup k souborům řeší jádro, kdežto ClamAV je běžný aplikační program. Řešeno je to tedy tak, že je v jádře speciální modul (Dazuko - původně vyvinutý firmou Avira právě pro účely virové kontroly), který umožňuje antivirovému programu získat řízení v okamžiku, kdy se někdo snaží otevřít soubor. Pak lze provést kontrolu a případně otevření souboru zabránit. Chceme-li tedy umožnit on-access skenování prostřednictvím ClamAV, je třeba vložit do jádra systému rozhraní modulu DAZUKO (ovladač dazuko.ko) pro řízení přístupu k souborům – viz Instalace Dazuko modulu.

Instalace Dazuko modulu (Dazuko & ClamAV & Debian GNU/Linux)

Jestliže chceme používat *On-Access* antivir pod Debianem, nejspíše sáhneme po Dazuko modulu. Modul se instaluje ze zdrojového kódu. Na oficiální stránce projektu Dazuko je návod pro OS Debian.

Stažení a kompilace

V Debianu je zdrojový kód dostupný pomocí „apt“. Na stránkách projektu je ale doporučení, stahovat modul přímo z oficiálního zdroje. Můžeme si tedy vybrat. Buď provedeme **'apt-get install dazuko-source module-assistent'** nebo modul stáhneme z The Dazuko project a nainstalujeme pomocí **'dpkg -i dazuko-source_xxxxx_all.deb'** (např. *dazuko-source_2.3.3-1_all.deb*). V případě ruční instalace nebudou splněny závislosti. Spustíme proto **'apt-get -f install'** a ještě doinstalujeme **'module-assistant'**.

Kompilaci provedeme pomocí **'sudo m-a a-i dazuko'**. Nyní je modul připraven k zavedení. Systém však zdaleka ještě ne.

Konfigurace systému

Aby se modul korektně zavedl, musí se spouštět před zavedením modulu *capability*. Pro okamžitý test funkčnosti provedeme:

```
sudo rmmod capability
sudo modprobe dazuko
sudo modprobe capability
```

Pokud vše proběhlo bez problémů a systém je stabilní, můžeme nastavit zavedení Dazuka při startu. Vytvoříme konfiguraci Dazuko modulu: **'gedit /etc/modprobe.d/dazuko'** a do souboru napíšeme:

```
install dazuko modprobe -r capability; \
modprobe -i dazuko; \
modprobe -i capability
```

Následně ještě na konec souboru **'/etc/modules'** dopíšeme *dazuko*. Tím máme funkční modul Dazuko pro libovolný antivir.

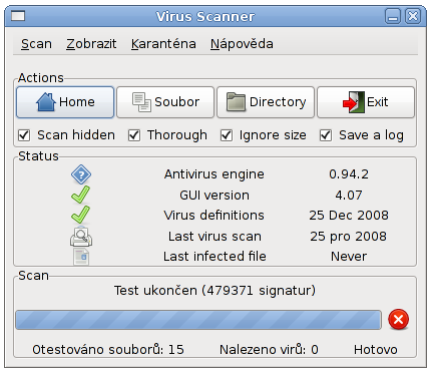
Modul `dazuko` komunikuje s ClamAV tak, že vytvoří speciální zařízení, které si ClamAV otevře. Na straně ClamAV toto zajišťuje komponenta Clamuko, která vytvoří zvláštní vlákno v rámci `clamd`.

Poznámka: U antivirového software je nutné zajistit minimálně chod modulu, starající se o nepřetržitou kontrolu souborů, se kterými uživatel nebo aplikace manipuluje (které otevírá / ukládá / spouští apod.). Takové moduly se většinou označují pojmy jako "rezidentní štít" či odborně "on-access skener". Je nutné si uvědomit, že tento modul je nejdůležitější součástí každého antivirového systému. Ve většině případů je sice nabízen i modul, který kontroluje přímo stahovanou či odesílanou poštu, ale i tento lze v nouzi oželet, jelikož tak jako tak by případné infekci zabránil on-access skener (až na některé výjimky způsobené zneužitím chyby v programu). Sice by virus neodchytil předtím, než bude zařazen do schránky doručené pošty, ale odchytil by ho v momentě, kdy by se uživatel pokusil otevřít - spustit infikovanou přílohu e-mailu. A jelikož příloha není nic jiného než soubor a otevírání a spouštění je činností, kterou sleduje on-access skener, není posléze co řešit.

Existují ještě další druhotné metody antivirové kontroly při přístupu. Patří mezi ně například mod_clamav (antivirový modul do HTTP serveru Apache) nebo Avfs (*An On-Access Anti-Virus File System* - souborový systém určený pro antivirovou kontrolu) [36].

5.4.1.7 Grafická uživatelská rozhraní

Pro ClamAV existují dvě grafická uživatelská rozhraní pro Linux: KlamAV a ClamTk. První z nich je určeno pro desktopové prostředí KDE, druhé pro GNOME a další prostředí založená na GTK+. Při přítomnosti příslušných knihoven lze programy samozřejmě spouštět i v "cizích" prostředích [36].




```
zdenek@debian:~$ clamscan --tempdir=/etc/home
/home/zdenek/.nessusrc.cert: OK
/home/zdenek/.bashrc: OK
/home/zdenek/.nessusrc: OK
     ...          ...
/home/zdenek/.Xauthority: OK
/home/zdenek/.ICEauthority: OK
/home/zdenek/.profile: OK

----------- SCAN SUMMARY -----------
Known viruses: 479371
Engine version: 0.94.2
Scanned directories: 1
Scanned files: 15
Infected files: 0
Data scanned: 0.43 MB
Time: 2.252 sec (0 m 2 s)
```

Obr. 40: Příklad antivirového programu v OS Linux (vlevo GUI, vpravo CLI – příkazový řádek). V uvedeném případě se jedná o grafický frontend pro ClamAV v prostředí GNOME (ClamTk).

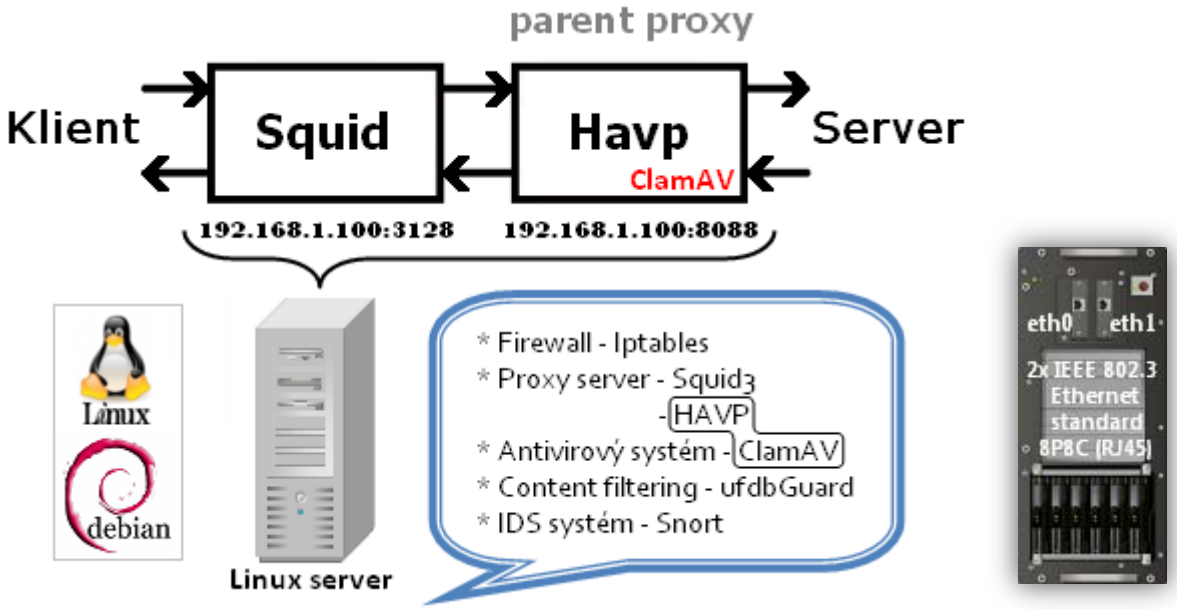
5.4.1.8 Shrnutí

Schopnosti a odladěnost programu jsou na více než dobré úrovni. Kvalitu detekce je vzhledem k chybějícím informacím těžké posuzovat. Pomohly by objektivní testy, pokud by je někdo udělal (resp. do běžných testů ClamAV zahrnul). ClamAV je například součástí řešení IBM Power Systems, takže lze předpokládat, že si dodavatelé určité testy důvěryhodnosti a spolehlivosti provedli, a ohodnotili produkt jako dostatečně způsobilý pro zmíněné nasazení. Hlavním polem působnosti programu ClamAV se stává průběžná kontrola pošty, kontrola při přístupu k souborům a také plánovaná kontrola souborů na disku. Díky tomu, že je ClamAV Open Source, je nasazován v mnoha enterprise serverových prostředích. Odpovídá to zaměření ClamAV spíše jako doplněk pro serverové produkty, kde je konzolová aplikace vhodnou volbou (ve výchozím nastavení se přistupuje pouze přes příkazovou řádku). Pro správné fungování rezidentního štítu ClamAV je nutný Dazuko modul pro kernel. V souhrnu se jedná o velice schopný antivirový program, jehož hlavní výhody jsou jednoduchost, rychlost a výkonnost s minimálními nároky na systémové zdroje (RAM, CPU).

5.4.2 HAVP (HTTP AntiVirus Proxy)

5.4.2.1 Konfigurace HAVP se Squid a ClamAV

HAVP (HTTP AntiVirus Proxy) je proxy s ClamAV antivirovým skenerem. Hlavním cílem je kontrola stahovaných souborů a skenování dynamického HTTP provozu. HAVP může být použit se Squidem (častá kombinace) anebo samostatně. Propojení ClamAV se Squidem (typicky nastaven jako transparentní proxy) lze uskutečnit právě přes HAVP, které umožňuje přímo využít ClamAV a sloužit Squidu jako rodičovská proxy. Ke své činnosti potřebuje mandatory locking. Je tedy třeba mít volnou diskovou oblast (partition) nebo dle manuálu namountovat kořen stromu „/“ v případě nedostatku volného místa. Na úvod bych chtěl poukázat na výhody zařazení dalšího proxy serveru (HAVP) do počítačové sítě. Předností tohoto produktu je, že dokáže kontrolovat (např. kromě e-mailů pomocí samotného ClamAV) i uživateli prohlížené stránky a stahované soubory z Internetu do vnitřní sítě na přítomnost virů a dalšího škodlivého kódu [41].



Zjednodušený chronologický postup [41]:

1.) Předpoklad již provozovaného, funkčního a správně nastaveného linuxového firewallu
2.) Instalace Squid, HAVP, ClamAV (pomocí apt-get install nebo kompilace binárního balíčku)
3.) Konfigurace Squid, HAVP, ClamAV (detailně uvedena níže u každého programu zvlášť)

Pro Debian postačuje v základě následující konfigurace proxy (pozn. jedná se o obecné nastavení):

- Konfigurační soubor HAVP (*/etc/havp/havp.config*)
  - BIND_ADDRESS 127.0.0.1 (ochrana proti přístupům mimo "localhost")
  - LOG_OKS false (zajímají nás pouze virové alerty)
  - odstranění řádku #REMOVETHISLINE deleteme //aktivace
- Konfigurační soubor SQUIDu (*/etc/squid(3)/squid.conf*)
  - http_port 3128 transparent
  - cache_peer 127.0.0.1 parent 8080 0 no-query no-digest no-netdb-exchange default
  - cache_peer_access 127.0.0.1 allow all

4.) Vytvoření příslušného pravidla v Iptables – je využíván transparentní režim proxy Squidu
5.) Vytvoření mandatory lock, následné připojení zařízení pomocí příkazu „mount“
6.) Restartování programů kvůli uplatnění změn v konfiguraci (/etc/init.d/*nazev_programu* restart)
7.) Otestování konfigurace a případné doladění systému

❖ Konfigurace cache proxy **Squid**

Squid bude všechny požadavky přesměrovávat na proxy server HAVP, který je pro Squid parentní proxy.

```
# port, který bude vyhrazen pro proxy-server
http_port 192.168.1.100:3128 transparent
acl all src 0.0.0.0/0.0.0.0
cache_peer 127.0.0.1 parent 8088 0 no-query no-digest no-netdb-exchange default
cache_peer_access 127.0.0.1 allow all
#pouze provoz HTTP bude skenován
acl Scan_HTTP proto HTTP
never_direct allow Scan_HTTP
# jméno počítače
visible_hostname proxy
# velikost paměti cache
cache_mem 32 MB
# nastavení uvolňování diskového prostoru
cache_swap_low 90
cache_swap_high 95
# nastavení velikosti místa na disku pro odkládací prostor
# číslo 100 je velikost v MB, další čísla 16 a 256 jsou rozsáhlost adresářové struktury
cache_dir ufs /var/spool/squid3 100 16 256
# zákaz logování přístupu (spotřebovává příliš mnoho diskového prostoru)
cache_access_log none
# logování informací o odkládacím prostoru
cache_log /var/log/squid3/cache.log
cache_store_log none
# nastavení uživatele, pod kterým se „cachuje“
cache_effective_user squid
cache_effective_group squid
http_access allow all
```

❖ Konfigurace antivirové proxy **HAVP**

```
USER havp
GROUP havp
DAEMON true
PIDFILE /var/run/havp/havp.pid
SERVERNUMBER 8
MAXSERVERS 100
ACCESSLOG /var/log/havp/access.log
ERRORLOG /var/log/havp/havp.log
LOG_OKS false
TRANSPARENT false
PORT 8088
BIND_ADDRESS 127.0.0.1
WHITELIST /etc/havp/whitelist
BLACKLIST /etc/havp/blacklist
FAILSCANERROR true
SCANIMAGES true
ENABLECLAMD true
CLAMDSERVER 127.0.0.1
CLAMDPORT 3310
```

Spuštění samotného proxy serveru se provede následujícím příkazem: havp -c /etc/havp/havp.conf

❖ Konfigurace antivirového systému **ClamAV**

Do souboru /etc/clamav/clamd.conf vložíme následující konfigurační řádky:

```
TCPSocket 3310
TCPAddr 127.0.0.1
```

Poté je nutné spustit daemona CLAMD pomocí příznačného příkazu "clamd".

❖ Nastavení firewallu **Iptables**

Jelikož je proxy server Squid použit jako transparentní proxy, je nutné nastavit Iptables tak, aby byl veškerý provoz na portu 80 (HTTP) automaticky přeposílán tomuto proxy serveru. Toho docílíme pomocí příkazu:

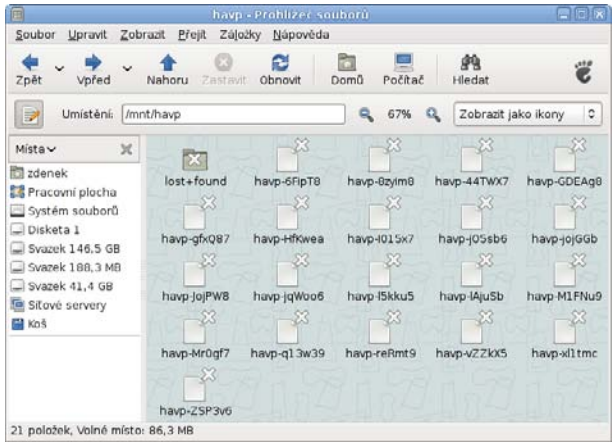
```
iptables -t nat -A PREROUTING -t tcp -i eht1 --dport 80 -j REDIRECT --to-port 3128
```

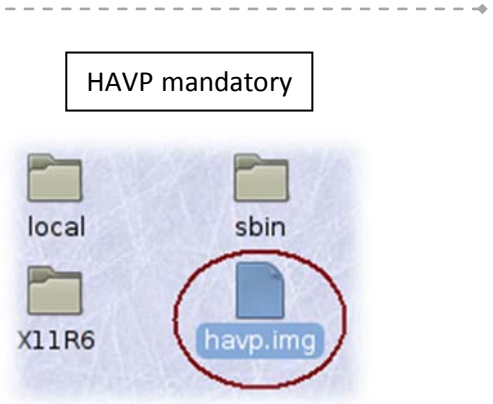
❖ Vytvoření odkládacího zařízení potřebného pro činnost HAVP

HAVP proxy potřebuje ke své práci souborový systém „přimountovaný“ v režimu mandatory lock. Takto je možno připojit některý diskový oddíl, nebo vytvořit virtuální disk v podobě souboru. K tomu poslouží příkazy:

```
# Vytvoření souboru (zařízení) o velikosti 4 GB. Parametr seek slouží k definici tzv. řídkého souboru.
# Díky tomuto parametru nezabírá soubor na disku celé 4 GB, ale podstatně méně (jeho velikost roste s objemem dat).
dd if=/dev/zero of=/usr/havp.img bs=1M count=1 seek=4096
# Ve vzniklém souboru vytvoříme souborový systém EXT3.
mkfs.ext3 /usr/havp.img
# Vzniklý virtuální disk připojíme jako LOOP zařízení v režimu mandatory lock do adresáře /var/tmp/havp
mount -o loop,mand /usr/havp.img /var/spool/havp
```

```
debian:/home/zdenek# mount -t ext3 /dev/loop0 /mnt/havp -o mand
```



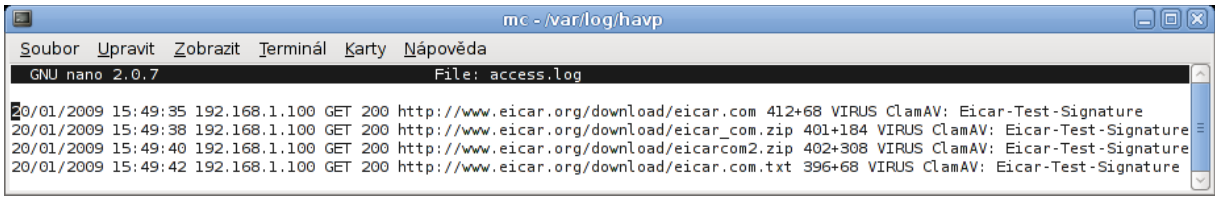


Zařízení	Adresář	Typ	Celkem	Volné	K dispozici	Použito	
/dev/sdb1	/	ext3	290,9 GiB	263,4 GiB	248,6 GiB	27,5 GiB	9 %
/var/lib/havp/havp.loop	/var/spool/havp	ext3	96,8 MiB	91,3 MiB	86,3 MiB	5,5 MiB	6 %

Na adrese 192.168.1.100 naslouchá na portu 3128 již ověřený Squid, nyní k němu přibyla antivirová proxy HAVP (implementační rozhraní pro antivirový program ClamAV) ve funkci předřazené parentní proxy, naslouchající na portu 8088 téhož stroje (lze aplikovat i IP adresu místní smyčky 127.0.0.1).

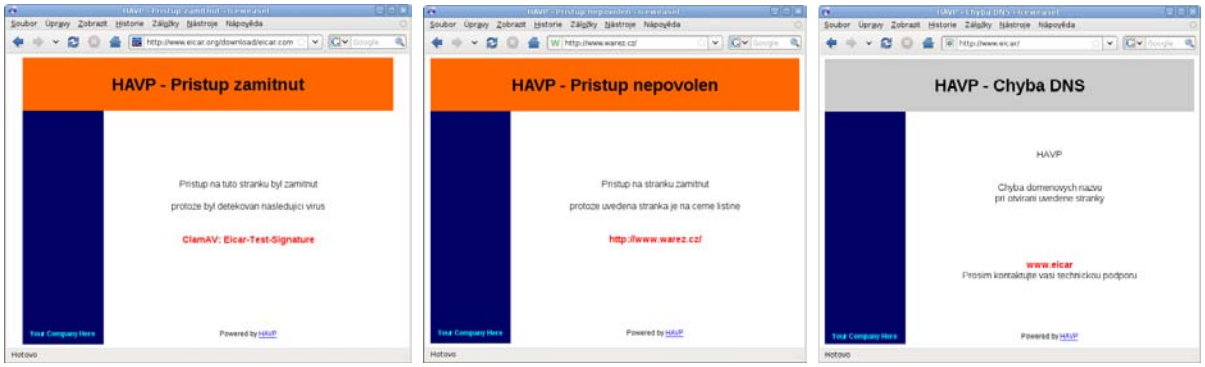
```
debian:/home/zdenek# netstat -tap
Aktivní Internetová spojení (servery a navázaná spojení)
tcp        0      0 192.168.1.100:3128      *:*     LISTEN
3718/(squid)
tcp        0      0 192.168.1.100:8088      *:*     LISTEN
3502/havp
```

Testování uplatněné konfigurace je možné pomocí EICAR řetězců. Ty je možné načíst z webové adresy bez ohrožení stanice, protože se jedná o fiktivní škodlivý kód. Pokud je konfigurace Squid & HAVP & ClamAV provedena správně, objeví se obrazovka podobná obrázku 42 (obrázek vlevo), tedy HAVP vypíše hlášku, že přístup ke stránce je zamítnut, protože je infikována virem. V opačném případě (dojde k otevření/stáhnutí souboru) není činnost v pořádku a je třeba hledat příčinu [41].



Obr. 41: Ukázka obsahu přístupového souboru HAVP proxy, kde dochází k zaznamenávání činností

Log přístupu na webové stránky s následným stáhnutím čtyř zavirovaných (testovacích) souborů je znázorněn na obrázku 41. K samotnému stáhnutí souborů (downloadu) ovšem nedošlo, v souborech byly pomocí antivirového programu ClamAV detekovány škodlivé kódy a přístup byl tedy prostřednictvím proxy HAVP zablokován. Uživatel je o této události informován přesměrováním www provozu na statickou HTML stránku s vysvětlujícími informacemi, jménem nalezeného viru a následnými pokyny k dalšímu pokračování. Některé z možností, které mohou nastat, a se kterými se uživatel může setkat (např. na rizikových WWW), jsou zaznamenány na obrázcích níže – viz Obr. 42.



Obr. 42: Ověření správnosti konfigurace – vlevo detekce viru následovaná zamítnutím přístupu, uprostřed zákaz přístupu na stránky obsahující warez, vpravo chybně napsaný doménový název (www.eicar.org), HAVP upozorní na chybu DNS

Jelikož HAVP nedisponuje tak vyspělým a sofistikovaným „content scanningem“, čili filtrováním internetového obsahu (webu) – poskytuje pouze základní formu filtrování, nebude pro tuto činnost použit. Ve srovnání s produkty jako Dansguardian, ufdbGuard (URLfilterDB) nebo SquidGuard, které se přímo specializují na content filtering, obstojí svými funkcemi pouze průměrně, a proto bude dána přednost některému ze jmenovaných, konkrétně velmi rychlému URL filteru pro Squid ufdbGuard.

HAVP je nutné (podobně jako Squid) dostatečně odladit. Vzhledem k tomu, že se ve výchozím nastavení kontroluje téměř vše, dochází občas k velkému zpoždění, které se projevuje nadměrnou odezvou mezi klientem a proxy. Uživatel toto vnímá jako měřitelné zpomalení síťových služeb. Proto je třeba vybrat konkrétní typy dokumentů a souborů, které budou procházet kontrolou, a které se naopak tomuto skenování vyhnou. Podpora formátů typu „rar“ a podobných není z důvodů autorských práv zahrnuta. Problematickým je i HTTPS protokol – přenos dat, tedy i škodlivého kódu, je realizován v zašifrované (pro potenciálního útočníka nečitelné) podobě. Šifrováním je zajištěna důvěrnost dat. Antivirový program není schopen rozpoznat virus v zašifrované stránce. Se stejným problémem se potýkají i některé z metod pro filtrování (regulaci) internetových zdrojů založené na obsahové kontrole, tj. vyhledávání definovaných slov na webové stránce. Kvůli zašifrovanému textu není kontrola na výskyt slov účinná. V dalším je uvedena názorná ukázka, která vše objasňuje.

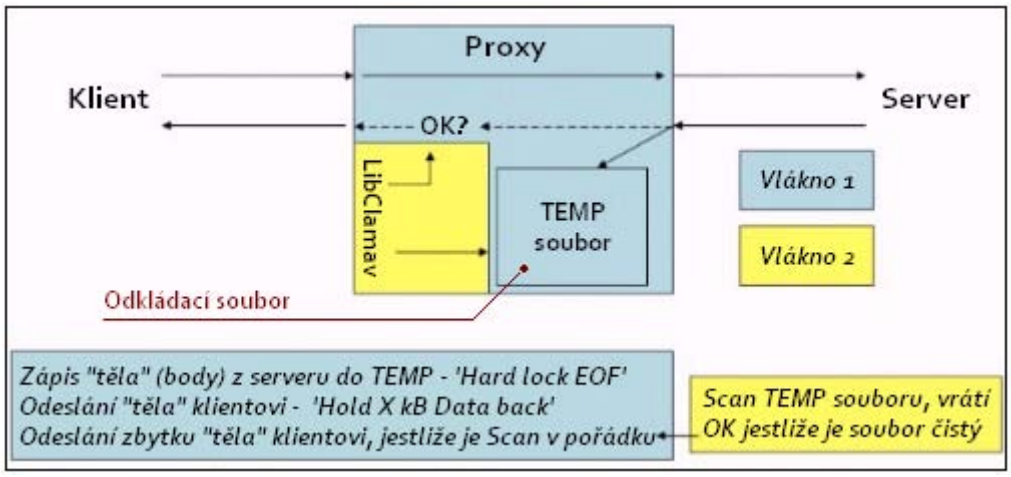
Ukázka: OT = text v otevřené podobě (plain-text), ŠT = text v zašifrované podobě (crypt-text)

OT: „Toto je ukázka textu zašifrovaná pomocí TLS, podobně jako u protokolu HTTPS.“
ŠT: iD8DBQFB3HDHQMFfY0HyOFURAhRjAJ42kzLuQ+xrxIxt5+wBSdZSImW5iACcC4/aiY9qrYQ/296gLLB5jXXkMWI

Antivirový program z této změti znaků (ŠT) nerozpozná škodlivý kód přenášený ze serveru na klienta.

V některých případech nemusí, i při správné konfiguraci parent proxy HAVP s cache proxy Squid a antivirem ClamAV, docházet k detekci škodlivého software. Na vině bývá většinou již zavirovaná cache Squidu nebo prohlížeče internetových stránek. V druhém případě je jednoduchá náprava v podobě vymazání dočasné paměti prohlížeče („clear browser cache“). První případ musí neprodleně řešit správce systému, aby nedošlo k postupné infikaci ostatních počítačů připojených na proxy.

HAVP obecně podporuje parentní proxy, umožňuje transparentní proxy mód a zajišťuje i částečně podporu protokolu FTP. Podporován naopak není HTTPS protokol, neboť je obtížné kontrolovat provoz zabezpečeným spojením (proxy neumí dešifrovat data přenášená komunikačním kanálem). Výhodou zprovozněného řešení je simultánní start skenování se začátkem stahování. Problematické jsou komprimované soubory, jež jsou extrahovány až po úspěšném downloadu stanicí. Nevýhodou je relativně pomalý skenovací proces u velkých souborů, kdy může dojít ke stavu „zadržování dat“. Známo je i několik problémů bezpečnostního charakteru, např. potenciální virová nákaza způsobená souborem, jehož velikost překračuje pole definující "hold back data". Pracuje se na zakomponování.



Obr. 43: Vnitřní struktura proxy HAVP sloužící k antivirové kontrole dat

U těchto a podobných řešení se jedná pouze o ochranu perimetru. Drobným handicapem HAVP proxy je, že nedokáže implementovat, resp. poskytnout rozhraní, např. k antispamovým filtrům a dalšímu bezpečnostnímu software (omezení je částečně kladeno i na antivirové produkty). Ale od toho jsou zde jiné programy na platformě Linuxu, doplňující mozaiku zabezpečení serverů a počítačových sítí.

```
Virus Dialer-160 in file /mnt/havp/havp-glxQ47 detected!
```

Výňatek záznamu výpisu z logu, kterému předcházelo úmyslné a kontrolované stáhnutí souboru obsahující nežádoucí dialer při surfování na záměrně vybraných www stránkách s pochybným obsahem. Záznam zobrazuje detekovaný škodlivý program dialer, měnící způsob přístupu na Internet prostřednictvím modemu (přesměruje vytáčení čísla pro připojení na linky s vysokými sazbami). Jelikož se v uvedeném případě jedná o plnohodnotný vir a zároveň není záměrem poškodit hostující OS, je k těmto experimentálním účelům zprovozněn virtualizovaný systém (rovněž systém Debianu).

5.5 Filtrování obsahu (content filtering)

Internet představuje téměř nevyčerpatelnou studnici informací nejrůznějšího charakteru, které jsou většinou přístupné bez jakéhokoliv omezení či restrikce. K dispozici však nejsou jen užitečné informace, ale jsou zde k nalezení i věci nevhodné pro výchovu dětí, materiály s rasistickými a xenofobními podtexty nebo stránky, které ne vždy souvisí s právě probíhající pracovní činností zaměstnanců (od multimediálních zábavných stránek, přes komunitní sítě, až po hazardní online hry - gamblerství). Z hlediska politiky firmy/organizace/instituce se nepochybně jedná o nevhodný obsah, který by měl být v pracovní době zakázán a příslušným filterem zavčas blokován [43].

Z každoročně prováděných průzkumů vyplynulo, že cca 65% firemní šířky pásma je obsazeno neproduktivním využíváním. Např. 65% všech přístupů na servery s pornografií se odehrává ve všední den v době od 9:00 do 17:00 hodin. Content filtering není jen o omezování přístupu k tomuto typu informací, ale mnoho webových stránek je infikováno spyware či samotnými viry. Aby bylo možné se vyvarovat těmto nástrahám, je nezbytně nutné zařadit do sítě speciální filtrovací zařízení (HW) nebo nainstalovat, nakonfigurovat a zprovoznit program (SW), který se umí s tímto problémem vypořádat.

V případě hardwarových zařízení se často jedná o (před)placené služby, ke kterým je získán přístup po zakoupení příslušné i-Karty (iCard), která obsahuje přístupový kód k databázi. Toto řešení je taktéž často integrováno do řady bezpečnostních produktů pro malé, střední i velké firmy. Zařízení obsahují možnost ručně zadat buď klíčová slova, podle kterých je obsah filtrován, nebo seznam nedůvěryhodných domén či IP adres. Samozřejmostí bývá plná podpora poskytovaná prodávající společností. Výjimečně se s ním lze setkat i v domácnostech, např. jako součást ADSL routeru, kdy je filtrování obsahu limitováno časovým omezením (např. 30 dní), nebo je proces aktualizace databáze přístupný jen do určitého počtu (např. 30x), poté je filter nadále funkční, ale se zastaralou (neaktuální) databází. V obou případech tedy pouze na vyzkoušení a prvotního seznámení se s funkčností. Software určený k filtrování obsahu budu (detailněji) probírat dále.

Obr. 44: Hardwarově založený content filtering určený pro nasazení na serveru (montáž do „racku“)

Principiálně se nejedná o nic složitého. Požadavek uživatele na zobrazení webové stránky je nejdříve porovnán se seznamem zakázaných URL a následně proveden rozbor jejího obsahu (i rozbor obsahu pozadí), zda nespadá do některé ze zakázaných kategorií. Pokud je některá z podmínek splněna, přístup je zablokován a objeví se varovné hlášení, které je většinou možné předem nadefinovat, popřípadě přesměrovat uživatele na vybranou statickou webovou stránku, např. s vysvětlujícími pokyny a důvodem blokace. Metody, prostřednictvím kterých lze provádět filtrování nevhodného obsahu webu, jsou představeny dále [43].

5.5.1 Filtrovací metody pro blokování nežádoucích webových stránek

Zde uvedu tři filtrovací metody, které se používají pro blokování nechtěného (nežádoucího) webového obsahu. Content filtering může probíhat na základě [44]:

- **Skenování obsahu** – blokování webových stránek v případě, že obsahují (soubor/sadu) „zakázaných“ (rozuměno nevhodných, nemravných, nemorálních atd.) slov
- **Umělá inteligence (AI)** – zlepšená, resp. inovovaná verze filtrace pomocí skenování obsahu
- **Černá listina (blacklist)** – blokace www stránek na základě vytvářeného seznamu kategorizovaných webových stránek (URL) do příslušných oblastí - např. pro dospělé, stránky s rasistickým podtextem, stránky s pornografickým obsahem, hry a zábava, násilí, apod.

Následující kritéria jsou použita pro porovnání filtrovacích metod [44]:

- Zkušenost uživatele: metoda musí být dostatečně rychlá a bezpečná pro jednotlivé uživatele.
- Nesprávné (chybné) blokování I: v případě, že je neoprávněně blokována www stránka zabývající se např. tematikou *rakoviny prsu*, nazýváme takovýto stav termínem pocházejícím z anglického jazyka „overblocking“ (čili jakési blokování za/nad určitou pomyslnou hranicí).
- Nesprávné (chybné) blokování II: opačný případ výše zmíněného, www stránka se sexuálním obsahem je chybně (či vůbec) klasifikována a zařazena do nesprávné/žádné kategorie. Stránka tedy není blokována a je zobrazena ve webovém prohlížeči. Jestliže filtr v takovém případě selže, je to nazýváno příslušným anglickým termínem „underblocking“ (čili blokování pod určitou pomyslnou hranicí).
- Blokování HTTPS: protokol HTTPS je nadstavba síťového protokolu HTTP, která umožňuje zabezpečit spojení mezi webovým prohlížečem a webovým serverem před odposlouchávání, podvržením dat a umožňuje též ověřit identitu protistrany. Přenášená data jsou šifrována pomocí SSL nebo TLS. Protože protokol používá šifrování, nemůže být obsah prohledán a prozkoumán na vyskytující se slova nebo známé fráze.
- Náklady infrastruktury: jednotlivé součásti jsou náročné na výpočetní výkon (procesorový čas, spotřeba paměti, ad.) a využití potřebné šířky pásma.

5.5.1.1 Metoda A: skenování (prohledávání) obsahu

V případě, že jsou webové stránky prohledávány vzhledem k nevhodnému obsahu, jsou nejprve staženy z Internetu, což však stojí nějaký čas i konektivitu (resp. určitou šířku pásma). Poté je obsah stránky skenován vůči „zakázaným“ slovům jako například "erotic", "racist", "games", "gambler", "s*x", "s*ck", "f*ck", "p??no" apod., která patří mezi často vyhledávaná. S výhodou lze využít hvězdičkové konvence pro větší účinnost či variabilitu slov. Metoda je postavena nejen na vyhledávání slov, ale i nevhodných slovních spojení či definovaných frází, vyskytujících se na některých uživatelem požadovaných stránkách.

V závislosti na provedeném nastavení, jedno či více slov spustí blokovací mechanismus a nežádoucí obsah může být blokován (při shodě dojde k blokaci celé domény www). Teorie zní velice jednoduše, v praxi je avšak mnoho webových stránek blokováno, protože byla zjištěna kombinace slov jako v případě "I don't like sex – *nemám rád sex [internetová poradna]*", "breast cancer – *rakovina prsu [odborný článek]*" či "playing computer games is bad for your health – *hraní počítačových her škodí zdraví [dokument-studie, průzkum]*". Tyto i další případy patří do již zmíněného označení „overblocking“.

Na druhé straně, www stránky se sexuálním podtextem, které jsou obvykle reprezentovány pouze obrázky (text může být součástí obrázků), nejsou blokovány, protože buď neobsahují žádné zakázané slovo vyskytující se v této kategorii filtrování obsahu webu (př. sexuality, adult, …), nebo nedisponují vůbec žádným doprovodným textem (v neposlední řadě může být text přímo vnořen do samotného obrázku – rastru, text špatně čitelný či nečitelný vůbec). Tyto omylem neblokované stránky jsou souhrnně nazývány pojmem „underblocking“ [44].

Čas potřebný k prohledání a odhadu (zařazení) typu obsahu webové stránky (i z kontextu) je u každé stránky různý (některé stránky na Internetu jsou velmi rozsáhlé a obsahově bohaté). Tato metoda je dostatečně rychlá pro jednotlivé (samostatné) uživatele. Nicméně, pro 250 či více uživatelů, je vyžadován výkonný počítač (s dostatečnou rezervou propustnosti) pro proxy server.

5.5.1.2 Metoda B: umělá inteligence

Internetové stránky blokované na základě umělé inteligence (AI - *Artificial Intelligence*) jsou také nejprve staženy ze vzdálených webových serverů umístěných na Internetu, a poté detailně skenovány a prozkoumávány. Tato metoda tedy rovněž spotřebovává šířku pásma internetového připojení a čas potřebný k získání oněch stránek. Rozličné AI metody jsou komplexnější verzí zmíněné metody A (resp. vhodná kombinace předchozího řešení s prvky umělé inteligence).

K redukci nedostatků způsobených „underblockingem“ a „overblockingem“ jsou všechna slova nacházející se na webové stránce ohodnocena, spolu s některými slovními spojeními. Podstatou je tedy vylepšení klasického skenování obsahu těla www stránky. V některých softwarových řešeních lze integrovat s různými přídavnými moduly (i experimentálními), z čehož plyne, že se dozajista jedná o účinnou a efektivní metodu, která se neustále zdokonaluje a dynamicky rozvíjí. V budoucnu půjde možná o velmi perspektivní a vyladěné řešení filtrace webu a s ním služeb spojených. Použít se může v situacích, při nichž je např. webový portál *www.**v**se**or**akovine**p**rsu.cz* neprávem blokován (pochopitelně by neměl být součástí seznamu černé listiny), protože při zjišťování „škodlivosti“ či nevhodnosti stránky bylo v textu nalezeno slovo *prsu*. V textu se míní obecně v titulku, hlavičce, těle, vnořené tabulce nebo grafu, adrese linkového odkazu, reklamě, pozadí apod. V tomto konkrétním případě se jedná o neoprávněné blokování – stránka pojednává o problematice rakoviny prsu nikoli o pornografii, má být tedy veřejně přístupná všem. Výsledkem prověrky by tedy bylo falešně pozitivní posouzení. V těchto a dalších případech nalézá uplatnění uvedená metoda [44].

Dalším zdokonalením je prohledávání obrázků a fotografií (běžné formáty *jpg*, *bmp*, *png*, *tiff* a další) vůči textu, textovým polím. Na text, který je součástí určitého úseku obrázku (analogicky jako „viditelný vodoznak“), se aplikují různé rozpoznávací techniky – v některých ohledech jistá podobnost s prvky OCR čili optickým rozpoznáváním znaků. Z principu vyplívá, že jde o pomalejší postup. Některé produkty se navíc pokouší zjistit, jestli je obsahem přenášených obrázků např. nahé lidské tělo, sledováním barev a vnořených odstínů. Zanáší tak vyšší úroveň přesnosti a zvyšují správnost rozhodování. Toto zdokonalení přesnosti a korektnosti blokování přichází ovšem s větší režií - je třeba více procesorového výkonu. Takže pro 100 a více uživatelů je vyžadován opravdu velmi výkonný počítač pro proxy server (zejména syntetický výkon PC). Prim hrají vícejádrové a víceprocesorové systémy.

Vize blízké budoucnosti je zřejmě i proxy spolehlivě filtrující video služby (např. různé audiovizuální nahrávky a videoklipy na serverech typu YouTube aj., sdílené pornografické filmy, rasistické spoty).

5.5.1.3 Metoda C: blacklisting („černá listina“)

Jestliže jsou webové stránky blokovány za použití blacklistu, tak k samotnému učinění rozhodnutí blokovat nebo propustit takovou stránku není třeba stahování z Internetu. Místo toho URL filter modul proxy serveru velmi rychle rozhodne na základě URL adresy: www.sex.com – tato doména je blokována, www.google.com – tato doména není blokována. Blokování webových stránek se tedy děje na základě řetězce znaků s definovanou strukturou – URL, které slouží k přesné specifikaci umístění zdrojů informací (ve smyslu dokument nebo služba) na Internetu. Další příklady blokovaných stránek mohou být *www.erotic.com*, *www.hazard-games.net*, *www.alkohol&drogy.cz*. URL filter činí rozhodnutí na základě databáze, která je často označována jako **Blacklist** (tzv. černá listina, specifický seznam blokovaných www, roztříděno do kategorií, nejčastější) nebo **Whitelist** (tzv. bílá listina, opak předchozího, vysloveně povolené, tj. neblokované www stránky, minoritní použití).

Vzhledem k nestálosti, dynamické povaze a neustálému rozvoji (spousta klasifikovaných webových stránek a služeb zaniká, velká skupina jich naopak vzniká) musí být blacklistový seznam pravidelně aktualizován a udržován. Získaný soubor je většinou „univerzální“, takže je třeba jej většinou modifikovat a přizpůsobit potřebám vnitřní sítě a konkrétním požadavkům firmy/organizace. Tato metoda je velmi rychlá, protože blokované www stránky nejsou vůbec stahovány. Například aplikace dále realizovaného filtru ufdbGuard zvládne vykonat 25 000 URL ověření za sekundu na běžném stroji s CPU Intel 2,8 GHz [44].

Z tabulky je patrné, proč je metoda "černé listiny" (tzv. blacklistu) nejlepší volbou.

metoda	**zkušenost uživatele**	**správnost blokování**	**blokování HTTPS**	**rozšiřitelnost (přes 500 uživatelů)**	**náklady infrastruktury**
skenování obsahu	±	-	--	-	±
umělá inteligence	±	±	--	--	-
černá listina (blacklist)	++	+	++	++	+

Tab. 2: Srovnání jednotlivých metod filtrace

ICAP

ICAP (Internet Content Adaptation Protocol) je objektově zaměřený protokol založený na „odlehčeném“ HTTP a navržený k tomu, aby kontrolou datové zátěže paketů specifikoval obsah k přiřazeným serverům, čímž pomáhá uvolňovat zdroje a upravuje způsoby implementace. Protokol ICAP je obvykle využíván proxy servery k začlenění produktů třetích stran jako např. antivirového software, software pro filtrování škodlivého obsahu webů, URL filtrů, apod.

Content Vectoring Protocol

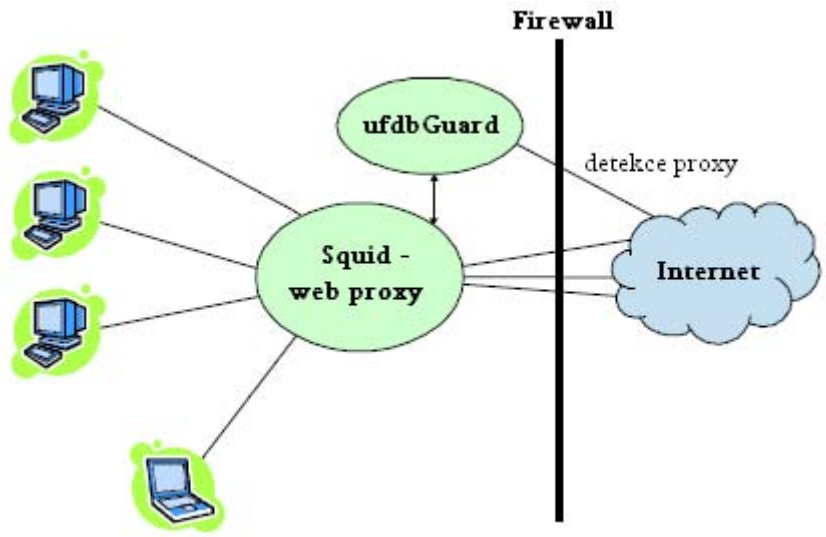
Jedním z častých řešení je umístění antiviru (a/nebo content filteringu) na firewall, případně proxy server na bázi CVP (content vectoring protocol), který slouží k přesměrování paketů na vyhrazený antivirový a/nebo content filteringový server umístěný v demilitarizované zóně. Ten má mechanismus založený na hodnocení obsahu stahovaných souborů a stránek. Protokol CVP přesměruje pakety na antivirový server či server zabývající se filtrováním obsahu, kde je provedena jejich kontrola. Následně jsou tyto pakety zaslány zpět firewallu a podle vyhodnocení jsou buďto firewallem propuštěny původně adresovanému příjemci, nebo ne. Přesměrovávání paketů probíhá na aplikační vrstvě a server, na kterém probíhá testování, by měl disponovat rychlými disky, procesorem, dostatkem paměti a být k firewallu připojen přímo.

Proxy server Squid verze 3.0 podporuje protokol ICAP a funkci ESI (Edge Side Includes).

5.5.2 UfdbGuard (URLfilterDB)

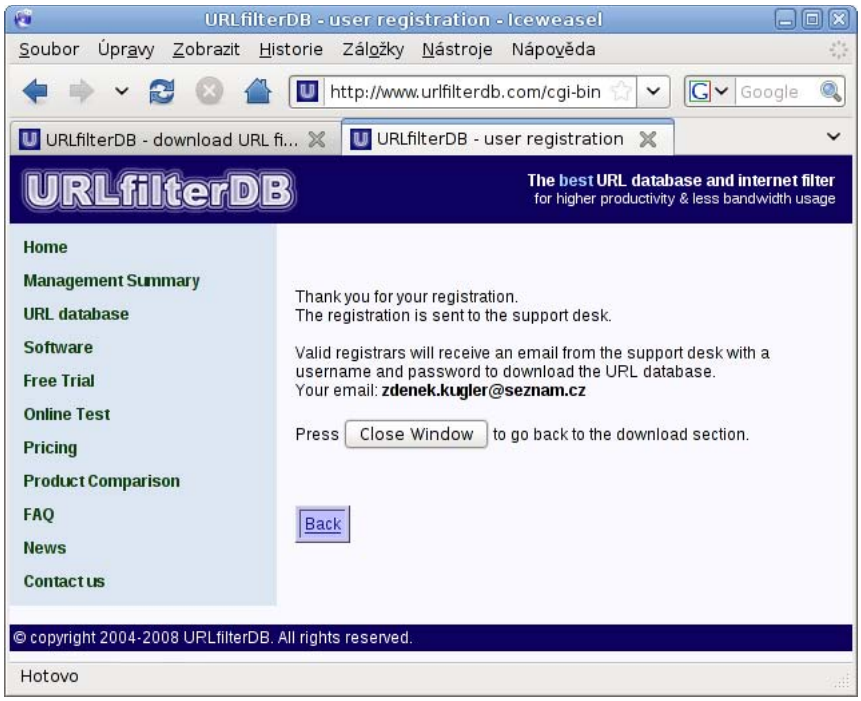
Bezpochyby zajímavou možností je rozšíření statického Squidu o další blacklisty. Jedním ze způsobů, jak toho dosáhnout, je pomocí URLfilterDB, který má oproti „základnímu“ SquidGuard několik výhod - je mnohonásobně rychlejší a umí obecně blokovat i HTTPS provoz. To umožňuje blokovat i stránky, které nemají certifikát vydaný známou důvěryhodnou certifikační autoritou (neboli certifikát je obvykle „self-signed“). Samozřejmostí je blokace IP adres místo doménových jmen. Na URLfilterDB je pozoruhodná také vlastnost, že program s blacklisty může běžet na úplně jiném serveru, než na jakém běží Squid [44].

URLfilterDB patří mezi jeden z nejčastěji používaných URL databázových a internetových filtrů (spolu s produkty DansGuardian, SquidGuard a jinými) pro zajištění vyšší pracovní produktivity, morálně-právní kázně a méně spotřebované šířky pásma (využité přenosové kapacity) komunikační linky. Software se skládá ze dvou komponent: programu, který je zdarma, a databáze, která je placená. Místo placené databáze lze však použít jakoukoliv databázi v textovém formátu (např. zdarma dostupné blacklisty, na které odkazuje SquidGuard). Volně dostupnou je alternativa ufdbGuard, což je v podstatě komerční produkt URLfilterDB bez placené databáze, kterou si lze obstarat i jinde. UfdbGuard lze bez URLfilterDB využívat, protože obsahuje nástroj, který umožňuje z otevřených textových souborů vygenerovat příslušné *.ufdb* soubory, kterým program rozumí. Naopak to neplatí, jelikož URL databáze odpovídá pouze formátu programu ufdbGuard. Squid je momentálně jediným proxy serverem, který ufdbGuard explicitně podporuje.



Obr. 45: Umístění a komunikace ufdbGuard v rámci proxy Squid

Někdy se může stát, že filtry (neoprávněně) blokují i některé „regulérní“ weby, jako např. http://www.jyxo.cz. Komplikacím se dá předejít tak, že program nejdříve (několik dní) běží v testovacím režimu, kdy nedochází ke skutečnému blokování stránek, ale v logu se vše zaznamenává. Pokud je třeba neprávem blokovanou stránku povolit, stačí jí obvykle přidat na seznam *alwaysallow* (tj. vždy povoleny). Toto řešení je mnohem snazší a méně časově náročné, než reportování (hlášení) chybných stránek v blacklistech jejich autorům. Opačnou situací je stránka, která je filterem propuštěna (vyhodnocena jako validní), a tedy internetovým prohlížečem zobrazena, ač by za normálních okolností měla být obsahem některé z filtrovacích kategorií, a tudíž blokována. Řešení je přiřazení na seznam *alwaysdeny* (tj. vždy zakázány).



Obr. 46: Registrace na webových stránkách programu URLfilterDB proběhla úspěšně (přináší s sebou výhody), nyní se stačí přihlásit uživ. jménem a heslem k účtu a obdržet databázi URL

Denní aktualizace databáze

Žádná metoda není dokonalá a nikdy stoprocentně nebude. Je to dáno rozsáhlým množstvím webových stránek vyskytujících se na Internetu, které jednoduše nemohou být všechny „ohodnoceny“ a bezchybně kategorizovány. Tato nedokonalost nečiní značný problém, pokud užívané metody blokují 99% nežádoucího obsahu a zbytečně neblokují obsah chtěný či validní. Aplikace ufdbGuard má vlastnost rozpoznávat URL *(Uniform Resource Locator – „jednotný lokátor zdrojů“)* adresy, které prozatím nejsou součástí URL databáze a poskytovat tyto adresy k tomu, aby mohly být zahrnuty v databázi již následující den [44].

Nejprve byla stáhnuta volně dostupná ukázková databáze přímo z domovských webových stránek programu (http://www.urlfilterdb.com), po vypršení zkušební licence poté databáze na adrese http://urlblacklist.com (soubor bigblacklist.tar.gz) - pouze na vyzkoušení a ověření funkčnosti. Stažení aktuální databáze URL filteru ufdbGuard příkazem „ufdbUpdate“ je zaznamenáno na řádcích níže.

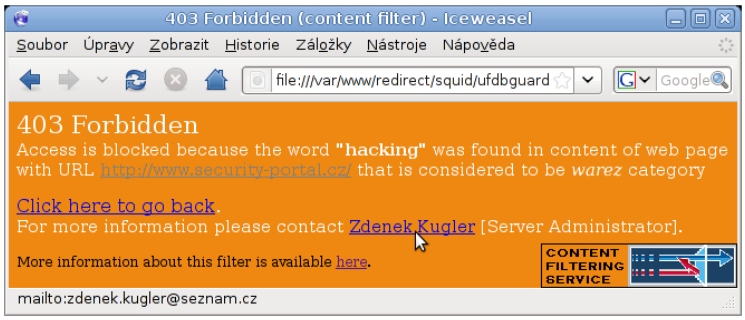
```
Download URL filter database and software
After registration as a trial user, a trial license may be issued by the support
desk. Follow the steps below for registration and implementation of URLfilterDB.
Download database *)use the ufdbUpdate command

debian:/usr/local/squid/bin# crontab -l
14 18 * * * /etc/webmin/cron/tempdelete.pl
00 22 * * * /usr/local/squid/bin/ufdbUpdate
15 6  * * * /usr/local/squid/bin/ufdbUpdate

new database downloaded:
-rw-r--r-- 1 root root 0 22. led 15.37 /tmp/urlfilterdb-latest.tar.gz
done.
```

Dále je možné využít volně dostupné databáze například MESD Blacklists nebo Shalla's Blacklists.

Tabulka představená dále uvádí přehled a popis programových součástí balíku ufdbGuard.

ufdbAnalyse	• nástroj pro statistickou analýzu	112,8 KB	Spustitelný soubor	So 17. leden 2009, 19:22:50 CET
ufdbgclient	• redirectory (lehké procesy)	270,0 KB	Spustitelný soubor	So 17. leden 2009, 19:22:49 CET
ufdbGenTable	• utilita ke generování .ufdb souborů	124,0 KB	Spustitelný soubor	So 17. leden 2009, 19:22:50 CET
ufdbguardd	• vícevláknový daemon	364,9 KB	Spustitelný soubor	So 17. leden 2009, 19:22:49 CET
ufdbhttpd	• "odlehčený" proces http daemon	168,0 KB	Spustitelný soubor	So 17. leden 2009, 19:22:50 CET
ufdbUpdate	• skript zajišťující aktualizaci	7,5 KB	Skript shellu	Čt 22. leden 2009, 15:37:46 CET

Ukázka pokusu o přístup k internetové stránce (cílem je klasické zobrazení stránky uživatelem), která se vyskytuje na seznamu zakázaných stránek (blacklistu) nebo je v datové struktuře stránky nalezeno některé z klíčových hledaných slov (uvedený případ) je zobrazena na obrázku 47.



Obr. 47: Přístup na *www.security-portal.cz* blokovaný na základě nálezu slova „hacking“ v obsahu

UfdbGuard je jeden z nejrychlejších obsahově zaměřených internetových filtrů, který může být využit v úzké součinnosti se Squidem (jako redirector, nastavení se provádí direktivou `redirect_program`, včetně souvisejících parametrů). Je poskytován zcela zdarma. Je možné jej doplnit databázemi třetích stran – komerčními i volně dostupnými. Program rovněž umožňuje dynamickou detekci HTTPS proxy tunelů, a proto zvyšuje bezpečnost na síti (protokol HTTPS může být využit různými typy proxy tunelů). Verze 2.0 bude podporovat protokol ICAP, všechny webové proxy pak mohou začít využívat ICAP k filtraci URL. Filter může být plně funkční a přitom pracovat v doplňkovém testovacím módu, který přesně ukazuje, které stránky mohou být potenciálně blokovány, aniž by k tomu skutečně došlo.

Kategorie filtrů
Inzeráty, reklama, tisk
Pornografie & sex
Audio & video
Chat & fórum
Osobní (citlivá) data
Drogy
Zábava
Externí aplikace
Finance & investice
Rasismus
Gamblerství
Hry (online)
Hacking & warez
Práce
Novinky
Obchody
Sport
Toolbary
Cestování
Násilí
Web Proxy & tunely
Blogy & sociální sítě
Internetový e-mail

Implementací internetového filtru lze dosáhnout blokování přístupu na stránky s pornografickým obsahem, reklamou, zábavou, zpravodajstvím, hazardem, rasistickým podtextem, chatovacími portály, aj., dále rozšíření bezpečnosti, poklesu rizika spojeného s porušováním zákonů, zvýšení produktivity zaměstnanců, redukce nákladů za infrastrukturu, omezení (snížení) virové nákazy, blokování HTTPS tunelů a vynucení užívání validních SSL certifikátů. Samostatnou kapitolou content filteringu, jakožto obsahově zaměřeného systému, je blokování reklamních bannerů na Internetu.

Při instalaci a konfiguraci programu ufdbGuard (verze 1.15) jsem přednostně vycházel z referenčního manuálu od tvůrců programu (soubor ReferenceManual.pdf), kde je vše přehledně popsáno. I když nelze zajistit 100% úspěšnost filtrování, aplikace regulujícího filteru se jistě vyplatí - vratná investice.

5.6 Systémy detekce průniků (IDS)

5.6.1 Úvod, představení IDS

Skenery: -systémové (přímo souvisejí s logy, analyzátory logů)
-síťové

Systémové skenery

U těchto skenerů je často jedná o formu analyzátoru logů (logwatch, logcheck) nebo komplexnějších skenerů (tiger, tripwire, aide). Kontrolují mimo jiné integritu souborového systému. Další možností jsou specializované skenery (sxid), samotnou částí jsou detektory rootkitů, jako např. chkrootkit.

Síťové skenery

Promiskuitní režim: -vyřazení MAC filtru na síťové kartě
-odchycení veškerého provozu na lokální síti
-příkladem sniffery: tcpdump, wireshark, dsniff a další

Analýza veškerého síťového provozu buď pomocí snifferu nebo přímo nástroje k tomu určenému, např. Nessus (také musí pracovat jako sniffer). Jednou z dalších možností je použití skenerů, např. nmap, NetCat, ad. Skenery lze detekovat například pomocí nástrojů scanlogd nebo portsentry.

IDS

Principem činnosti IDS je sběr informací z různých částí IS a analýza informací o výskytu příznaků bezpečnostních problémů. V podstatě obdoba detektorů skenování s možností podnikání dalších kroků. Rozdělují se na pasivní (pouze informace o skenování) a aktivní, reaktivní (informace o skenování + nápravné kroky – jako je zavedení příslušných pravidel do firewallu nebo routovací tabulky). Mohou pracovat i centralizovaně (senzory <-> server) s možností definování pravidel a výjimek. V linuxovém prostředí je nejznámějším zástupcem program Snort, kterému bude dále věnována pozornost, nebo např. PSAD [38].

Pouze velmi krátce bych zmínil program *DenyHosts*, což je jeden ze specializovanějších zástupců IDS. Spolupracuje s SSH, kontroluje počet nesprávných přihlášení na superuživatele a pokusy o přihlášení na neexistující uživatele. Nekorektní pokusy blokuje prostřednictvím /etc/hosts.deny. Lze jej s úspěchem využít při obraně proti útokům hrubou silou („brute force attack“).

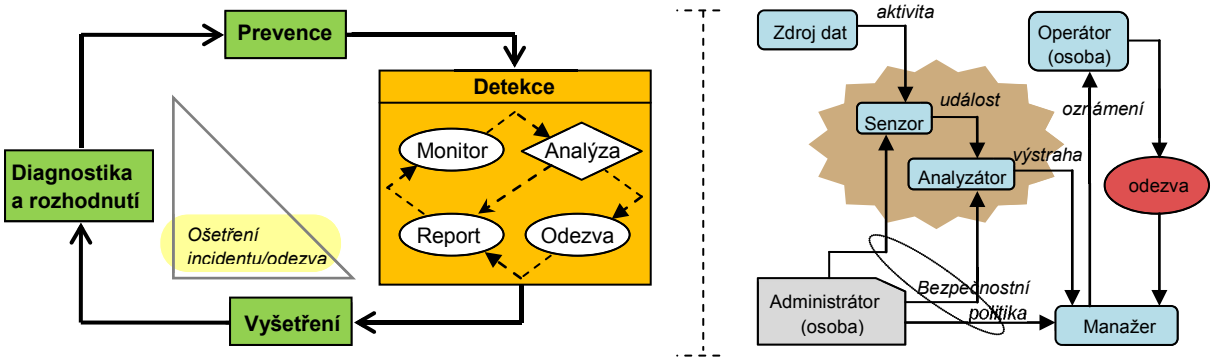
5.6.2 Systémy k detekci průniku („Intrusion Detection System“ – IDS)

Je to souhrn hardwarových a softwarových prvků k detekci neautorizovaného (tj. nepovoleného) přístupu do chráněného systému. Jejich cílem je zjistit průnik nebo pokus o průnik do informačního systému (nejčastěji se jedná o průnik do sítě nebo do konkrétního výpočetního systému). Bezpečnostní mechanismy těchto systémů mohou být preventivní nebo detekční.

IDS jsou součástí bezpečnostního systému, tj. komplexního bezpečnostního zajištění IS. Slouží k zajištění spolehlivosti (povoleny pouze autorizované přístupy k objektům), integrity, dostupnosti, autentičnosti a důvěrnosti. Mohou být provozovány samostatně, nebo jako elementy firewallů. Představují významný artikl na trhu informačních a komunikačních technologií (ICT) [38].

Struktura IDS:

- senzory: rozmístěny na vhodných místech chráněného systému a slouží k detekci událostí, které jsou z hlediska bezpečnosti důležité; generátory událostí (agenti, čidla)
- vyhodnocovací jednotka: zpracovává a vyhodnocuje informace od senzorů, archivuje je
- řídící konzola: slouží k řízení a konfiguraci celého systému, správní komponenta



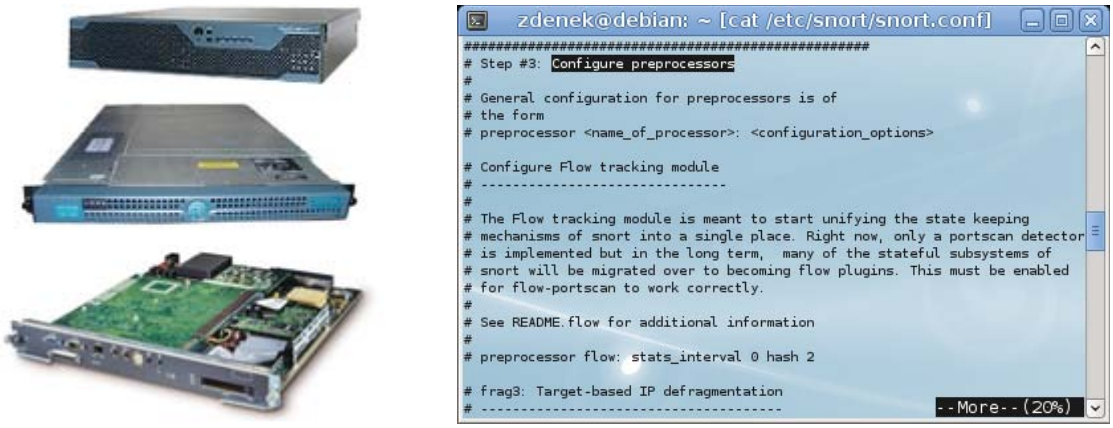
Obr. 48: Schéma obecné struktury detekce průniku, na IDS systém navazující modul protiopatření

Typy IDS:

- pasivní IDS: na útok reagují upozorněním operátora na řídící konzole a záznamem relevantních dat o útoku
- aktivní IDS: navíc provádějí protiopatření (např. IP adresa útočníka je okamžitě zanesena do tabulky firewallu => znemožnění útoku

Senzory IDS - možnosti řešení:

- softwarové: SW implementace, *výhody: cena (většinou zdarma), implementace *nevýhody: rychlost, značné nároky na výpočetní výkon (často vyžadován dedikovaný výpočetní systém)
- hardwarové: HW implementace, *výhody: výkonnost, rychlost *nevýhody: cena (příliš drahé)
- Senzory obsahují rozsáhlé seznamy detekčních příznaků – signatur. Pokud je v některém z paketů naleznou, informují o tom vyhodnocovací jednotku, která pak učiní patřičné kroky.



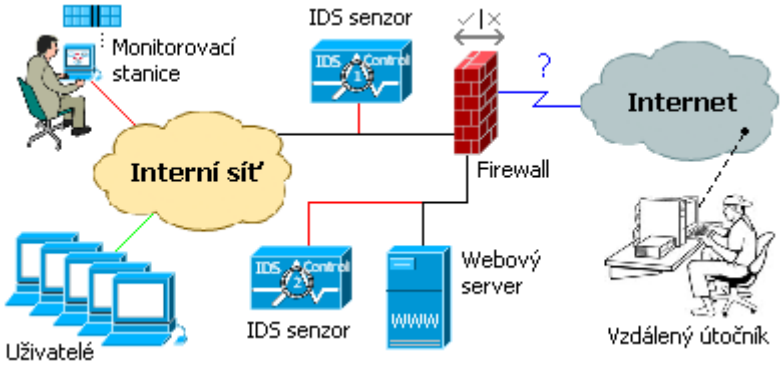
Obr. 49: Ukázka HW IDS produktů od firmy Cisco [25] a SW produkt Snort pod OS Linux (konfigurace)

Druhy senzorů IDS dle nasazení (oblast, ve které se provádějí sběr informací a analýza):

- HIDS (*Host IDS* - hostitelské prvky IDS): typicky vykonávají dohled nad přístupem k serverům. Kromě inspekce serverů se také může jednat o software na jednotlivých počítačích sítě, který sleduje podezřelou aktivitu daného počítače (např. zatížení procesoru, aktivitu pevného disku, manipulaci se soubory, opakované přihlašování uživatele, apod.)
- NIDS (*Network IDS* - síťové prvky IDS): monitorují a vyhodnocují provoz na počítačové síti. Specializovaná zařízení (podobná HW firewallům; někdy kombinace FW+IDS → interoperabilita) umísťovaná na hranicích sítě a v demilitarizovaných zónách. Sledují komunikaci v daném segmentu sítě s cílem detekovat v přenášených datech příznaky útoku.

IDS systémy jsou často implementovány s dalšími bezpečnostními nástroji:

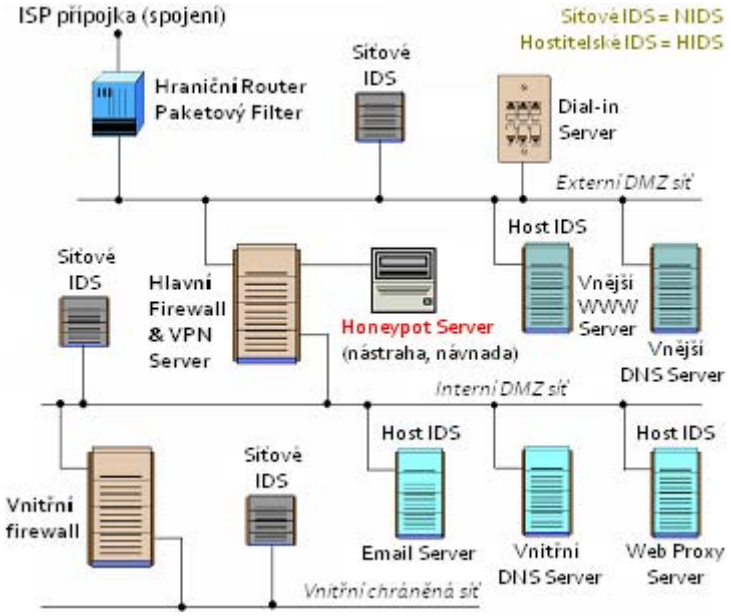
- systémy pro analýzu (hodnocení) zranitelnosti *(vulnerability assessment systems)* - ověřují odolnost na známé útoky, generují „snímky“ bezpečnostních stavů systému v určitém čase pro možnost zjištění problémů v důsledku lidských chyb nebo pro potřeby auditu
- systémy pro kontrolu integrity souborů *(integrity checkers systems)* - používají kryptografické kontrolní sumy pro kontrolu kritických datových objektů. *Integrity checkers* mohou preventivně ochránit systém před útokem (příprava útočníka často spočívá v předběžné modifikaci určitých systémových souborů).
- "Hrnce s medem" *(Honey Pot)* a "Vycpané cely" *(Padded Cell)*.



Obr. 50: Nasazení systému IDS v lokální síti

5.6.3 Klamné segmenty sítě (návnada, "*Honeypot*"); představení programu Snort

Návnada je segment sítě, který daná organizace nepoužívá, a který obsahuje mylné informace. Vnější firewall při detekci útoku umožní útočníkovi přístup do tohoto segmentu sítě. Ten zde pokračuje v útoku na jednotlivé prvky. Ztrácí tak čas a energii v útocích na bezcenné cíle a IDS systém současně získává cenné informace o útočníkovi a případně i o nových typech útoků. V operačním systému Linux lze pomocí daemona *honeyd* nalákat potenciální útočníky na virtuální zranitelný systém. Jeho prostřednictvím je možné současně simulovat různorodé systémy, za něž se bude vydávat [38].



Obr. 51: Prvky IDS rozmístěné v síťové infrastruktuře, Honeypot server, Firewally

V současné době existují dva typy NIDS. První typ detekuje útoky tím způsobem, že ve sledovaném provozu vyhledává známé vzorce, které odpovídají známým útokům. NIDS pracující tímto způsobem se označují jako **systémy založené na signaturách**. Druhým typem NIDS jsou statistické monitory. Ty rovněž sledují provoz v síti, nevyhledávají ale konkrétní obrazce nebo signatury. Místo toho si udržují statistický přehled o paketech, které sítí procházejí, a upozorní nás v okamžiku, kdy se provoz v síti vymkne běžnému obrazci. NIDS tohoto typu se označují jako **systémy založené na anomáliích** [30].

V následujících dílčích kapitolách popíši, jak nastavit systém Snort, což je IDS založený na signaturách a zároveň jeden z nejlepších volně šiřitelných detekčních programů (vyvíjen jako Open Source). Jedna z věcí, díky nimž je Snort tak úspěšný, je jeho architektura, tvořená jádrem detekujícím známé signatury, snadno rozšiřitelná prostřednictvím doplňujících modulů a preprocesorů. Díky tomu lze Snort rozšiřovat prakticky libovolným způsobem. Vyplívá z toho rovněž skutečnost, že pokud se objeví nějaký nový typ síťového útoku, nemusíme čekat, až někdo napíše pravidla pro jeho detekci – se základními znalostmi protokolu TCP/IP si taková pravidla můžeme snadno a rychle napsat sami. To je pravděpodobně nejužitečnější vlastnost Snortu, protože nové útoky se objevují neustále. Navíc má Snort velmi pružný mechanismus hlášení incidentů, které může zaznamenávat do *syslogu*, vlastních souborů nebo přímo do databáze [40].

Komponenty Snortu

Snort je logicky rozdělen do několika komponent. Tyto komponenty pracují společně, detekují tak jednotlivé útoky a generují výstup v požadovaném formátu. IDS Snort sestává z hlavních komponent:

- ✓ Jednotka paketového záchytu
- ✓ Zásuvné moduly preprocesoru
- ✓ Detekční jednotka
- ✓ Systém logování a výstrah
- ✓ Výstupní zásuvné moduly

Pravidla Snortu

Každé pravidlo se skládá ze dvou části - hlavičky a volby (options). Hlavička pravidla obsahuje akci pro vykonání, dále protokol, ke kterému se pravidlo vztahuje, zdrojové a cílové adresy včetně portů. Volby pravidla nám dovolují vytvořit popisnou zprávu spojenou s pravidlem, a také kontrolovat různé druhy dalších atributů paketu, které Snort používá v rozsáhlé knihovně zásuvných modulů [40].

Takto vypadá obecný tvar pravidla Snortu:

```
action protokol zdroj_ip zdroj_port směr cíl_ip cíl_port (options)
```

Když přijde paket, porovná se jeho zdrojová i cílová IP adresa a porty s pravidly v množině pravidel. Jestliže některé pravidlo odpovídá paketu, pak se vyhodnotí options. Jestliže souhlasí všechna porovnání, vykoná se příslušná akce.

Snort poskytuje několik vestavěných akcí, které můžeme použít, když pravidla vytváříme:

1) **pass** (předání) - ignoruje paket,
2) **log** (zaznamenání) - zaznamená paket,
3) **alert** (výstraha) - vygeneruje výstrahu a paket zaznamená,
4) **activate** (aktivace) - vygeneruje výstrahu a vyvolá k otestování další pravidlo,
5) **dynamic** (dynamika) - akce „dynamic“ jsou vyvolávány pouze dalšími pravidly užitím akce „activate“,
6) **drop** (zahození) - přidá do Iptables pravidlo pro zahození paketu a paket zaznamená,
7) **reject** (odmítnutí) - přidá do Iptables pravidlo pro zahození paketu, paket zaznamená a pošle TCP reset, jestliže je protokolem TCP nebo ICMP zprávu o nedostupnosti portu, jestliže je protokolem UDP,
8) **sdrop** - přidá do Iptables pravidlo pro zahození paketu, ale paket nezaznamená.

Implicitně se jako první testují pravidla Aktivace, poté pravidla Dynamiky, následují pravidla Výstrahy, poté přijdou pravidla Předání a končí se pravidly Zaznamenání. Nicméně pořadí není pevně určené.

5.6.4 Snort, Barnyard, Oinkmaster, Sguil, SnortCenter, ACID (BASE), Swatch a další

Snort – postup instalace, konfigurace, spuštění a další

Snort je program sloužící k detekci útoků (neautorizovaných přístupů) a odposlechu v síti, kde hledá vzorky známých útoků a v případě nalezení je schopen provádět různé akce. Má obrovskou databázi stop několika tisíců různých útoků, která se neustále aktualizuje a pravidelně udržuje. Není to firewall, což znamená, že detekuje již vzniklé problémy, ale nijak jim nezabraňuje (alespoň v defaultním nastavení, více viz Snort_inline nebo SnortSam) [39]. V dalším uvedu několika detailními kroky postup konfigurace a zprovoznění Snortu včetně dalšího souvisejícího event. podpůrného SW.

5.6.4.1 Instalace příbuzného software

Nejprve nainstalujeme (a spustíme) SSH Server:

```
# apt-get update
# apt-get install ssh
```

Poté nainstalujeme všechny potřebné softwarové balíčky tak, aby byly splněny závislosti:

```
# apt-get install apache-ssl apache-common libapache-mod-php4 mysql-server mysql-common \
mysql-client php4-mysql libnet1 libnet1-dev libpcre3 libpcre3-dev autoconf automake1.9 \
libpcap0.8 libpcap0.8-dev libmysqlclient15-dev php4-gd php4-pear libphp-adodb vim gcc \
make php4-cli libtool libssl-dev gcc-4.1 g++
```

5.6.4.2 Úprava firewallu

V tomto kroku je třeba nastavit povolení firewallu (obecně založeného na Iptables) pro příchozí spojení na portech 22 (*ssh*) a 443 (*apache-ssl*). Místní síť LAN je považována za důvěryhodnou.

5.6.4.3 Instalace systému Snort, pravidel VRT a přidání uživatele i skupiny snort

Další krok zahrnuje samotnou instalaci Snortu, vytvoření uživatele a skupiny snort a také instalaci detekčních pravidel narušení VRT.

```
************************************ Snort 2.X ************************************
# cd /usr/local/src                                          //změna adresáře           #
# wget http://snort.org/dl/current/snort-2.X.X.tar.gz        //stáhnutí Snortu          #
# tar xvzf snort-2.X.X.tar.gz                                //rozbalení z archivu      #
# cd snort-2.X.X                                             //přepnutí do adresáře     #
# ./configure --with-mysql --enable-dynamicplugin            //kompilace a instalace    #
# make & make install                                                                    #
***************************************************************************************
```

Snort instalace – okno zobrazené při nastavení sledovaného rozhraní (např. eth0, eth1, apod.)

Tato hodnota obyčejně bývá „eth0“, ale to nemusí být v některých síťových prostředích vhodné. Pro vytáčené připojení může být vhodnější „ppp0“ (viz výstup příkazu '/sbin/ifconfig'). Typicky je to stejné rozhraní jako je výchozí směrování („default route“). Které rozhraní se pro toto používá, můžeme zjistit spuštěním příkazu '/sbin/route -n'.

Také není neobvyklé použít rozhraní bez IP adresy nastavené do promiskuitního módu. V tomto případě vybereme rozhraní systému fyzicky připojené k síti, a které chceme sledovat. Následně povolíme promiskuitní mód a ujistíme se, že je síťový provoz posílán na toto rozhraní (buď připojené k port mirroring/spanning portu přepínače, k hubu nebo k tap rozhraní). Můžeme také zadat více rozhraní, jednoduše zapsáním více než jednoho jména rozhraní, oddělených mezerami. Každé rozhraní může mít své specifické nastavení [39].

Poznámka: současná stabilní verze Snortu má označení 2.8.3.2 -> stažení náležejícího komprimovaného balíčku *snort-2.8.3.2.tar.gz*.

Dále provedeme následující příkazy:

```
# mkdir /etc/snort      //vytvoření adresáře programu /etc/snort
# mkdir /var/log/snort      //vytvoření adresáře pro logování /var/log/snort
# groupadd snort      //vytvoří skupinu
# useradd -g snort snort      //vytvoří uživatele
# chown snort:snort /var/log/snort      //změna vlastníka
```

```
zdenek@debian:~$ cat /etc/passwd        // Výpis obsahu textového souboru „/etc/passwd“ obsahující seznam uživatelů a dalších parametrů. Důležité a související účty jsou vyznačeny.
root:x:0:0:root:/root:/bin/bash        // Superuživatel (root)
zdenek:x:1000:1000:Zdenek Kugler,,,:/home/zdenek:/bin/bash        // Uživatel Zdenek
clamav:x:109:116::/var/lib/clamav:/bin/false        // ClamAV antivirus
havp:x:110:117::/var/run/havp:/bin/false        // HAVP antivirová proxy
squid:*:1555:1555:squid user:/usr/local/squid:        // Squid cache proxy
snort:x:111:118:Snort IDS:/var/log/snort:/bin/false        // Systém detekce průniků IDS Snort
testname:x:1001:1001:Virtuální uživatel,,,:/home/testname:/bin/bash        // Vytvořený uživatel, sloužící k testovacím účelům na stroji s Debianem
```

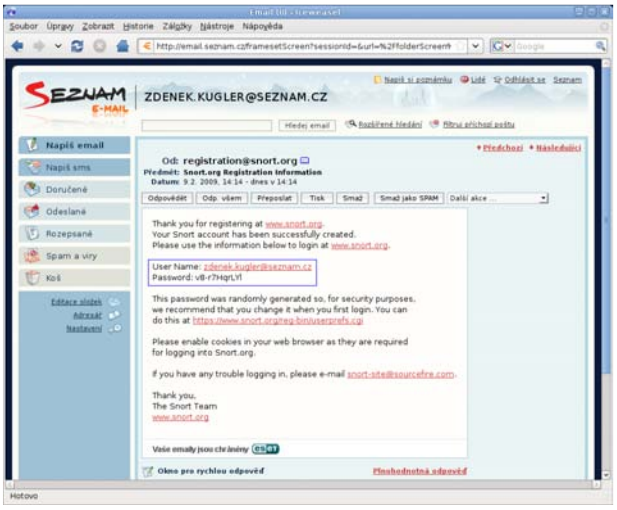
Stručný souhrn pravidel VRT z adresy http://www.snort.org/vrt/:

Zdrojové certifikované VRT pravidla jsou oficiálními pravidly Snortu. Každé pravidlo bylo nejprve pečlivě (opakovaně) testováno a teprve poté nabídnuto cílovým zákazníkům.
Existují tři způsoby k získání těchto pravidel:

- Předplatitelé přijímají pravidla v reálném čase, jakmile je aktualizace k dispozici
- Registrovaní uživatelé mohou přistupovat k pravidlům po 5 dnech od vydání
- Neregistrovaní uživatelé obdrží statickou aktualizaci pravidel s každým významnějším vydáním Snortu (nové verze Snortu)

Samozřejmou funkcí je vytváření a přidávání uživatelských (doplňujících či nadrámcových) pravidel. Pravidla mohou být distribuována – spadají pod licenci GPL a volně poskytována ostatním uživatelům Snortu (open source) [39].

Vytvořením uživatelského účtu na webové adrese http://www.snort.org se lze stát registrovanými uživateli a využívat jejich výhod. Získaný (stažený) soubor se jmenuje `snortrules-snapshot-CURRENT.tar.gz` (současný, aktuální) nebo `snortrules-snapshot-2.X.tar.gz` (pro konkrétní verzi).



Thank you for registering at www.snort.org.
Your Snort account has been successfully created.
Please use the information below to login at www.snort.org.

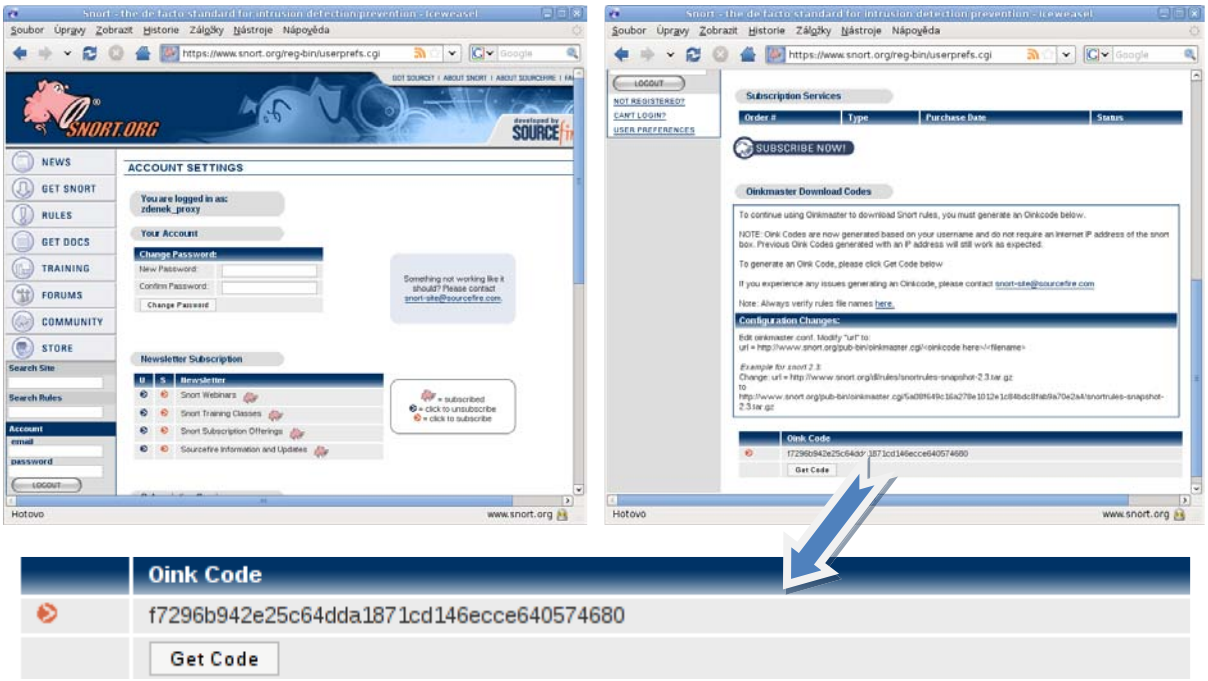
User Name: zdenek.kugler@seznam.cz
Password: v8-r7HqrLYl

This password was randomly generated so, for security purposes, we recommend that you change it when you first login. You can do this at https://www.snort.org/reg-bin/userprefs.cgi

Please enable cookies in your web browser as they are required for logging into Snort.org.

Obr. 52: Snort registrace – obsah e-mailu

Jakožto registrovaný uživatel mám nárok na aktuální VRT certifikovaná pravidla pro Snort. Stažený komprimovaný soubor s pravidly („SnortRules file“) se jmenuje: *snortrules-snapshot-CURRENT.tar.gz.*



Následně provedeme příkazy:

```
# cd /etc/snort
# mv /root/snortrules-snapshot-2.X.tar.gz .
# tar xvzf snortrules-snapshot-2.X.tar.gz
# cp /usr/local/src/snort-2.X.X/etc/*.conf* .
# cp /usr/local/src/snort-2.X.X/etc/*.map .
```

5.6.4.4 Konfigurace a spuštění programu Snort

Nyní je čas nakonfigurovat a poté spustit Snort. Jako první potřebujeme editovat hlavní konfigurační soubor [39]:

```
# vim /etc/snort/snort.conf
```

Pro začátek upravíme pouze tyto řádky:

```
var HOME_NET [192.168.1.0/24,192.168.2.0/24]   //192.168.1.100, 192.168.2.100
var EXTERNAL_NET !$HOME_NET     //any
var RULE_PATH /etc/snort/rules   //cesta k pravidlům Snortu
(var PREPROC_RULE_PATH /etc/snort/preproc_rules)
```

Tyto předdefinované proměnné jsou důležité zejména kvůli „ladění“ senzorů a musíme je uzpůsobit naší síti. Nyní si můžeme vytvořit jednoduché místní pravidlo za účelem okamžitého získání varovných hlášení Snortu při jeho prvním spuštění:

```
# vim /etc/snort/rules/local.rules   //lokální definovaná pravidla
```

Vytvoříme jednoduché pravidlo pro testování:

```
alert icmp any any -> $HOME_NET any (msg:"ICMP test"; dsize:8; itype:8; sid:10000001;)
```

nebo skutečně vygenerujeme několik výstrah tímto místním pravidlem:

```
alert tcp any any -> any any (msg:"test"; sid:1000002;)
```

Jako další otestujeme, zda je námi navržená konfigurace v pořádku. Spustíme následující příkaz:

```
# snort -T -c /etc/snort/snort.conf   //spustí Snort v testovacím módu
```

Snort vypíše případné nalezené chyby a skončí. Výsledkem by měl být přibližně výstup:

--== Initialization Complete ==-- ***Pozn. Ve výpisu zachycena jen konečná část (kvůli délce).***

```
 ,,_     -*> Snort! <*-
o"  )~   Version 2.8.3.2 (Build 22)
 ''''    By Martin Roesch & The Snort Team: http://www.snort.org/team.html
         (C) Copyright 1998-2008 Sourcefire Inc., et al.
         Using PCRE version: 7.8 2008-09-05

         Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 1.9  <Build 15>
         Preprocessor Object: SF_SSH  Version 1.1  <Build 1>
         Preprocessor Object: SF_SMTP  Version 1.1  <Build 7>
         Preprocessor Object: SF_FTPTELNET  Version 1.1  <Build 10>
         Preprocessor Object: SF_DNS  Version 1.1  <Build 2>
         Preprocessor Object: SF_DCERPC  Version 1.1  <Build 4>

Snort sucessfully loaded all rules and checked all rule chains!
Snort exiting
```

Pokud se žádné chyby neobjeví, můžeme Snort definitivně spustit příkazem:

```
# /usr/local/bin/snort -Dq -u snort -g snort -c /etc/snort/snort.conf
```

Snort by se měl úspěšně inicializovat. Více zjistíme ze souboru /var/log/syslog, kde najdeme řádek zhruba podobný tomuto:

```
Feb 23 19:05:43 debian snort[19165]: Snort initialization completed successfully (pid=19165)
```

V souboru /var/log/messages lze navíc dohledat přepnutí síť. rozhraní do tzv. promiskuitního módu.

```
Feb 23 19:03:37 debian kernel: [ 8760.197776] device eth0 entered promiscuous mode
```

K ověření, zda Snort běží na pozadí jako daemon či nikoliv, poslouží příkaz ps aux. Dojde k výpisu všech procesů, které má spuštěny uživatel snort. Využito je přesměrování výstupu rourou (pipe).

```
# debian:/home/zdenek# ps aux | grep snort
root   5062  6.8  3.9  58904 35868 ?   Ss  20:09  0:00 snort -c /etc/snort/snort.conf -i eth0 -D
```

Nyní v systému povolíme/spustíme odposlouchávající „sniffing“ rozhraní (pokud je potřeba):

```
# ifconfig eth0 up  (ifconfig eth1 up)
```

Nezapomeňme ukončit současný proces Snortu v systému (*kill*) před spuštěním nového (např. s modifikovanou konfigurací a vstupními parametry).

Dobrým předpokladem k odposlouchávání dat na domácí síti („sniff live traffic“) může být starý 10Mbit/s 4portový rozbočovač, do kterého bude zapojen síťový kabel RJ45 (resp. 8P8C) od ISP poskytovatele připojení. Může se jednat např. o propůjčené zařízení v podobě ADSL modemu. Stejně tak bude k rozbočovači připojen průchozí směrovač/domácí brána, většinou již ve vlastnictví majitele lokální sítě. K třetímu portu rozbočovače bude připojeno snímající rozhraní – Snort. Celkově na rozbočovači tedy budou tři přípojky. Veškerý síťový provoz do (ze) sítě bude vysílán na zbylé porty hubu – provoz tak může být odposlechnut snímajícím rozhraním Snortu, pracujícím v promiskuitním režimu čili s vyřazeným hardwarovým filtrem (síťová karta předává nadřízené vrstvě nejen rámce s jí příslušející fyzickou MAC adresou, ale i všechny ostatní zachycené rámce určené pro zbylá zařízení).

Snort může běžet ve třech módech [39]:

- „Sniffer“ mód - chová se jako standardní sniffer, odchytává pakety ze sítě a tiskne je na obrazovku.
- „Packet logger“ mód - to samé, navíc ale loguje (zaznamenává) pakety do souboru na disku.
- „NIDS“ mód - nejvýznamnější mód programu, v tomto módu Snort odchytává síťová data a analyzuje je v kontextu s uživatelem definovanými pravidly a provádí různé akce podle nálezu.

Sniffer mód

Zobrazení IP a TCP/UDP/ICMP hlaviček: `snort -v`
Pro vyzkoušení tohoto módu stačí spustit program ping z jiného PC. Např.: `ping 192.168.1.100`

Odchycené pakety:

```
03/18-14:51:38.994878 192.168.2.1 -> 192.168.1.100
ICMP TTL:128 TOS:0x0 ID:351 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:13  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

03/18-14:51:38.994939 192.168.1.100 -> 192.168.2.1
ICMP TTL:64 TOS:0x0 ID:17249 IpLen:20 DgmLen:60
Type:0  Code:0  ID:1  Seq:13  ECHO REPLY
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Zobrazení hlaviček linkové (spojové) vrstvy: `snort -ve`

```
03/18-14:54:47.334210 0:1B:24:E:73:40 -> 4C:0:10:7B:6D:5E type:0x800 len:0x4A
192.168.2.1 -> 192.168.1.100 ICMP TTL:128 TOS:0x0 ID:382 IpLen:20 DgmLen:60
Type:8  Code:0  ID:1   Seq:17  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

03/18-14:54:47.334283 4C:0:10:7B:6D:5E -> 0:1B:24:E:73:40 type:0x800 len:0x4A
192.168.1.100 -> 192.168.2.1 ICMP TTL:64 TOS:0x0 ID:17253 IpLen:20 DgmLen:60
Type:0  Code:0  ID:1  Seq:17  ECHO REPLY
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Zobrazení včetně datové části paketu: `snort -ved`
Z „útočníkova počítače“ spustíme příkaz: `telnet 192.168.1.100`

Po navázání TCP spojení (první tři pakety –> třístavový handshake: SYN, SYN+ACK, ACK), dochází pak již jen k přenosu samotných dat (příznak PUSH) a potvrzování (příznak ACK). Ukončení relace (session) se provede čtyřstavovým handshakem oběma komunikujícími stranami (příznaky FIN, ACK; FIN, ACK) – ve výpise již nezachyceno. Z datové zátěže paketů lze vyčíst následující:

```
03/18-15:38:59.731798 0:1B:24:E:73:40 -> 4C:0:10:7B:6D:5E type:0x800 len:0x42
192.168.2.1:49639 -> 192.168.1.100:23 TCP TTL:128 TOS:0x0 ID:3981 IpLen:20 DgmLen:52 DF
******S* Seq: 0x677D2183  Ack: 0x0  Win: 0x2000  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP WS: 2 NOP NOP SackOK
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

03/18-15:38:59.731864 4C:0:10:7B:6D:5E -> 0:1B:24:E:73:40 type:0x800 len:0x42
192.168.1.100:23 -> 192.168.2.1:49639 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:52 DF
***A**S* Seq: 0x3EFE8C84  Ack: 0x677D2184  Win: 0x16D0  TcpLen: 32
TCP Options (6) => MSS: 1460 NOP NOP SackOK NOP WS: 6
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

03/18-15:38:59.732369 0:1B:24:E:73:40 -> 4C:0:10:7B:6D:5E type:0x800 len:0x3C
192.168.2.1:49639 -> 192.168.1.100:23 TCP TTL:128 TOS:0x0 ID:3982 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x677D2184  Ack: 0x3EFE8C85  Win: 0x4029  TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

03/18-15:41:17.412396 4C:0:10:7B:6D:5E -> 0:1B:24:E:73:40 type:0x800 len:0x47
192.168.1.100:23 -> 192.168.2.1:49640 TCP TTL:64 TOS:0x10 ID:2936 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0xAE54B154  Ack: 0xF281B471  Win: 0x5C  TcpLen: 20
7A 64 65 6E 65 6B 40 64 65 62 69 61 6E 3A 7E 24  zdenek@debian:~$
20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Packet logger mód

Při specifikování adresáře pro logování Snort automaticky přejde do módu logování:

```
snort -vde -l /var/log/snort
```

V sítích s velkým provozem je výhodnější logovat do binárního souboru:

```
snort -vde -l /var/log/snort -b
```

Formát binárního souboru je stejný jako u programu tcpdump. Proto může být přečten programy Tcpdump, Wireshark a také Snortem:

```
snort -vd -r packet.log
```

Význam použitých parametrů:

- „-r“: čte a zpracovává tcpdump soubor
- „-l“: loguje do (uvedeného) adresáře

NIDS mód

Pro zapnutí NIDS módu je třeba specifikovat konfigurační soubor s definovanými pravidly.

```
snort -c /etc/snort/snort.conf -l /var/log/snort
```

V dalším bude pojednáváno zejména o práci Snortu v NIDS. Výstup Snortu může vypadat např. takto:

[**] [119:13:1] (http_inspect) NON-RFC HTTP DELIMITER [**]
01/12-13:08:20.834286 192.168.1.100:1111 -> 82.114.56.132:80
TCP TTL:128 TOS:0x0 ID:4916 IpLen:20 DgmLen:134 DF
****AP*** Seq: 0x10A5E061 Ack: 0x74054544 Win: 0x40B0 TcpLen: 20*

První řádek je hlavička zprávy, která obsahuje některé zajímavé informace. První číslo v hranaté závorce (119) je generátor ID, to říká, která komponenta Snortu vygenerovala tuto zprávu. Hlavička obsahuje stručný popis varování. Pod hlavičkou jsou detailnější informace o paketu, který varování vyvolal. Od verze Snortu 1.5 a výše v něm byly definované tzv. preprocesory. Kód preprocesoru je spuštěn potom, co je zavolán detekční engine, ale předtím, než je paket dekódován. Díky této modulární struktuře umožňují uživatelům jednoduše rozšiřovat funkcionality Snortu. Zde je popis některých základních preprocesorů:

Frag3 – je modul zajišťující IP defragmentaci paketů a detekci útoků využívající IP defragmentaci. Je tzv. target-based. Různé operační systémy implementují svůj IP stack („zásobník“) různými způsoby, to znamená, že pro různé operační systémy může IP defragmentace probíhat odlišně, a útoky využívající IP fragmentaci se tedy budou lišit podle toho, na jaký operační systém jsou namířeny. Target based tedy značí, že místo toho, aby Snort detekovat útoky čistě na základě paketů, které přes něj prochází, bude modelovat chování cíle, pro které jsou pakety určeny a defragmentovat je podle stejných pravidel, jako to dělá cílový systém. To, jak se má modul chovat, se nastaví pomocí volby `policy <type>`, kde `type` může být `first`, `last`, `bsd`, `bsd-right`, `linux`.

Stream4 – tento modul umožňuje znovu sestavovat TCP stream („datový tok“) a dovoluje mu aplikovat stavovou analýzu, zda je vše v pořádku. Díky tomu může Snort detekovat různé bezestavové (stateless) útoky.

HTTP Inspect – je generický HTTP dekodér, tzn., že pracuje na aplikační úrovni síťového spojení. Data ukládá do bufferu (vyrovnávací paměti), který poté dekóduje, najde HTTP hlavičky a prověří celkovou strukturu HTTP, zda odpovídá definici RFC (standardu, specifikaci). HTTP Inspect funguje jak na klientské dotazy, tak i na serverové odpovědi.

Spousta útoků vykazuje ten rys, že v přenášených datech je nějaký statický řetězec. Například tyto pravidla zachycují vir *ILOVEYOU*, označovaný též jako *VBS/Loveletter*, což je počítačový červ vytvořený v jazyce VBScript, ohrožující pouze počítače s operačním systémem z rodiny Microsoft Windows. Rozsah a následky infekce tímto virem byly po celém světě nesmírně rozsáhlé.

```
alert tcp any 110 -> any any (msg:"Incoming Love Letter Worm";\
content :"rem barok -loveletter"; content:"@GRAMMERSoft Group";)
```

```
alert tcp any 143 -> any any (msg:"Incoming Love Letter Worm";\
content :"rem barok -loveletter"; content:"@GRAMMERSoft Group";)
```

```
alert tcp any any -> any 25 (msg:"Outgoing Love Letter Worm";\
content :"rem barok -loveletter"; content:"@GRAMMERSoft Group";)
```

Snort poskytuje velice široké možnosti při definování pravidel. Většina pravidel jsou jednořádkové řetězce a jsou rozděleny do dvou logických sekcí, hlavička pravidla a nastavení pravidla. Header (čili hlavička) obsahuje prováděnou akci, protokol, zdrojové a cílové IP adresy a síťové masky. Část s nastavením pak obsahuje definici varovných zpráv a informace o tom, na kterou část paketu má být pravidlo provedeno (obsah, hlavička apod.).

Vezměme si pro ilustraci jedno pravidlo:

```
alert tcp $EXTERNAL_NET any ->
$HOME_NET 21 (msg:"FTP EXPLOIT
wu-ftpd 2.6.0 bsd"; content:
"|31c0 50 50 50 b07e cd80 31db
31c0|"; flags: A+; depth: 32;
reference:arachnids,228; classtype:
attempted-admin; sid:343; rev:1;)
```

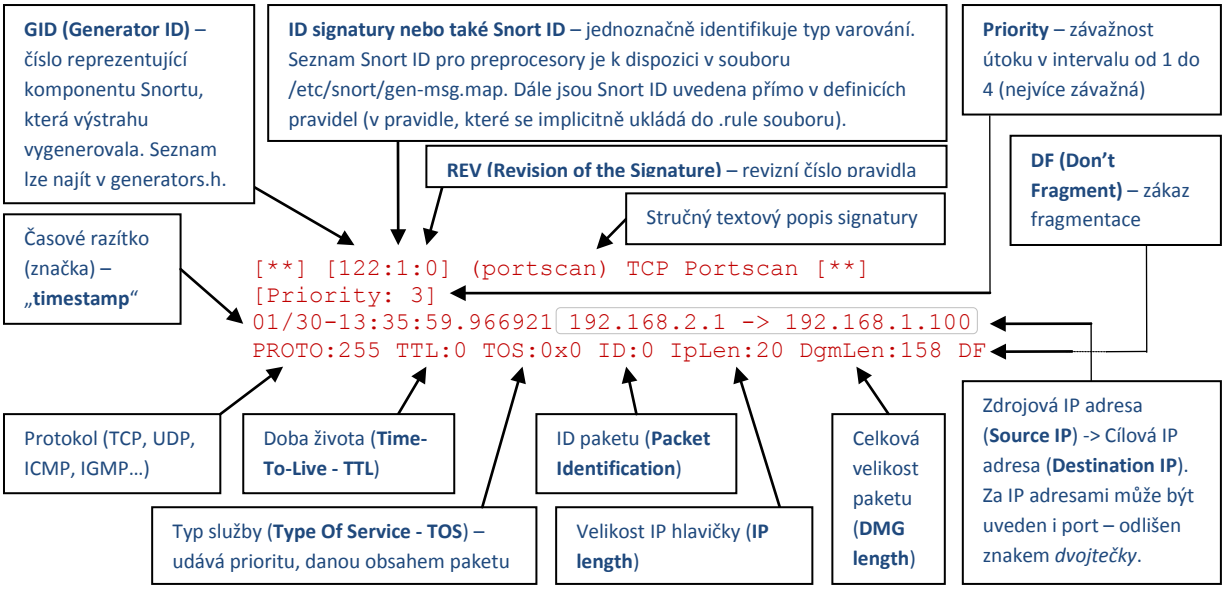
V případě takového útoku pak můžeme v logu nalézt informaci:

```
[**] [1:343:1] FTP EXPLOIT wu-ftpd 2.6.0 bsd [**]
[Classification: Attempted Administrator
Privilege Gain]
[Priority: 10] 01/16-10:41:18.031375 {PROTO006}
195.168.1.141:4870 -> 129.10.11.2:21
```

Tímto pravidlem definujeme, že pokud se objeví nějaký TCP paket z vnější sítě, který bude mířit na port 21 v našem lokálním systému a bude obsahovat sekvenci 31c0 50 50 50 b07e cd80 31db 31c0 s příznakem ACK, bude se jednat o pokus získání oprávnění administrátora a Snort vyvolá poplach.

Formát výstrahy z logovacího souboru

Pro popis formátu zaznamenané výstrahy byla náhodně vybrána jedna výstraha ze souboru „/var/log/snort/alert“, kterou dále celou vysvětlím.



Příklady spuštění Snortu s různými parametry

snort -dv -i eth0 -A cmg -k none
snort -evi eth0 // poběží v „packet dump“ módu
snort -A fast -i eth0 -b -k none
snort -Dd -z est -c /etc/snort/snort.conf

Základní parametry Snortu	
parametr	*význam*
-l	Do logu bude přidáno i označení síťového rozhraní, pro nějž byl útok rozpoznán.
-i	Definuje rozhraní, kde budeme útoky sledovat.
-d	Při dumpování (výpisu) podezřelého paketu budeme dumpovat také aplikační data.
-e	Tato data rozšíříme o hlavičku příslušné vrstvy.
-A	Specifikuje formát zápisu do logu (v našem případě fast = čas, zpráva, IP adresa a port).
-c	Specifikuje umístění konfiguračního souboru.
-l	Udává adresář, do něhož budeme logovat (pro každé rozhraní je vhodné nastavit vlastní adresář).
-b	Pakety budeme dumpovat ve tvaru tcpdump (bezproblémová funkčnost i na 100Mbit/s síti).
-p	V průběhu činnosti Snortu nebudeme přepínat síťovou kartu do promiskuitního módu.
-S	Zde uvedeme hodnoty všech proměnných z konfiguračního souboru. Používáme-li stejný konfigurační soubor, musíme specifikovat pouze hodnotu HOME_NET.

Vyzkoušejme si konfiguraci na praktickém příkladu. Budeme předpokládat, že je naše vnitřní síť chráněna linuxovým firewallem (např. Iptables), a chceme zjistit, zda někdo na náš server (např. lokální Proxy server) náhodou nezkouší něco, co by zkoušet neměl – ať už úspěšně, nebo neúspěšně. Předpokládejme také, že náš server bude mít dvě síťová ethernetová rozhraní (ethX):

eth0 (vnější rozhraní): IP adresa: 192.168.1.100, maska sítě: 255.255.255.0 (/24), výchozí brána: 192.168.1.1
eth1 (vnitřní rozhraní): IP adresa: 192.168.2.100, maska sítě: 255.255.255.0 (/24)

U obou síťových rozhraní jde o masku třídy C, logickým vynásobením masky a IP adresy stanice získáme adresy sítí.

Snort spustíme startovacím skriptem s následujícími parametry:

```
#/etc/snort/src/snort -I -i eth0 -de -A fast –c /etc/snort/snort.conf -l /var/log/snort/eth0/ -b -p -S
HOME_NET=192.168.1.100/32   //adresa této sítě je 192.168.1.0
```

Pro druhé rozhraní pak spustíme další proces s přizpůsobenými parametry:

```
#/etc/snort/src/snort -I -i eth1 -de -A fast –c /etc/snort/snort.conf -l /var/log/snort/eth1/ -b -p -S
HOME_NET=192.168.2.100/32   //adresa této sítě je 192.168.2.0
```

Jednotlivé parametry a jejich argumenty jsou uvedeny v tabulce. Máme tedy dvě síťová rozhraní. Snort na vnitřním rozhraní kontroluje útoky z vnitřní sítě (dle statistik CERT je jich až 60 %) a proces na vnějším rozhraní kontroluje útoky prováděné odněkud z prostředí Internetu. Teď už jen zbývá pozorně sledovat logy. To lze provádět buď manuálním procházením, anebo je možné nasadit nějaké sledování pomocí cronu apod. Snort však disponuje také řadou doplňků a rozšíření, jež umožňují mimo jiné velmi efektivně sledovat právě tento alert.log. Jedním z takových nástrojů je SnortSnarf. Program SnortSnarf slouží ke zpracování údajů z logovacích souborů Snortu do podoby přehledných a velmi dobře srozumitelných HTML stránek, ze kterých je ihned zřejmá aktuální situace a provoz v síti.

Výsledné shrnutí při ukončení činnosti Snortu

```
===============================================================================
Packet Wire Totals:
Snort Received       626294 packets
      Analyzed:      440759(70.376%)
      Dropped:       185441(29.609%)
      Outstanding:   94(0.015%)
===============================================================================
Breakdown by protocol:
      TCP: 436080     (98.938%)
      UDP: 2655       (0.602%)
     ICMP: 877        (0.199%)
      ARP: 230        (0.052%)
    EAPOL: 0          (0.000%)
     IPv6: 71         (0.016%)
  ETHLOOP: 0          (0.000%)
      IPX: 0          (0.000%)
     FRAG: 0          (0.000%)
    OTHER: 824        (0.187%)
  DISCARD: 22         (0.005%)
InvChkSum: 0          (0.000%)
===============================================================================
Action Stats:
ALERTS: 4258
LOGGED: 4258
PASSED: 0
===============================================================================
```

5.6.4.5 Nastavení MySQL serveru

Dalším krokem je nastavení MySQL serveru (resp. databáze). Pomocí „apt-get install“ stáhneme a nainstalujeme balíčky programů Apache a MySQL na systém Debianu. Programy se dále spouští inicializačními skripty při startu – „bootování“ OS. Nejdříve nastavíme heslo roota (superuživatele) MySQL databáze příkazem [39]:

```
# mysqladmin -u root password "heslo_roota"
```

Vstup do příkazového řádku systému MySQL:

```
# mysql -u root –p (zde vložíme heslo "heslo_roota" abychom se dostali dále)
```

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 59
Server version: 5.0.77-1 (Debian)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

Vytvoření databáze Snortu:

```
mysql> create database snort;
```

Query OK, 1 row affected (0.02 sec)

Vytvoření uživatele snort a přidělení oprávnění (potřebných práv):

```
mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort@localhost;
```

Query OK, 0 rows affected (0.05 sec)

Nastavení hesla uživatele snort pro databázi:

```
mysql> SET PASSWORD FOR snort@localhost=PASSWORD('moje_heslo');    //moje_heslo=841024
```

Query OK, 0 rows affected (0.03 sec)

Ukončení relace:

```
mysql> exit
```

Bye

Nyní importujeme schéma programu Snort:

```
# cd /usr/local/src/snort-2.X.X/schemas/
# mysql -u root -p < create_mysql snort
```

Přihlášení se k MySQL serveru a zhlédnutí vytvořených tabulek:

```
# mysql -u root -p (znovu vložíme heslo)
mysql> use snort;
```

Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

```
mysql> show tables;
```

```
+------------------+
| Tables_in_snort  |
+------------------+
| data             |
| detail           |
| encoding         |
| event            |
| icmphdr          |
```

```
| iphdr             |
| opt               |
| reference         |
| reference_system  |
| schema            |
| sensor            |
| sig_class         |
| sig_reference     |
| signature         |
| tcphdr            |
| udphdr            |
+-------------------+
16 rows in set (0.01 sec)
```

V této chvíli bychom měli vidět seznam nových importovaných tabulek.

5.6.4.6 Konfigurace Snortu k logování do MySQL a test

V další části nakonfiguruji Snort k logování do MySQL databázového serveru a celý systém otestuji. Výstražné alerty Snortu tak nejenže zaznamenáme do logovacího souboru, ale i do připravené databáze, která poskytuje větší přehlednost, lepší orientaci mezi jednotlivými útoky a protokoly, různé třídění do kategorií, statistiky, grafy, vyhodnocení, apod. Výše zmíněné provedeme nastavením výstupního pluginu (zásuvného modulu) pro logování do databáze [39]:

```
# vim /etc/snort/snort.conf
```

V textu nalezneme řádek související s výstupem do databáze, odkomentujeme, a přidáme (resp. upravíme) příslušné hodnoty tak, aby se shodovali s námi vytvořenou databází (MySQL):

```
output database: alert, mysql, user=snort password=841024 dbname=snort host=localhost
```

Restartujeme Snort a ověříme jeho zápis do MySQL databáze. Nejjednodušším způsobem potvrzení je vstup do mysql a zápis "select * from event", s jehož pomocí jsme schopni zjistit, ale hlavně vidět, vzniklé události v případě, že je proces (instance) Snortu v systému nadále aktivní, a je aplikováno nejméně jedno z jednoduchých lokálně definovaných (testovacích) pravidel Snortu (viz local.rules). V takovém případě budou zaznamenány minimálně všechny ICMP nebo TCP pakety – lze poznat také podle náležející zprávy „ICMP test“ popř. jen „test“. Další možností je spuštění následujícího příkazu:

```
# mysql –u root –p heslo_roota -D snort -e "select count(*) from event"
```

```
+----------+
| count(*) |
+----------+
|     5    |
+----------+
```

5.6.4.7 Instalace webového serveru Apache 2 (Apache-SSL)

Konfigurace webového serveru Apache 2 (nebo Apache s podporou SSL) je na stroji Debianu triviální. Po provedení instalační procedury (`apt-get install nazev_balicku`) uvedené v prvním bodě je server Apache nainstalován a běží (na pozadí, jako daemon). Eventuální soubor s certifikátem (*SSLCertificateFile*) je vytvořen při instalaci. Kořenový adresář (*DocumentRoot*) je /var/www.

Editace konfiguračního souboru apache2 (apache-ssl):

```
# vim /etc/apache2/apache2.conf     // /etc/apache-ssl/httpd.conf
```

Řádky, které odkomentujeme kvůli podpoře parsování obecně php souborů:

```
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```

Taktéž musíme povolit rozšíření extension=mysql.so v souboru /etc/php5/apache2/php.ini:

```
# vim /etc/php5/apache2/php.ini    // /etc/php4/apache/php.ini
```

Zrušení komentáře u řádku:

```
extension=mysql.so
```

Nakonec restart Apache 2 (Apache-SSL) serveru pro uplatnění provedených změn:

```
# /etc/init.d/apache2 restart    // /etc/init.d/apache-ssl restart
```

5.6.4.8 Instalace a konfigurace BASE (Basic Analysis and Security Engine)

BASE je aplikace poskytující webovou nástavbu („web front-end“) programu Snort. Podrobný návod ke zprovoznění a nastavení BASE v1.4.1 (lara) je uveden zde [39]:

```
# cd /var/www    //přesun do www adresáře (adresář /var = proměnná data systému)
# rm index.html  //smažeme výchozí HTML stránku Apache [http://IP_stroje(:port)/index.html]
# wget http://internap.dl.sourceforge.net/sourceforge/secureideas/base-1.4.1.tar.gz
# tar xvzf base-1.4.1.tar.gz    //rozbalení z archivu tar.gz
# mv base-1.4.1 base    //přejmenování adresáře
# chmod 777 base    //udělená práva pouze pro tuto chvíli
```

Otevřeme webový prohlížeč a do adresního řádku napíšeme příslušnou IP stroje, na kterém Apache běží a vyžádáme adresář „/base“. Příklad: *http(s)://192.168.1.100/base*, *http://127.0.0.1/base*.

V dalším zobrazeném okně zvolíme patřičné jazykové možnosti (English,…) a vložíme cestu k abstraktní knihovně pro PHP (a Python) ADOdb: /usr/share/php/adodb.

```
Database Name: snort
Database Host: localhost
Database Port: blank (default)
Database User Name: snort
Database Password: 841024
```

Vložením správných hodnot pro vstupní autentizaci pokračujeme kliknutím na pole „create baseag“, čímž přidáme tabulky pro rozšíření databáze Snortu k podpoře funkcionality BASE.

Nyní by mělo být vše potřebné nastaveno. Přihlášením se podobně jako v 5.6.1.5 jsme schopni sledovat nově vznikající alarmující události a potvrdit tak funkčnost zachytávání senzorů Snortu. Restartem Snortu, okomentováním vlastních pravidel z local.rules a smazáním záznamů v databázi je systém připraven k odladění v síti.

Poté nastavíme s pomocí příkazu chmod 755 přístupová práva adresáři /base, který se nachází ve /var/www. Další věcí, kterou je nutné k sestavování grafů provést, je:

```
# rm /etc/alternatives/php
# ln -s /usr/bin/php5 /etc/alternatives/php   //symbol. přilinkování php na Debianu k php5
```

Následně spustíme příkaz:

```
# pear config-set preferred_state alpha   //PEAR je framework a distr. systém pro PHP
komponenty; umožňuje stahovat, instalovat, updgradovat a odinstalovávat PHP skripty
```

Zbývá zrušit komentáře u rozšíření extension=gd.so v souboru /etc/php5/cli/php.ini a ověřit v příkazové řádce pearu pomocí „php-cli“ skriptu nezbytné závislosti.

```
# vim /etc/php5/cli/php.ini
```

Odkomentování řádku pro potřebné rozšíření:

```
extension=gd.so
```

Spuštění příkazů pro instalaci *Image_Color*, *Image_Canvas* a *Image_Graph*:

```
# pear install Image_Color
# pear install Image_Canvas
# pear install Image_Graph
```

Předtím, než bude odkaz (link) na sestavení grafů zcela funkční, je třeba vykonat poslední část, a tou je restart Apache.

```
# /etc/init.d/apache2 restart    // /etc/init.d/apache-ssl restart
```

5.6.4.9 Instalace Barnyard a konfigurace Snortu pro rychlé unifikované logování

Samotný Snort je vhodný pro sledování malých sítí nebo sítí s malými objemy provozu, v náročnějším nasazení však nevyhovuje. Problém nespočívá ani tak v detekčních mechanismech Snortu, ale ve skutečnosti, že jde o jednovláknovou aplikaci. Kdykoliv je tedy detekován provoz, který si vyžaduje generování poplachu nebo záznam do logu, musí Snort nejprve zajistit doručení poplachu nebo zápis do logu, a teprve potom může pokračovat v analýze paketů. Pokud je Snort nastaven tak, aby zapisoval jen do souboru, nepředstavuje to zásadní problém. Potíže mohou vznikat v případech, kdy Snort zapisuje do databáze, protože vložení záznamu do tabulky může trvat relativně dlouho. Nemluvě o vzdáleném databázovém serveru [39].

K řešení těchto potíží byla napsána aplikace Barnyard. Funkčně je Barnyard ekvivalentní všem výstupním doplňkům Snortu koncentrovaným do jedné aplikace. Barnyard čte vstup z jediného souboru vytvářeného Snortem, a následně jej rozesílá do databází a na jiná cílová místa tak, jak by to normálně dělal přímo Snort. Jedinou nevýhodou Barnyardu je omezená databázová podpora.

Nejprve vykonáme následující příkazy:

```
# cd /usr/local/src    //přepnutí se v adresářové struktuře
# wget http://www.snort.org/dl/barnyard/barnyard-0.2.0.tar.gz    //stáhnutí souboru
# tar xvzf barnyard-0.2.0.tar.gz    //rozbalení souboru
# cd barnyard-0.2.0    //vstup do vytvořeného adresáře

# ./configure --enable-mysql    //překlad Barnyardu s podporou MySQL
# make    //přeložení
# make install    //instalace
# cp /usr/local/src/barnyard-0.2.0/etc/barnyard.conf /etc/snort    //zkopírování souboru
```

Následuje modifikace souboru snort.conf:

```
# vim /etc/snort/snort.conf
```

Okomentování řádku:

```
output database: alert, mysql, user=snort password=841024 dbname=snort host=localhost
```

Zrušení komentářů u řádků:

```
# output alert_unified: filename snort.alert, limit 128
# output log_unified: filename snort.log, limit 128
```

Ukončení existujícího procesu Snortu (*kill pid*) a start nového, kvůli uplatnění současných změn.

Nastavení konfiguračního souboru Barnyardu (provedeny pouze minimální změny):

```
# vim /etc/snort/barnyard.conf

config hostname: debian    # název počítače (hostitelské jméno)
config interface: eth0    # nastavení rozhraní, na kterém se poslouchá
```

Konfigurace výstupu pro zápis do databáze se provede komentováním všech výstupních pluginů vyjma tohoto:

```
output alert_acid_db: mysql, sensord_id 0, database snort, server localhost, user snort,
password 841024    # modul alert_acid_db slouží pro zápis poplachů do DB
```

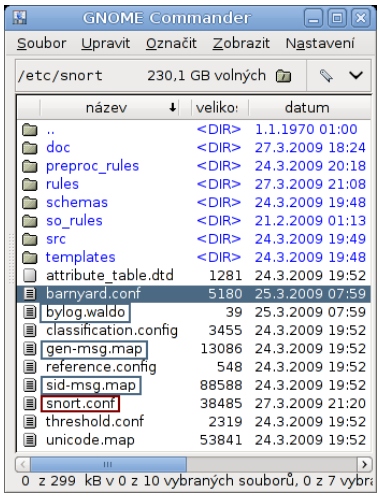
Dále je vytvořen soubor „waldo“, s jehož pomocí je Barnyard užíván nepřetržitě:

```
# cd /etc/snort
# vi bylog.waldo
```

Do souboru bylog.waldo vložíme následující obsah:

```
------------------
/var/log/snort     //Pozn. Třetí řádek tvoří časovou značku (timestamp) asociovanou
snort.alert        //s unifikovanými formáty jednotlivých logovacích souborů Snortu
1240322921         //ve /var/log/snort, příslušný soubor snort.alert => do databáze
0                  //zaznamenány poplachy (Snortem, resp. senzory generované alerty)
------------------
```

Po startu Snortu s novými změnami je vhodné vložit novou časovou značku do tohoto souboru (a přepsat tak třetí zmíněný řádek aktuální hodnotou). Soubor uložíme a ukončíme.



Spuštění programu Barnyard (na pozadí) příkazem:

```
# /usr/local/bin/barnyard -c /etc/snort/barnyard.conf -g \
/etc/snort/gen-msg.map    -s    /etc/snort/sid-msg.map    -d
/var/log/snort -f \
snort.alert -w /etc/snort/bylog.waldo &
```

Pro ujištění, zda Barnyard korektně vkládá zaznamenané události do MySQL databáze, poslouží příkaz:

```
# mysql –u root –p heslo_roota -D snort -e "select count(*)
from event"
+----------+
| count(*) |
+----------+
|    11    |
+----------+
```

5.6.4.10 Automatická aktualizace pravidel Snortu skriptem Oinkmaster

Pokud máme nainstalováno jen několik senzorů IDS, je aktualizace pravidel Snortu rychlý a snadný proces. S rostoucím počtem senzorů se ale aktualizace komplikuje. Naštěstí lze použít Oinkmaster, což je perlový skript, který toho dokáže mnohem více, než jen stáhnout nejnovější pravidla.

Nově stažená pravidla dokáže modifikovat podle již existujících pravidel a umožňuje selektivní vypnutí pravidel, což je užitečné zejména v situacích, kdy máme standardní pravidla Snortu upravena tak, aby lépe vyhovovala užitému prostředí, nebo pokud vypínáme určité typy pravidel, které generují velké množství falešných poplachů. Samozřejmostí je záloha starších souborů s pravidly před přepisem novými. Základní instalace docílíme spuštěním následujících příkazů [39]:

```
# cd /usr/local/src
# wget http://internap.dl.sourceforge.net/sourceforge/oinkmaster/oinkmaster-2.0.tar.gz
# tar xvzf oinkmaster-2.0.tar.gz
# cd oinkmaster-2.0
# cp oinkmaster.pl /usr/local/bin    //zkopírujeme skript Oinkmasteru do /usr/local/bin
# mkdir /usr/local/etc    //vytvoříme v /usr/local adresář etc
# cp oinkmaster.conf /usr/local/etc    //zkopírujeme do něj konfig. soubor Oinkmasteru
```

V konfiguračním souboru je mnoho možností k nastavení, které Oinkmaster nabízí. Cílem bude prosté stažení balíčku pravidel z domovské stránky Snortu (http://www.snort.org).

```
# vim /usr/local/etc/oinkmaster.conf
```

Vyhledáme tuto sekci v konfiguračním souboru oinkmaster.conf:

```
-----------------------------------------------------------------------------------------------------------------------------------------
# As of March 2005, you must register on the Snort site to get access
# to the official Snort rules. This will get you an "oinkcode".
# You then specify the URL as
# http://www.snort.org/pub-bin/oinkmaster.cgi/<oinkcode>/<filename>
# For example, if your code is 5a081649c06a277e1022e1284b and
# you use Snort 2.4, the url to use would be (without the wrap):
# http://www.snort.org/pub-bin/oinkmaster.cgi/
# 5a081649c06a277e1022e1284bdc8fabda70e2a4/snortrules-snapshot-2.4.tar.gz
# See the Oinkmaster FAQ Q1 and http://www.snort.org/rules/ for
# more information.
# URL examples follows. Replace <oinkcode> with the code you get on the
# Snort site in your registered user profile.
# Example for Snort-current ("current" means cvs snapshots).
url = http://www.snort.org/pub-bin/oinkmaster.cgi/f7296b942e25c64dda1871cd146ecce640574680/
snortrules-snapshot-CURRENT.tar.gz
-----------------------------------------------------------------------------------------------------------------------------------------
```

Po získání vlastního „oink“ kódu (viz krok 5.6.1.3) jsme schopni automatizovat činnost správy pravidel. Oinkmaster bere výstupní adresář jako argument s parametrem -o. S jeho pomocí ví, kam stažený soubor s pravidly rozbalit.

```
# mkdir /tmp/oinktest
# /usr/local/bin/oinkmaster.pl -o /tmp/oinktest
```

Loading /usr/local/etc/oinkmaster.conf
*Downloading file from http://www.snort.org/pub-bin/oinkmaster.cgi/*oinkcode*/snortrules-snapshot-CURRENT.tar.gz... done.*
Archive successfully downloaded, unpacking... done.
Setting up rules structures... done.
Processing downloaded rules... disabled 0, enabled 0, modified 0, total=13173
Setting up rules structures... done.
Comparing new files to the old ones... done.
Updating local rules files... done.

[***] Results from Oinkmaster started 20090302 16:57:17 [***]

[*] Rules modifications: [*]
None.

[*] Non-rule line modifications: [*]
None.

[+] Added files (consider updating your snort.conf to include them if needed): [+]

-> attack-responses.rules -> backdoor.rules -> bad-traffic.rules -> chat.rules -> content-replace.rules -> ddos.rules -> dns.rules -> dos.rules -> experimental.rules -> exploit.rules -> finger.rules -> ftp.rules -> icmp-info.rules -> icmp.rules -> imap.rules -> info.rules -> misc.rules -> multimedia.rules -> mysql.rules -> netbios.rules -> nntp.rules -> open-test.conf -> oracle.rules -> other-ids.rules -> p2p.rules -> policy.rules -> pop2.rules -> pop3.rules -> porn.rules -> rpc.rules -> rservices.rules -> scada.rules -> scan.rules -> shellcode.rules -> smtp.rules -> snmp.rules -> specific-threats.rules -> spyware-put.rules -> sql.rules -> telnet.rules -> tftp.rules -> virus.rules -> voip.rules -> VRT-License.txt -> web-activex.rules -> web-attacks.rules -> web-cgi.rules -> web-client.rules -> web-coldfusion.rules -> web-frontpage.rules -> web-iis.rules -> web-misc.rules -> web-php.rules -> x11.rules

Pomocí příkazu ls (list) s parametrem -al vypíšeme detailní obsah adresáře /tmp/oinktest/:

```
# ls -al /tmp/oinktest/
```

S přepínačem -b provede Oinkmaster zálohu současných pravidel na předem určené místo. Nyní jsou pravidla zkopírována v adresáři /tmp/oinktest, pokusíme se je nejprve zálohovat předtím, než budou opětovně získána. Je třeba modifikovat nejméně jedno z existujících pravidel, například takto:

```
# vim /tmp/oinktest/pop3.rules (nahradíme číslo portu 110 za jiné náhodné; port 110 = POP3)
```

Nato spustíme tyto příkazy:

```
# mkdir /tmp/OINKBACK
# /usr/local/bin/oinkmaster.pl -o /tmp/oinktest -b /tmp/OINKBACK
```

Na výstupu bude vypsáno hlášení „Modified Active Rules“ a provedena záloha v /tmp/OINKBACK.

```
-rw-r--r-- 1 root root 256K 2009-04-11 13:28 rules-backup-20090411-185361.tar.gz
```

5.6.4.11 Startovací skripty pro Snort a Barnyard

Vytvoření startovacího skriptu [39]:

```
#vim /etc/init.d/snort-barn
```

Do souboru snort-barn vložíme následující obsah:

```
-------------------------------------------------------------
#!/bin/bash
/sbin/ifconfig eth0 up
/etc/snort/src/snort -Dq -u snort -g snort -c \
/etc/snort/snort.conf -i eth0
/usr/local/bin/barnyard -c /etc/snort/barnyard.conf -g \
/etc/snort/gen-msg.map -s /etc/snort/sid-msg.map -d \
/var/log/snort -f snort.alert -w /etc/snort/bylog.waldo &
-------------------------------------------------------------
```

Přidat práva pro spuštění (učinění vykonatelným, proveditelným – „*eXecutable*“):

```
#chmod +x /etc/init.d/snort-barn
```

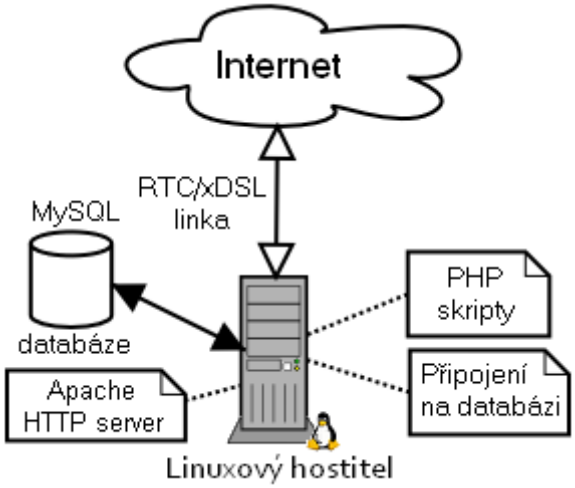
Příkazem update-rc.d nastavíme linky (symb. odkazy) mezi soubory v adresářích runlevelů /etc/rc?.d:

```
#update-rc.d snort-barn defaults 95
```

Nezbývá než restartovat systém a ověřit funkčnost.

5.6.4.12 Podpora z oblasti třetích stran

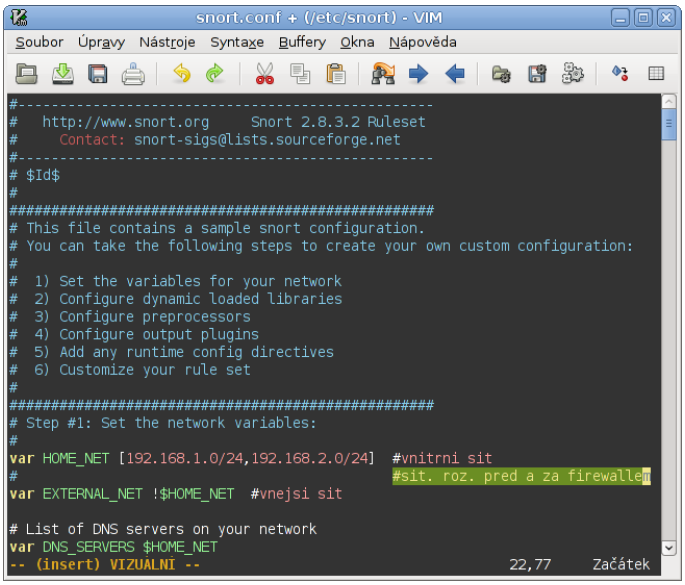
Prezentace výsledků a zhodnocení stanovených závěrů IDS systému Snort podpořili průvodní technologie jako Apache HTTP server, databázový systém MySQL a skriptovací programovací jazyk hypertextového preprocesoru PHP (serverové skriptování).

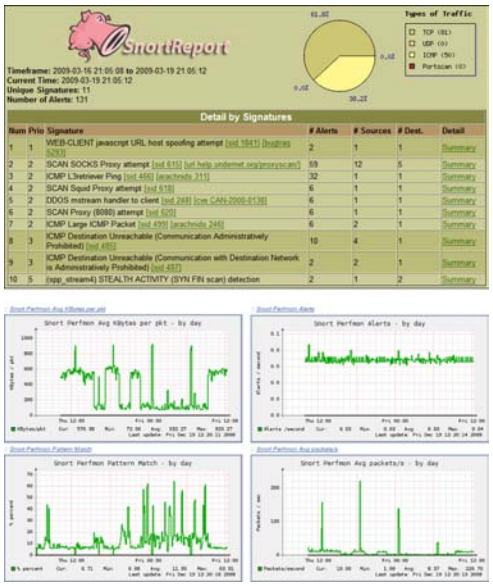


Obr. 53: Kombinace trojlístku Apache, PHP a MySQL jako základního software webového serveru je velmi častá a v praxi běžně nasazovaná

5.6.4.13 Závěr, zhodnocení

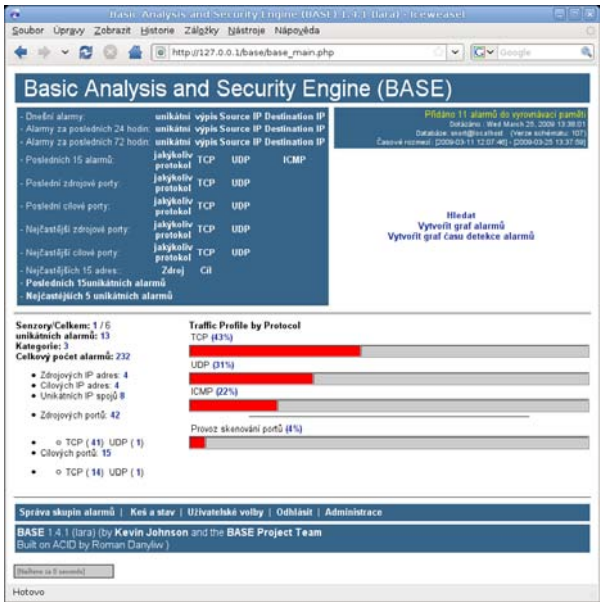
Síťová bezpečnost je závažnou problematikou, stále se vyvíjející a zdokonalující, která si zaslouží zvláštní pozornost. Použitím tak výkonného a mocného operačního systému, jakým Debian bezesporu je, spolu s bezprostředním softwarem jako např. Snort, Barnyard, Oinkmaster, ACID jsme schopni vybudovat spolehlivé řešení IDS systému, a s jeho pomocí monitorovat síťové segmenty na přítomnost nežádoucího (tj. neoprávněného) vniknutí podniknutého útočníkem.

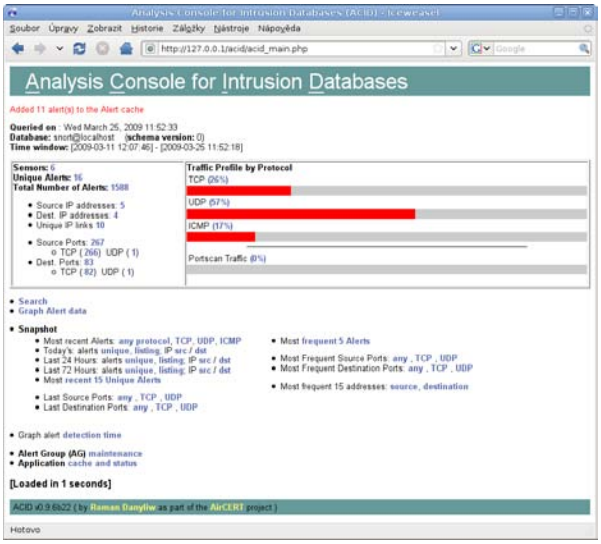


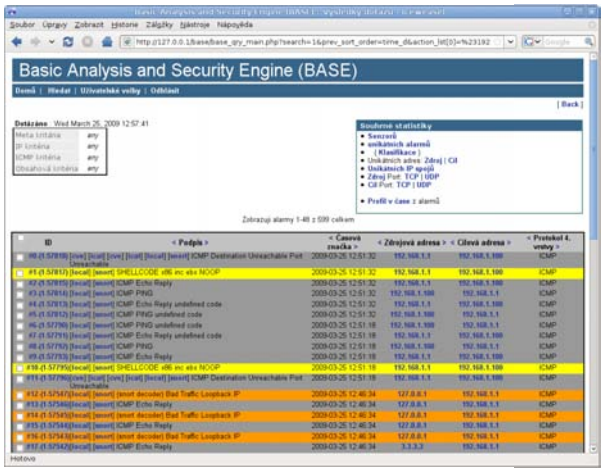


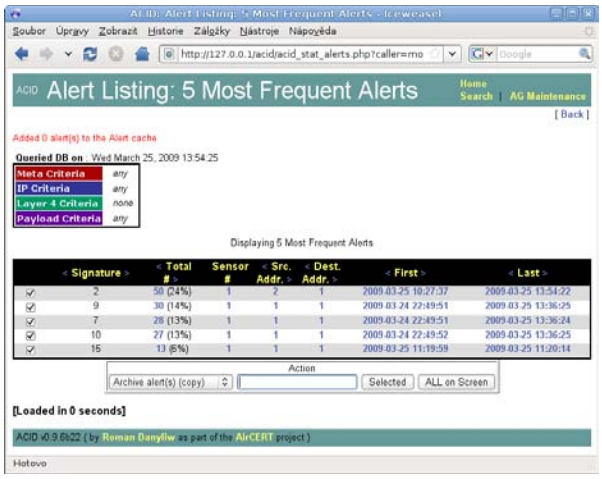
Pozn. Pořízené grafy jsou ze systémové utility Munin, sbírající a generující grafické statistiky různého druhu informací - jedná se tedy o generátor sestav a statistik. SnortReport je přídavný modul pro Snort, generující aktuální webové reporty o zaznamenaných činnostech v reálném čase.

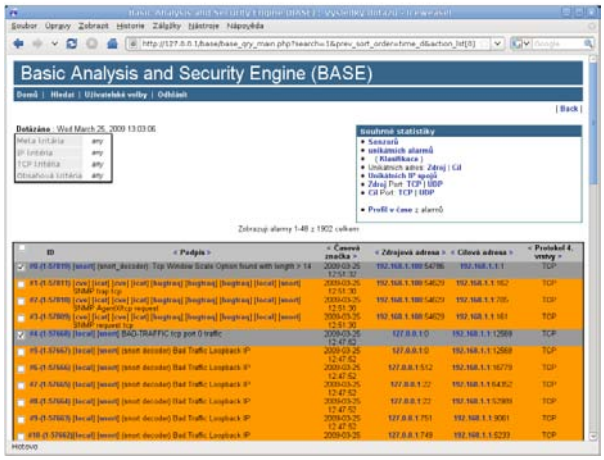
Basic Analysis and Security Engine (BASE), Analysis Console for Intrusion Databases (ACID)

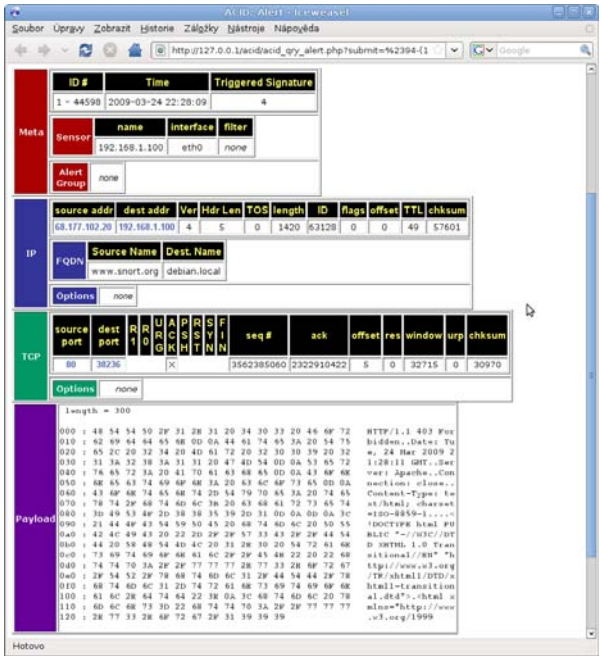




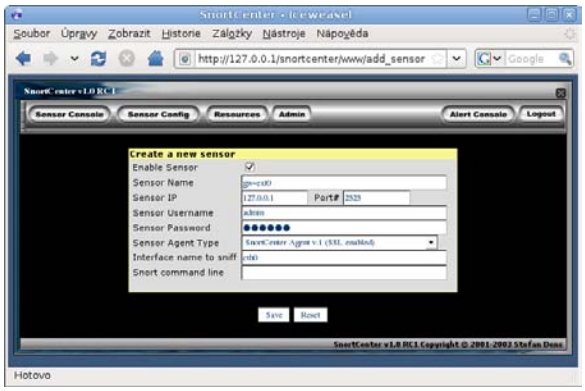


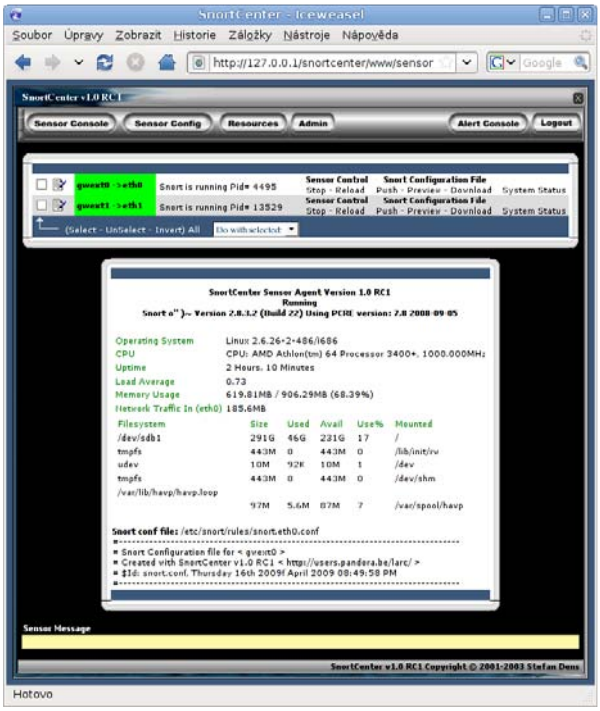






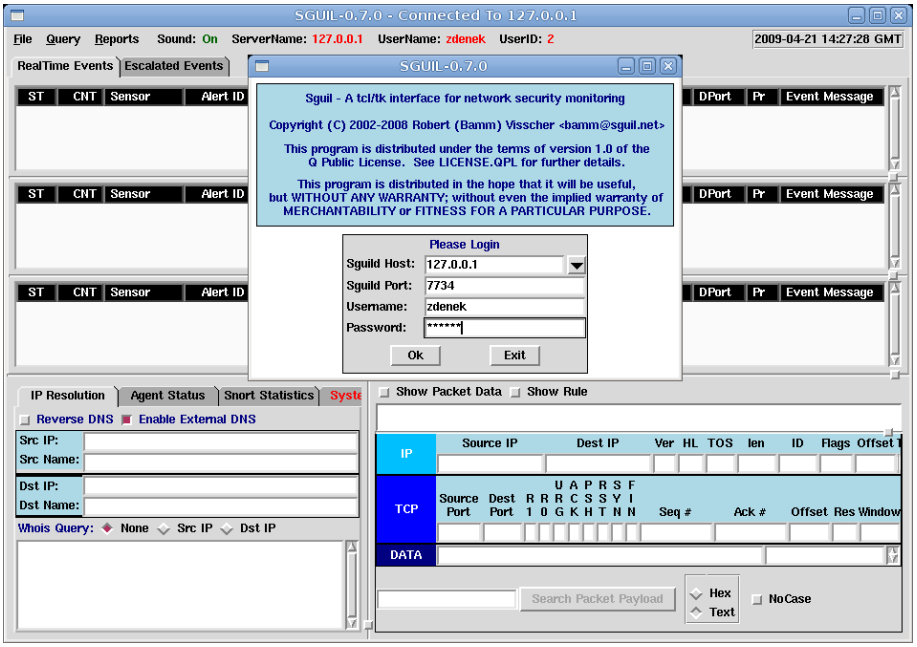
SnortCenter





Přípravu programu SnortCenter (SSL certifikát, generování klíče, senzory, agenti, atd.) je možné zhlédnout v souboru *Snort – doplňky.pdf* umístěném na médiu CD-ROM, které je přílohou práce.

Sguil



Programy BASE, ACID, SnortCenter a Sguil zde nebudou textově rozepisovány kvůli nedostatku potřebného prostoru. U každého programu je uveden příslušný obrázek, který graficky vyjadřuje jeho činnost a je z něj zřetelné nasazení v praxi (v součinnosti se Snortem). U všech uvedených lze říci, že obecně rozšiřují působnost nebo zpřehledňují a zefektivňují činnost programu Snort. BASE a ACID umožňují přístup k databázi Snortu, Sguil sledování v reálném čase, SnortCenter správu sítě senzorů.

Swatch

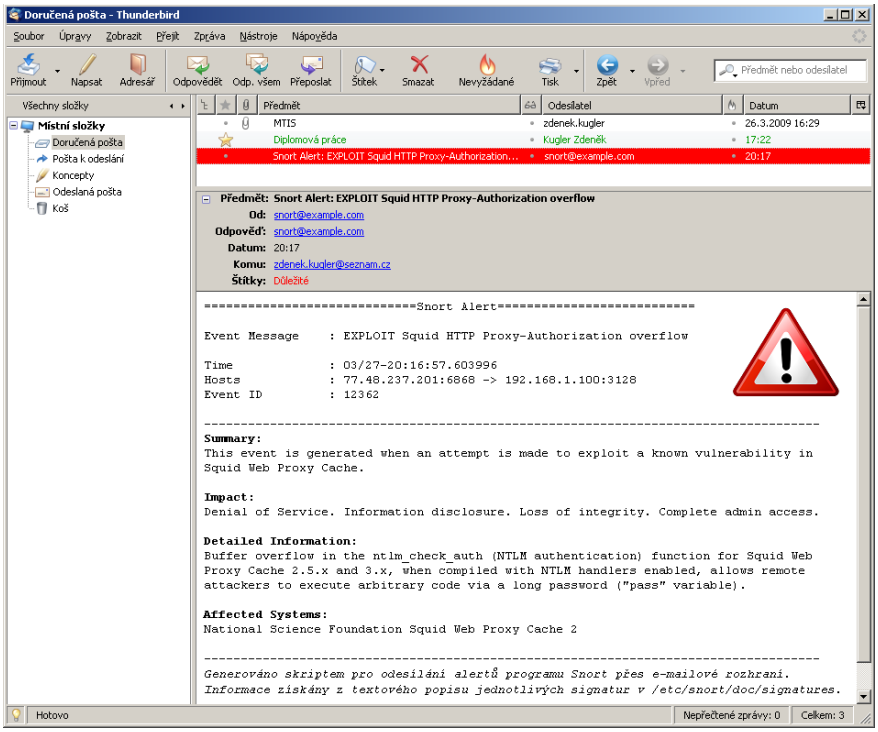
Pomocí programu Swatch (analýza systémových logů) můžeme jednoduše přeposílat alerty - výstrahy Snortu na e-mail, popř. pager:

```
debian:/#swatch -c /root/.swatchrc -t /var/log/snort/alert
*** swatch version 3.2.3 (pid:10386) started at Po bře 23 17:51:41 CET 2009
-------------
Mail sent.
```

Program umí číst logy dávkově nebo průběžně, umí i sledovat výstupy libovolných příkazů. Instalace vytváří konfigurační soubor v /home/user/.swatchrc. Podle nastavení může reagovat výpisem (echo), zvonkem (bell), spuštěním příkazu (exec), e-mailovou zprávou (mail), přeposláním zkoumaného řádku uživateli (write), předáním řádku na standardní vstup zadaného programu (pipe) nebo odlišit reakci při opakování události (throttle). Konfigurační soubor obsahuje vzorce a skupiny akcí. Protože jde o program napsaný v Perlu, vzorcem musí být perlový regulární výraz.

Jednotlivé konfigurační kroky spočívají v instalaci programu Swatch (perl Makefile.PL, make, make test, make install, make realclean), instalaci potřebných CPAN modulů, dále spuštění Snortu v NIDS módu (snort -c /etc/snort/snort.conf -l /var/log/snort), startu programu Swatch (viz příkaz výše) a konečně dodatečné nastavení POP3 účtu k správnému přijetí e-mailů [40].

```
#---------------#    // Test doručení varovného e-mailu do elektronické schránky administrátora.
# EXPLOIT RULES #    // Ten po příjmu e-mailu může zavčas zareagovat na vzniklé události v síti
#---------------#    // (firewall) a konkrétním postupem zabránit např. škodám na aktivech firmy.

ATTACK: EXPLOIT Squid HTTP Proxy-Authorization overflow
ATTACK TYPE: attempted-user

alert tcp $EXTERNAL_NET any -> $HOME_NET 3128 (msg:"EXPLOIT Squid HTTP Proxy-Authorization overflow";
flow:established,to_server; content:"Proxy-Authorization|3A| NTLM TlRMTVNTUAADA"; content:!"AAAGAAYA";
within:8; reference:bugtraq,10500; reference:cve,2004-0541; classtype:attempted-user; sid:12362; rev:1;)
```



Obr. 54: Příchozí varovné hlášení informuje vzdáleného uživatele o hrozícím nebezpečí

Konfigurační soubor programu Swatch „/root/.swatchrc“ modifikovaný pro Snort a linuxový perlový skript „/usr/local/bin/snort_email_alert.pl“ pro odesílání alertů Snortu e-mailem jsou přílohou CD-ROM.

5.7 Princip fungování, sestavení a součinnost systémů

Na linuxovém stroji s Debianem spolu vzájemně kooperují nebo se doplňují tyto systémy:

1) Linuxový firewall Iptables v1.4.2 – framework Netfilter, integrace v jádru, stavová inspekce
2) Proxy cache server Squid 3.0 – stěžejní klíčová část systému; základ, na kterém se „staví“
3) Antivirová proxy HAVP (HTTP Antivirus Proxy) 0.89 – rozhraní mezi Squidem a ClamAV
4) Antivirový program ClamAV 0.94.2 - „modul“ pro HAVP
5) Content filter ufdbGuard 1.15 (založen na softwaru URLfilterDB) – internetový filter, blokace www
6) Systém detekce průniků (IDS) Snort 2.8.3.2 – detektor rozpoznávající síť. útoky či pokusy o napadení

Pozn.: program ufdbGuard je nutné doplnit ještě obsáhlou databází – blacklistem (nebo též černou listinou). K dispozici jsou volně dostupné i komerční seznamy s různými úrovněmi a zaměřením blokovaných kategorií. Velmi jednoduše lze vytvářet nové či upravovat současné kategorie filtrů. Podporován je zápis doménových jmen (názvů) i IP adres. Velmi důležité z hlediska efektivity a aktuálnosti blokace jsou aktualizace, které jsou vydávány téměř denně (zbytečná blokace domény, která je již dlouho nefunkční, zatímco nově vzniklé projdou).

Ve finální fázi je funkčnost celého řetězce systému následující. Uživatelé vnitřní sítě jsou pomocí HTTP proxy Squid připojeni k Internetu. Veškerá komunikace od uživatelů směřující na port 80 (cílový port – implicitně využívaný HTTP protokolem pro službu WWW) s libovolnou IP adresou cíle a zdrojovou IP adresou v rozsahu místní sítě je přesměrována s použitím pravidel firewallu (resp. nástroje) Iptables na registrovaný „bind“ port Squidu - 3128, na kterém aktivně naslouchá příchozím požadavkům. Jak plyne z kontextu, je aplikován tzv. transparentní režim Squidu, ve kterém není třeba žádných změn v internetových prohlížečích uživatelů (žádná síťová nastavení či změny parametrů). O vše se postará výše zmíněný firewall, který mimo jiné inicializuje databázi modulů, zajistí obranu před DoS útoky, zamezí IP spoofingu, zaručí základní kvalitu služeb (QoS), a hlavně svým nasazením a zásahem do komunikace odfiltruje nepovolené pakety v síťovém provozu. Jeho prostřednictvím je aktivováno taktéž směrování paketů. Poté Squid s použitím externího redirectoru – konkrétně content filteru (systému pro regulaci internetového obsahu) ufdbGuard zkontroluje, zdali se požadovaná URL adresa nenachází na seznamu blacklistu (tzv. seznam černé listiny zakázaných entit). V případě, že vyskytuje, je přístup okamžitě zamítnut a uživatel o situaci informován chybovým hlášením např. v podobě varovné HTML stránky (Error 403 - "Access Denied/Forbidden"; přístup k požadovanému dokumentu/stránce zamítnut v důsledku nedostatečných přístupových práv, zakázaný přístup!, atp.) s odůvodněním blokace požadavku klienta a dalšími souvisejícími informacemi (např. do jaké filtrované kategorie přísluší, kontakt na administrátora systému v případě neoprávněné blokace, a možné další). Veškeré vzniklé události jsou navíc zaznamenány (textový soubor, databáze), takže je možné zpětné dohledání => prokazatelnost dat. V opačném případě je HTTP požadavku vyhověno a pokračuje se kontaktováním vzdáleného webového serveru (spojení od klienta ukončeno, navázáno nové spojení směrem k WWW serveru). Proxy tak přebírá funkci klientské strany, vůči serveru se tváří jako prostý klient (např. PC1). Tím, že proxy vystupuje za klienta, se řeší nejen překlad adres z určitého počtu privátních na jednu veřejnou, ale i významně zvyšuje zabezpečení. Po vyřízení požadavku serverem je navrácená odpověď (než je předána klientovi, který o ní žádal) podrobena prostřednictvím parentní (rodičovské) proxy HAVP antivirové kontrole, které se ujímá antivirový program ClamAV. Obsah odpovědi (může se jednat o www stránku, online dokument, binární nebo textový soubor, atd.) je testován na přítomnost virů, červů, trojských koní, včetně podvržených stránek – phishing a dalšího malware, který je podporován (např. rootkity, které jsou využívány zejména pro ukrytí určitého kódu před konvenčními bezpečnostními nástroji, jako jsou antiviry, IDS).

Funkčnost on-access skeneru („rezidentního štítu“) antiviru ClamAV byla otestována na EICAR souborech obsahujících fiktivní virový vzorek, a to reakcí přerušením spojení při přístupu k těmto souborům s příponami .com, .txt a dvakrát archivní formát .zip. Standardní testovací soubory EICAR jsou přístupné na internetové adrese http://www.eicar.org/. Při detekci virové infekce je uživatel opět informován zástupnou HTML stránkou o vzniklých událostech a hrozícímu nebezpečí virové nákazy. Následuje zpětná obsahová kontrola. Kromě URL filteru (zmiňovaného při odchozím spojení, kdy mohl zareagovat pouze tzv. blacklisting) mohou být uplatněny také další používané metody a režimy content scanningu, jako např. metoda prohledávání obsahu, kdy jsou vyhledávána a blokována definovaná slova (např. drogy, sex, warez), slovní spojení (např. dětská pornografie), vymezené fráze (např. rasistické hanlivé fráze) apod. vyskytující se na stránce, dále soubory určitých přípon (zip, rar, avi, jpg a další) nebo pokročilá blokace internetových stránek na základě umělé inteligence (AI), která si dokáže velmi slušně poradit například i s obrázky a videem (obecně rozlišení a kategorizace multimediálního obsahu). I když se nyní nemusí jednat o úplně stoprocentní spolehlivé řešení, v budoucnu se může stát velmi perspektivní technologií. Inspekce je prováděna při příchozím spojení, kdy je již známa struktura a datový obsah například klientem požadované webové stránky. Pokud je v obsahu např. požadované www stránky nalezeno některé ze zakázaných slov, slovních spojení, atp., dojde opět k blokaci celé domény. Vše ovšem záleží na aktuálním nastavení, přesněji, která filtrovací metoda je použita (k dispozici jsou kromě osvědčených i experimentální v podobě doposud vyvíjených a stále se zlepšujících technik). Rovněž ne vždy a za všech okolností se vyplácí blokovat všechno veškerými filtry, ideálním řešením je v dané situaci nalezení vhodného kompromisu. Teprve po provedení všech těchto obsahových kontrol a skenování je čistá/prověřená HTTP odpověď (rozuměno bez virů a nepovolených či zakázaných znaků) předána klientovi, který o ní původně žádal a nyní s ní může dle libosti naložit. Zde uváděné kontroly se bez výjimky prováděly na předřazeném serveru. Je více než pravděpodobné, že serverem ověřená a posléze klientem získaná odpověď bude podrobena případné (opakované) kontrole např. firewallem, antivirem, antispywarem, antispamem, a to u hostitele (byť se samozřejmě nejedná o stejný bezpečnostní software jako na serveru – např. linuxovým serverem ověřená odpověď (FW, AV, CF) je předána klientovi s OS Windows či Mac OS).

Veškeré informace uvedené výše platí pouze za situace, u níž se požadovaná odpověď nenachází v lokální proxy cache, např. při prvotním kontaktu daného serveru (při opakovaném už tomu tak není). V případě nalezení odpovědi na hostitelský požadavek v cache paměti Squidu je tato popsaná procedura podstatně zkrácena a zjednodušena. Nedochází k navázání spojení se serverem, odpověď, pakliže je stále aktuální, je navrácena klientovi ihned => výhody, jako např. rychlost, odezva, úspora přenosového pásma, hospodárnost při využívání dostupné kapacity komunikační linky. Čím více uživatelů sítě se „ptá“ na stejný požadavek (v určitém časovém horizontu), tím větší je efektivita využití cache.

Problém nastává s příchodem dynamických stránek, v případě, že jejich autoři práci s cache neřeší. Potom proxy server neví, jak dlouho může uchovat uloženou stránku, aniž by zastarala. V takovém případě musí vždy stáhnout stránku znovu, aby klientovi zajistil vždy aktuální stránku a ne týden staré "aktuality". V takovém případě místo úspory a zrychlení dojde k mírnému zpomalení, protože na stažení se podílí další server.

Proxy se také může používat pro záznam provozu na Internetu, přesněji co, kdo, kdy a kde dělal. Proxy server, jakožto prostředník, zná celé URL, a může je tedy zaznamenávat do logovacích souborů. Oblast využití proxy serverů je nejčastěji ve školách, firmách, organizacích, úřadech, ale i ISP.

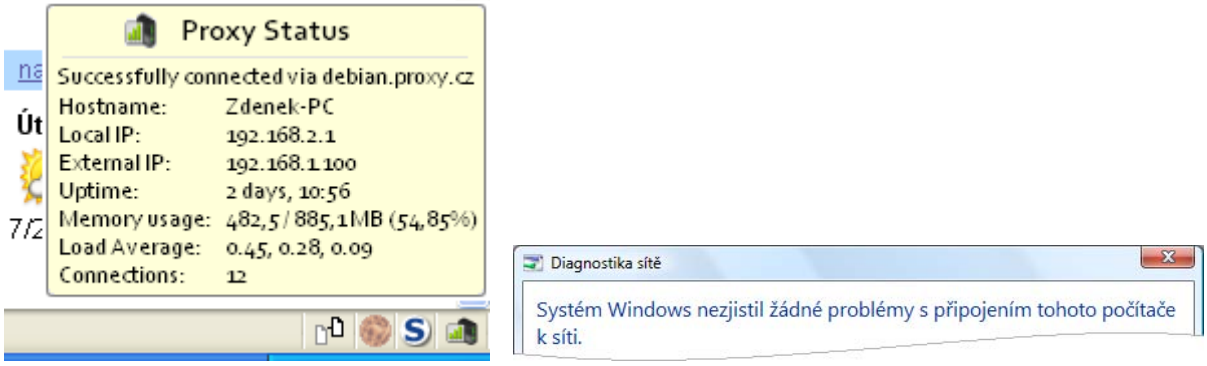
Ještě bych zmínil několik důležitých aspektů. • Lokální proxy cache je provozována za parentní antivirovou proxy HAVP, která spolupracuje s produktem ClamAV. • Protože se jedná o parentní proxy, tak HAVP kontroluje jen to, co jde z Internetu k nám (downlink). Máme-li cache už zavirovanou, HAVP to neřeší. • HAVP neumí a nekontroluje SSL spojení - zde je možný potenciál zavirování. • HAVP se dá použít i s jinými antiviry – AVG, Avast!, NOD32, Kaspersky atd. • Jelikož je ClamAV poměrně schopný a výkonný antivir, leckdy má i rychlejší aktualizaci databáze (update), než třeba řada komerčních řešení.

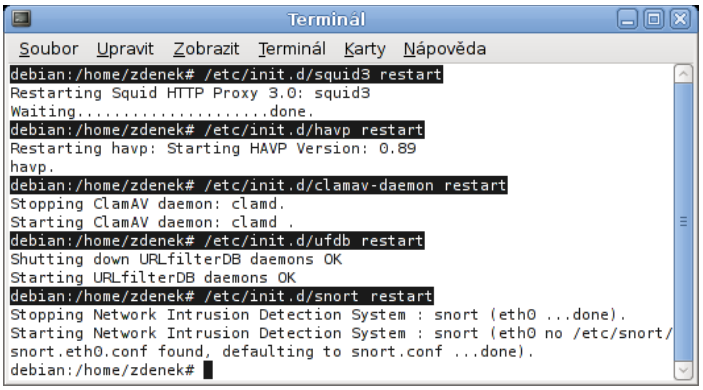
Posledním provozovaným bezpečnostním systémem na stroji s Debianem je Snort. Snort patří mezi systémy detekce průniků (IDS – *Intrusion Detection System*) a jeho cílem je doplnit a podpořit ostatní aplikované systémy, a to detekcí neautorizovaných přístupů a monitorováním (odposloucháváním) provozu na lokální síti. Je možné ho nepřímo zařadit také mezi systémy zvyšující zabezpečení v počítačových sítích (PC, serverů). Nemusí se vždy bezpodmínečně jednat pouze o detekci síťového útoku, ale i obranná protiopatření v reálném čase (např. aktivní samočinná obrana při útoku hackera). U těchto případů je obvyklá velmi úzká spolupráce s firewally a podobným bezpečnostním softwarem. Důležitou roli zde mimo jiné hraje také rychlost reakce administrátora systému na detekovanou probíhající činnost, proto jsou alerty s vyšší prioritou útoků přeposílány na e-mailovou schránku nebo pager.

Zkompilovaný a nainstalovaný Snort je funkčně nakonfigurován s dvěma senzory, jedním před proxy firewallem (adresa sítě 192.168.1.0) a druhým za proxy firewallem (adresa sítě 192.168.2.0). Jelikož Snort ve výchozím nastavení generuje nadměrné množství planých výstražných alertů neúnosných pro logovací činnost a snadné vyhledávání mezi nalezenými a zaznamenanými útoky, byly po důkladném zvážení použity jen některé vhodné preprocesory a aplikována pouze některá z pravidel. Z výše uvedeného je patrné, že Snort není univerzálním programem, který je schopný ihned odvádět kvalitativní hodnotnou práci, bez jakéhokoliv nastavení a základní konfigurace - je nutné provézt optimalizaci vzhledem k místním podmínkám a topologii sítě dlouhodobým testováním, upravováním a dolaďováním pravidel (kromě oficiálních pravidel lze získat i doplňující komunitní čili uživatelská pravidla). Více informací viz přiložený konfig. soubor programu Snort a obecný popis systémů IDS.

U zprovozněného IDS Snort byly odzkoušeny a stručně popsány všechny tři režimy práce - Sniffer, Logger i NIDS. Přednostně ovšem bylo pojednáváno o nejvýznamnějším režimu NIDS, kterému byla věnována největší pozornost – veškeré doprovodné obrázky, popisy, grafy, testy, provedené audity atd. se týkaly právě tohoto módu.

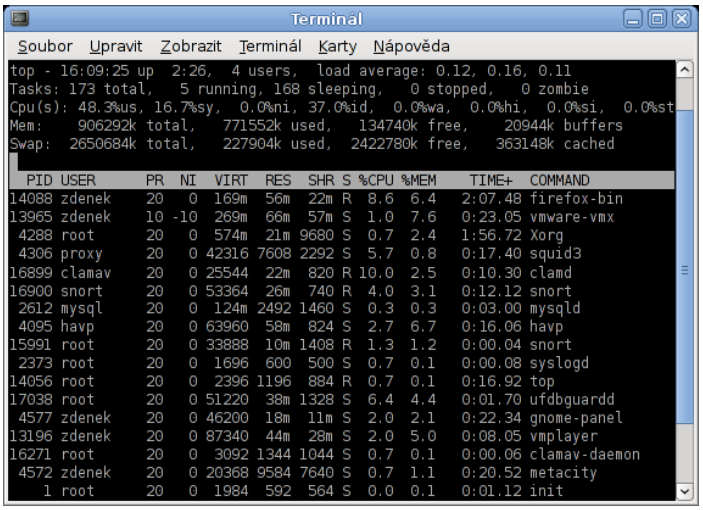
Vše (Squid, Havp, ClamAV, ufdbGuard, Snort a firewall Iptables) ověřeno a odzkoušeno několikaměsíčním provozem v domácím prostředí na třech počítačích bratrů (s různými operačními systémy a internetovými prohlížeči) a dvoudenním ostrým provozem u místního poskytovatele internetového připojení (ISP) na cca 180 online uživatelích obce. Kontaktovaná osoba pan Petr Kubeš.



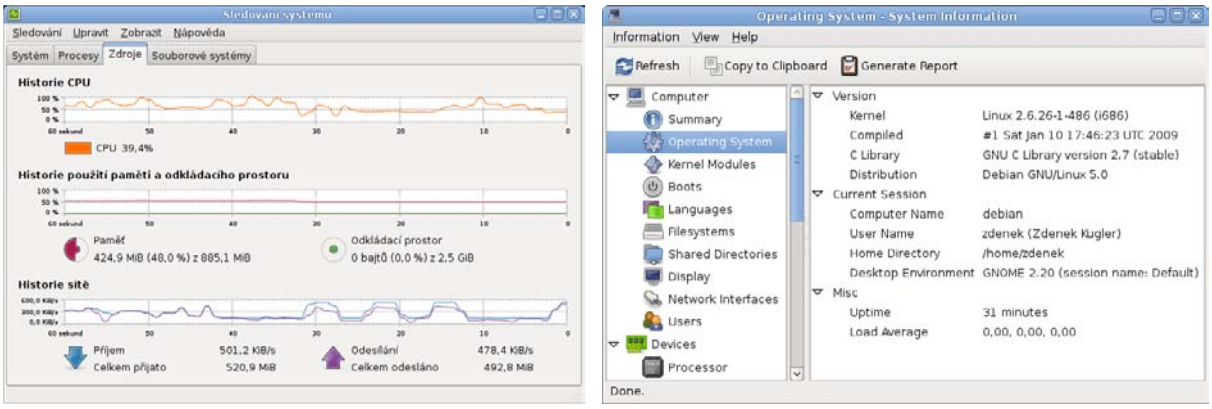


Obr. 55: Restart daemonů a ověření správnosti instalace programů (v případě chyby výpis)

Příkaz (resp. program) top vytvoří pravidelně se obnovující (defaultně po 5 sekundách) tabulku s procesy seřazenými dle zvolených kritérií. Programu lze zadávat parametry jak při spuštění (v příkazovém řádku) tak i interaktivně při běhu programu a tím získat informace přesně podle potřeby. Z programu lze také posílat signály různým procesům. Díky dynamickému charakteru lze v reálném čase sledovat změny stavu procesů, jejich boj o procesor, současnou velikost paměti, kterou procesy alokují a další. Zaznamenaný obrázek 56. zachycuje všechny aktuální procesy, důležité jsou zejména instance procesů (chronologicky) squid3, clamd, snort, havp, ufdbguardd a clamav-daemon.



Obr. 56: Příkaz „top“ vypisuje aktuálně běžící procesy a další důležité systémové informace



Obr. 57: Zatíženost systému (a ostatní parametry včetně historie CPU, RAM, SWAP a sítě) lze s přehledem sledovat a vyhodnocovat v okně Sledování systému -> položka Zdroje

5.8 Prezentace výsledků dosažené práce

Výsledky dosažené práce výstižně charakterizují jednotlivé obrázky doprovázené popisy, ze kterých je zřejmá hlavní podstata a účel nasazení v praxi. Jedná se o proxy server, antivirus, obsahový filter, IDS.

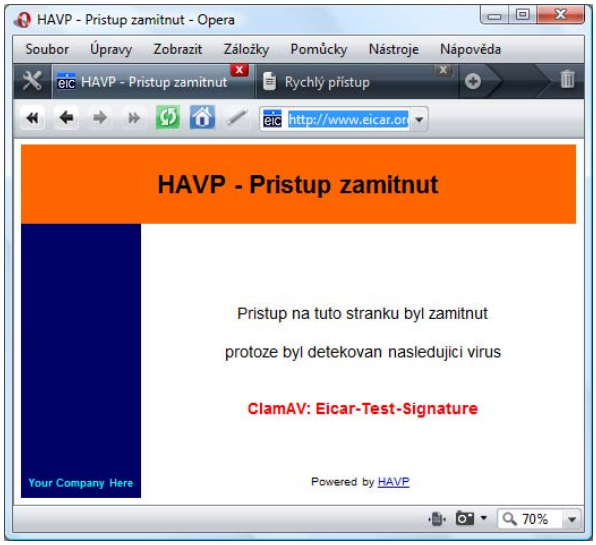
CacheHost: localhost
ErrPage: ERR_ACCESS_DENIED
Err: [none]
TimeStamp: Wed, 28 Jan 2009 12:21:13 GMT

ClientIP: 192.168.2.1
ServerIP:

HTTP Request:
GET / HTTP/1.1
Host: www.freefoto.cz
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; cs; rv:1.9.0.5) Gecko/2008120122 Firefox/3.0.5
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/*;q=0.8*
Accept-Language: cs,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1250,utf-8;q=0.7,;q=0.7*
Keep-Alive: 300
Proxy-Connection: keep-alive

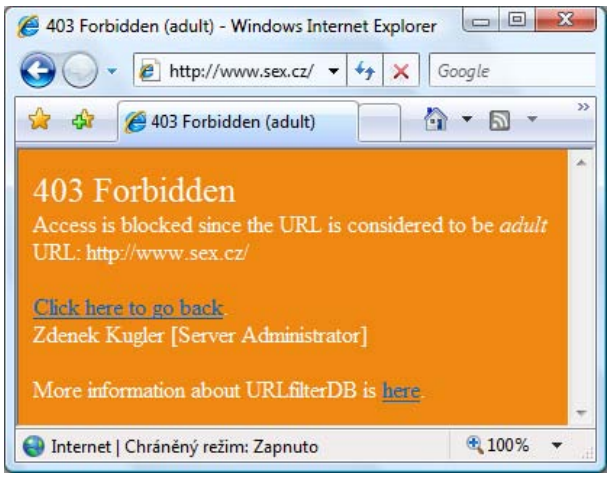
***Squid 3.0** (použit webový prohlížeč Mozilla Firefox)*

Klienti (uživatelé) lokální sítě jsou k Internetu připojeni přes prostředníka v podobě forward proxy serveru Squid. Jsou jim poskytnuty veškeré dostupné výhody, které toto řešení v současnosti nabízí. Proxy je nastavena jako transparentní, kdy veškeré požadavky směřující na port 80 (HTTP) jsou pomocí Iptables přesměrovány na port příslušející Squidu (tj. port 3128, na kterém naslouchá) a není tedy třeba žádných změn či úprav na straně klientů. Squid je doplněn antivirovou kontrolou, kterou zajišťuje GNU/GPL program ClamAV. ClamAV je se Squidem propojen přes rozhraní antivirové proxy HAVP (HTTP Antivirus Proxy), která umí přímo využít ClamAV a slouží Squidu jako „parent“ (čili rodičovská) proxy. Po konfiguraci a restartu Squidu, HAVP a ClamAV je proxy otestována vstupem na fiktivní zavirované stránky a požadavkem stažení souboru (bin, text) s virovou infekcí.

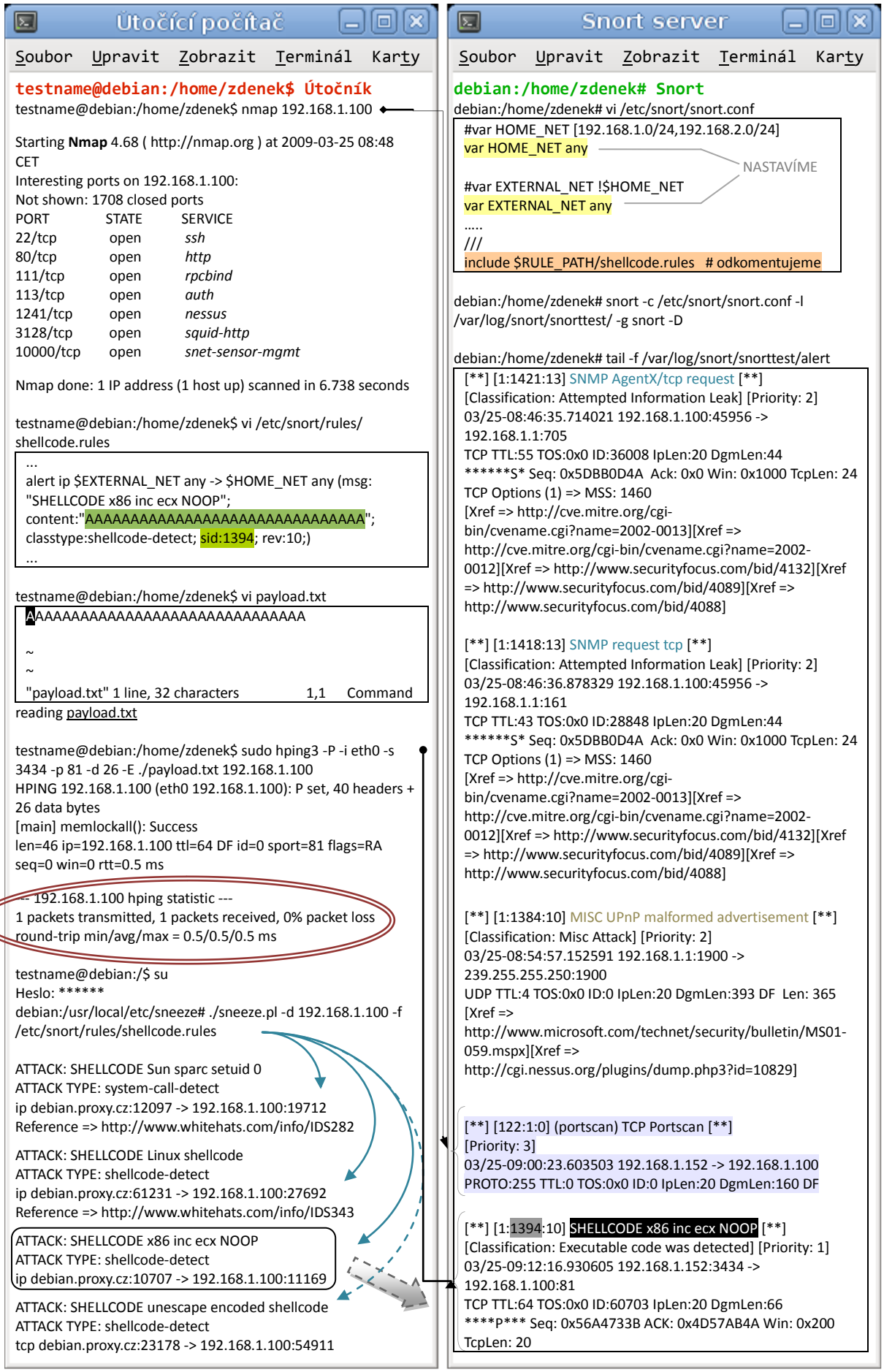


***HAVP 0.89 & ClamAV 0.94.2** (použit webový prohlížeč Opera)*

V pořadí dalším nasazeným systémem, starajícím se o regulaci internetového obsahu, a tedy disponujícím možností blokovat či povolovat příslušné zdroje na Internetu (www stránky, dokumenty, atd.) na základě provedení inspekce obsahu a porovnání se seznamem černých (tj. zakázaných) stránek – blacklistem, je program ufdbGuard doplněný bezplatnou databází. Posledním provozovaným bezpečnostním nástrojem je systém sloužící k detekci průniků do sítě – IDS Snort. Jeho činnost je podobná síťovým snifferům. Na základě identifikátorů se snaží odlišit činnost útočníka (narušitele) od běžné činnosti uživatelů. Identifikuje a hlásí neschválené, neautorizované či podezřelé aktivity. Je provozován v NIDS módu. Kromě souboru zapisuje i do databáze.



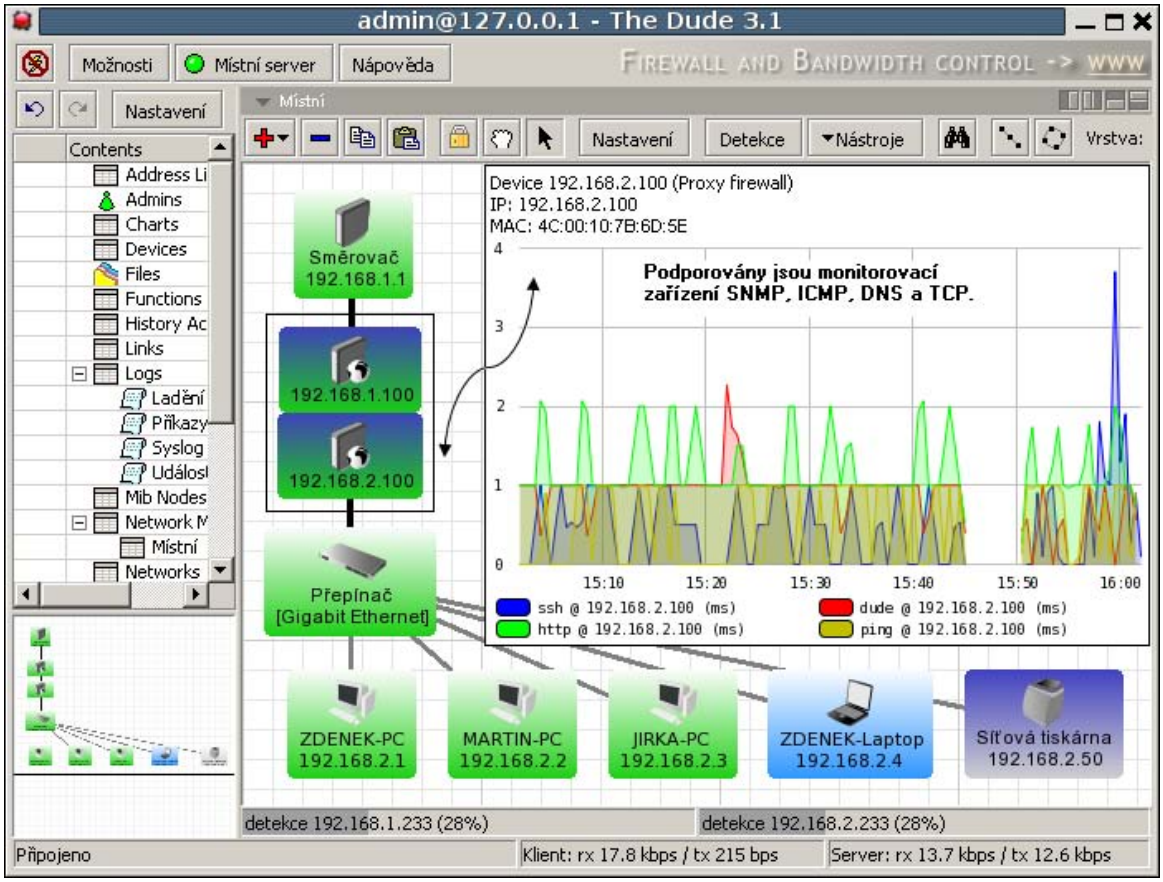
***ufdbGuard 1.15** (použit webový prohlížeč Internet Explorer)*


Útočící počítač
Soubor Upravit Zobrazit Terminál Karty
testname@debian:/home/zdenek$ Útočník
testname@debian:/home/zdenek$ nmap 192.168.1.100
Starting Nmap 4.68 (http://nmap.org) at 2009-03-25 08:48 CET
Interesting ports on 192.168.1.100:
Not shown: 1708 closed ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http
111/tcp open rpcbind
113/tcp open auth
1241/tcp open nessus
3128/tcp open squid-http
10000/tcp open snet-sensor-mgmt
Nmap done: 1 IP address (1 host up) scanned in 6.738 seconds
testname@debian:/home/zdenek$ vi /etc/snort/rules/shellcode.rules
...
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"SHELLCODE x86 inc ecx NOOP";
content:"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";
classtype:shellcode-detect; sid:1394; rev:10;)
...
testname@debian:/home/zdenek$ vi payload.txt
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
~
~
"payload.txt" 1 line, 32 characters 1,1 Command
reading payload.txt
testname@debian:/home/zdenek$ sudo hping3 -P -i eth0 -s 3434 -p 81 -d 26 -E ./payload.txt 192.168.1.100
HPING 192.168.1.100 (eth0 192.168.1.100): P set, 40 headers + 26 data bytes
[main] memlockall(): Success
len=46 ip=192.168.1.100 ttl=64 DF id=0 sport=81 flags=RA seq=0 win=0 rtt=0.5 ms
--- 192.168.1.100 hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.5/0.5/0.5 ms
testname@debian:/$ su
Heslo: ******
debian:/usr/local/etc/sneeze# ./sneeze.pl -d 192.168.1.100 -f /etc/snort/rules/shellcode.rules
ATTACK: SHELLCODE Sun sparc setuid 0
ATTACK TYPE: system-call-detect
ip debian.proxy.cz:12097 -> 192.168.1.100:19712
Reference => http://www.whitehats.com/info/IDS282
ATTACK: SHELLCODE Linux shellcode
ATTACK TYPE: shellcode-detect
ip debian.proxy.cz:61231 -> 192.168.1.100:27692
Reference => http://www.whitehats.com/info/IDS343
ATTACK: SHELLCODE x86 inc ecx NOOP
ATTACK TYPE: shellcode-detect
ip debian.proxy.cz:10707 -> 192.168.1.100:11169
ATTACK: SHELLCODE unescape encoded shellcode
ATTACK TYPE: shellcode-detect
tcp debian.proxy.cz:23178 -> 192.168.1.100:54911
Snort server
Soubor Upravit Zobrazit Terminál Karty
debian:/home/zdenek# Snort
debian:/home/zdenek# vi /etc/snort/snort.conf
#var HOME_NET [192.168.1.0/24,192.168.2.0/24]
var HOME_NET any
NASTAVÍME
#var EXTERNAL_NET !$HOME_NET
var EXTERNAL_NET any
.....
///
include $RULE_PATH/shellcode.rules # odkomentujeme
debian:/home/zdenek# snort -c /etc/snort/snort.conf -l /var/log/snort/snorttest/ -g snort -D
debian:/home/zdenek# tail -f /var/log/snort/snorttest/alert
[**] [1:1421:13] SNMP AgentX/tcp request [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/25-08:46:35.714021 192.168.1.100:45956 -> 192.168.1.1:705
TCP TTL:55 TOS:0x0 ID:36008 IpLen:20 DgmLen:44
******S* Seq: 0x5DBB0D4A Ack: 0x0 Win: 0x1000 TcpLen: 24
TCP Options (1) => MSS: 1460
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012][Xref => http://www.securityfocus.com/bid/4132][Xref => http://www.securityfocus.com/bid/4089][Xref => http://www.securityfocus.com/bid/4088]
[**] [1:1418:13] SNMP request tcp [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/25-08:46:36.878329 192.168.1.100:45956 -> 192.168.1.1:161
TCP TTL:43 TOS:0x0 ID:28848 IpLen:20 DgmLen:44
******S* Seq: 0x5DBB0D4A Ack: 0x0 Win: 0x1000 TcpLen: 24
TCP Options (1) => MSS: 1460
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0013][Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0012][Xref => http://www.securityfocus.com/bid/4132][Xref => http://www.securityfocus.com/bid/4089][Xref => http://www.securityfocus.com/bid/4088]
[**] [1:1384:10] MISC UPnP malformed advertisement [**]
[Classification: Misc Attack] [Priority: 2]
03/25-08:54:57.152591 192.168.1.1:1900 -> 239.255.255.250:1900
UDP TTL:4 TOS:0x0 ID:0 IpLen:20 DgmLen:393 DF Len: 365
[Xref => http://www.microsoft.com/technet/security/bulletin/MS01-059.mspx][Xref => http://cgi.nessus.org/plugins/dump.php3?id=10829]
[**] [122:1:0] (portscan) TCP Portscan [**]
[Priority: 3]
03/25-09:00:23.603503 192.168.1.152 -> 192.168.1.100
PROTO:255 TTL:0 TOS:0x0 ID:0 IpLen:20 DgmLen:160 DF
[**] [1:1394:10] SHELLCODE x86 inc ecx NOOP [**]
[Classification: Executable code was detected] [Priority: 1]
03/25-09:12:16.930605 192.168.1.152:3434 -> 192.168.1.100:81
TCP TTL:64 TOS:0x0 ID:60703 IpLen:20 DgmLen:66
****P*** Seq: 0x56A4733B ACK: 0x4D57AB4A Win: 0x200
TcpLen: 20

Snort 2.8.3.2 *– útočník v levém okně (představující vzdálený počítač hackera) se snaží zjistit co nejvíce informací a zaútočit na server v lokální síti. Pravé okno představuje server Snortu, kde je veškerá aktivita útočníka zaznamenána a podrobena analýze zkoumání. Uvedená situace je pouze modelová, v praxi se často jedná o mnohem sofistikovanější útoky, naplánované a detailně promyšlené s předstihem. Útočníci mnohdy disponují velmi dobrou znalostí bezpečnostní problematiky sítí, proto jsou kromě IDS systémů (ať aktivních nebo pasivních) nasazovány také návnady v podobě HoneyPotů („Hrnců s medem"), simulující síť zranitelných systémů. Zatímco útočník ztrácí energii i čas na bezcenných cílech segmentů sítě, dochází k získávání cenných informací o útočníku a jeho metodách => následuje další zabezpečení sítě, protože útočníkovi se podařil průnik skrze (vnější, hlavní) firewall [15].*

Pozn. Správně nasazený IDS systém je nedílnou součástí bezpečnostního perimetru, který v předstihu upozorňuje administrátora na některé bezpečnostní události, které jsou nebo mohou být reálnou hrozbou pro organizaci či firmu.

5.8.1 Jednoduchý dohled a správa sítě



Obr. 58: Program Dude slouží pro detailní monitorování a snadnější správu sítě. Aplikace automaticky skenuje všechna dostupná síťová zařízení a podporuje i zobrazení struktury sítě v přehledné mapě.

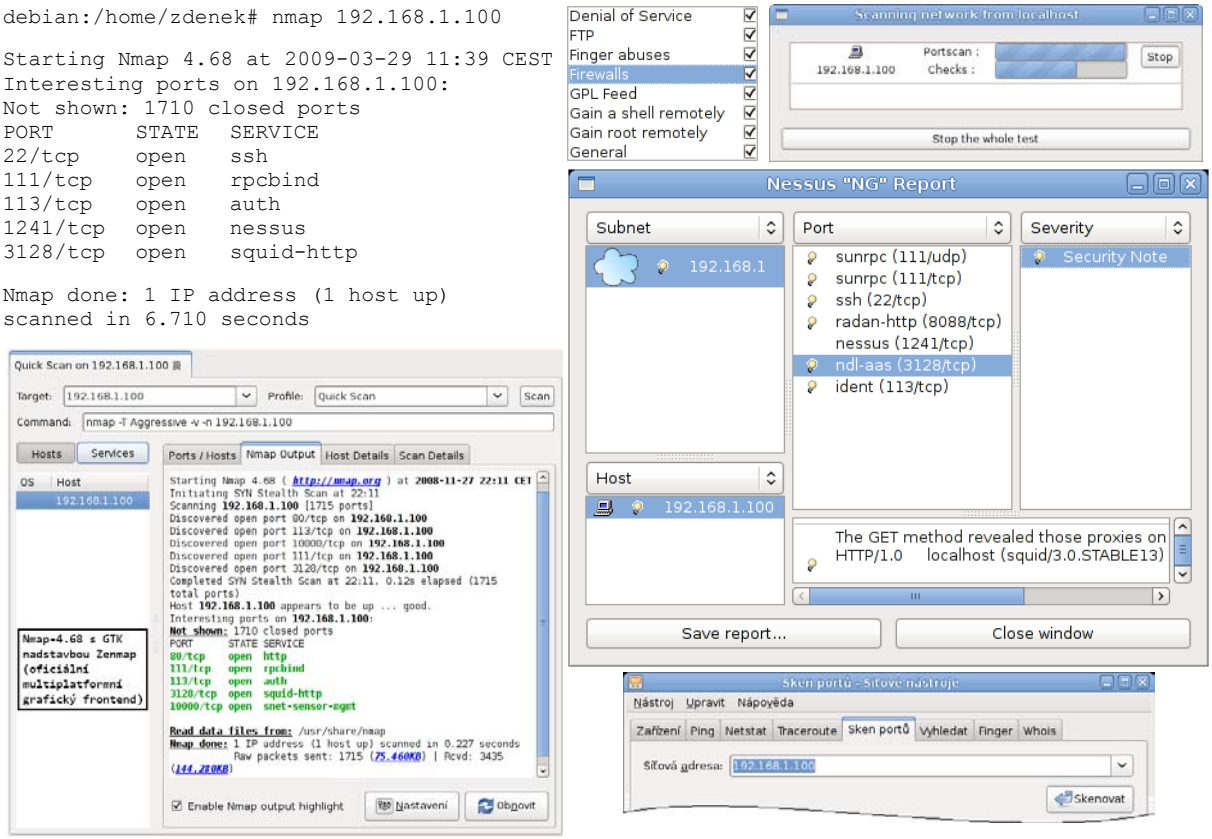
Bezpochyby zajímavým (z hlediska oblasti mé práce) je Open Source projekt IPCop, integrující linuxový firewall včetně IDS, monitoringu a správy přes webové rozhraní. Jedná se o minidistribuci (velikost v desítkách MB) určenou zejména pro firewally a směrovače, která v kostce nabízí router, firewall, IDS (Snort) a některé další užitečné programy a funkce. Mimo jiné nabízí služby jako SSH, VPN, NTP aj. Díky své komplexnosti a spoustě variant a možností nastavení lze IPCop použít jak pro ochranu malých sítí (domácnosti, malé firmy), tak i pro ochranu rozsáhlých a správou složitých sítí.

5.8.2 Bezpečnostní audit firewallu a IDS

5.8.2.1 Otestování aplikovaných pravidel firewallu Iptables

Nmap **Nessus**

```
debian:/home/zdenek# nmap 192.168.1.100

Starting Nmap 4.68 at 2009-03-29 11:39 CEST
Interesting ports on 192.168.1.100:
Not shown: 1710 closed ports
PORT       STATE   SERVICE
22/tcp     open    ssh
111/tcp    open    rpcbind
113/tcp    open    auth
1241/tcp   open    nessus
3128/tcp   open    squid-http

Nmap done: 1 IP address (1 host up)
scanned in 6.710 seconds
```



Obr. 59: Otestování firewallu Iptables programy Nmap a Nessus vůči otevřeným portům

Některé z těchto otevřených portů byly využívány službami, které na proxy (firewallu) nemají v žádném případě místo, ovšem z důvodů využívání pouze jediného fyzického serveru a snahou poskytnout dosažené výsledky a souhrnné přehledy např. i ve formě databáze ukládající se na webový server (příklad IDS Snortu), nebo v případě aplikované kombinace ClamAV-AMaViS-MAIL server, ve které antivirová ochrana pro poštovní server kontroluje přílohy e-mailových zpráv tzv. "on-fly" bylo zvoleno toto dostupné (v praxi typicky neakceptovatelné) řešení. Silným argumentem je možné zneužití těchto poskytovaných služeb na serveru, získání vyšších práv superuživatele (roota) a poté "volná" cesta dále sítí (únos, mystifikace, zneužití firemních dat, aj.) či zneužití serveru pro distribuovaný útok např. proti konkurenční firmě nebo páchání trestně právní činnosti [2].

5.8.2.2 Otestování systému k detekci průniků Snort

Činnost IDS Snort byla ověřena a důkladně otestována těmito specializovanými nástroji a skripty – Sneeze, Stick, Snot a Metasploit (nepočítaje základní nástroje Nmap, popř. Nessus, které již byly využity v souvislosti s firewallem a jeho pravidly; viz penetrační testy). Jednotlivé nástroje zde nebudu detailně rozepisovat, některé slouží jako falešný pozitivní generátor Snortu, jiné pracují obecně na principu generování různých útoků typu DoS, DDoS, buffer overflow, Mitnick attack, IP spoofing, atd., nebo se jedná o důmyslnější techniky založené např. na vyhledání a zneužití nebezpečných míst softwaru exploity, apod.

Důležité je tedy s pomocí těchto vyjmenovaných programů otestovat detekční schopnosti Snortu, zjistit chyby a nefunkční části, úpravou konfigurace sjednat nápravu, opětovně otestovat a doladit případné nalezené nedostatky. Výstupem je pak spolehlivě pracující systém Snortu připravený k reálnému nasazení v konkrétních podmínkách, odladěný podle současných pravidel a připravený na nové hrozby ze strany útočníků. Žádný systém ovšem nepracuje se stoprocentní přesností, tak i zde vzniká určité riziko planých poplachů, které se snažíme minimalizovat [11].

Některé Snortem zaznamenané útoky, provedené prostřednictvím programu Nessus (bezpečnostní skener):

```
[**] [122:1:0] (portscan) TCP Portscan [**]
[Priority: 3]
03/29-10:40:43.775690 192.168.2.100 -> 192.168.2.7
PROTO:255 TTL:0 TOS:0x4 ID:0 IpLen:20 DgmLen:161 DF

[**] [1:1504:7] MISC AFS access [**]
[Classification: Misc activity] [Priority: 3]
03/29-10:40:45.313382 192.168.2.100:46460 -> 192.168.2.7:7001
UDP TTL:64 TOS:0x0 ID:23849 IpLen:20 DgmLen:60 DF
Len: 32

[**] [1:1413:13] SNMP private access udp [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/29-10:40:45.376860 192.168.2.100:48782 -> 192.168.2.7:161
UDP TTL:64 TOS:0x0 ID:23865 IpLen:20 DgmLen:74 DF
Len: 46

[**] [1:2049:6] SQL ping attempt [**]
[Classification: Misc activity] [Priority: 3]
03/29-10:40:58.282399 192.168.2.100:60344 -> 192.168.2.7:1434
UDP TTL:64 TOS:0x0 ID:27091 IpLen:20 DgmLen:29 DF
Len: 1

[**] [1:1616:10] DNS named version attempt [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/29-10:41:04.350354 192.168.2.100:42333 -> 192.168.2.7:53
UDP TTL:64 TOS:0x0 ID:28608 IpLen:20 DgmLen:58 DF
Len: 30
```

```
[**] [1:524:8] BAD-TRAFFIC tcp port 0 traffic [**]
[Classification: Misc activity] [Priority: 3]
03/29-10:41:15.441769 192.168.2.100:54728 -> 192.168.2.7:0
TCP TTL:64 TOS:0x0 ID:16745 IpLen:20 DgmLen:40
******S* Seq: 0x2D8BB349  Ack: 0x0  Win: 0x200  TcpLen: 20

[**] [1:634:4] SCAN Amanda client-version request [**]
[Classification: Attempted Information Leak] [Priority: 2]
03/29-10:41:15.468359 192.168.2.100:53596 -> 192.168.2.7:10080
UDP TTL:64 TOS:0x0 ID:31388 IpLen:20 DgmLen:97 DF
Len: 69

[**] [1:1893:6] SNMP missing community string attempt [**]
[Classification: Misc Attack] [Priority: 2]
03/29-10:41:15.549249 192.168.2.100:53154 -> 192.168.2.7:161
UDP TTL:64 TOS:0x0 ID:31408 IpLen:20 DgmLen:70 DF
Len: 42

[**] [1:236:9] DDOS Stacheldraht client check gag [**]
[Classification: Attempted Denial of Service] [Priority: 2]
03/29-10:41:18.424306 192.168.2.100 -> 192.168.2.7
ICMP TTL:64 TOS:0x0 ID:4660 IpLen:20 DgmLen:39
Type:0  Code:0  ID:668  Seq:1  ECHO REPLY

[**] [116:150:1] (snort decoder) Bad Traffic Loopback IP [**]
[Priority: 3]
03/29-10:41:49.318438 127.0.0.1 -> 192.168.2.7
ICMP TTL:64 TOS:0x10 ID:0 IpLen:20 DgmLen:28 DF
Type:0  Code:0  ID:0  Seq:0  ECHO REPLY
```

Sneeze – příklad automatizovaného generování útoků typu DoS na cílovou IP 192.168.1.100 (proxy)

```
debian:/usr/local/etc/sneeze# ./sneeze.pl -d 192.168.1.100 -f /etc/snort/rules/dos.rules

ATTACK:
 debian.proxy.cz:20793 -> 192.168.1.100:44327

ATTACK: DOS IGMP dos attack
ATTACK TYPE: attempted-dos
ip debian.proxy.cz:40368 -> 192.168.1.100:91
Reference => http://www.microsoft.com/technet/security/bulletin/MS99-034.mspx
Reference => http://cve.mitre.org/cgi-bin/cvename.cgi?name=1999-0918
Reference => http://www.securityfocus.com/bid/514

ATTACK: DOS Cisco attempt
ATTACK TYPE: web-application-attack
tcp debian.proxy.cz:58630 -> 192.168.1.100:80

ATTACK: DOS squid WCCP I_SEE_YOU message overflow attempt
ATTACK TYPE: attempted-user
udp debian.proxy.cz:46271 -> 192.168.1.100:2048
Reference => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2005-0095
Reference => http://www.securityfocus.com/bid/12275
.....
```

```
debian:/etc/framework-3.2# ./msfconsole
                  _                  _
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ _ __
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \ '__|
| | | | | |  __/ || (_| \__ \ |_) | |  __/ |
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___|_|
                            |_|

       =[ msf v3.2-release
+ -- --=[ 320 exploits - 217 payloads
+ -- --=[ 20 encoders - 6 nops
       =[ 99 aux
msf > session -i 1
[*] Starting interaction with 1...
Meterpreter > use stdapi
Loading extension stdapi...success.
Meterpreter >
```

5.8.3 Další využitelné systémy (navržené řešení pro budoucí rozvoj)

Úspěšně nasazené a na proxy serveru provozované technologie prozatím jsou – stavový firewall, antivirový program, filter regulující internetový obsah a systém k detekci průniků. Z dalších technik a programů, které by bylo vhodné na serveru nasadit a uplatňovat z bezpečnostních či jiných důvodů (kvalita služeb - QoS, VPN, autentizace, virtualizace, pokročilý monitoring, zdokonalení logovací činnosti apod.) jsou důležité zejména: **Traffic management** (např. Policy-based Traffic Shaping, Priority-bandwidth Utilization), některé další prvky pro **správu** a **řízení provozu**, **AMaViS** (A Mail Virus Scanner), který kontroluje přílohy e-mailových zpráv a **SpamAssassin** jako mailový filter proti spamu.

6 Závěr

V diplomové práci jsem podrobně popsal princip činnosti proxy firewallů, stavových firewallů a paketových filtrů. Následně je mezi sebou vzájemně porovnal, u každého řešení zmínil hlavní přednosti a oblasti použití, spolu s typickými představiteli každé kategorie. O každém jmenovaném představiteli bych se nyní velmi krátce zmínil a shrnul ty nejpodstatnější rysy.

Firewally jsou jakési „kontrolní stanoviště“ mezi privátní sítí na jedné straně a jednou nebo více veřejnými sítěmi na straně druhé. Firewall je tedy bránou, která selektivně rozhoduje, co smí a nesmí do privátní sítě vstoupit nebo ji naopak opustit. V rámci své činnosti kontroluje každý datový paket a ověřuje, jestli vyhovuje pravidlům bezpečnostní politiky, pro jejichž dodržování je naprogramován. Firewall tak tvoří určitou „hráz“, záměrně postavenou do cesty síťového provozu, a trvale monitoruje dodržování bezpečnostních pravidel ze strany síťových spojení mezi vnitřním a vnějším světem. Pojmem firewally označujeme poměrně širokou skupinu systémů lišících se tím, zda jsou primárně zaměřeny na filtrování nebo monitoring, na jaké úrovni síťový provoz sledují, nebo třeba zdali podporují autentizaci uživatelů, apod.

Paketové filtry jsou nejjednodušší a nejzákladnější forma firewallu. Rozhodují se pouze na základě hlaviček paketů (zdrojová/cílová adresa/port). Pracují na síťové vrstvě referenčního modelu ISO/OSI. Stavové firewally jsou rozšířením paketových filtrů. Udržují navíc informace o (stavu) spojeních, takže umožňují hlídat souvislosti mezi jednotlivými pakety. Kvůli paketovému filtru ani stavovému firewallu není nutné měnit stávající aplikace. Aplikační proxy je ochrana na vrstvě aplikační, kde je daleko lépe porozuměno obsahu paketů. Inspekci jsou tedy podrobována samotná data. Jedná se většinou o nějaký program, se kterým komunikují klienti a posílají požadavky na servery přes něj. Příkladem aplikační proxy je *HTTP proxy* a *FTP proxy*. Pro každou službu tedy musí být samostatná proxy, pro spolupráci je třeba navíc upravit klientské programy (existují i výjimky). Aplikační proxy umožňují také identifikaci uživatelů. Tyto proxy zpravidla nepotřebují podporu jádra systému. Proxy servery obecně slouží k poskytování speciálních služeb, např. bezpečnostní izolace sítě, meziukládání dat, atd.

Zmíněny byly také ostatní související technologie příbuzné výše uvedeným, například technika překladu síťových adres, které doplňují problematiku síťové bezpečnosti v celkový komplex, složený právě z dílčích vrstvených obranných prvků a pomyslných barikád, primárně nebo sekundárně sloužících k ochraně počítačových sítí, serverů, hostitelských systémů a jiných důležitých aktiv.

Jako další jsem navrhnul praktické řešení proxy firewallu postavené na operačním systému Linux. Jako vhodným kandidátem se jevila, a byla zvolena, osvědčená linuxová distribuce Debian, která svými charakteristickými vlastnostmi a dlouholetou tradicí tvoří solidní základ pro spolehlivou serverovou činnost a obecně profesionální užití. Dalšími programy a potřebnými nástroji byly po zvážení zvoleny: multiplatformní cachující proxy server Squid (v aktuální verzi 3.0), síťový provoz omezující linuxový firewall Iptables, antivirový program ClamAV, HTTP antivirová proxy HAVP, content filter (filter internetového obsahu) UfdbGuard a systém k detekci neautorizovaných průniků (IDS) Snort. Všechny jmenované programy jsou Open Source Software šířené pod licencí GNU GPL. Výstupem (řešením) praktické části je šest komentovaných souborů. Jedná se o nově vytvořený linuxový skript pro (automatickou) aplikaci pravidel firewallu Iptables s názvem *firewall-iptables.sh*, dále modifikované (vhodně upravené) konfigurační soubory Squidu (*squid.conf*), HAVP (*havp.config*), ClamAV (*clamd.conf*), UfdbGuard (*ufdbguard.conf*) a Snortu (*snort.conf*).

Každý je uplatňován samostatně a nezávisle při (re)startu systému. Prostřednictvím již podotknutého HAVP je zajištěna kontrola stahovaných souborů na přítomnost virů. V podstatě se jedná o redirect (rozhraní) mezi Squidem a ClamAV, tedy obecně mezi proxy a antivirem.

Zprovozněné řešení nabízí ochranu vnitřní sítě před útoky směřujícími z Internetu (např. DoS, DDoS, UDP & SYN floods, Port Scan attack, Ping of Death, Exploits a spoustu dalších; čili útoky typu odepření služby, zneužití bezpečnostních chyb, zjišťování informací, dezinformační útoky, atd.). Ochrana sítě spočívá ve včasném varování, filtrování paketů, hloubkové inspekci paketů, ochraně proti některým známým útokům, filtrováním obsahu, detekci a prevenci virů, trojanů, rootkitů a nebezpečného spyware, skenováním e-mailů na branách, aplikací antispamových filtrů, apod.

Samozřejmostí by měl být aktualizovaný (záplatovaný) systém včetně nejnovějších (stabilních) verzí instalovaných programů – primárně kvůli bezpečnosti, důležité je ovšem i hledisko stability, odladěnosti, integrace nových funkcí a další. Nové verze s sebou často přináší kromě zmíněného i vyšší rychlost. Bezpodmínečně nutné jsou pravidelné aktualizace databází virových vzorků (ClamAV), signatur útoků (Snort), aktuálních blacklistů (ufdbGuard), aj. Firewall by měl být taktéž dostatečně adaptabilní, schopný přizpůsobovat se aktuálním hrozbám Internetu a čelit novým důmyslnějším nástrahám ze strany útočníků. Nelze opomenout ani riziko napadení síť. prostředí zneužitím špatně nastavených přístupových práv, např. příliš benevolentní režim (právo ke čtení, zápisu nebo spuštění souboru má jak vlastník, tak i skupina či dokonce všichni ostatní); práva roota (superuživatele) používat pouze v nezbytně nutných případech (př. změna současné konfigurace, instalace/odstranění systémových balíčků, apod.).

Zejména kvůli doplňujícím názorným obrázkům, ze kterých je zřejmá podstata věci, a doprovodným přílohám se stala práce poměrně rozsáhlou, na druhou stranu však splnila veškeré cíle a požadavky kladené v zadání. Nad rámec práce je uvedeno několik aplikací a nástrojů, především ve spojení s programem Snort, které rozšiřují a prohlubují jeho oblast působnosti (snaha o přiblížení se podmínkám reálné praxe, kde jsou nasazovány převážně jednoduché, jednoúčelové programy, vzájemně spolupracující a tvořící celistvý komplex). Tedy alespoň co se týče operačního systému Linux, kde je tato filosofie zastávána. V poslední řadě je zmíněno několik dalších rozšíření (modulů) nebo bezpečnost doplňujících balíčků a přídavných prvků pro správu a řízení.

Vytvořená práce je poměrně komplexní, stále je zde ovšem prostor pro další vylepšení zabezpečení. Následujícími kroky, kterými bych se v práci dále ubíral, by zřejmě byli (po vzoru moderních firewallových řešení): rozdílné chování k paketům a pokročilé řízení provozu (TOS, QoS, Bandwidth management), podpora virtuálních privátních sítí VPN a virtuálních lokálních sítí VLAN (služby tunelování a šifrování), automatické blokování aplikací typu IM/P2P, zákaz sociálních sítí (např. MySpace, Facebook, YouTube) apod.

Při psaní dokumentace a textových popisů k programům, jako například Iptables, bylo ve velké míře využito materiálů dostupných v manuálových stránkách (popř. uživatelských příručkách) k těmto programům.

Diplomová práce rozšířila možnosti proxy firewallu o součást antivirové ochrany, content filteringu a systému detekce průniků (IDS). Všechny uvedené systémy výrazným způsobem zvyšují úroveň zabezpečení jak samotného proxy serveru v síti, tak i v důsledku klientských koncových stanic. Na bezpečnost lze nahlížet jako na stav (neustále se vyvíjí), nikoliv na vlastnost, která je samozřejmá.

7 Seznam literatury

[1] NEMETH, Evi, SNYDER, Garth, HEIN, Trent R. *Linux : Kompletní příručka administrátora*. Brno : Computer Press, 2004. 880 s. ISBN 80-722-6919-4.

[2] TOXEN, Bob. *Bezpečnost v Linuxu : Prevence a odvrácení napadení systému*. Brno : Computer Press, 2003. 876 s. ISBN 80-7226-716-7.

[3] NORTHCUTT, Stephen, et al. *Bezpečnost počítačových sítí : Kompletní průvodce návrhem, implementací a údržbou zabezpečené sítě*. Brno : CP Books, 2005. 592 s. ISBN 80-251-0697-7.

[4] THOMAS, Thomas M. *Zabezpečení počítačových sítí bez předchozích znalostí*. Brno : CP Books, 2005. 344 s. ISBN 80-251-0417-6.

[5] STREBE, Matthew, PERKINS, Charles. *Firewally a proxy-servery : Praktický průvodce*. Brno : Computer Press, 2003. 472 s. ISBN 80-7226-983-6.

[6] VELTE, Toby J., VELTE, Anthony T. *Síťové technologie Cisco : Velký průvodce*. Brno : Computer Press, 2003. 800 s. ISBN 80-7226-857-0.

[7] LOCKHART, Andrew. *Bezpečnost sítí na maximum : 100 tipů a opatření pro okamžité zvýšení bezpečnosti vašeho serveru a sítě*. Brno : CP Books, 2005. 280 s. ISBN 80-251-0805-8.

[8] KRČMÁŘ, Petr. *Linux : tipy a triky pro bezpečnost*. Praha : Grada Publishing, 2004. 208 s. ISBN 80-247-0812-4.

[9] HORÁK, Jaroslav. *Bezpečnost malých počítačových sítí : praktické rady a návody*. Praha : Grada Publishing, 2003. 200 s. ISBN 80-247-0663-6.

[10] KOSTRHOUN, Aleš. *Stavíme si malou síť*. Praha : Computer Press, 2001. 216 s. ISBN 80-7226-510-5.

[11] BIGELOW, Stephen J. *Mistrovství v počítačových sítích : Správa, konfigurace, diagnostika a řešení problémů*. Brno : Computer Press, 2004. 992 s. ISBN 80-251-0178-9.

[12] HERBORTH, Chris. *Unix a Linux : Názorný průvodce*. Brno : Computer Press, 2006. 288 s. ISBN 80-251-0978-X.

[13] HONTANóN, Ramón J. *Linux praktická bezpečnost*. Praha : Grada Publishing, 2003. 440 s. ISBN 80-247-0652-0.

[14] DOSTÁLEK, Libor, KABELOVÁ, Alena. *Velký průvodce protokoly TCP/IP a systémem DNS*. Praha : Computer Press, 2000. 426 s. ISBN 80-7226-323-4.

[15] SCAMBRAY, Joel, MCCIURE, Stuart, KURTZ, George. *Hacking bez tajemství*. Praha : Computer Press, 2002. 646 s. ISBN 80-7226-644-6.

[16] DOSTÁLEK, Libor, et al. *Velký průvodce protokoly TCP/IP : Bezpečnost*. Brno : Computer Press, 2003. 592 s. ISBN 80-7226-849-X.

[17] SMITH, Roderick W. *Linux ve světě Windows : průvodce administrátora heterogenních sítí*. Praha : Grada Publishing, 2006. 460 s. ISBN 80-247-1470-1.

[18] JEŘÁBEK, Jan. *Pokročilé komunikační techniky*. Brno : [s.n.], 2007. 165 s.

[19] NORIS, Ivan. *Příručka systémového administrátora* [online]. 2002-2007 [cit. 2008-11-09]. Dostupný z WWW: <http://deja-vix.sk/sysadmin/>.

[20] *Abclinuxu.cz : Linux na stříbrném podnose* [online]. 1999-2008 [cit. 2008-11-22]. Dostupný z WWW: <http://www.abclinuxu.cz/>. ISSN 1214-1267.

[21] PŘIBYL, Adam. *Linux.cz : České stránky systému GNU/Linux* [online]. 2007 [cit. 2008-11-12]. Dostupný z WWW: <http://www.linux.cz/>.

[22] KURE, Miroslav, KUBELKA, Juraj. *Debian : Univerzální operační systém* [online]. 1997-2008 [cit. 2008-11-01]. Dostupný z WWW: <http://www.debian.org/>.

[23] *Linux.com* [online]. 1999-2008 [cit. 2008-10-26]. Dostupný z WWW: <http://www.linux.com/>.
[24] PARKANSKY, Keith. *Debian Linux Tutorial : Beginners Guide To Linux Servers and Networking Installation and Set Up with Instructions On How To Configure A Home Server* [online]. 2003-2008 [cit. 2008-11-05]. Dostupný z WWW: <http://www.aboutdebian.com/>.
[25] *Cisco Systems, Inc.* [online]. 1992-2008 [cit. 2008-12-10]. Dostupný z WWW: <http://www.cisco.com/cz/index.shtml>.
[26] *Squid : Optimising Web Delivery* [online]. 2008 [cit. 2008-10-24]. Dostupný z WWW: <http://www.squid-cache.org/>.
[27] WELTE, Harald, AYUSO, Pablo Neira. *Netfilter/iptables project homepage : The netfilter.org project* [online]. 1999-2008 [cit. 2008-10-29]. Dostupný z WWW: <http://www.netfilter.org/>.
[28] KRČMÁŘ, Petr. *Root.cz : linux, open source a free software* [online]. 1998-2008 [cit. 2008-10-13]. Dostupný z WWW: <http://www.root.cz/>. ISSN 1212-8309.
[29] ZANDL, Patrick, et al. *LUPA : Server o českém Internetu* [online]. 1998-2009 [cit. 2009-02-16]. Dostupný z WWW: <http://www.lupa.cz/>. ISSN 1213-0702.
[30] KLAŠKA, Luboš, et al. *Svět sítí : o počítačových sítích nejen pro administrátory a specialisty* [online]. 2000-2008 [cit. 2008-11-19]. Dostupný z WWW: <http://www.svetsiti.cz/>.
[31] Impossible. *Linuxzone.cz : server o Linuxu pro programátory, administrátory a fanoušky* [online]. 2002 [cit. 2008-11-28]. Dostupný z WWW: <http://www.linuxzone.cz/>. ISSN 1213-8738.
[32] *D-Link : Building Networks for People* [online]. 2008 [cit. 2008-12-09]. Dostupný z WWW: <http://www.dlink.com/>.
[33] *Živě.cz : O počítačích, IT a internetu* [online]. 2008 [cit. 2008-12-03]. Dostupný z WWW: <http://www.zive.cz/>. ISSN 1212-8554.
[34] *Počítačové sítě : Intranet, proxy a gateway* [online]. 2007 [cit. 2008-11-07]. Dostupný z WWW: <http://pc-site.owebu.cz/>.
[35] MATOUŠEK, Petr. DOBRÉHO NESPÁLÍ : Test firewallů pro sítě do 150 uživatelů. *Connect!*. 1.1.2006, č. 06, s. 26-29.
[36] JELÍNEK, Lukáš, et al. *Linux E X P R E S* [online]. 2008 [cit. 2009-03-11]. Dostupný z WWW: <http://www.linuxexpres.cz/>. ISSN 1801-3996.
[37] *The Linux Home Page at Linux Online* [online]. 1994-2008 [cit. 2008-11-23]. Dostupný z WWW: <http://www.linux.org/>.
[38] BURDA, Karel. *Návrh, správa a bezpečnost počítačových sítí*. Brno : [s.n.], 2008.
[39] FIRMAN, Andy. 11 step guide to build a Debian based Intrusion Detection Sensor (IDS) with Snort 2.4.5 or Snort 2.6. *SNORT.ORG - Docs : Setup Guides* [online]. 2006 [cit. 2009-04-10]. Dostupný z WWW: <http://www.snort.org/docs/>.
[40] Sourcefire. *Snort : the de facto standard for intrusion detection/prevention* [online]. 2009 [cit. 2009-04-04]. Dostupný z WWW: <http://www.snort.org/>.
[41] HLOBIL, Petr. *Konfigurace HAVP se SQUID a CLAMAV* [online]. 2007 [cit. 2009-04-16]. Dostupný z WWW: <http://www.hlobil.net/>.
[42] HÁK, Igor. *VIRY.CZ : Antivirové systémy* [online]. 1998-2007 [cit. 2009-04-25]. Dostupný z WWW: <http://www.viry.cz/>. ISSN 1213-4694.
[43] RAMPULA, Michal. *Filtrování obsahu aneb Content Filtering* [online]. 2005 [cit. 2009-04-28]. Dostupný z WWW: <http://www.100mega.cz/>.
[44] *URLfilterDB : Internet filter, URL Filter, Content Filtering, Web blocking with Squid* [online]. 2004-2008 [cit. 2009-04-06]. Dostupný z WWW: <http://www.urlfilterdb.com/>.

Seznam použitých zkratek a symbolů

ACK *(Acknowledgement)*; potvrzovací paket (slouží k potvrzení přijatých dat)
ACL *(Access Control List)*; seznam pro řízení přístupu (přístupových práv)
API *(Application Program Interface)*; rozhraní pro programování aplikací (OpenGL a DirectX)
APT *(Advanced Packaging Tool)*; balíčkovací systém užívaný v Debian GNU/Linuxu a jeho derivátech
BSD *(Berkeley Software Distribution)*; odvozenina Unixu distribuovaná Kalifornskou univerzitou v Berkeley mající počátky v 70. letech 20. století
CGI *(Common Gateway Interface)*; protokol pro propojení externích aplikací s webovým serverem
CLI *(Command Line Interface)*; způsob uživatelského rozhraní prostřednictvím tzv. příkazového řádku
DDoS *(Distributed Denial of Service)*; distribuované odmítnutí síťových služeb (distribuované DoS)
DEB *(souborový formát)*; archiv balíčkovacího systému distribuce Debian
DHCP *(Dynamic Host Configuration Protocol)*; automat. přidělování IP adres koncovým stanicím v síti
DMZ *(Demilitarized Zone)*; demilitarizovaná zóna
DNS *(Domain Name System)*; hierarchický systém doménových jmen
DoS *(Denial of Service)*; odmítnutí síťových služeb
FTP *(File Transfer Protocol)*; protokol aplikační vrstvy z rodiny TCP/IP určený pro přenos souborů
GNU *(GNU's Not Unix)*; GNU není Unix, projekt zaměřený na svobodný software
GNU GPL *(GNU General Public License)*; všeobecná veřejná licence GNU
GNU/Linux; svobodný operační systém tvořený velkým množstvím svobodných programů
GTK *(GIMP Toolkit)*; jedna ze dvou nejpopulárnějších open source knihoven pro vytváření grafických uživatelských rozhraní
GUI *(Graphical User Interface)*; uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků
HTML *(HyperText Markup Language)*; značkovací jazyk pro hypertext
HTTP *(HyperText Transfer Protocol)*; internetový protokol určený pro výměnu hypertextových dokumentů ve formátu HTML
HTTPS *(HyperText Transfer Protocol Secure)*; šifrovaná varianta internetového protokolu HTTP pro přenos webových stránek
HW *(Hardware)*; technické vybavení počítače
ICMP *(Internet Control Message Protocol)*; protokol sloužící pro přenos chybových a řídících zpráv mezi uzly a směrovači sítě TCP/IP
ID *(Identification)*; identifikace
IDS *(Intrusion Detection System)*; systém detekce průniku
IETF *(Internet Engineering Task Force)*; komise, která vyvíjí a podporuje internetové standardy a úzce spolupracuje s konsorciem W3C a s orgány ISO/IEC
IMAP *(Internet Message Access Protocol)*; protokol pro přístup k e-mailovým schránkám (IMAP4)
IP *(Internet Protocol)*; datový protokol používaný pro přenos dat přes paketové sítě
IPSec *(IP security)*; bezpečnostní rozšíření IP protokolu
IPv4 *(Internet Protocol version 4)*; jednoznačná identifikace konkrétního zařízení (typicky počítače) v prostředí Internetu, adresou je 32bit. číslo (zapisované po jednotlivých bajtech, oddělených tečkami)
IPv6 *(Internet Protocol version 6)*; síťová vrstva pro mezisíťový přenos paketů, následník IPv4**,** používá 128bitové IP adresy (v budoucnu nahradí stávající IPv4)
ISO/OSI *(International Standards Organization / Open Systems Interconnection)*; sedmivrstvý referenční komunikační model

ISP *(Internet Service Provider)*; poskytovatel internetového připojení
Klient-Server architektura; síťová architektura, která odděluje klienta a server, kteří spolu komunikují přes počítačovou síť
LAN *(Local Area Network)*; místní/lokální počítačová síť
MAC *(Media Access Control)*; jedinečný identifikátor síťového zařízení, fyzická adresa
NASL *(Nessus Attack Scripting Language)*; interní skriptovací jazyk (vyvinutý v rámci projektu Nessus)
NAT *(Network Address Translation)*; překlad síťových adres
NIC *(Network Interface Controller)*; síťový adaptér (hardware), instalovaný v pracovní stanici nebo serveru, umožňující komunikaci po počítačové síti
OS *(Operating System)*; operační systém
OSI *(Open Source Initiative)*; podpůrná organizace pro open source software; předepisuje soubor pravidel, podle kterých lze konkrétní licenci považovat za "Open Source"
OSS *(Open-Source Software)*; počítačový software s otevřeným zdrojovým kódem
PAT *(Port Address Translation)*; překlad adres portů, podmnožina NAT
PC *(Personal Computer)*; osobní počítač
POP *(Post Office Protocol)*; internetový protokol, který se používá pro stahování e-mailových zpráv ze vzdáleného serveru na klienta (POP3)
QoS *(Quality of Services)*; řízení datových toků v síti
RFC *(Request for Comments)*; označení dokumentů, které popisují standardy, doporučení a informace o internetových protokolech a službách
RPM *(RPM Package Manager, původně Red Hat Package Manager)*; balíčkovací systém pro Linux
SMTP *(Simple Mail Transfer Protocol)*; internetový protokol určený pro přenos zpráv elektronické pošty mezi přepravci elektronické pošty (MTA)
SNMP *(Simple Network Management Protocol)*; protokol pro správu síťových prvků
SSH *(Secure Shell)*; klient/server protokol v síti TCP/IP umožňující bezpečnou komunikaci mezi dvěma počítači pomocí šifrování přenášených dat
SSL *(Secure Sockets Layer)*; protokol, resp. vrstva vložená mezi vrstvu transportní a aplikační, která poskytuje zabezpečení komunikace šifrováním a autentizaci komunikujících stran
SW *(Software)*; programové vybavení počítače
SYN *(Synchronization)*; synchronizační paket (slouží k navázání spojení)
TCP *(Transmission Control Protocol)*; nejpoužívanější přenosový protokol v IP sítích, poskytující spolehlivé, plně duplexní přenosy s navazováním spojení
TCP/IP *(Transmission Control Protocol/Internet Protocol)*; primární transportní protokol - TCP/protokol síťové vrstvy - IP
TOS *(Type of Service)*; specifikace požadavků na přenos datagramu
TTL *(Time To Live)*; doba života IP paketu
UDP *(User Datagram Protocol)*; protokol přenosové vrstvy bez navazování spojení ze sady protokolů TCP/IP používající protokol IP
URL *(Uniform Resource Locator)*; jednoznačný lokátor zdrojů, řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací na Internetu
VPN *(Virtual Private Network)*; virtuální privátní síť
WAN *(Wide Area Network)*; rozsáhlá počítačová síť (např. Internet)
WWW *(World Wide Web)*; celosvětová síť („pavučina“), distribuovaný soubor dokumentů v Internetu, který lze hypertextově prohledávat
XML *(Extensible Markup Language)*; rozšiřitelný značkovací jazyk (vyvinut konsorciem W3C)

8 Seznam příloh

8.1 Praktická část – přiložené skripty & konfigurační soubory

8.1.1 /etc/init.d/firewall-iptables.sh (součást přílohy DP + CD-ROM)

8.1.2 /etc/squid3/squid.conf (součást přílohy DP + CD-ROM)

8.1.3 /etc/havp/havp.config (pouze CD-ROM)

8.1.4 /etc/clamav/clamd.conf (pouze CD-ROM)

8.1.5 /etc/snort/snort.conf (pouze CD-ROM)

8.1.6 /usr/local/squid/etc/ufdbguard.conf (pouze CD-ROM)

Uvedený skript firewallu Iptables **firewall-iptables.sh** a konfigurační soubor Squidu3 **squid.conf** byli záměrně zvoleni jako přímá nedílná součást diplomové práce, zejména kvůli příkladné (demonstrační) ukázce, která zřetelně vystihuje postup a zpracování všech dalších souborů, se kterými bylo v práci manipulováno. Ostatní konfigurační soubory jsou kvůli jejich délce, resp. nadměrnému počtu řádků, umístěny na médiu CD-ROM. Konkrétně se jedná o soubory **havp.config** (konfigurační soubor antivirové proxy *HAVP*), **clamd.conf** (konfigurační soubor antivirového programu *ClamAV*), **snort.conf** (konfigurační soubor IDS *Snortu*) a **ufdbguard.conf** (konfigurační soubor filteru regulujícího internetový obsah *ufdbGuard*). U každého souboru jsem se snažil slovně popsat klíčové pasáže, důležité pro správnou konfiguraci daného programu v souvislosti se zvoleným řešením, a to komentáři uvedenými za těmito příkazy nebo direktivy. Zahrnuty jsou vždy pouze oblasti lišící se od původního (originálního) konfiguračního souboru, tzn. modifikované úseky. Je toho dosaženo za pomocí příkazu Diff, který hledá rozdílné a shodné části ve dvou souborech (_original, _modified).

8.2 Soubory se zpracovanými materiály, pro které již v práci nezbyl prostor, případně materiály přesahující rámec DP (CD-ROM; formát pdf)

8.3 CD-ROM s elektronickou verzí diplomové práce a přiloženými soubory

8.4 Popisný soubor závěrečné práce (metadata)

9 Přílohy

9.1 /etc/init.d/firewall-iptables.sh

```
#!/bin/sh
#
# firewall-iptables.sh - PRAVIDLA PRO FIREWALL
# ----------------------------------------------
# Poznámky:
#
# Tohle je funkční konfigurace firewallu nad iptables.
#
# Předpokládané nasazení na firewallovém serveru se dvěma síťovými kartami
# eth0 a eth1. Firewall zajišťuje konektivitu vnitřní sítě pomocí SNATu.
#
# Na firewallu kromě toho neběží žádné doplňkové či jiné služby.
#
# Požadavky na HTTP jsou transparentně směrovány na proxy-cache Squid.
# ******************************************************************************

# Naše IP adresa a vnější rozhraní
INET_IP="192.168.1.100"    # IP adresa vnějšího rozhraní není veřejná =>
INET_IFACE="eth0"          # předřazení hraničního směrovače/brány do sítě WAN

# IP a broadcast adresa a rozhraní vnitřní sítě
LAN1_IP="192.168.2.100/24"      # IP adresa vnitřní sítě je 192.168.2.0,
LAN1_BCAST="192.168.2.255/24"   # maska sítě 255.255.255.0, klientské stanice
LAN1_IFACE="eth1"               # dostanou přiřazenu IP adresu od DHCP serveru (.1 ~ .50)

# Lokální loopback rozhraní
LO_IP="127.0.0.1/8"    # /localhost/
LO_IFACE="lo"

# Cesta k programu iptables
IPTABLES="/sbin/iptables"

# Inicializace databáze modulů
/sbin/depmod -a

# Zavedeme moduly pro nestandardní cíle
/sbin/modprobe ipt_LOG
/sbin/modprobe ipt_REJECT
/sbin/modprobe ipt_MASQUERADE

# Modul pro FTP přenosy
/sbin/modprobe ip_conntrack_ftp
/sbin/modprobe ip_nat_ftp

# Zapneme směrování paketů
echo "1" > /proc/sys/net/ipv4/ip_forward

# Zapneme ochranu proti synflood útoku (součást kernelu)
echo "1" > /proc/sys/net/ipv4/tcp_syncookies

# rp_filter na zamezení "IP spoofingu"
for interface in /proc/sys/net/ipv4/conf/*/rp_filter; do
   echo "1" > ${interface}
done

# Implicitní politikou je zahazovat nepovolené pakety
$IPTABLES -P INPUT DROP    # vstup (input)
$IPTABLES -P OUTPUT DROP    # výstup (output)
$IPTABLES -P FORWARD DROP    # přesměrování (forwarding)

#                                                          #
# ---------- Řetězec PREROUTING v NAT tabulce ---------- #
#                                                          #
```

```
# Odchozí HTTP požadavky na port 80 budou přesměrovány na Squida ve funkci
# transparentní proxy cache (IP adresa: 192.168.1.100, port: 3128).

# $IPTABLES  -t nat  -A PREROUTING  -p tcp  -i ! $INET_IFACE  -d ! $ INET_IP  --dport 80  -j
REDIRECT --to-port 3128
# REDIRECT přesměruje provoz na jiný port, ale IP adresa zůstává stejná.
# Oproti tomu DNAT slouží k přesměrování i na jinou IP - náš případ.

$IPTABLES -t nat -A PREROUTING -p tcp -i ! $INET_IFACE -d ! $INET_IP --dport 80 -j DNAT -
-to-destination 192.168.1.100:3128

#                                                          #
# ---------- Řetězec POSTROUTING v NAT tabulce ---------- #
#                                                          #

# IP maškaráda - SNAT
# NATujeme
$IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j SNAT --to $INET_IP

#
# Přídavné řetězce pro snazší kontrolu na rezervované adresy
#

# Zahazovat a logovat (max. 5 x 3 pakety za hodinu)
$IPTABLES -N logdrop
$IPTABLES  -A logdrop   -m limit   --limit  5/h  --limit-burst  3  -j LOG   --log-prefix
"Rezervovana adresa: "
$IPTABLES -A logdrop -j DROP

# V tomto řetězci se kontroluje, zda příchozí pakety nemají nesmyslnou IP adresu
$IPTABLES -N IN_FW
$IPTABLES -A IN_FW -s 192.168.0.0/16 -j logdrop # rezervováno podle RFC1918
$IPTABLES -A IN_FW -s 10.0.0.0/8 -j logdrop     # rezervováno podle RFC1918
$IPTABLES -A IN_FW -s 172.16.0.0/12 -j logdrop  # rezervováno podle RFC1918
$IPTABLES -A IN_FW -s 96.0.0.0/4 -j logdrop     # rezervováno podle IANA

# TOS flagy slouží k optimalizaci datových cest. Pro ssh, ftp a telnet
# požadujeme minimální zpoždění. Pro ftp-data zase maximální propustnost.
$IPTABLES -t mangle -A PREROUTING -p tcp --sport ssh -j TOS --set-tos Minimize-Delay
$IPTABLES -t mangle -A PREROUTING -p tcp --dport ssh -j TOS --set-tos Minimize-Delay
$IPTABLES -t mangle -A PREROUTING -p tcp --sport ftp -j TOS --set-tos Minimize-Delay
$IPTABLES -t mangle -A PREROUTING -p tcp --dport telnet -j TOS --set-tos Minimize-Delay
$IPTABLES  -t mangle  -A PREROUTING  -p tcp  --sport  ftp-data  -j T OS  --set-tos Maximize-
Throughput

#                                      #
# ---------- Řetězec FORWARD ---------- #
#                                      #

# Navazování spojení
# Jestliže paket navazuje spojení, ale nemá nastavený příznak SYN - zahodit jej.
$IPTABLES -A FORWARD -p tcp ! --syn -m state --state NEW -j DROP

# Portscan s nastaveným SYN,FIN
$IPTABLES -A FORWARD -p tcp -i $INET_IFACE --tcp-flags SYN,FIN SYN,FIN -j LOG -m limit --
limit 10/m --log-prefix="Falesny paket: "
$IPTABLES -A FORWARD -p tcp -i $INET_IFACE --tcp-flags SYN,FIN SYN,FIN -j DROP

# Nechceme rezervované adresy na internetovém rozhraní
$IPTABLES -A FORWARD -i $INET_IFACE -j IN_FW

# Směrování zevnitř sítě ven neomezujeme
$IPTABLES -A FORWARD -i $LAN1_IFACE -j ACCEPT

# Směrování zvenku dovnitř sítě pouze pro navázaná spojení (stavový firewall)
$IPTABLES -A FORWARD -i $INET_IFACE -o $LAN1_IFACE -m state --state ESTABLISHED,RELATED -
j ACCEPT

# Ostatní pakety budou zahozeny, tak je budeme logovat (max. 12 x 5 paketů/hodinu)
$IPTABLES -A FORWARD -m limit --limit 12/h -j LOG --log-prefix "FORWARD drop: "
```

```
#                                     #
# ---------- Řetězec INPUT ---------- #
#                                     #

# Navazování spojení
# Jestliže paket navazuje spojení, ale nemá nastavený příznak SYN - zahodit jej.
$IPTABLES -A INPUT -p tcp ! --syn -m state --state NEW -j DROP

# Portscan s nastaveným SYN,FIN
$IPTABLES -A INPUT -p tcp -i $INET_IFACE --tcp-flags SYN,FIN SYN,FIN -j LOG -m limit --
limit 10/m --log-prefix="Falesny paket: "
$IPTABLES -A INPUT -p tcp -i $INET_IFACE --tcp-flags SYN,FIN SYN,FIN -j DROP

# Nejprve se zbavíme nežádoucích adres
$IPTABLES -A INPUT -i $INET_IFACE -j IN_FW

# Službu AUTH (AUTH server) není dobré filtrovat pomocí DROP,
# protože to může vést k prodlevám při navazování některých spojení.
# Proto jej sice zamítneme, ale tak, aby nedošlo k nežádoucím prodlevám (REJECT).
$IPTABLES -A INPUT -i $INET_IFACE -p TCP --dport 113 -m limit --limit 12/h -j LOG
$IPTABLES -A INPUT -i $INET_IFACE -p TCP --dport 113 -j REJECT --reject-with tcp-reset

# Propouštíme pouze ICMP ping
$IPTABLES -A INPUT -i $INET_IFACE -p ICMP --icmp-type echo-request -j ACCEPT

# Loopback není doporučeno omezovat
$IPTABLES -A INPUT -i $LO_IFACE -j ACCEPT

# Stejně jako pakety z lokální sítě, jsou-li určeny pro nás
$IPTABLES -A INPUT -i $LAN1_IFACE -d $LAN1_IP -j ACCEPT
$IPTABLES -A INPUT -i $LAN1_IFACE -d $INET_IP -j ACCEPT

# Broadcasty na lokálním rozhraní jsou také naše
$IPTABLES -A INPUT -i $LAN1_IFACE -d $LAN1_BCAST -j ACCEPT

# MS klienti mají chybu v implementaci DHCP
# (musíme povolit udp port 67, aby služba fungovala správně)
$IPTABLES -A INPUT -i $LAN1_IFACE -p udp --dport 67 -j ACCEPT

# Pakety od navázaných spojení jsou v pořádku
$IPTABLES -A INPUT -d $INET_IP -m state --state ESTABLISHED,RELATED -j ACCEPT

# Všechno ostatní je zakázáno, logujeme tedy (max. 12 x 5 paketů/hodinu)
$IPTABLES -A INPUT -m limit --limit 12/h -j LOG --log-prefix "INPUT drop: "

#                                      #
# ---------- Řetězec OUTPUT ---------- #
#                                      #

# TOS flagy slouží k optimalizaci datových cest. Pro ssh, ftp a telnet
# požadujeme minimální zpoždění. Pro ftp-data zase maximální propustnost.
$IPTABLES -t mangle -A OUTPUT -o $INET_IFACE -p tcp --sport ssh -j TOS --set-tos Minimize-Delay
$IPTABLES -t mangle -A OUTPUT -o $INET_IFACE -p tcp --dport ssh -j TOS --set-tos Minimize-Delay
$IPTABLES -t mangle -A OUTPUT -o $INET_IFACE -p tcp --sport ftp -j TOS --set-tos Minimize-Delay
$IPTABLES -t mangle -A OUTPUT -o $INET_IFACE -p tcp --dport ftp -j TOS --set-tos Minimize-Delay
$IPTABLES -t mangle -A OUTPUT -o $INET_IFACE -p tcp --dport telnet -j TOS --set-tos Minimize-Delay
$IPTABLES -t mangle -A OUTPUT -o $INET_IFACE -p tcp --sport ftp-data -j TOS --set-tos Maximize-
Throughput

# Povolíme odchozí pakety, které mají naše IP adresy
$IPTABLES -A OUTPUT -s $LO_IP -j ACCEPT
$IPTABLES -A OUTPUT -s $LAN1_IP -j ACCEPT
$IPTABLES -A OUTPUT -s $INET_IP -j ACCEPT

# Povolíme DHCP broadcasty na LAN rozhraní
$IPTABLES -A OUTPUT -o $LAN1_IFACE -p UDP --dport 68 --sport 67 -j ACCEPT

# Ostatní pakety logujeme (neměly by být žádné takové)
$IPTABLES -A OUTPUT -j LOG --log-prefix "OUTPUT drop: "

-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
```

9.2 /etc/squid3/squid.conf

K odstranění řádků komentářů konfiguračního souboru Squidu lze použít (kvůli přehlednosti a délce dokumentu) následující příkaz(y):

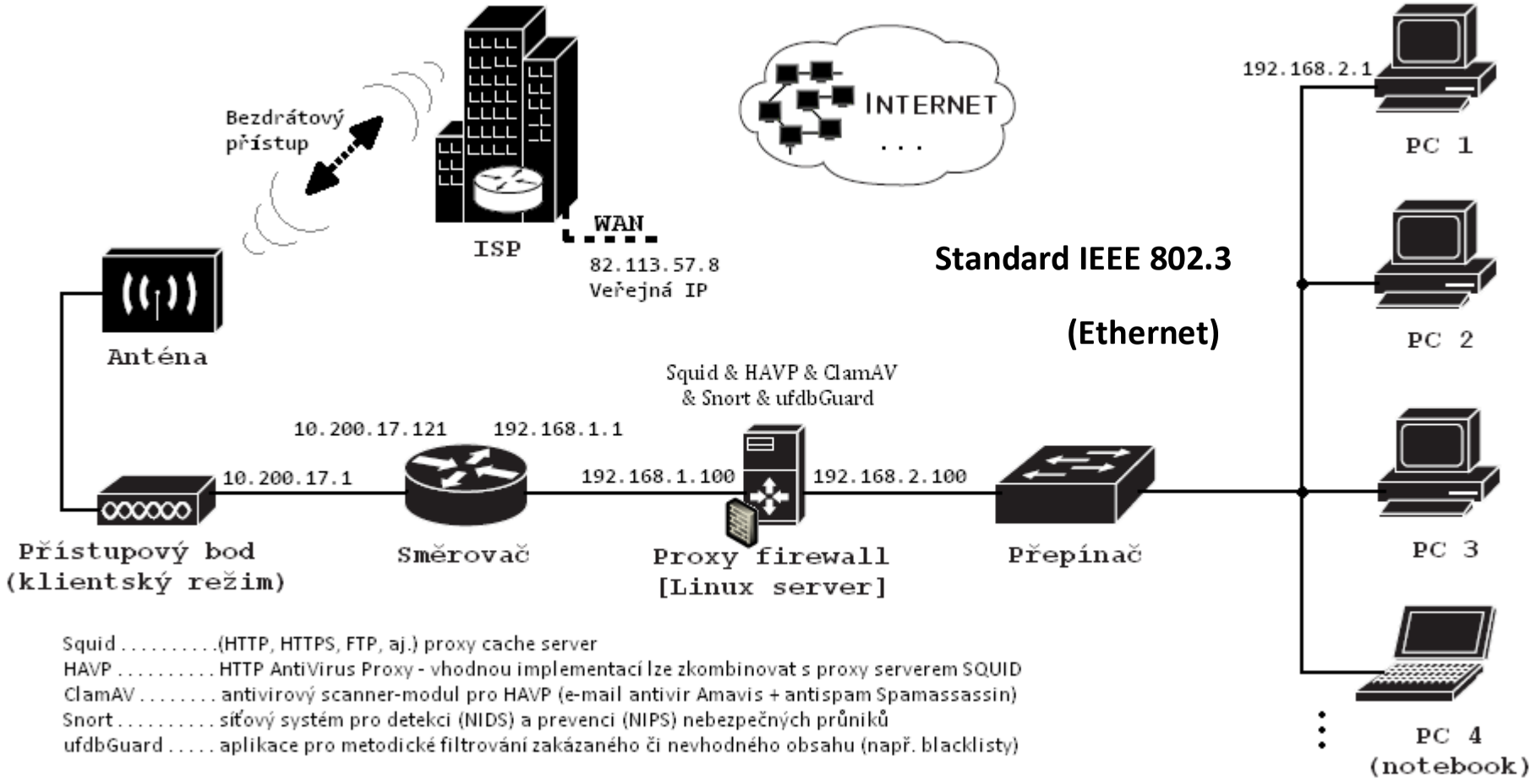
*# Tento příkaz odstraní komentáře z konfiguračního souboru **squid.conf**:*
cat /etc/squid3/squid.conf | egrep -v "^\s*(#|$)"

 # případně je také použitelné: grep -v ^$ squid.conf.org |grep -v ^# > squid.conf

```
# squid.conf
#          ### ----------- KONFIGURAČNÍ SOUBOR SQUIDU ----------- ###
#
# Podrobně komentovaná funkční konfigurace Squidu, v praxi nasazená a otestovaná
# na linuxovém serveru Debian, pracujícím jako proxy firewall.
#
# Poznámka: Squidův konfigurační soubor (/etc/squid3/squid.conf) je velmi obsáhlý,
# a proto zde není uveden celý, ale pouze mnou upravené/změněné části kódu oproti
# originálnímu souboru. Ostatní neuvedené parametry jsou brány jako výchozí nastavení.
# ******************************************************************************

# Nastavení ACL (Access Control List, seznam pro řízení přístupu)
acl manager proto cache_object  # Třída manager jsou požadavky, které používají speciální
acl localhost src 127.0.0.1/32  # Squidův protokol a užívají se jen pro administr. účely.
acl to_localhost dst 127.0.0.0/8 # Třída localhost jsou požadavky z IP adresy 127.0.0.1,
                                 # tedy z našeho loopbacku. Výsledkem tedy je:
                                 # "protokolem cache_object jen z lokálního počítače".

# Naše adresa sítě bude 192.168.2.0/255.255.255.0 (IP adresa sítě/maska sítě).
# Klientské (hostitelské) počítače v rozsahu adres 192.168.2.1 ~ 192.168.2.254,
# vyjma IP adresy 192.168.2.100 (výchozí brána).
acl localnet src 192.168.2.0/24 # Dle RFC1918 jedna z možných interních (vnitřních) sítí.

# Zakázané adresy (vybrané kritické domény z oblastí pornografie, online her a rasismu)
acl forbiddenpages dstdomain www.freefoto.cz www.freeonlinegames.com www.racism.net

# Definujeme povolené ("bezpečné") porty, potažmo služby
acl Safe_ports port 80          # http
acl Safe_ports port 443         # https
acl Safe_ports port 21          # ftp
acl Safe_ports port 22          # ssh
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535  # neregistrované porty
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
acl Safe_ports port 591         # filemaker
acl Safe_ports port 777         # multiling http

# Metoda spojení
# HTTP podporuje několik metod spojení - get, post a connect. Get se používá
# pro download, post pro zasílání informací a connect zejména pro ssl spojení.
acl CONNECT method CONNECT

# Jestliže požadavek spadá do ACL třídy manager a zároveň do třídy localhost, má
# jej povolit. Tak zkontroluje, zda požadavek do těchto tříd patří. Pokud ano, je
# požadavek povolen a tím vyhodnocování končí, pokud ne, přejde se k dalšímu řádku.
http_access allow manager localhost

# Vše, co patří do třídy manager, se má zakázat. Spolu s výše uvedeným řádkem to znamená:
# "Pokud požadavek patří do manager, povol jej pouze tehdy, pokud patří i do localhost".
http_access deny manager

# K zamezení přístupu na ostatní porty použijeme (vykřičník = logická negace)
http_access deny !Safe_ports

# Blokujeme metodu connect pro jiná než ssl spojení
http_access deny CONNECT !SSL_ports
```

```
# Zakážeme ("deny") nepovolené "forbiddenpages" stránky (resp. URL adresy)
http_access deny forbiddenpages

# Povolíme ("allow") naši síť (localnet) a localhost
http_access allow localnet
http_access allow localhost

# Vše ostatní zakážeme (tzn. co není výslovně povoleno, to je zakázáno!)
http_access deny all   # "Default policy" je deny (odmítnutí, zákaz)

# Určuje IP adresu a port, na kterém Squid naslouchá příchozím požadavkům
# (standardně port 3128) + nastavení transparentního režimu Squidu
http_port 192.168.1.100:3128 transparent

# Direktiva k nastavení rodičovské proxy, stejný stroj jako Squid (IP), port 8088
cache_peer 127.0.0.1 parent 8088 0 no-query no-digest no-netdb-exchange default
cache_peer_access 127.0.0.1 allow all   # "cache_peer" pro nastavení A-V proxy HAVP

hierarchy_stoplist cgi-bin ?   # Nedotazování se souseda na vše (veškeré podněty)

# Udává velikost fyzické paměti, kterou Squid použije jako cache v paměti.
# Neurčuje celkovou paměť zabranou Squidem, ale jen tu, kterou použije navíc.
# Reálně Squid zabere paměti ještě více. Je jasné, že čím více paměti mu dáme,
# tím větší výkon může podat. Musíme ale ponechat dostatečně velkou rezervu
# pro systém a ostatní procesy. Swapující počítač by nebyl vhodným řešením.
cache_mem 64 MB

# Maximální velikost objektu (souboru), který je ukládán do cache paměti.
# Pokud například uživatel stahuje obrázek, který má 200 kB, uloží se do cache.
# Pokud ovšem stahuje video, které má 20 MB, není už cachováno.
maximum_object_size_in_memory 4096 KB

# Totéž, jen určuje nejmenší velikost objektů ukládaných do cache.
minimum_object_size 0 KB

access_log /var/log/squid3/access.log squid   # Logování přístupů

# Redirector pro Squid - program ufdbGuard, požadovaná lokace a logovací adresář
url_rewrite_program /usr/local/squid/bin/ufdbgclient -l /usr/local/squid/logs
url_rewrite_children 16   # Počet vyvolaných filtrů (procesů k vyvolání jiného)

# Nastaví, po jakou dobu má Squid uchovávat "nacachované" stránky ve svém úložišti
refresh_pattern ^ftp:           1440   20%    10080
refresh_pattern ^gopher:        1440   0%     1440
refresh_pattern (cgi-bin|\?)    0      0%     0
refresh_pattern .               0      20%    4320

# Uživatel, pod kterým poběží Squid, pokud bude spuštěn s právy roota
# (dojde k automatickému přepnutí ihned po startu).
cache_effective_user proxy

# Totéž, ovšem s určením skupiny.
cache_effective_group proxy

# Určuje port, na kterém Squid vysílá požadavky pro vzdálené cache.
# Tuto funkci lze vypnout zadáním "0" místo čísla portu.
icp_port 0

# Adresář s chybovými hlášeními. Zde uvedená direktiva zabezpečí zobrazování
# českých chybových hlášení.
error_directory /usr/share/squid3/errors/Czech

# Squid zašle požadavek na server a nebude žádat "peera". Odpověď se "cachuje" tak,
# jak je běžné.
always_direct allow all

# Zakážeme (kvůli anonymitě) X-Forwarded-For hlavičku.
# Nezjistí se tak IP adresy počítačů v lokální síti schované za proxy firewallem.
forwarded_for off

# "coredump_dir" nastavujeme adresář, kde bude Squid zanechávat soubory "core"
coredump_dir /var/spool/squid3
```

Linuxový proxy firewall



* Konkrétní navržené řešení proxy firewallu testováno a provozováno v těchto (domácích) podmínkách. Veškerá přenášená komunikace je navíc filtrována linuxovým firewallem se stavovou inspekcí – **Iptables** (kontrola na základě zdrojové/cílové IP adresy a portu, stavu probíhajícího spojení).