ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA JADERNÁ A FYZIKÁLNĚ INŽENÝRSKÁ

Obor: Inženýrská informatika
Zaměření: Tvorba softwaru

GNU/Linux a bezpečnost v akademických sítích

REŠERŠNÍ PRÁCE

Autor: Bc. Josef Kadlec
Vedoucí rešeršní práce: Mgr. Jiří Fišer, Ph.D.

Rok 2005

Prohlášení

Prohlašuji, že jsem tuto rešeršní práci vypracoval samostatně a uvedl jsem veškerou použitou literaturu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 zákona 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským.

V Praze dne 25. dubna 2005 Josef Kadlec

Poděkování

Tímto bych chtěl poděkovat Mgr. Jiřímu Fišerovi, Ph.D. za jeho rady a věcné připomínky k mé práci.

Anotace

Rešeršní práce analyzuje hrozby a možnosti operačního systému GNU/Linux k jejich eliminaci s přihlédnutím na sítě v akademickém prostředí. Práce je rozdělená do čtyř základních tématických částí, kde se postupně seznamujeme s problematikou bezpečnosti na úrovni operačního systému, přes bezpečnostní problémy spojené se sítí až k fyzické bezpečnosti. V poslední části je stručně popsán a graficky znázorněn příklad sítě, která by se mohla v akademickém prostředí vyskytnout. Na problematiku je pohlíženo ze strany administrátora, takže je analýza bezpečnostních rizik popisována s jistou mírou teoretičnosti a přesné metody zneužití jsou skryty. Použité programové vybavení je většinou na bázi Open Source, což obnáší výhody i nevýhody, které jsou zmíněny v úvodní kapitole. Tato práce je také jakási příručka administrátora, pomocí které lze krok za krokem vyřešit bezpečnost nejen akademické sítě. Autor zároveň usiluje o to, aby při vytváření informačních řešení byl na problematiku bezpečnosti brán stejný ohled, který je brán například na funkčnost. Pro správné pochopení problematiky, kterou se zabývá tato práce, je nutná dobrá znalost administrace některého systému z rodiny Unixů – nejlépe však GNU/Linuxu.

Annotation

This research thesis analyzes threats of the operating system GNU/Linux and its abilities of their elimination in relation to the network in an academic environment. The work is divided into four basic logical parts, where we in sequence get acquainted with the issue of security at an operating system level, from problems of network security to physical security. In the final part an example of network is described and graphically demonstrated, which could appear in an academic environment. The issue is watched from an administrator´s side so that the analysis of security risks is described theoretically up to the point and exact exploiting methods are hidden. The software that is used is mostly Open Source-based, which has pros and cons, that are mentioned in the opening chapter. This work is an administrator guide which helps to find a solution of security not only of an academic network step by step. The author strives for taking the same regard for an issue of security which is taken for funcionality in the course of creating information solution. For right understanding of the issue solved in this work it is necessary to have good knowledge of administration of a system from the Unix family – preferably GNU/Linux.

Obsah

Předmluva

Cílem práce je analyzovat současné možnosti zabezpečení operačního systému GNU/Linux s přihlédnutím na prostředí akademických sítí. Na problém má být nahlíženo z pohledu administrátora – práce se nemá zabývat nízkoúrovňovou bezpečností. Popisovaná problematika má postihovat jak obecné problémy, tak současnou situaci v oblasti informační bezpečnosti. Práce má popsat prostředky zabezpečení od těch nejjednodušších až po komplexní softwarová řešení. Pokud to bude možné, měly by být k řešení bezpečnostních problémů používány Open Source projekty. Struktura a způsob zpracování práce má být uzpůsoben k použití coby uživelské příručky administrátora.

Kapitola 1

Úvod

V současné době, kdy počítače zasahují prakticky do každé sféry našeho života, jsme se na nich stali závislí. Asi si těžko představíme život bez jejich asistence – ať už jde o sféru vědeckou, komerční nebo jen o náš domácí osobní počítač. Počítač nám zprostředkovává mnoho forem komunikace, provádí výpočty všeho druhu nebo prostě jen slouží k zábavě.

Na samotném začátku vzniku počítačů a jejich následného spojování do větších sítí (například Arpanet[1], později pak Internet[2], nebo prostě jen místní datové sítě) nikdo netušil, jakým fenoménem se informační technologie stanou a jak obrovského uplatnění dosáhnou. Ovšem tyto, na první pohled dokonalé, stroje stvořil člověk, který dělá chyby a proto ani tyto stroje nejsou dokonalé a obsahují chyby, které jsou v tomto nedokonalém světě zneužívány z mnoha důvodů. Jedna z příčin, která stála za vznikem oboru informační bezpečnosti, byl jistě vznik sítě Internet, který najednou vzájemně spojoval ne pouze důvěryhodné subjekty, jako tomu bylo na počátku vzniku sítí, kdy spolu měla možnost komunikovat jen hrstka počítačů, které byly navíc dostupné pouze malé skupině uživatelů.

1.1 Obor informační bezpečnosti

Pokud bychom to chtěli hodně zjednodušit, tak obor informační nebo také počítačové bezpečnosti, v poslední době diskutovaný v mnoha pádech, se zabývá ochranou dat nebo jiných hodnot před neoprávněným přístupem. Co myslíme těmi jinými hodnotami? Může to být například výpočetní výkon počítače, diskový prostor nebo pásmo internetového připojení. Tento obor je velmi mladý a samotná informační bezpečnost ještě nevešla zcela do povědomí lidí. Touto prací bych se chtěl také zasloužit o to, aby byla počítačová bezpečnost brána skutečně vážně. Aby byla při vytváření nejen softwarových řešení brána v potaz stejně jako například funkčnost.

[1] http://www.dei.isep.ipp.pt/docs/arpa.html
[2] http://www.isoc.org/internet/history/

Ještě bych měl zmínit jeden aspekt, který významně ovlivnil a stále ovlivňuje počítačovou bezpečnost a to je fakt, že v dnešní době stále používáme například protokoly, které byly vytvořeny v době, kdy se na bezpečnost nebral ohled a vše bylo vytvářeno do jisté míry otevřeně. Příkladem z dnešní doby může být například protokol POP, který slouží k manipulaci s elektronickou poštou. Hesla jsou zde přenášena v textové podobě, nijak nechráněna proti zachycení jinou osobou. Kdo by to dříve tušil, že například tento protokol, usnadňující lidem život, bude takhle zneužíván.

1.2 Proti čemu se bráníme

Systém je nejvýše tak bezpečný, nakolik bezpečné je jeho nejslabší místo. Toto je velmi důležité pravidlo, které musíme mít na paměti. Představme si zloděje, který se snaží vlámat do domu. Určitě si nevybere cestu přes zabezpečené dveře, když je pootevřené přízemní okno. Naprosto stejně to funguje u operačních a jiných systémů.

Ovšem musíme mít také na paměti druhé pravidlo a to princip nejsnazšího průniku, který říká, že o narušiteli musíme předpokládat, že využije každý možný způsob průniku, přičemž to nemusí být ten nejsnazší ani ten, proti kterému byla přijata nejdokonalejší ochranná opatření.

Útoky, proti kterým se budeme bránit rozdělíme do několika skupin. Prvním typem útoku je, že narušitel čte naše data. To znamená, že se vetřelec dostal k více, či méně důvěrným datům a může s nimi naložit jak se mu zlíbí. V případě, že se jedná například o hesla, čísla kreditních karet nebo složení nového farmaceutického výrobku, je situace jistě zcela vážná. Samotná data nemusí být získána pouze čtením z pevných disků, ale i z jiných nosičů informací jako je například ethernetový kabel nebo pouze vzduch, kterým se šíří bezdrátový přenos. Z těchto médií, kterými se data přenášejí, mohou být informace zachyceny – tento způsob se nazývá *sniffing*. Dalším, na první pohled jistě zvláštním, ovšem ne zcela ojedinělým (zvláště v USA), způsobem získání informací je tzv. *trash-diving* (nebo jen *trashing*), kdy jsou prohledávány odpadky většinou významnějších institucí a hledány například záznamy o uživatelských kontech, manuály k ústřednám nebo jiné zajímavé informace. Samozřejmě tyto dva útoky nelze stavět do stejné roviny.

Druhým typem útoku je, že narušitel modifikuje data. O tomto útoku se mluví jako o jednom z nejzákeřnějších. Útočník může změnit konstrukční a výrobní plány letadel nebo léků nebo zdravotní stav pacienta, což může vést v nejhorším případě i ke ztrátě na životech. Modifikovat lze také samozřejmě přenášená data. Proti tomuto útoku se lze účinně bránit kontrolou integrity dat.

Dalším typem útoku je mazání dat útočníkem, což bez zvládnuté zálohovací politiky, může vést k finančním ztrátám.

Mezi další typy útoků patří zneužití jiných hodnot, než dat. Patří sem například využití početního výkonu k lámání přístupových hesel nebo využití celého systému a jeho síťového připojení k napadání jiných systémů. Specifickým příkladem napadení systému je tzv. DoS (*Denial of Services* nebo česky odepření služby). Jedná se o útok, kdy je určitá služba zahlcena velkým množstvím požadavků, které není schopna zvládnout. Cílem je zpomalení služby nebo její úplné odpojení. Jsou známy i případy, kdy DoS může sloužit ke zvýšení privilegií útočníka. Sofistikovanější formou útoků odepření služby je DDoS (*Distributed Denial of Services*), kde je do útoku na cíl zapojeno paralelně více útočníků, což celý proces zefektivňuje.

Trochu zde opomíjím fyzickou bezpečnost. Samozřejmě útočník může získat vaše data tak, že vám odcizí notebook nebo sejme záření monitoru. Ovšem my se v této práci zaměříme především na bezpečnost na úrovni OS (operační systém) a sítě.

1.3 Proti komu se bráníme

Vyhnu se ne příliš duchaplnému rozdělení útočníků na špióny, extrémisty nebo nespokojené zaměstnance. Vetřelce si můžeme rozdělit na ty, co útočí z "vlastních řad" – například útok na firewall ze sítě, kterou odděluje od okolního světa a na útočníky, kteří nás napadají z okolního světa – například z Internetu. Paradoxem však je, že většina útoků pochází právě z "vlastních řad".

Tito vetřelci jsou všeobecně označováni jako crackeři nebo hackeři. Jedním ze zásadních problémů je, že lidé si často pletou významy slov "hacker" a "cracker". Zatímco hackery se označovali programátoři (nazývaní také "Opravdovými programátory") ze slavné éry laboratoří MIT[3], tzv. "Zlaté éry hackingu", crackery byli a jsou označovány osoby, které úmyslně škodí pro vlastní prospěch nebo jen tak pro zábavu. Bohužel význam slova "hacker" postupem doby zmutoval a dnes si pod ním většina lidí představí právě toho, kdo napadá systémy pro vlastní prospěch. Více se o skutečném hackerství dočtete v dokumentu *The Jargon File*[4], který je stále aktualizovaný Ericem S. Raymondem[5].

[3] Massachusetts Institute of Technology

[4] http://www.catb.org/jargon/

[5] Autor stati *Cathedral and market*, jejíž český překlad lze nalézt na této URL – http://www.zvon.org/translations/cathedral-bazaar/Output/index_cs.html

Za zmínku stojí ještě termín "script kiddies", jinak také "wannabies", kterým jsou označovány osoby, které se snaží bez jakékoliv znalostí napadat systémy často pouze pomocí programů, které vytvořili lidé se znalostí dané problematiky. Poslední termín, se kterým se můžete setkat při studování materiálů o informační bezpečnosti, je termín "phreacker", což je člověk, který využívá telefonních linek k dosahování různých cílů. Můžeme se setkat i s některými dalšími označeními.

1.4 Bezpečnostní vrstvy

Pro zjednodušení celého popisu zavádím tři bezpečnostní vrstvy.

1.4.1 Bezpečnost na úrovni OS

První vrstvou je bezpečnost na úrovni OS. Tu dále můžeme rozdělit na další tři základní vrstvy. S popisem začneme od té nejnižší úrovně. Nejnižší vrstvou, která komunikuje až s hardwarem, je bezpečnost na úrovni jádra. Sem patří samotné jádro operačního systému, které zajišťuje především správu paměti, plánování procesů, atd. a dále ovladače, které zajišťují komunikaci s hardwarem. Tato vrstva je těsně spjata s celou stabilitou systému a dojde-li například k špatnému přidělení nebo ochraně paměťového prostoru pro uživatelský proces, může dojít k zhroucení celého systému.

Druhou vrstvou je bezpečnost na aplikační úrovni. Aplikace by se měla chovat přesně tak, k čemu byla původně vytvořena bez vedlejších nechtěných funkcí. Dále je velmi důležitá správná implementace programového kódu, aby nedocházelo k zápisům mimo přidělenou paměť a z toho plynoucích následků (ať už se jedná o pád programu nebo spuštění jiné aplikace), čehož může záškodník přímo využít. Častou chybou je také neošetření vstupu od uživatele. Bezpečné programování je kapitola sama o sobě a zájemce o tuto problematiku odkazuji na dokument *Secure Programming for Linux and Unix HOWTO*[6], který se zabývá bezpečným vytvářením aplikací v OS GNU/Linux a v systémech z rodiny Unixů. Tímto tématem se v této práci přímo nezabývám, protože se k problematice bezpečnosti stavím spíše z pohledu administrátora než programátora.

Poslední nejvyšší vrstvou je bezpečnost na uživatelské úrovni. Zde se jedná především o chování administrátora systému a samotných uživatelů systému. Administrátor by měl zajistit uživatelům správné chování systému a uživatel by se měl chovat podle stanovených pravidel. Důležitou roli zde hraje lidský faktor.

[6] http://www.dwheeler.com/secure-programs/

1.4.2 Bezpečnost na úrovni sítě

Do této skupiny patří především ohrožení, které je důsledkem špatné topologie sítě, špatně nakonfigurovaným firewallem[7] nebo jinými nedostatky. Patří jsem hrozby jako IP spoofing, ARP cache poisoning, odposlech přenosu, únos relace, ale řadím sem i problematiku virů[8].

1.4.3 Fyzická bezpečnost

Do této sekce patří ohrožení, při kterém útočník překonává nedostatky, ke kterým je nutný fyzický přístup. Zahrnuji sem například zabezpečení boot managera[9] nebo důvěrně známý problém "horkých" kláves Ctrl+Alt+Del, kterými může jakýkoliv uživatel kdykoliv vyvolat nežádoucí restart systému. Samozřejmě sem patří i ne příliš sofistikované útoky, kdy vám útočník odcizí notebook (mimochodem opravdu častá metoda získávání informací, v roce 2001 bylo v USA zcizeno asi 591 000 notebooků[10]) nebo použije požární sekeru k přerušení provozu vašeho serveru – těmito metodami se samozřejmě zabývat nebudu.

1.5 Proč GNU/Linux

Systémy na bázi GNU/Linuxu využívající jádra Linuxu se zvláště v poslední době stávají velmi populární a v mnoha sférách (komerční, státní, akademická, apod.) na něj přecházejí. Jedná se o operační systém s volně šiřitelným otevřeným zdrojovým kódem, což ho zásadně odlišuje od proprietárních systémů. Vyznačuje se především svojí stabilitou a podporou široké palety hardwaru. I když se původně začal používat především na serverech, dnes ho často potkáte i na pracovních stanicích – pro tuto platformu je již k dispozici nesčetné množství aplikací jako například kancelářské balíky (OpenOffice.org, KOffice), nástroje pro přehrávání multimédií (Mplayer, Xine, XMMS), práce s grafikou (Gimp, Blender) atd.

Ještě zmíním jméno původního autora kernelu Linuxu a tím je Linus Torvalds. V dnešní době se správě a dalšímu rozšiřování zdrojového kódu kernelu Linuxu podílí několik vývojářů z celého světa. Zakladatelem GNU je Richard Matthew Stallman[11].

[7] Více o firewallech v kapitole "4 Bezpečnost na úrovni sítě".
[8] Více o síťových hrozbách v kapitole "4 Bezpečnost na úrovni sítě".
[9] Zavaděč systému
[10] http://www.pcguardiananti-theft.com/Additional_Resources/Computer_Security_Threat_11_02.pdf
[11] http://www.gnu.cz/rms.html

1.5.1 Open Source a bezpečnost

Open Sourcem[12] označujeme něco, co je volně šiřitelné a k čemuž máme k dispozici zdrojový kód[13]. Kdokoliv může v tomto kódu provádět změny. Do rodiny Open Source nepatří pouze Linux, ale i další projekty jako například webový server Apache[14] nebo program pro správu elektronické pošty Sendmail[15] a mnoho dalších.

Ale podívejme se na Open Source z pohledu bezpečnosti. Je samozřejmé, že pokud máme přístup ke zdrojovému kódu, můžeme provést mnohem hlubší analýzu aplikace. Zastánci Open Source tvrdí, že otevřenost kódu dělá aplikace bezpečnější. Mnoho vývojářů z celého světa se může podílet na odhalování a opravování chyb. A skutečně to tak funguje. Je faktem, že doba od nalezení chyby a vydáním opravy je mnohem kratší než u uzavřeného softwaru, kde má ke zdrojovému kódu přístup jen omezené množství lidí. Hlavním argumentem odpůrců otevřeného softwaru je fakt, že tvůrci a správci tohoto softwaru se mu věnují pouze ve svém volném čase a pokud se jim nebude chtít, tak otevřený software nebude vznikat, vyvíjet se, nebudou vznikat bezpečnostní záplaty apod. V dnešní době už to nelze takto úplně paušalizovat. Naštěstí tomu tak není a vůle ze strany vývojářů tu je a rozhodně to nevypadá, že Open Source v nejbližší době zanikne. Tohle bych nepovažoval až zas tak za velkou hrozbu, rozhodně ne z hlediska bezpečnosti.

Podle mě mnohem hrozivějším faktem je to, že každý může přispět svojí částí kódu, kterou maintaineři[16] zaintegrují do svého projektu (samozřejmě nemusí). Ovšem co když člověk, který přispívá svojí částí kódu (může jít například o novou featuru[17]) nemá tak úplně dobré úmysly. Co když se v kódu nachází na první pohled neviditelná chyba – může se jednat o chyby přetečení zásobníku nebo mnohem snáze objevitelné logické a race condition chyby. Co když se útočníkovi podaří takovou část kódu propašovat například do projektu Apache, což je nejrozšířenější webový server tvořící asi dvě třetiny všech webových serverů a nebo do samotného kernelu Linuxu. Záškodník si tak vytvoří velmi sofistikovaný backdoor[18], kterým může dosáhnout až práv roota[19]. Samozřejmě maintaineři svých projektů kód kontrolují, ale zde už záleží jen na jejich pečlivosti a znalostech takové chyby rozpoznat.

[12] http://www.opensource.org/
[13] Nezaměňovat se svobodným softwarem (tzv. *free software*).
[14] http://www.apache.org/
[15] http://www.sendmail.org/
[16] Vedoucí projektu
[17] Rys
[18] Zadní vrátka
[19] Superuživatel v systémech na bázi Unixu

1.6 Prostředí akademických sítí

Jedná se o velmi různorodé prostředí, kde se samozřejmě nemusí nacházet pouze operační systémy GNU/Linux. Síť může tvořit více, či méně počítačů (omezení může tvořit počet přidělených IP adres nebo adresní rozsah privátních IP adres), které mohou být pospojované různými fyzickými zařízeními do LAN[20], WAN[21], nebo i jiných struktur. Málokdy v tomto prostředí narazíme na vytáčené spojení.

Uživatele těchto sítí musíme považovat za nedůvěryhodné a je nutné zavést správnou bezpečnostní politiku a eliminovat jakékoliv hrozby jak ze strany uživatelů, tak z vnějšího prostředí – Internetu. Na rozdíl od běžných sítí, kde bud' nejsou žádní nebo malé množství neměnících se uživatelů, stálý příliv nových studentů v prostředí akademických sítí zvyšuje riziko ohrožení bezpečnosti počítačů v těchto sítích.

Nutno také upozornit, že v tomto prostředí se, vedle fluktuace studentů, vyskytuje fluktuace zaměstnanců, která je oproti jiným sítím relativně vyšší. Na toto by měl být brán ohled při vytváření pevné bezpečnostní politiky – například pokud odejde zaměstnanec, který znal superuživatelské heslo stroje v akademické síti, toto heslo by mělo být samozřejmě s příchodem nového zaměstnance změněno.

Akademické sítě patří obecně mezi sítě s velkým počtem počítačů. Větší množství počítačů na síti je nutné rozdělit na menší segmenty za účelem zvýšení bezpečnosti a zjednodušení správy. Mezi segmenty by měla být nakonfigurována pečlivá kontrola přístupu, ale v každém segmentu může být jiná politika zabezpečení. Segmentem můžeme označovat například počítačovou laboratoř, vědecké pracoviště nebo webový server. Samozřejmě jiná konfigurace týkající se bezpečnosti bude aplikována na pracovní stanice a jiná na servery nebo routery[22].

Tyto sítě také disponují vysokorychlostním připojením, které tvoří potencionální riziko pro ostatní počítače vně akademické sítě (počítače v Internetu). Samozřejmě každý uživatel za běžných podmínek by měl mít k dispozici pouze část pásma internetového připojení, což by mělo riziko snížit. Riziko spočívá v tom, že pokud by si někdo zvýšil privilegia a mohl užívat větší nebo celou část tohoto pásma, mohl by s velkou úspěšností realizovat útoky DoS na sítě se slabším připojením, které by nápor paketů doslova odstřihl od Internetu.

[20] Local Area Network
[21] Wide Area Network
[22] Směrovače

Kapitola 2

Základní hardening GNU/Linuxu

Bezpečnost samotného operačního systému má v akademických sítích, kde se nachází mnoho uživatelů, velký význam. Ať už jsou uživatelé k narušování bezpečnosti jakkoliv motivováni, měli bychom jakýmkoliv zvyšováním privilegií ze strany uživatelů na všech počítačích striktně zabránit. Systém GNU/Linux nabízí jakési základní (nebo nativní) mechanismy lokálního zabezpečení a dále jsou k dispozici nástroje, které poskytují nadstandardní možnosti hardeningu operačního systému.

V této kapitole si probereme právě zmíněné nativní mechanismy a také hrubé chyby, kterých se můžeme dopustit při administraci linuxových systémů a jak se těmto chybám vyvarovat.

2.1 Správné nastavení přístupových práv

Dalo by se říci, že problematika souborových práv a uživatelů je základním kamenem bezpečnosti v OS GNU/Linux. Při zabývání se problematikou bezpečnosti se bez znalosti této problematiky neobejdeme. GNU/Linux je víceuživatelský operační systém. Tzn. že se do systému může přihlásit více uživatelů – a to i ve stejný moment. V rámci zabezpečení uživatelů a nakonec i celého systému je proto nutné souborová práva zavést.

To jak fungují přístupová práva v Linuxu vysvětluji v další části této podkapitoly. Ovšem je také potřeba tyto znalosti použít k vytvoření správné politiky přístupových práv. Situaci nám usnadňují vývojáři distribucí, kteří v distribuci určitou bezpečností politiku zavedou, takže nám stačí většinou pouze modifikovat současná přístupová práva. U větších distribucí jako Mandrake, Slackware, Fedora, Debian, atd. by se nám nemělo stát, že po nainstalování budou přístupová práva nějak katastrofálně nastavena. Ovšem je potřeba se vyvarovat nebezpečným zásahům do tohoto nastavení jako například byt'jen nechtěné nas-

tavení práva zápisu pro vlastníka, skupinu i ostatní na soubor `/etc/passwd`, čímž bychom vlastně úplně odstranily systém zastíněných hesel[1]. Toto byl však případ, který spíše vypadá na sabotáž ze strany superuživatele, než nechtěný krok, ale vážné problémy častěji plynou z mnohem prostších situací jako například špatné nastavení práv na domovských adresářích, které se může lehce vyskytnout, pokud přidáváme další uživatele. Dále se například doporučuje nastavovat práva pro adresář `/boot` jen pro čtení. V tomto adresáři by se měl nacházet zkompilovaný kernel a pár dalších souborů jako například `System.map`. S těmito soubory je většinou manipulováno pouze v případě, kdy se mění samotný kernel. Poškození, záměna nebo modifikace těchto souborů může znamenat pohromu pro systém nebo pro bezpečnost systému. Dalším problémem může být nastavení přístupových práv souboru `/dev/tty`, který by měl mít nastaven mód 666, aby se zamezilo čtení cizích terminálů. Neměli bychom také zapomenout na vhodné nastavení výchozích práv příkazem `umask`.

Akademická síť je víceuživatelské prostředí, kde může dosahovat počet uživatelských kont relativně vysokého čísla. Tato konta (tzn. domovské adresáře a záznamy o uživatelích) jsou uložena na serveru a pracovní stanice je využívají pomocí služby NIS (Network Information Service), která dokáže sdílet soubory po síti. Ať si sednete k jakémukoliv počítači, můžete se z něj přihlásit a pracovat se svými soubory. Ovšem implicitně je těmto uživatelům dovoleno se na tento server vzdáleně přihlásit a pokoušet tak zabezpečení tohoto stroje. Proto je nutné držet tato práva co možná nejstriktněji. Pomocí ACL[2] lze navíc docílit mnohem jemnějšího rozdělení různých přístupových práv pro různé uživatele. Zvláštním případem je adresář `/tmp`, do kterého mají implicitně plná práva (`drwxrwxrwt`) všichni uživatelé. Všimněme si, že na tomto adresáři je aplikován *sticky bit*[3]. Proto také bývá tento adresář využíván k různým nekalým činnostem – od úložiště velkého množství dat, což může způsobit zaplnění diskového oddílu až po místo, odkud lze lehce spouštět soubory jako například shelly, čehož se hojně využívá v různých exploitovacích technikách. Podobná omezení platí samozřejmě i pro pracovní stanice. Je fakt, že pokud se naruší provoz pracovní stanice, nic to neznamená ve srovnání s narušením provozu serveru. Správným nastavením přístupových práv se samozřejmě nechráníme jen proti "našim" uživatelům, ale i proti jiným nebezpečím.

[1] Více o zastíněných heslech dále v této podkapitole.
[2] Více o ACL dále v této podkapitole.
[3] Více o sticky bitu dále v této podkapitole.

2.1.1 /etc/passwd

V souboru /etc/passwd jsou uchovány informace o všech uživatelích daného linuxového systému. Zde je názorná ukázka toho, jak může vypadat tento soubor:

```
root:x:0:0::/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/log:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/:
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
operator:x:11:0:operator:/root:/bin/bash
games:x:12:100:games:/usr/games:
ftp:x:14:50::/home/ftp:
smmsp:x:25:25:smmsp:/var/spool/clientmqueue:
mysql:x:27:27:MySQL:/var/lib/mysql:/bin/bash
rpc:x:32:32:RPC portmap user:/:/bin/false
gdm:x:42:42:GDM:/var/state/gdm:/bin/bash
pop:x:90:90:POP:/:
nobody:x:99:99:nobody:/:
sshd:x:33:33:sshd:/:
pepa:x:1000:100:Josef Kadlec,413,098 876,435 876:/home/user:/bin/bash
```

Každý řádek souboru je záznamem o jednom uživateli. Superuživatel root může tento soubor libovolně editovat – buď textovým editorem nebo lépe utilitou vipw, která sníží riziko při nečekané události během provádění daných změn v tomto adresáři. K přidání dalších uživatelů může superuživatel použít utilit adduser nebo useradd. K odstranění pak příkaz userdel a k editaci nastavení uživatele příkaz usermod.

Jednotlivé informace o každém uživateli jsou odděleny znakem dvojtečky. První údaj je uživatelské jméno unikátní pro každý linuxový stroj. Prvním typem uživatele je superuživatel, který má přístup ke všem souborům a může spouštět všechny programy. Jeho UID se rovná hodnotě 0. Na jednom systému by se měl vyskytovat pouze jeden superuživatel a zpravidla to bývá root. Další skupinou jsou systémoví uživatelé, kteří nereprezentují žádnou fyzickou osobu – nepřihlašují se do systému, většinou nemají svůj domovský adresář. Jejich UID je

menší než 1000. Slouží především ke spouštění služeb (angl. *daemon*) – například web serveru nebo ftp serveru. Poslední skupinou jsou běžný uživatelé, kteří se vyznačují hodnotou UID 1000 a více[4] a v souboru `/etc/passwd` jsou řazeni za systémové uživatele.

Druhý údaj představuje heslo uživatele. Ovšem u mého ukázkového souboru `/etc/passwd` jsou použita zastíněná hesla, což znamená, že zašifrovaná hesla uživatelů jsou uložena v souboru `/etc/shadow`, ke kterému má přístup pouze `root`. Uživatel si může změnit heslo příkazem `passwd`.

Třetí částí je UID (User Identification Number) – uživatelské identifikační číslo. Opět je na každém stroji jedinečné a systém pomocí něj rozlišuje přístupová práva k souborům u jednotlivých uživatelů.

Čtvrtou částí je GID (Group Identification Number) – identifikační číslo skupiny. Skupinou můžeme definovat určitý počet uživatelů, kteří budou mít stejné vlastnosti – především stejná přístupová práva k souborům. Skupiny a k nim patřící uživatelé jsou definovány v souboru `/etc/group`. K přidání dalšího uživatele do skupiny slouží příkaz `gpasswd`. K vytvoření, odstranění a modifikaci nastavení skupiny slouží příkazy `groupadd`, `groupdel` a `groupmod`. Opět lze použít utilitu podobnou `vipw` a to `vigw`.

Pátou částí souboru `/etc/passwd`, tentokráte nepovinnou, jsou informace o uživateli, které si každý uživatel může editovat příkazem `chfn`. Běžně je tato položka obsazena pouze skutečným jménem uživatele, ale zmíněný příkaz se ptá ještě na další informace, které poté oddělí čárkou. Tato položka je také označována jako tzv. záznam GECOS.

Šestou položkou je uživatelův domovský adresář. Zde se nacházejí soubory uživatele a některé konfigurační soubory, týkající se jeho profilu.

Poslední položkou je shell, který se uživateli spustí po přihlášení. Příkazem `chsh` se můžeme přepínat mezi shelly. Toto ovšem platí především pro superuživatele a běžné uživatele. Pokud budete chtít někomu zamezit přístup k shellu, vyplňte poslední položku hodnotou `/dev/null` – tohoto lze také docílit umístěním znaku "*" nebo znaků "!!" do pole, kde je heslo nebo znak `x`, který značí zastíněnná hesla.

Integritu tohoto souboru lze zkontrolovat příkazem `pwck`, který zkontroluje jedinečnost UID a GID, existenci domovského adresáře, atd.

[4] Toto neplatí pro všechny distribuce.

Zastíněná hesla

Protože soubor `/etc/passwd` mohou číst běžní uživatelé, dává jim to, v případě nezapnutých zastíněných hesel, možnost číst digitální otisky (angl. *hash*) hesel ostatních uživatelů a následného podrobení těchto otisků slovníkovým útokům nebo útokům hrubou silou, kde jsou postupně zkoušeny kombinace znaků. V případě zapnutí zastíněných hesel jsou otisky hesel uchovávány v souboru `/etc/shadow`, který má právo číst pouze superuživatel (přesněji všichni s hodnotou UID 0). Soubor `/etc/shadow` může obsahovat další informace jako datum poslední změny hesla, počet dní, které zbývají do další možné změny hesla, počet dní, které zbývají do další nutné změny hesla, doba, se kterou bude uživatel v předstihu upozorněn na změnu hesla nebo počet dní, které zbývají do nutné změny hesla, jejíž neprovedení bude mít za následek zablokování účtu. Tyto údaje lze měnit příkazem `chage`.

2.1.2 Jak fungují přístupová práva souborů

Přístupová práva souborů umožňují uživateli omezení přístupu k souborům a adresářům[5].

Zde vidíme ukázku zobrazení souborových práv:

```
pepa@workstation$ ls -la | grep .tuxracer
drwxr-xr-x 2 pepa users 4096 lis 19 14:03 .tuxracer
```

Nás bude především zajímat první část výpisu `drwxr-xr-x`. Důležité je také si všimnout, že vlastníkem souboru je uživatel `pepa` a k souboru se váže skupina `users`. Ostatní položky nás zatím nezajímají – samozřejmě kromě poslední položky, což je jméno souboru či adresáře.

Rozeznáváme tři druhy práv. První je právo na čtení (znak `r`) – u adresáře znamená prohlížení jmen souborů. Druhé je právo zápisu (znak `w`) – u adresáře znamená přidávání a odstraňování souborů. Třetí právo je právo ke spouštění (`x`) – v případě adresáře znamená toto právo vstup do adresáře – přístup k souborům. Pokud chci smazat soubor, musím mít v daném adresáři právo `w`. Pokud chci smazat adresář, musím mít právo `w` na všechny jeho podadresáře.

Sekvenci znaků `drwxr-xr-x` můžeme rozdělit na čtyři části. První část označuje typ souboru a nabývá základních hodnot `d` pokud se jedná o adresář (jako v našem případě), "`-`" pokud se jedná o normální soubor, `l` pokud se jedná o symbolický odkaz a `s` pokud se jedná o schránku.

[5] Což jsou v podstatě také soubory, které odkazují na další soubory, proto to někdy nebudu rozlišovat.

Za typem souboru následují tři trojice znaků, které reprezentují přístupová práva pro vlastníka souboru, skupinu, která se váže k souboru a ostatní uživatele. Pokud pozice obsahuje znak "-" je právo odepřeno. Z našeho příkladu můžeme vyvodit, že vlastník souboru, kterému patří první trojice `rwx` má právo na prohlížení jmen souborů v adresáři, vytváření a mazání souborů a právo vstupu do adresáře. Následující trojice znaků `r-x` nám říká, že skupina vázaná k souboru má právo na prohlížení jmen v adresáři a právo vstupu do adresáře. Právo na vytváření a mazání souborů jí je odepřeno. Zbytek světa neboli ostatní, reprezentováni třetí trojicí `r-x` mají v našem případě stejná práva jako skupina.

Přístupová práva může měnit vlastník souboru nebo superuživatel (příkaz `chmod`). Vlastníka souboru může měnit pouze superuživatel (příkaz `chown`). Skupinu, která se váže k souboru může měnit buď superuživatel nebo vlastník souboru, pokud patří do původní i cílové skupiny (příkaz `chgrp`).

Důležité je popsat algoritmus, jakým se rozhodne, zda má uživatel to určité právo (`r`, `w` nebo `x`). Funguje to takto: nejdříve ověříme, zda ten, kdo chce daného práva využít je superuživatel. Pokud ano, má povolen přístup. Pokud ne, ověříme, zda-li je vlastníkem souboru. Pokud ano, tak ověříme, jestli má povolené dané právo a podle toho povolíme přístup. Pokud ne, ověříme, jestli patří do skupiny, na kterou se váže soubor. Pokud ano, tak ověříme právo skupiny na danou akci a podle toho povolíme přístup. Pokud ne, tak ověříme, jestli má zbytek světa (ostatní) právo na akci a povolíme nebo zamítneme přístup.

Práva souborů mohou být reprezentována i číselnými soustavami – konkrétně se používá třech čísel v osmičkové soustavě. Všimněme si, že právo je zadané tak, že je buď povolené nebo zakázané. Z toho vyplývá, že naší sekvenci `rwxr-xr-x` můžeme přepsat na 111101101 a po převedení všech třech trojic `111`, `101` a `101` do osmičkové soustavy dostaneme 755. Tuto hodnotu můžeme používat k měnění přístupových práv příkazem `chmod`.

Jak již bylo řečeno, změna práv se provádí příkazem `chmod`. V případě použití reprezentace práv v osmičkové soustavě je situace jednoduchá:

```
chmod 751 file.txt
```

Tento příkaz změní přístupová práva souboru `file.txt` na `rwxr-x-x`.

Pokud chceme například přidat právo na spuštění pro vlastníka, skupinu i ostatní u souboru `file.sh` provedeme to příkazem:

```
chmod +x file.sh
```

Pokud bychom například chtěli přidat právo zápisu jen pro skupinu u souboru `file.sh`, provedli bychom to takto:

```
chmod g+w file.sh
```

Pokud bychom toto právo chtěli odebrat, vypadalo by to následovně:

```
chmod g-w file.sh
```

Vidíme, že přidání přístupového práva se realizuje pomoci symbolu "+" a odebírání probíhá pomocí symbolu "-". Vlastník souboru je reprezentován symbolem `u`, skupina symbolem `g` a zbytek světa symbolem `o`.

Speciálním případem je *sticky bit*, který se aplikuje na adresáře. Pokud je na adresáři nastaven *sticky bit*, může uživatel mazat pouze ty soubory, které v tomto adresáři vlastní. Toto je výhodné použít například na adresář `/tmp` nebo `/var/tmp`, kam mají vetšinou možnost zápisu všichni uživatelé. Sticky bit můžeme přidat příkazem `chmod` takto:

```
chmod +t /tmp
```

Adresář vybavený tímto bitem poznáme podle znaku `t` v místě nastavení práva provádění (`x`) pro ostatní.

Je nutné ještě zmínit, podle čeho se určuje výchozí nastavení práv souborů – tzn. pokud vytvoří uživatel nový soubor, jaká bude mít tento soubor přístupová práva. Toto se určuje příkazem `umask`, ke kterému přidáme parametr, který představuje přístupová práva v osmičkové soustavě.

Další informace lze nalézt v manuálových stránkách příkazu `chmod` (`man chmod`).

ACL

ACL (Access Control List) je komplexnější řešení přístupových práv v Linuxu (samozřejmě i na jiných operačních systémech). Umožňuje nám nastavit taková práva, která bychom s běžnými právy v Linuxu těžko realizovali nebo bychom to realizovali velmi složitě. Např. pokud budeme chtít přidělit souboru právo zápisu pro uživatele `lukas`, `karel` a `jachym` a ostatním uživatelům ze skupiny `studenti` přístup zakázat. Superuživatel by to zřejmě řešil tak, že by vytvořil další skupinu, kam by tyto tři uživatele umístil. Ale pokud nemáte práva superuživatele, tak nemůžete přidávat další skupiny, takže pro běžného uživatele je zvolení politiky přístupových práv neuskutečnitelné. ACL nezavádí žádné převratné změny v modelu přístupových práv, ale spíše rozšiřuje model základní.

Model ACL podle specifikace POSIX 1003.1E, která je pro unixové systémy typická, podobně jako u běžných linuxových práv, umožňuje definovat pro každého uživatele či skupinu tři druhy práv a to právo čtení, zápisu a spouštění – adekvátně u adresářů. Dalším atributem je maska, která symbolizuje údaj nejvyšších práv, která lze nastavit pomocí ACL a vztahuje se pouze na skupinu. Tato maska se transformuje na masku v klasickém modelu a je využívána aplikacemi, které ACL nepodporují, přičemž toto nemůže daný soubor zbavit práv, která mu byla přidělena. Dále jsou specifikována tzv. standardní ACL (angl. *default ACL*) práva, která se uplatňují při vytváření nového souboru nebo adresáře uvnitř adresáře, kde jsou tato práva definována. Samotná ACL práva podadresářů a podsouborů jsou zděděna po rodičovském adresáři. Podadresáře dědí i standardní ACL práva.

Podpora ACL pro souborové systémy Ext2fs a Ext3fs přišla s linuxovým kernelem 2.5.46, což je samozřejmě vývojové jádro. Stabilní jádra řady 2.6 podporu pro tyto souborové systémy ACL obsahují. Na jádra řady 2.4 je nutno použít patch, který lze sehnat na domovských stránkách projektu ACL[6]. Co se týče ostatních souborových systémů, tak ReiserFS a XFS je na linuxových jádrech řady 2.4 podporováno a na JFS je nutné použít patch[7]. Některé distribuce, které nepoužívají Vanilla jádra[8], mohou obsahovat podporu ACL i v řadě 2.4 – například SuSE či Mandrake.

K zobrazení ACL práv se používá utilita `getfacl`. Použití může být například takovéto:

```
getfacl /etc/file
```

Nebo pokud bychom chtěli vypsat standardní ACL práva adresáře, tak by zápis vypadal takto:

```
getfacl -d /etc/directory
```

Samotné nastavení a modifikace ACL práv se realizuje pomocí utility `setfacl`. Volba, kterou budeme zřejmě nejvíce používat je volba `-m`, která slouží k modifikaci ACL práv. Alternativou k této volbě je volba `-set`, která smaže současná ACL práva a vytvoří nová. Nesmíme zapomenout, že tato volba vyžaduje zadání práv pro vlastníka, skupinu i ostatní. Volbou `-x` lze ACL práva odstranit. Použití `setfacl` může vypadat například takto:

```
setfacl -m u:pepa:rw /etc/file
```

[6] http://acl.bestbits.at

[7] http://oss.software.ibm.com/developerworks/patch/download.php?id=409

[8] Jádro hlavní vývojové větve dostupné z http://www.kernel.org/.

Vidíme, že v našem případě má hodnota volby `-m` tři části. První část specifikuje, zdali se jedná o uživatele (`u` nebo `user`), skupinu (`g` nebo `group`), masku (`m` nebo `mask`) nebo ostatní (`o` nebo `others`). Druhá hodnota představuje UID v případě uživatele nebo GID v případě skupiny. V ostatních případech je pole prázdné. A poslední část charakterizuje samotná přístupová práva, jejichž symbolika je již více než jasná. Druhým parametrem utility `setfacl` je název souboru či adresáře.

Jedním z problémů je implementace ACL v NFS[9] klientech. Především u NFS klientů verze 2, kdy je rozhodováno u cachovaných souborů podle klasických unixových práv. Následky si každý dokáže představit. U NFS verze 3 je situace lepší. Kontrola práv probíhá na straně serveru pomocí tzv. ACCESS RPC volání, které ovšem nemusejí podporovat všechny implementace NFSv3. NFSv4 už s ACL umí pracovat, ale podpora pro Linux není úplná. Jak se dozvíme dále, je mnoho důvodů, proč NFS nepoužívat a toto je jistě jeden z nich. Situace u projektu Samba[10] je úplně opačná. ACL bylo v Sambě zabudováno dávno předtím, než se objevila implementace ACL v Linuxu. Proto lze Sambu bez jakýchkoliv potíží nasadit.

Další informace lze opět nalézt v příslušných manuálových stránkách.

2.1.3 Atributy souborů

Atributy souborů jsou úzce spjaty se samotnými právy souborů a rozšiřují základní rámec bezpečnosti souborů. Atributy souborů můžeme zobrazit příkazem `lsattr`. Každý atribut reprezentuje znak, podobně jako tomu bylo u samotných práv v Linuxu. V tabulce 2.1 jsou vypsány atributy, které mohou být změněny při použití souborového systému Ext2 či Ext3.

Změnu těchto atributů můžeme provést pomocí příkazu `chattr` například takto:

```
chattr +i /etc/passwd
```

Nebo vrátit do původního stavu takto:

```
chattr -i /etc/passwd
```

Je zřejmé, že přidání atributu se provádí znakem "+" a sundání atributu znakem "-".

[9] http://nfs.sourceforge.net/
[10] http://www.samba.org/

Znak	Význam
A	Neaktualizuje `atime` souboru.
a	Otevírá soubor pouze v přidávacím módu.
c	Soubor je na disku automaticky komprimován.
i	Soubor není možno nijak pozměnit, vymazat nebo přejmenovat.
d	Ochrání soubor proti vymazání čistícím programem.
s	Soubor je odstraněn bez možnosti navrácení obsahu.
S	V okamžiku modifikace souboru je soubor zapsán na disk.
u	Pokud je soubor odstraněn, jeho obsah je zachován.

Tabulka 2.1: Atributy souborů a jejich význam

Další informace lze nalézt v manuálových stránkách použitých příkazů.

EA

EA (Extended Attributes) rozšiřují klasický model atributů v Linuxu. Umožňují libovolně nadefinovat další atributy souboru – název atributu a hodnota atributu. Zobrazení těchto atributů se provádí příkazem `getfattr`. K vytvoření nového atributu můžeme použít příkaz `setfattr` například takto:

```
setfattr -n autor -v Josef /home/pepa/bak.tex
```

Podporu EA pro různé souborové systémy zajišťují patche, které lze nalézt například na URL:

```
http://acl.bestbits.at/download.html
```

2.2 Zapnutí zastíněných hesel

Důvody použití zastíněných hesel a to jak fungují jsem popsal v sekci "2.1.1 /etc/passwd". Rozhodně bychom měli podporu zastíněných hesel zapnout – resp. ji nevypínat. Ve většině distribucí se tak již děje implicitně při instalaci – v mé oblíbené distribuci Slackware Linux již od verze 3.4.

Určitě jako administrátoři v prostředí akademických sítí nechceme, aby uživatelé (převážně studenti) pokoušeli sílu našich hesel nebo hesel jiných uživatelů, přes které by poté mohli takřka anonymně provádět nekalou činnost.

2.3 Problém setuid a setgid programů

Setuid[11] jsou programy, které po spuštění uživatelem běží s právy vlastníka (v případě setgid s právy skupiny, která se k souboru vztahuje). Takovéto programy mohou být zvláště nebezpečné pokud je jejich vlastník superuživatel `root` (tzv. *root-setuid* nebo *root-setgid* programy) a navíc mohou být spouštěny kýmkoliv. Jedná se například o programy `/usr/bin/at`, `/usr/bin/passwd`, `/bin/ping`, `/bin/mount`, `/bin/umount`, `/usr/bin/chfn`, atd. Root-setuid programy (čili ty nejnebezpečnější) můžeme najít příkazem `find` například takto:

```
find / -perm -4000 -uid 0
```

Jak vidíme, mezi tyto programy patří soubory, které používáme velmi často. V těchto programech se v minulosti objevilo spoustu chyb, které často, i když v některých případech nepřímo, vedly k získání nejvyšších pravomocí. Příkladem může být chyba v programu `/usr/bin/man`, která vedla k získání práv skupiny `man`. Tyto soubory bychom měli omezit tak, aby je mohla spouštět jen omezená část uživatelů. Toto můžeme realizovat dvěma způsoby. První je ten, že vytvoříme skupinu, která bude mít jediná přístup k danému setuid programu. Druhům způsobem je použití programu `sudo`[12], což je komplexnější řešení než první způsob. Konfigurace `suda` může být zprostředkována pomocí souboru `/etc/sudoers`, který může vypadat například takto:

```
Host_Alias LOCAL = localhost
Cmnd_Alias MOUNT = /bin/mount /dev/hda1 /mnt/hda1
pepa LOCAL=NOPASSWD:MOUNT
```

Tato konfigurace umožní uživateli `pepa`, který je přihlášen z localhostu připojení diskového oddílu `/mnt/hda1` příkazem:

```
/bin/mount /dev/hda1 /mnt/hda1
```

a `sudo` nebude vyžadovat heslo. Pokud daný setuid (setgid) program nepotřebujeme, měli bychom ho odstranit. Pokud můžeme sejmout setuid bit, opět bychom tak měli učinit v rámci odstranění bezpečnostních rizik. Ale bohužel ve většině případů znamená tato akce nesprávnou funkčnost nebo úplnou nefunkčnost daného programu.

[11] Tyto programy mají v masce práv nastaven tzv. efektvní bit s.
[12] http://www.courtesan.com/sudo/

2.4 Špatně nastavená PATH

Někteří uživatelé, v horším případě i superuživatelé si zjednodušují práci tím, že si přidávají adresář "." do proměnné systémového prostředí `$PATH`. Výhody to má takové, že jim potom stačí psát pouze `mujprogram` místo `./mujprogram` či `sh mujprogram`. Co když někdo vytvoří zákeřný program s názvem běžně užívaného příkazu jako například `ls`, `who` nebo `ps`. Pokud je adresář "." v proměnné `$PATH` před adresářem, ve kterém se nachází program, jehož názvu chceme zneužít a uživatel napíše příkaz, jehož názvu zneužíváme - například `ls`, znamená to, že se spustí náš zákeřný program, který v případě spuštění uživatelem nebo superuživatelem může vést k přímé kompromitaci systému s nejvyššími právy a záleží jen na útočníkově fantazii, jak bude vypadat kód zákeřného programu - může posílat obsah souboru `/etc/passwd` na určitou e-mailovou adresu nebo může například vytvořit setuid shell v adresáři `/tmp` a umožnit sobě a ostatním uživatelům používaní shellu s právy superuživatele. Zákeřný kód může obsahovat pasáž, kde se vykoná původní záměr uživatele a ten nemusí vůbec nic poznat.

Je možno zneužít i situaci, kdy se "." nachází na konci proměnné `$PATH` a to například využitím názvu programu, který není v systému nainstalován a uživatel se ho pokouší spustit nebo využitím toho, že se uživatel někdy překlepne a napíše například místo příkazu `nice` příkaz `ncie`.

Obrana je jasná, nepoužívat adresář "." v proměnné `$PATH`! Superuživatel by měl také zajistit odstranění všech výskytů přidání tohoto adresáře do cesty především v souborech `/etc/profile`, `/etc/bashrc` a v domovských adresářích v souborech `.bashrc` a `.bash_profile`. Bezradný bude v hledání a odstraňování tohoto zavedení proměnné `$PATH` v binárních aplikacích.

2.5 Omezení uživatelů pomocí kvót a limitů

Někteří uživatelé se domnívají, že jsou v systému sami a může dojít například k tomu, že uživatel využije celý procesorový čas stroje a tím znemožní práci ostatním uživatelům. Samozřejmě k tomu může dojít i nechtěně, jako se stalo například mně. Zaplnil jsem celý diskový prostor v oddílu `/home` v důsledku nesprávného běhu mého programu. Tak nebo tak je potřeba těmto situacím předcházet a uvalit na uživatele limity a kvóty.

Pokud chceme zamezit právě tomu, aby někdo zabral velkou část diskového prostoru využijeme kvót. Abychom mohli kvóty používat musíme v souboru `/etc/fstab` přidat parametr `usrquota` či `grpquota` k oddílu, na kterém chceme kvóty používat. Dále na daném oddíle vytvoříme soubory `quota.user` a `quota.group` nejlépe s právy 600. To jestli je vše správně nastaveno si můžeme ověřit příkazem:

```
quotacheck -avug
```

Samotné zapnutí kvót se provádí příkazem `quotation` bez parametrů. Přidání kvóty například pro uživatele `pepa` může vypadat takto:

```
edquota -u pepa
```

Tento příkaz spustí textový editor, který máte určený systémovou proměnnou `$EDITOR` (v mém případě je to důvěrně známý editor `vim`), ve kterém vidíte informace se současnými kvótami, které můžete editovat. Nastavit můžete překročitelný (soft) limit, při kterém bude uživatel jen upozorněn a nepřekročitelný (hard) limit, který překročen být nemůže. Hodnota nula znamená, že uživatel nemá nastavenou kvótu. Pokud bychom chtěli nastavit stejné kvóty jako má uživatel `pepa` například pro uživatele `karel`, provedli bychom to takto:

```
edquota -p pepa karel
```

Dále si představíme příkaz `ulimit`, který nám umožňuje omezit kromě velikosti datového segmentu například množství procesorového času, počet otevřených souborů, velikost souborů core, počet spuštěných procesů, atd. Omezení lze provést záznamem do souboru `/etc/profile` a tím omezit všechny uživatele přihlašující se do systému. Další možností editování limitů je editace souboru `/etc/security/limits.conf`, kde jsou informace seřazené ve formátu:

```
domain type item value
```

`Domain` je uživatelské jméno nebo název skupiny, který po znaku "`@`" nebo znaku "`*`" znamená všechny uživatele uvedené skupiny. `Type` může nabývat hodnot `soft` nebo `hard`, které jsou totožné s významem u kvót. `Item` označuje zdroj, který chcete omezit – například `cpu`, `nproc`, `maxlogins`, atd. `Value` je pak hodnota příslušného omezení.

Jak jsem již zmínil, omezení pro uživatele by mělo být co možná nejstriktnější. Abychom zbytečně neomezovali důvěryhodné uživatele, je možné vytvořit více skupin s různými omezeními. Skupina, která by se mohla příznačně jmenovat `studenti`, by měla mít největší omezení, ale samozřejmě ne na úkor použitelnosti. Jako systémoví správci tím předejdeme mnoha problémům.

2.6 Omezení práv superuživatele

S právy superuživatele bychom měli spouštět opravdu jen procesy, které to vyžadují. Co byste měli přímo zakázat, je používání programu `telnet` a FTP klienta, aby někdo nemohl odchytit nezašifrované heslo superuživatele. Tento zákaz lze realizovat například přes TCP wrappers[13]. Našli bychom i další příklady restrikce práv superuživatele. Některé nám nabízí projekt Bastille[14].

2.6.1 /etc/securetty

Tento soubor umožňuje definovat, ze kterých TTY a VC (virtuální konsole) zařízení se může superuživatel přihlásit. Pokud nechcete, aby dané zařízení bylo funkční, zakomentujte příslušný řádek v tomto souboru. Program `/bin/login` bude akceptovat pouze nezakomentované řádky a k němu příslušná zařízení.

2.7 Vypnutí a odstranění nepotřebných služeb a softwaru

Každá služba představuje jakési pomyslné dveře do systému, které může vetřelec využít k průniku. Je velmi pravděpodobné, že po instalaci vaší distribuce najdete služby, které potřebovat nebudete. Může se jednat například o web server, FTP server či drobné služby jako ECHO a chargen. Vidíme, že se jedná především o síťové služby. Pokud danou službu nepotřebujete, měli byste ji vypnout. Pokud víte, že službu nikdy potřebovat nebudete, můžete ji vymazat a tím znemožnit spuštění nepřítelem, který by si třeba chtěl takto zajistit přístup do systému. Seznam naslouchajících procesů můžeme vypsat příkazem `netstat` nejlépe takto:

```
netstat -atuvp
```

Další možností je použití příkazu `ps`, který může vypsat všechny běžící procesy a následné důkladné prozkoumání každé z nich, zdali je potřebná, či nikoli.

To jestli daná služba bude spuštěna, či nikoli můžete ovládat pomocí spouštěcích skriptů (spouštěcí skripty unixového typu System V se nacházejí v adresáři `/etc/rc.d/`) a další především nativnější služby se ovládají pomocí `/etc/inetd.conf` (nebo `/etc/xinetd.d/*`). Vyšší bezpečnosti docílíme spuštěním minima služeb.

Níže uvedené služby jsou služby, které znamenají vysokou míru rizika, nedají se nijak účinně zabezpečit a na systému, který je označen jako bezpečný, by se rozhodně objevit neměly. Tyto programy již nejsou v současné době nutné a nebo našly své bezpečnější alternativy.

[13] Více o TCP wrappers v kapitole "4 Bezpečnost na úrovni sítě".
[14] Více o projektu Bastille v sekci "3.4 Bastille."

finger

Problém služby `finger` (resp. jeho stejnojmenného příkazu) je ten, že o uživateli prozrazuje příliš mnoho informací. Pomocí příkazu `finger` může cizí uživatel zjistit mnoho informací o uživatelích vzdáleného, cizího systému – všechny terminály z nichž je momentálně přihlášen, některé informace o elektronické poště, obsah záznamu GECOS v souboru `/etc/passwd`, přihlašovací cestu k adresáři, jaký shell tento uživatel používá a specializované soubory `.plan` a `.project`. Význam tohoto programu je již dávno pryč a proto bychom měli zajistit vypnutí (popř. úplné odstranění) tohoto programu, což se provede editací příslušného záznamu v `/etc/inetd.conf` (popř. editací skriptu démonu `xinetd` patřící službě `finger`).

telnet

`Telnet` slouží k vzdálenému přihlašování do systému. Problém je v tom, že heslo je přenášeno v textové podobě, nijak nechráněno zrakům těch, co mohou sledovat provoz kudy tato data procházejí. Další nevýhodou je to, že je zranitelný vůči útoku s "člověkem uprostřed"[15], což znamená že třetí osoba naruší integritu odeslaných a přijatých dat, mezi stroji, u kterých probíhá spojení. Tento program má však svojí náhradu a tou je protokol ssh[16], který komunikaci šifruje, takže zde není žádný důvod, proč `telnet` používat a ohrožovat tím vlastní bezpečnost.

R* služby

Mezi tzv. R* služby patří `rlogin`, `rsh`, `rcp` a `rexec`. `Rlogin` má stejný význam jako `telnet`, ale obsahuje ještě jednu vlastnost, která ještě více ohrožuje bezpečnost a tou je, že umožňuje činnost bez hesla. Řešení je stejné jako v případě služby `telnet` a tím je použití bezpečného shellu ssh.

Služby `rsh`, `rcp` a `rexec` slouží k vzdálenému vykonávání procesů. Zde je možností průniku několik – například nainstalování podvrženého souboru `.rhost`), ale nebudu je zde podrobné probírat, protože tyto programy by v současné době měly být nahrazeny jejich bezpečnějšími alternativami.

echo, chargen

Tyto služby slouží pro testování sítě a systému. Služba `echo` testuje propojení sítě tak, že opakuje zpět odeslaná data. `Chargen` (character generation) vypisuje posloupnost znaků a slouží tedy pro testování terminálových zařízení `tty`. Tyto služby jsou náchylné vůči útokům DoS. Útočník navíc může při zfalšování jeho zdrojové adresy (IP spoofing) paketovou smrští zahltit třetí systém.

[15] Více o útoku MITM v sekci "4.12 Síťové hrozby".
[16] Secure Shell – více o ssh v sekci "4.7.2 OpenSSH, SSH".

talk, ntalk, ytalk

Tyto démoni umožňují vzájemnou komunikaci dvou i více uživatelů na jednom stroji. Tyto služby jsou opět lokálně náchylné vůči útokům DoS a vzhledem k existenci moderních komunikačních protokolů jako ICQ, ICR, Jabber, apod. je zbytečné tyto služby povolovat.

rwhod

Démon `rwhod` umožňuje vzdálenému uživateli zjistit, kdo je v systému přihlášen. I když lze službu `rwhod` (k němu příslušný příkaz `rwho`) používat jen na místní síti, neměli bychom tohoto démona pouštět. Lze ho zneužít například k zjišťování jmen uživatelů.

rwalld

Tento démon umožňuje vzdálenému uživateli posílat zprávy, které budou vypisovány na obrazovkách všech uživatelů. Riziko je zřejmé a je tím je opět neodolnost vůči útoku odepření služby.

TFTP

TFTP (Trivial FTP) je protokol navržený před mnoha lety, kdy byla snaha vytvořit protokol podobný FTP pro zavádění systému přes síť u počítačů bez lokálního disku. To již dnes ztrácí význam, takže je zbytečné tento nebezpečný protokol protokol používat. Náhradou může být protokol FTP, který ovšem není v dnešní době také bezpečný. Proto se doporučuje používat program `sftp`, který je součástí balíku SSH2.

X window

Používaní X Window znamená opravdu obrovské bezpečnostní riziko. Když řeknu, že X Window mají přístup ke každému stisknutí kláves od všech uživatelů a k veškerému výstupu na obrazovku a běží jako root-setuid, tak je ohrožení zcela zřejmé. Na počítači, který slouží jako server, router či firewall by X Window rozhodně běžet neměla. Na pracovních stanicích jsou však většinou nutné. Abychom zamezili tomu, že se na nás vetřelec skrze X Window připojí, musíme omezit přístup na firewallu nebo spouštět X Window přes ssh.

Emulátory

Do této skupiny programů řadím především software VMware, Win4Lin, Wine, Winex, DOSemu, atd. Tyto programy umožňují provozovat cizí operační systém (nebo jen cizí program) na systému, kde emulátor běží (v našem případě tedy

GNU/Linux). K tomu, aby toto mohlo fungovat je potřeba, aby tyto programy emulovaly hardware, k čemuž potřebují být spuštěny pod superuživatelem. Což znamená, že náš systém nemůže být bezpečnější, než je spouštěný cizí operační systém, nemluvě o chybách v samotných emulovacích programech.

Znám případy, kdy bylo potřeba tyto programy na GNU/Linuxu používat i na serverové straně. Jednalo se především o nutnost použití proprietárních programů, pro které neexistovala náhrada nebo portace pro GNU/Linux. Ale s nárůstem programů vytvářených pro OS GNU/Linux se bez těchto programů většinou obejdeme.

2.7.1 Exploitování smazaného programu

Na tento problém jsem narazil v článku[17], kde je popisován problém, který řešila skupinka administrátorů. Začalo to tím, že jim někdo kompromitoval systém na práva superuživatele, na čemž není ještě nic divného. Podivnější však je už to, že útočník celou akci zopakoval po tom, co byl program aktualizován na nejnovější verzi a byla u něj nastavená přístupová práva pouze pro vlastníka a pro ostatní uživatele byla sejmuta. Co je však nejpodivnější, když správce systému použil program `/bin/rm` k odstranění programu a útočník opět systém úspěšně napadl skrze tento program. Jak je to možné? Problém je v příkazu `rm`, který neodstraní celkový obsah souboru na disku, ale pouze zruší odkaz a soubor pak není viditelný. Ke všemu útočník dávno předtím, než správce zjistil, že má aplikace bezpečností chybu, vytvořil pevný odkaz (angl. *hard link*), takže při upgradu kernel neuvolnil místo a link vedl opět k starému kódu aplikace. Celou reakci autora článku na tento problém naleznete například na stránce:

`http://cert.uni-stuttgart.de/archive/isn/2003/12/msg00056.html`

Prevencí je používání programů pro kontrolu integrity, které dokáží kontrolovat odkazy nebo umisťovat pro uživatele zapisovatelné adresáře na jiný oddíl a tím zabránit vytváření odkazů.

2.8 Udržování aktuálních verzí softwaru

Samozřejmě i v GNU/Linuxu se nacházejí chyby – jinak by tato práce snad ani nemohla vzniknout. Každou chvíli jsou nacházeny chyby v softwaru nebo samotných jádrech operačních systémů. Může se jednat o různě závažné chyby. Některé chyby mohou vést rovnou k získání nejvyšších oprávnění, některé například k získání práv skupiny `man`. Dále bychom mohli rozdělit chyby na lokálně a vzdáleně zneužitelné. Lokálně zneužitelné chyby jsou chyby, kterých lze zneužít

[17] http://www.hackinglinuxexposed.com/articles/20031214.html

pouze v případě, pokud máme na daném systému uživatelské konto. Vzdáleně zneužitelné chyby pak může zneužít prakticky kdokoliv vzdáleně.

Minimem, které by každý linuxový administrátor měl splnit, je pravidelná instalace bezpečnostních záplat (angl. *patches*). V případě větších distribucí jsou tyto záplaty většinou pravidelně umisťovány na jejich stránkách. Pokud máte nějaký software, který přímo neposkytuje vaše distribuce, je potřeba sledovat domovské stránky daného softwaru. Spoustu informací o aktuálních problémech lze také nalézt ve specializovaných konferencích a poštovních konferencích (mailing listech). Jednou z nejznámějších konferencí, kde je probírána problematika bezpečnosti, je konference *Bugtraq*[18].

Po aplikování záplaty není většinou potřeba restartovat systém, kromě aplikace některých jaderných záplat.

2.9 Editace bannerů

Bannery značíme zprávy, které se zobrazí jakémukoliv i cizímu uživateli při pokusu o přihlášení do systému. V GNU/Linuxu se jedná zejména o soubory `/etc/issue` a `/etc/issue.net`. V těchto souborech často prozrazujeme citlivé informace jako například používanou verzi linuxového jádra nebo co jsme vůbec za instituci. Toto samozřejmě může útočníka motivovat k napadení našeho systému – například pokud v těchto souborech upozorňujeme na to, že se jedná o akademický vědecký server – to už by mohlo někoho zajímat. Informace o tom, jaké jádro (nebo obecněji jaký operační systém) používáme, je pro útočníka cenná informace.

Samozřejmě bychom měli vhodným způsobem editovat bannery i u subsystémů – FTP server, SMTP server, atd. U FTP serverů se nacházejí bannery v souboru `~ftp/welcome.msg` nebo `~ftp/.message`. K přečtení bannerů může vetřelec jednoduše použít `telnet`, kterým se připojí k počítači na portu příslušící dané službě. Bannery jsou crackery často využívány k hromadným útokům na servery, kdy jsou pomocí skriptů či programů proskenovány široké rozsahy IP adres a jejich bannery jsou filtrovány podle výskytu určitého řetězce – například řetězec "Sendmail 8.12.10", podle kterého bychom mohli získat databázi strojů používajících právě tuto verzi programu Sendmail. Pokud je tato verze děravá, může cracker velmi efektivně napadat nalezené servery. Nebo si může vytvořit jakousi databázi a uchovávat v ní bannery softwaru, dávno předtím než se v něm objeví chyba a pokud se objeví chyba v určitém softwaru, může si jednoduše prohledat svojí databázi, jestli náhodou neobsahuje nějaký server s touto děravou verzí softwaru.

[18] Archiv konference přístupný z webu: http://www.securityfocus.com/archive/1

Obrana je jasná a to odstranit bannery nebo v nich neprozrazovat žádné konkrétní informace. Silnější povahy mohou modifikovat bannery tak, že se budou tvářit jako jiná verze softwaru či úplně jiný operační systém. Samozřejmě toto není stoprocentní ochrana, protože jsou jiné techniky, kterými lze skutečně používaný software poznat. Mám na mysli především metodu rozpoznávání tzv. otisků (angl. *fingerprints*). Osvědčený je například program Xprobe2[19] k rozpoznávání operačních systémů. K tomuto úkolu může dobře posloužit i důvěrně známý Nmap[20]. Samozřejmě můžete obelstít i tuto crackerovu metodu, protože existují programy, které dokáží tyto otisky simulovat[21].

2.10 Problém adresáře /lost+found

Distribuce Red Hat (či Fedora), Mandrake a Slackware a další vytvářejí v současných verzích adresář `/lost+found` s přístupovými právy 755. Do tohoto adresáře se ukládají případné ztracené soubory nalezené programem `fsck` po náhlém pádu systému. Souborový systém Ext2 totiž nemá žurnál, který této události dokáže předcházet. Problém je, že tomuto adresáři je povolen přístup pro všechny uživatele. Řešením je restrikce práv na mód 700 nebo používání žurnálovacích souborových systémů – například Ext3.

2.11 Volba bezpečných hesel

Pokud budete vy a vaši uživatelé používat slabá systémová hesla, velmi usnadníte útočníkovi přístup do vašeho systému, protože je to jedna z věcí, kterou crackeři zkoušejí nejdříve. Pokud se bude jednat o slabé heslo ke kontu superuživatele, je situace alarmující. Heslo by měla znát opravdu jen osoba (může být i více osob), která má oprávnění ho používat a samozřejmě by nemělo být lehce uhodnutelné. Cracker může vaše heslo podrobit slovníkovému útoku nebo útoku hrubou silou, kde většinou záleží na délce a složitosti (paletě použitých znaků) hesla, jak bude odolné – čím delší a složitější, tím déle bude trvat crackerovi prolomení tohoto hesla. Existují i další specializované útoky.

2.11.1 Kontrola hesel uživatelů

Jedním ze způsobů, jak kontrolovat to, zdali si uživatel zvolil bezpečné heslo, je ten, že se vžijeme do role crackera a pokusíme se hesla uživatelů prolomit. K tomu můžeme použít například program John the Ripper[22]. Samozřejmě záleží na

[19] http://www.sys-security.com/html/projects/X.html
[20] http://www.insecure.org/nmap/
[21] Tzv. honeypots – např. Honeyd – http://www.honeyd.org/
[22] http://www.openwall.com/john/

výpočetním výkonu počítače, který by útočník použil, ale určitý obrázek o stavu hesel uživatelů na našem systému si můžeme udělat. Jen bych chtěl upozornit, že k tomuto kroku byste si měli vyžádat písemný souhlas vašich nadřízených, abyste se sami nedostali do problémů.

Existují i moduly do programu `passwd`, které se snaží automaticky heslo uživatele prolomit.

Další možností je použít knihovnu `cracklib` z PAM[23]. Tato knihovna analyzuje hesla a snaží se je prolomit. Pokud chceme tuto možnost aktivovat, je potřeba do souboru `/etc/pam.d/passwd` napsat:

```
passwd password requisite /usr/lib/security/pam_cracklib.so retry=3
passwd password required /usr/lib/security/pam_pwdb.so use_authtok
```

Na mnou používaném Slackware Linuxu je ještě potřeba nainstalovat příslušný slovník a zapsat do souboru `/etc/login.defs`:

```
CRACKLIB_DICPATH /var/cache/ceacklib/cracklib_dict
```

V akademickém prostředí, kde se přirozeně vyskytuje mnoho uživatelů (které navíc o informační bezpečnosti nemusí nic vědět), je velmi pravděpodobné, že se naleznou uživatelé (nejen studenti), kteří budou mít lehce uhodnutelná nebo slabá hesla. Abychom zabránili z toho plynoucím nežádoucím důsledkům, je potřeba tyto jedince donutit k používání bezpečných hesel (nebo striktně slabá hesla nepovolovat) a také samotnému zacházení s tímto heslem, což by mělo být zmíněno v případné bezpečnostní politice dané sítě.

2.12 Příznaky připojování

Samotnému příkazu `mount` nebo v souboru `/etc/fstab` můžeme definovat příznaky (angl. *flags*) pro zvýšení bezpečnosti. Prvním příznakem je `nodev`, který řekne jádru, aby nerozpoznávalo žádné soubory zařízení. Toto je užitečné především u jednotek CD- či DVD- ROM nebo NFS. Dalším flagem je `noexec`, který zakáže na příslušném oddílu provádění spustitelných souborů, které vyvolává jádro. Příznak `nosuid` zapříčiní to, že nebudou akceptovány příznaky set-UID a set-GID. Příznakem `ro` se připojí daný oddíl pouze pro čtení.

[23] Více o PAM v sekci "3.1 PAM".

2.12.1 Připojování /boot s příznakem ro

Jak už jsem probíral v sekci "2.1 Správné nastavení přístupových práv", adresář `/boot` by měl mít přístupová práva jen pro čtení. Pokud máme tento adresář na zvláštním oddílu, což je relativně časté, můžeme elegantně využít příznak `ro` a docílit tak stejného efektu.

2.13 Odstranění starých účtů

Jak studenti (zaměstnanci) přicházejí a dostávají konta, tak také odcházejí a konta by jim měla být odstraňována. Čím více účtů v systému, tím větší je riziko, že se přes ně do systému někdo dostane. Proto bychom každé konto, které již v našem systému nemá co dělat, měli odstranit – včetně uživatelova domovského adresáře a všech souborů, jejichž je vlastníkem a dalších záznamů například v aplikacích pro elektronickou poštu.

2.14 /etc/aliases

Tento soubor umožňuje nastavit jiné uživatelské jméno u konta než u e-mailové adresy. Toto se hodí, pokud nechceme, aby se někdo jednoduše dozvěděl název konta na našem systému, u kterého by mohl například zkoušet uhodnout heslo. Obecně platí pravidlo, že čím méně informací o sobě (obecně o počítačích, používaných službách a softwaru, uživatelích, atd.) prozradíme, tím vyšší bezpečnost zajistíme. Samozřejmě toto platí pouze pokud se nachází poštovní server a uživatelská konta na jednom stroji.

V akademickém prostředí, kde se většinou volí jméno uživatelského podle skutečného jména studenta (a ostatních zaměstnanců), není těžké odhadnout uživatelské jméno podle skutečného jména uživatele. Je k tomu však potřeba znát formu tohoto uživatelského jména (např. `josef.k`, `josef.kadlec`, apod.). Vůbec nejlepší je použít formu, která by se nedala jednoduše odhadnout – to můžeme docílit přidáním náhodného čísla (např. `josef.kadlec638`, apod.). A abychom zbytečně neprozrazovali tuto formu nebo samotné uživatelské jméno, je vhodné použít soubor `/etc/aliases`.

2.15 /etc/mailcap

Jedná se o metasoubor voleb pošty a většina poštovních klientů ho používá. Obsahuje informace o zpracování jednotlivých datových typů ve zprávách. Problém je v tom, že náš pečlivě zkonfigurovaný soubor může každý uživatel potlačit vytvořením souboru `.mailcap`, což může být nežádoucí. Obrana může být

taková, že nainstalujeme náš `.mailcap` do domovských adresářů uživatelů a nastavíme jim příznak nezměnitelnosti příkazem `chattr +i /home/*/.mailcap`. Uživatel však může nastavit jiné umístění pomocí systémové proměnné `$MAILCAPS`. Takže bychom zase museli nastavit bit nezměnitelnosti i pro soubory příkazového interpretu. Řešením může být pravidelné, zautomatizované kontrolování těchto faktorů a jejich následné odstraňování. Nebo použít klienta, který soubor `/etc/mailcap` nepoužívá.

2.16 Bezpečné odstraňování souborů

Už jsme se o této problematice lehce zmínil v sekci "2.7.1 Exploitovani smazaného programu". Problém je v tom, že když v GNU/Linuxu (na souborovém systému Ext2 nebo Ext3) vymažete soubor, Linux pouze označí datové bloky, které soubor zabíral, jako volné, ale skutečný obsah na disku stále je, dokud ho nepřepíše nějaký jiný. To si můžeme ověřit příkazem `grep`, který prohledává neformátované diskové zařízení a hledá v blocích vámi nadefinovaný text.

Jedním ze způsobů, jak odstranit obsah souborů, které jsme již smazali, je zaplnění volné kapacity disku. To lze provést například příkazem:

```
cat /dev/urandom » velky_soubor
```

Tento příkaz začne plnit soubor `velky_soubor` náhodnými znaky ze zařízení `/dev/urandom` a tím dojde postupně k zaplnění celého oddílu. Je třeba si dávat pozor na limity, kterými můžeme mít určenou maximální velikost souboru. Velikost souboru také určuje souborový systém Ext2 a to na hodnotu 2 GB, takže v případě oddílu, který je větší než 2 GB, je potřeba vytvořit souborů s náhodnými znaky více, než dojde k úplnému zaplnění kapacity oddílu. Toto je ovšem zejména účinné, pokud tento soubor vytváříme s právy superuživatele. Jen bych chtěl upozornit, že zaplněním celé diskové kapacity, můžete ohrozit některé běžící procesy a proto by toto mělo být vykonávané, pokud zrovna není nikdo přihlášen v systému.

Kapitola 3

Pokročilý hardening GNU/Linuxu

Tato kapitola se věnuje pokročilejším technikám zabezpečování operačního systému GNU/Linux.

3.1 PAM

Myšlenka modulů PAM (Pluggable Authentication Modules) spočívá v tom, že místo toho, aby určitá aplikace četla soubor s hesly, požádá PAM, aby autorizaci provedl. PAM umožňují komplexnější autorizaci než jednoduché ověření hesla a hlavně celou operaci ověřování uživatele dělají univerzální a aplikace si tyto moduly mohou pouze volat a nemusejí si vytvářet své vlastní ověřovací metody, které by mohly být nekvalitní. Pokud aplikace potřebuje autorizovat uživatele, tak se nejprve snaží najít příslušnou funkci v konfiguračních souborech dané aplikace. Pokud tam žádné nejsou, použije se implicitní konfigurační soubor modulů PAM, kde aplikace dostane instrukce, jak má autorizaci provést. Aplikace se pak dozví, jestli byl uživatel úspěšně autorizován, či nikoliv.

Konfigurační soubory modulů PAM jsou umístěny implicitně v adresáři `/etc/pam.d`. Vyhodnocování v těchto souborech probíhá po řádcích – tzn. že na každém řádku probíhá určité kritérium ověřování uživatele. Každý soubor má následující tvar:

```
module_type control_flag module_path arguments
```

`Module_type` může být jedna ze čtyř hodnot. První je hodnota `Auth`, která přikazuje aplikaci, aby vyzvala uživatele k zadání hesla a uděluje práva uživatelů a skupin. Druhou hodnotou je `Account`, která provádí ověřování na základě jiných vlastností než je heslo a to například čas nebo místo (konsole, apod.). Další hod-

notou je Session, která určuje, jaká akce se má provést před a po přihlášení uživatele. A poslední modul Password, umožňuje uživateli změnu hesla.

Control_flag nám umožňuje určit, jak bude naloženo s úspěchem, či neúspěchem autorizace daného modulu. Může nabývat hodnoty Required, což znamená, že autorizace musí být provedena úspěšně. Druhou hodnotou je Requisite, která je podobná hodnotě Require s tím rozdílem, že při neúspěchu se další moduly nevolají a aplikace dostane zprávu o neúspěchu autorizace. Hodí se, pokud chceme ověřit určité faktory ještě před přihlášením uživatele. Další hodnotou je Sufficient, která v případě úspěchu a toho, že se v souboru nenacházejí další příznaky Required nebo Sufficient, vrátí úspěch. Posledním příznakem je Optional, který dovoluje i v případě neúspěchu kontrolovat další moduly.

Module_path určuje platnou cestu do adresáře modulu. Kompletní seznam modulů naleznete na adrese:

http://www.kernel.org/pub/linux/libs/pam

Arguments může u každého modulu nabývat jiných voleb, proto zde uvedu jen ty, které jsou pro všechny společné. Příznak Debug zasílá informace o ladění programu do logovacího systému. Volba no_warn zajistí, aby se aplikaci nepředala žádná varovná zpráva. Use_first_pass zabrání tomu, aby bylo heslo po uživateli požadováno dvakrát. Při volbě try_first_pass bude požadováno opětovné zadání hesla pouze v případě, že první heslo vrátí neúspěch. Use_mapped_pass zapříčiní, že se heslo předá z předchozího modulu do aktuálního modulu a použije se k vygenerování například šifrovacího klíče.

Již zmíněným implicitním souborem, který se načte pokud aplikace neobsahuje svůj konfigurační soubor pro PAM, je /etc/pam.d/other.

Pokud budeme chtít donutit uživatele, aby zadali bezpečné heslo o minimální délce 8 znaků, provedeme to úpravou souboru /etc/pam.d/passwd, který by mohl vypadat nějak takto:

```
auth required /lib/security/pam_stack.so service=system-auth
account required /lib/security/pam_stack.so service=system-auth
password required /lib/security/pam_cracklib.so retry=3D3 minlen=8
password sufficient /lib/security/pam_unix.so nullok use_authtok md5 shadow
password required /lib/security/pam_deny.so
```

Kromě běžné rutiny je důležitý především třetí řádek, kde je použita PAM knihovna cracklib k ověření prolomitelnosti hesla a také minimální délky hesla. Ovšem funguje tu tzv. kreditní systém. Tzn. že délka určená příznakem minlen je

do jisté míry dynamická. Pokud například použijete v hesle jako jeden ze znaků číslici nebo nealfanumerický znak, zkrátí se vám povinná minimální délka hesla o jeden znak, takže budete moci změnit heslo, i když jeho skutečná délka nebude dosahovat hodnoty určené příznakem `minlen`. Pokud uživatel nezadá správné heslo, nebude mu dovoleno toto heslo změnit. Pro úplnost bych měl ještě dodat, jak bude vypadat soubor `/etc/pam.d/system-auth`:

```
auth required /lib/security/pam_env.so
auth sufficient /lib/security/pam_unix.so likeauth nullok
auth required /lib/security/pam_deny.so
account required /lib/security/pam_unix.so
session required /lib/security/pam_limits.so
session required /lib/security/pam_unix.so
```

Je nutno upozornit, že superuživatel `root` může být vůči těmto omezením imunní. To že superuživatel může ledacos přepsat nebo obejít je taková vlastnost všech operačních systémů unixového typu.

Jsou k dispozici moduly pro autentizační metody jako například LDAP[1], SecurID[2], Kerberos[3], atd. PAM nám umožní použít různé autentizační metody pro různé služby, proto se stává velmi účinným autorizačním nástrojem, který lze použít při omezování uživatelů.

3.2 chroot

Chroot je zkratka pro "change root" a je tím myšleno to, že kořenový adresář je změněný nebo spíše posunutý. Pomocí této systémové funkce (resp. stejnojmenného příkazu) můžeme vytvořit uzavřené prostředí, ve kterém můžeme nechat běžet nebezpečné aplikace. Výsledek bude takový, že daná aplikace a nebo vetřelec, který se zmocní dané aplikace, uvidí pouze soubory v tomto uzavřeném prostředí (což jsou většinou pouze soubory nutné k chodu dané aplikace) a neohrozí tím celý systém, protože k tomu, aby se dostal do skutečného kořenového adresáře, by musel překonat samotný chroot – je známo několik způsobů, jak toto udělat. Tímto způsobem nemusíme omezovat jen aplikace, ale i samotné domovské adresáře uživatelů.

K vytvoření chroot prostředí je nutné si nejprve vytvořit adresář, kam zkopírujeme všechny potřebné knihovny a soubory k běhu dané aplikace, přičemž musíme zachovat strukturu adresářů – tzn. že v našem vytvořeném adresáři vytvoříme například adresář `/etc`, kam zkopírujeme potřebné soubory, které byly původně

[1] http://ldap-abook.sourceforge.net/
[2] http://www.unc.edu/ais/systems/security/securid.html
[3] http://web.mit.edu/kerberos/www/

v adresáři `/etc` skutečného kořenového adresáře. A pak je samozřejmě nutné upravit konfigurační soubory a spouštěcí soubory tak, aby pracovaly s novým prostředím.

Do tohoto prostředí by měly být umisťovány především problematické síťové služby jako například BIND nebo Apache. Uzavření domovského adresáře do prostředí chroot může být značně problematické, protože je potřeba zabránit připojení segmentu sdílené paměti vytvořený mimo chroot, připojení unixového soketu mimo chroot, ovládání a manipulace procesů mimo chroot, zvedání priority procesů v chrootu vzhledem k procesům vně chrootu, atd. Některé z uvedených problémů můžeme řešit pomocí IDS (Intrusion Detection System) jako například grsecurity[4] nebo LIDS[5].

Omezování pomocí syscallu `chroot()` není zas až tak běžné, protože konfigurace programů, které se běžně v chroot prostředí nepouštějí, nemusí být vůbec bezproblémová. Druhým problémem je to, že udělat chroot prostředí opravdu bezpečné není zase až tak lehké a v minulosti se našlo pár způsobů, jak se z prostředí chroot vymanit a na těchto chybách bylo založeno například exploitování FTP serveru wu-ftpd v2.4.2-beta18[6].

K vytváření chroot prostředí lze také využít nadstavbu a tou je Jail[7].

K zajištění vyšší bezpečnosti se jistě vyplatí nechat některé služby jako například web server či name server běžet v chroot prostředí, ale pro akademické sítě bych viděl největší výhodu v tom, že můžeme jednotlivé uživatele uzamknout do tohoto prostředí a tím jim vlastně zabránit v pohybu mimo jejich domovský adresář. Tito uživatelé jsou pak úplně izolováni od samotného systému a to jim brání v získávání informací o tomto systému – takže i v případné nekalé činnosti.

3.3 Openwall patch

Zkráceně řečeno, Openwall patche[8] by měly řešit jisté bezpečnostní problémy na úrovni jádra za účelem zvýšení bezpečnosti. Při aplikování záplaty a následné konfiguraci kompilace jádra se volby záplaty Openwall objeví v sekci "Security Options". V nynější době lze tento patch aplikovat na linuxová jádra řady 2.0, 2.2 a 2.4, přičemž aktuální stabilní jádro je již řady 2.6.

[4] http://www.grsecurity.net/
[5] http://www.lids.org/
[6] Exploit využívající tuto chybu: http://www.securityfocus.com/archive/1/12962
[7] http://www.jmcresearch.com/projects/jail/
[8] http://www.openwall.com/linux/

Záplata Openwall přidá vlastnosti jako například nespustitelné zásobníky (angl. *non-executable user stack area*), což zabrání spouštění kódu umístěného v zásobníku. Takový kód je většinou vytvářen crackery, kteří zneužívají chybu v přetečení vyrovnávací paměti. Další vlastností je omezení odkazů v adresáři `/tmp` – zabraňuje například uživatelům ve vytváření odkazů na soubory, jejichž nejsou vlastníkem a některá další omezení. Dalším omezením v adresáři `/tmp/` je zákaz vytváření pojmenovaných rour (angl. *pipes*), které mohou být využity k přesměrování dat mezi uživateli. Dále jsou upravena přístupová práva v `/proc` tak, že uživatelé nevidí procesy jiných uživatelů, pokud nejsou ve speciální skupině (tohoto lze také dosáhnout například pomocí již zmiňovaného grsecurity). A některá další bezpečnostní vylepšení spojená především se správou paměti – například odalokování paměti, která nesměřuje na žádný proces aj.

Samozřejmě toto vás neochrání na sto procent. Existují exploity, které v jistých situacích dokáží obejít ochrany vytvořené patchem Openwall. Co se týče přetečení haldy (angl. *heap overflow*), není schopen zabránit vůbec – vyhnu se přesnému vysvětlování těchto pojmů, což je součástí nízkoúrovňové bezpečnosti a tou se v této práci nezabývám.

Součástí této záplaty je i program `stacktest`, kterým si lze vyzkoušet, zdali je náš systém náchylný k přetečení zásobníku. Po úspěšném aplikování patche Openwall by jste měli po spuštění programu dostat hlášku "Segmentation fault" místo "Succeeded.", což značí, že k tomuto útoku náchylní jste.

Do této kapitoly můžeme zařadit i příbuzné projekty jako například Medusa DS9 Security Systems[9], LoMaC[10], SELinux[11], RSBAC[12] a také mnohem komplexnější programy jako LIDS[13], grsecurity[14] a další.

3.4 Bastille

Původně měla z projektu Bastille[15] vzniknout celá linuxová distribuce, která by měla jako hlavní prioritu vlastní bezpečnost. Ovšem to bylo časově náročnější, než autoři původně zamýšleli a tak radši vytvořili sadu modulů, které tenkrát měly upevnit bezpečnost nově nainstalované distribuce Red Hat. Tato utilita se však nyní dá použít nejen bezprostředně po instalaci, ale kdykoliv. A jsou již podporovány i jiné distribuce než je Red Hat, pro který byl tento program navržen

[9] http://medusa.fornax.sk/
[10] http://opensource.nailabs.com/lomac/
[11] http://www.nsa.gov/selinux/
[12] http://www.rsbac.de/
[13] http://www.lids.org/
[14] http://www.grsecurity.org/
[15] http://www.bastille-linux.org/

původně. Bastille lze aplikovat na Mandrake, Debian, SuSE nebo operační systémy unixového typu jako HP-UX či Mac OS X.

Tento program se ovládá sérií textových menu. Každé z těchto menu popisuje určitou situaci, kde hrozí potencionální bezpečnostní riziko a program se vás ptá, zdali si přejete tuto situaci zabezpečit. Tato menu se spouští skriptem `InteractiveBastille.pl`. Po projití této konfigurace se nastavení uloží do souboru `BackEnd.pl`. Tento soubor lze pak použít na jiném systému, kde chceme aplikovat stejné bezpečnostní úpravy. Stačí překopírovat starý `BackEnd.pl` a spustit skript `AutomatedBastille.pl`.

Bastille nabízí ochrany jako omezení root-setuid binárních souborů, které bude moci spouštět pouze superuživatel. Dále umožňuje nastavit platnost hesla na 180 dní. Nabídne vám několik možností přísnějších nastavení `umask`. Dále je umožněno omezit přihlašování superuživatele `root` – například ho donutit, aby se nemohl přihlašovat přímo z konzole a musel použít program su po přihlášení pod běžným uživatelem. Pomocí omezení práv zamezí používání R* služeb, pokud se na systému nacházejí. Zvláště užitečnou věcí je návrh omezení uživatelů v používání nástroje `cron` pomocí souborů `/etc/cron.allow` a `/etc/cron.deny`. Také umožňuje použít heslo v zavaděči systému (pokud máte fyzický přístup k počítači) – toto je záležitost především fyzické bezpečnosti. Pokud by totiž měl útočník přístup k vašemu počítači, mohl by nabootovat jiný operační systém například z CD nebo diskety a tím se dostat k vašim datům. Dalším bezpečnostním prvkem, týkající se fyzické bezpečnosti je ten, že Bastille umožňuje nadefinovat akci, která se provede po stisknuti kombinace kláves Ctrl+Alt+Del. Nadefinovat lze také heslo pro jednouživatelský režim. Dále umožňuje konfiguraci TCP wrappers[16]. Bastille vám také pomůže editovat bannery, o kterých sem zde již psal. Dále můžete vypnout `telnet`, zamezit uživatelům používat kompilátor, nastavit zabezpečení subsystémů jako web server, mail server či DNS server a mnohé další.

3.5 libsafe

Knihovna `libsafe`[17] ochrání náš systém proti přetečení zásobníku (angl. *buffer overflow*), ale ne na úrovni jádra, jako tomu je v případě záplaty Openwall. `Libsafe` funguje jako sdílená knihovna. Ještě před spuštěním daného programu se zkontrolují možná přetečení zásobníku ve funkcích jako `strcat()`, `getwd()`, `gets()`, `scanf()`, `sprintf()`, `vsprintf()`, atd. Výhoda je jistě v tom, že není potřeba znovu kompilovat kernel.

[16] Více o TCP wrappers v sekci "4.1 TCP wrappers."
[17] http://www.research.avayalabs.com/project/libsafe/

Podobně funguje i program StackGuard[18], kterým můžete zkontrolovat kompilovaný kód. Funguje to tak, že StackGuard vloží speciální identifikátor do zásobníku před spuštěním funkce. Po spuštění se zkontroluje integrita tohoto identifikátoru a pokud byl identifikátor přepsán, proces se ukončí. Nevýhodou však je, že aplikace, které chcete kontrolovat musíte kompilovat právě pomocí programu StackGuard, k čemuž potřebujete zdrojový kód, který nemusíte mít vždy dostupný.

3.6 Šifrované souborové systémy

Toto je první kapitola, která se týká problematiky šifrování, proto bychom si měli vysvětlit, co to je šifrování a proč vlastně šifrovat. Šifrování je prostředek přímo určený k řešení problémů spojených s bezpečností. Říká se, že i kdyby nás mělo šifrování stát navíc jednu jednotku strojového času, impulz internetového připojení nebo jinou protihodnotu, měli bychom šifrovat. Samozřejmě existují případy, kdy je šifrování vyloženě zbytečné, ale v dnešní době je možnost šifrovat tak dostupná, že přemýšlet o tom, jestli nešifrovat je takřka zbytečná. Věda, která se zabývá šifrováním – tj. tedy utajováním informací, se nazývá kryptografie. Luštěním šifer se zase zabývá kryptoanalýza. Obě tyto disciplíny se řadí pod pojem kryptologie.

Šifrované souborové systémy jsou určeny k ochraně dat na pevném disku. Pokud někdo získá fyzický nebo vzdálený přístup k datům, která budou tímto způsobem ochráněna, budou pro něj nečitelné. Tento způsob je zvláště vhodný pro šifrování většího množství dat. Další šifrovací mechanismy, probírané v kapitole "4 Bezpečnost na úrovni sítě" jako například GPG[19], by se pro tento účel příliš nehodily. Zašifrování a rozšifrování dat by oproti šifrovaným souborovým systémům trvalo dlouho (v závislosti na objemu dat, použité šifry a její síly). To by nám bránilo v pohodlném používání dat.

Tento mechanismus šifrování bude v akademickém prostředí účinný především na straně serveru nebo na strojích, které slouží jako skladiště dat. Lze zde chránit data nejrůznějšího typu – od nejrůznějších dokumentů až po některé systémové soubory, které chceme zvláště ochránit.

[18] http://immunix.org/stackguard.html
[19] Více o GPG v sekci "4.8 GnuPG".

3.6.1 Linux CryptoAPI

Linux CryptoAPI[20] je sada kernelových patchů, která implementuje šifrování dat na disku, ale i síťového přenosu v podobě IPSec[21]. V jádrech řady 2.6 je již součástí (konkrétně již v řadě 2.5, která je ovšem vývojová). Záplata (angl. *patch*), který obsahuje CryptoAPI (jakožto holou implementaci šifrovacích algoritmů) a další potřebné záplaty, se nazývá `patch-int`. Tato záplata obsahuje šifry jako AES, Blowfish, DES, GHOST, IDEA, MARS, RC5, RC6, Twofish, 3DES, atd. Dále také hashovací funkce jako například SHA či MD5. Ale také, pro naše účely velmi potřebnou, implementaci loopback zařízení (konkrétně se nám jedná především o utilitu `losetup`), která je součástí balíku `util-linux`. Bohužel i v případě jádra řady 2.6, kde je již CryptoAPI součástí, je nutné nainstalovat novější verzi nebo aplikovat záplatu na příslušnou verzi `util-linux`[22] (toto neplatí pouze pro jádra 2.4.21 a nižší). Při konfiguraci jádra v sekci "Block devices" je potřeba povolit příslušné položky pro loopback zařízení, které nám bude sloužit jako mezivrstva nad diskem, pomocí které budeme moci data šifrovat – po příslušném nastavení v sekci "Cryptography support (CryptoAPI)". Dále je potřeba povolit "ramdisk" a "initrd" (v té samé sekci jako jsme nastavili loopback zařízení), které použijeme jako úložiště dat při rozšifrování oddílu. V případě potřeby můžeme povolit podporu pro více souborových systémů.

Pokud tedy máme kernel 2.4.21 a nižší, je potřeba samotný Vanilla kernel, na který aplikujeme příslušnou záplatu `patch-int` - například příkazem:

```
bunzip2 -c ../patch-int-2.4.21.0.bz2 | patch -p1
```

Povolení příslušných položek v konfiguraci kernelu bude vypadat takto (výpis z menuconfigu):

```
Block devices  --->
  <*> Loopback device support
  <*> RAM disk support
  [*]   Initial RAM disk (initrd) support

Cryptography support (CryptoAPI)  --->
  <*> CryptoAPI support
  [*] Cipher Algorithms
  <*>  AES (aka Rijndael) cipher
  <*>  Loop Crypto support
    [*]   Loop IV hack
```

Poté už jen kernel zkompilujeme a po nabootování tohoto jádra můžeme

[20] http://ftp.kerneli.org/pub/linux/kernel/crypto/
[21] Více o IPSec v sekci "4.7.1".
[22] http://www.kernel.org/pub/linux/kernel/people/hvr/util-linux-patch-int/

používat podporu CryptoAPI.

U kernelu 2.4.22 je situace trochu odlišná. Je zde verze CryptoAPI z jader řady 2.6, na kterou je ovšem nutné použít novější verzi `util-linux`. Poté aplikujeme správnou verzi záplaty cryptoloop (například `cryptoloop-jari`). Povolení příslušných položek ve výpisu menuconfigu z konfigurace kernelu bude vypadat takto:

```
Block devices  --->
  <*> Loopback device support
    <*>  CryptoLoop support
  <*> RAM disk support
  [*]   Initial RAM disk (initrd) support

Cryptography support (CryptoAPI)  --->
  <*> CryptoAPI support
  [*] Cipher Algorithms
  <*>  AES (aka Rijndael) cipher
```

Následuje samotná kompilace kernelu.

U kernelů řady 2.6 je situace o poznání jednodušší, protože CryptoAPI je již jeho součástí. Jak již bylo řečeno, je opět potřeba použít novější verzi `util-linux`. Výpis menuconfigu by měl vypadat takto:

```
Block devices  --->
  <*> Loopback device support
    <*>  CryptoLoop support
  <*> RAM disk support
  [*]   Initial RAM disk (initrd) support

Cryptography support (CryptoAPI)  --->
  --- CryptoAPI support
  --- Cipher Algorithms
  <*>  AES (aka Rijndael) cipher
```

Jak si můžete všimnout, preferuji šifru AES-Rijndael (Advanced Encryption Standard). Vyznačuje se vysokou rychlostí šifrování a také samotnou sílou šifrovacího algoritmu.

Tímto máme zprovozněné CryptoAPI. Nyní předvedu základní použití tohoto mechanismu. Cílem bude vytvořit šifrovaný svazek na existujícím oddílu nebo vytvořit šifrovaný svazek ze samotného oddílu. Pomocí `dd` vytvoříme příslušně velký soubor na disku takto:

```
dd if=/dev/zero of=crypto_file count=100 bs=1M
```

Vytvořili jsme soubor `crypto_file` o velikosti 100 MB. Nyní použijeme utilitu `losetup` k aplikování šifrování na námi vytvořený soubor `crypto_file`:

```
losetup /dev/loop2 crypto_file -e aes256
```

Je vidět, že jsem vybral právě šifru AES - Rijndael s velikostí šifrovacího klíče 256 bitů. Nyní již stačí naše looback zařízení `/dev/loop2` upravit tak, abychom s ním mohli pracovat – chová se stejně jako jiné blokové zařízení. Vytvoříme na něm souborový systém:

```
mke2fs /dev/loop2
```

Vytvoříme například adresář `/mnt/crypto_dir`, do kterého loopback zařízení připojíme:

```
mkdir /mnt/mnt/crypto_dir; mount /dev/loop2 /mnt/crypto_dir
```

Místo vytvořeného souboru je vhodnější šifrovat celé diskové oddíly – klidně i oddíl swap. Šifrovat lze i celý kořenový svazek, ale většinou k uspokojení i vyšších nároků na bezpečnost stačí zašifrovat pouze některé svazky jako například `/var`, `/home` či swap oddíl. Připojovat šifrované svazky nemusíme jen pomocí `losetup`, ale i příslušným zápisem do souboru `/etc/fstab`. Pro náš příklad by jeden řádek, týkající se našeho šifrovaného oddílu, mohl vypadat takto:

```
/crypto_file /mnt/crypto_dir ext2 noauto,encryption=AES,keybits=256,
phash=SHA512 0 0
```

Další velmi zdařilý projekt, který slouží ke stejnému účelu se nazývá loop-AES[23]. Tato utilita slouží pouze k šifrování souborových systémů a používá k tomu šifru AES - Rijndael. Implementace pro procesory x86 je přímo v assembleru, takže je velmi rychlá. Přesný postup, jak pomocí loop-AES šifrovaný oddíl vytvořit naleznete v mém dvojčlánku Cryptoroot pro nejvyšší bezpečnost[24], který vás provede celým procesem krok za krokem. Ovšem s aktuálností linuxových jader řady 2.6 nutno zmínit, že se začíná od cryptoloopu, který nebyl úplně tak bezchybný, upouštet a nahrazuje ho dm-crypt[25].

Nakonec bych chtěl upozornit na to, že v souvislosti s šifrovanými souborovými systémy, je oblíbené používat tzv. tokeny. Jedná se o přenosné médium, na kterém je nějakým způsobem uloženo heslo (někdy také klíč), který po připojení do počítače zpřístupní šifrované svazky. Výhodou je to, že si nemusíte pamatovat

[23] http://loop-aes.sourceforge.net/
[24] http://www.root.cz/clanky/cryptoroot-pro-nejvyssi-bezpecnost-1/
[25] http://www.root.cz/clanky/zaostreno-na-dm-crypt/

heslo – v případě uložení klíče je výhodou to, že máte klíč stále u sebe. Výroba jednoho takového velmi levného tokenu byla představena Rodolfem Markem na akci *CryptoFest 2003*[26]. Protože samotná výroba takového tokenu se vymyká tématu této práce, odkázal bych na článek *Byli jste na CryptoFestu?*[27] a na samotný abstrakt[28], který se týká přednášky o výrobě tokenu. Jiné informační zdroje kromě video záznamu přednášky o tomto konkrétním tokenu k dispozici nejsou.

[26] http://www.cryptofest.cz/
[27] http://www.root.cz/clanek/1924
[28] http://www.cryptofest.cz/slajdy/ssl/Cryptofest.pdf

Kapitola 4

Bezpečnost na úrovni sítě

Toto téma je, co se týče bezpečnosti, velmi důležité, z určitých hledisek ještě důležitější než bezpečnost na úrovni OS, protože síť je ta frontová linie, kterou musí útočníci většinou překonat k dosahování dalších cílů. Asi nejdůležitějším a nejkomplexnějším prvkem zabezpečení sítí je firewall[1]. Ale abychom dobře porozuměli tomu, jak takový firewall správně nakonfigurovat, je potřeba vědět, jak se počítače připojují k jiným počítačům, jak si vyměňují data a v jaké podobě, apod. K tomuto všemu je potřeba alespoň dobře rozumět protokolu TCP/IP, který je nejrozšířenější a skrývá v sobě mnoho výhod – je založený na paketech, takže přes jedno síťové propojení může komunikovat více počítačů. V sítích TCP/IP může každý počítač iniciovat nebo přijmout síťová připojení nezávisle na ostatních počítačích. Umožňuje decentralizované řízení, takže pokud organizace dostane blok IP adres, může je dále distribuovat bez jakýchkoliv technických omezení. TCP/IP je nezávislý na přenosovém médiu – funguje na médiích jako Ethernet, Token Ring, ARCNet, FDDI, USB, sériové spoje, paralelní spoje, krátkovlnné radiokomunikační sítě (AX.25) a dalších. Také je směrovatelný, robustní a je otevřeným standardem (všechna dokumentace je volně dostupná), který si každý může implementovat zdarma. Bohužel, i když návrháři předpovídali jistý pokrok a navrhli TCP/IP s možností přidávání dalších vlastností, má tento protokol mnoho nedostatků – jak bezpečnostních, tak například již nedostačující 32 bitová adresace IPv4. Mezi další, ne příliš rozšířené, síťové protokoly patří například IPX/SPX či SNA.

Nebudu se zde zabývat celým TCP/IP protokolem úplně do podrobností, protože je to téma značně rozsáhlé, ale zaměřím se především na bezpečnostní aspekty. TCP/IP je rozdělena do vrstev, přičemž každá vrstva spoléhá na služby, které poskytuje vrstva pod ní i nad ní. Vrstvy popisuje tabulka 4.1.

Fyzickou vrstvou jsou myšleny ty nejzákladnější struktury a to fyzická zařízení, kterými se můžeme připojit do sítě. Propojení na fyzické vrstvě lze rozdělit

[1] Více o firewallech v sekci "4.2 Firewall".

<table>
<tr><td>Aplikační vrstva</td><td colspan="3" rowspan="3">Aplikace
FTP, SNMP, Ping, DNS,...</td></tr>
<tr><td>Prezentační vrstva</td></tr>
<tr><td>Relační vrstva</td></tr>
<tr><td>Transportní vrstva</td><td colspan="2">TCP</td><td>UDP</td></tr>
<tr><td>Síťová vrstva</td><td>ICMP</td><td>IP</td><td>IGMP</td></tr>
<tr><td>Linková vrstva</td><td colspan="3">Ovladače zařízení</td></tr>
<tr><td>Fyzická vrstva</td><td colspan="3">Síťový adaptér</td></tr>
</table>

Tabulka 4.1: TCP/IP vrstvy

do tří kategorií – vytáčené spojení, LAN (Local Area Network) a do další skupiny patří WAN (Wide Area Network) a MAN (Metropolitan Area Network). Vytáčené spojení je ohroženo především fyzicky – provoz klasického modemu i ISDN modemu lze vyvést do odposlouchávajícího počítače. Do skupiny projení LAN patří především hojně používaný Ethernet, dále pak Token Ring, FDDI, ARCNet apod. Slabina tohoto připojení je v tom, že může naslouchat všem komunikacím, které přes něj procházejí. Do poslední skupiny propojení WAN a MAN, kam patří především přenos mikrovlnami, radiovými vlnami či laserem. Zde je slabým místem ještě snazší odposlech, než u sítí LAN. V poslední době se začal hojně používat Ethernet 802.11 (rychlost 2 Mb/s), 802.11b (rychlost 11 Mb/s) a 802.11a (54 Mb/s). Viděl sem praktické použití tohoto standardu na VUT Brno[2], kde pokryli signálem část areálu školy, kde se mohli studenti se svými notebooky připojovat do místní sítě. Jedna z největších bezdrátových sítí v ČR CZFree.net[3] pokrývá velkou část Prahy a další lokality. Ovšem tato technologie má spousty bezpečnostních nedostatků – využití nativního šifrování WEP (Wired Equivalent Privacy) nemá prakticky smysl, protože bylo prolomeno. WEP by měla nahradit nová implementace šifrování WPA (Wi-Fi Protected Access) pro bezdrátové sítě, která by měla být mnohem odolnější. Ale bezdrátové sítě lze napadnout mnoha dalšími způsoby. První problém je v samotném fyzickém zabezpečení sítě. Je potřeba si dát pozor na to, aby nám signál zbytečně nepřesahoval plochu, kterou chceme pokrýt a dával tím možnost někomu cizímu pokoušet bezpečnost této sítě (například pokud má signál pokrývat jednu místnost, aby nezasahoval mimo budovu). Říká se, že současná bezpečnost bezdrátových sítí se rovná vystrčenému ethernetovému kabelu na ulici. Pro nejvyšší bezpečnost nestačí vypnout pouze DHCP, změnit implicitní identifikátor SSID, změnit přístupové heslo na přístupovém bodu (tzv. AP - Access Point), ale je potřeba použít pokročilejší metody

[2]Vysoké učení technické v Brně
[3]http://www.czfree.net/home/index.php

zabezpečení jako jsou VPN[4] – v případě bezdrátových sítí například IPSec[5] nebo CIPE[6]. V souvislosti s Wi-Fi je dnes aktuální chyba, kterou lze omezit (úplně odstavit) provoz fyzického zařízení vysíláním tzv. jam signalu (informace o kolizi v síti). Horší je to, že tomuto lze jen těžko zabránit, protože chyba vychází ze samotné podstaty Wi-Fi. Mezi další propojení WAN a MAN patří ještě DSL a kabelové modemy, které jsou zase náchylné na odposlech.

Pomocí linkové vrstvy komunikuje systém se samotným síťovým adaptérem. Používá k tomu podvrstvu MAC, se kterou manipuluje těsně předtím než přemění data na elektrické a optické signály. Nebudu zde rozebírat samotnou strukturu rámců Ethernet a propojení PPP (point-to-point). Jsou zde možné útoky DoS nebo změna zdrojové MAC adresy a budeme se jimi zabývat ještě v této kapitole.

Do síťové vrstvy patří samotné pakety a určuje tedy způsob, jakým budou data přenášena. Útoky na tyto vrstvy a další nadřazené vrstvy se budu zabývat v další části této kapitoly.

Zájemce, kteří by měli zájem o velmi podrobné informace o protokolu TCP/IP odkazuji na knihy z použité literatury číslo [3] a [4].

Důležité je též říct, že je zde velký rozdíl v zabezpečení serveru oproti pracovní stanici. Na nějaké internetové konferenci jsem kdysi narazil na jedno takové pořekadlo:

> A busstation is a place where busses stop.
> A trainstation is a place where trains stop.
> A workstation is a place...

Toto pořekadlo odlehčenou formou naráží na problém zabezpečení pracovních stanic, na kterých přirozeně běží spousta aplikací. Ovšem spousta těchto programů nemusí být dlouho aktualizovaná. Je pravdou, že pracovní stanice nemusejí být pro útočníky tolik zajímavé, protože oproti serverovým počítačům jsou často restartovány, vypínány, přenášeny (zejména notebooky, laptopy, atd.) a hlavně nejsou základním kamenem dané sítě, takže jejich odstavení většinou nemá na chod celé sítě vliv. Ale samozřejmě, pokud vetřelec dosáhne zvýšení privilegií nějaké takovéto stanici, může jí použít k narušení bezpečnosti celé sítě. Na pracovní stanici by neměly běžet žádné serverové aplikace a pro jistotu bychom měli striktně zakázat jakékoliv přihlašování klientských programů – to lze uskutečnit například pomocí TCP wrappers[7] či iptables[8].

[4] Více o VPN v sekci "4.4 VPN".
[5] Více o IPSec v sekci "4.4.1 IPSec".
[6] http://sites.inka.de/sites/bigred/devel/cipe.html
[7] Více o TCP wrappers v sekci "4.1 TCP wrappers".
[8] Více o iptables v sekci "4.2.2 iptables".

Zabezpečení serverů (tj. i firewallů či routerů) by mělo odpovídat politice, kterou jsem se zabýval v kapitole "3 Pokročilý hardening GNU/Linuxu" – spouštění jen opravdu potřebných služeb, jejich pravidelná aktualizace, atd.

4.1 TCP wrappers

TCP wrappers (balíček `tcp_wrapper`) tvoří vrstvu na úrovni neformátovaného TCP či UDP soketu, jež je spravován démonem `inetd` (nebo `xinetd`) a jednotlivými aplikacemi. Tato vrstva dokáže kontrolovat, který z uživatelů má přístup k dané službě. Tímto způsobem tedy dokážeme omezit jednotlivé uživatele nebo dokonce celé systémy v přístupu ke službám na našem systému. S použitím knihovny libwrap lze omezovat i služby, které nejsou spouštěny démonem `inetd` (nebo `xinetd`).

Ke konfiguraci TCP Wrappers slouží dva soubory a to `/etc/hosts.allow` a `/etc/hosts.deny`. Tyto soubory by měly mít přístupový mód 600, abychom zabránili tomu, že útočník získá přehled o tom, kterým systémům důvěřujeme a kterým ne. Využitím těchto informací by mohl nejprve získat kontrolu nad systémy, kterým důvěřujeme a přes tyto systémy kompromitovat nás.

Spouštění většiny služeb z `inetd` (nebo `xinetd`) zajišťuje `/usr/sbin/tcpd`, kterému se předá jako první parametr samotný název aplikace. Program `tcpd` si zjistí IP adresu a název klientského počítače. Dále se zjistí uživatelské jméno zaslané službou `ident` od počítače, jenž požaduje danou službu. Tato informace však nemusí být zaslána. Služba `tcpd` pak zkontroluje dva již zmíněné konfigurační soubory a podle získaných údajů rozhodne, jestli službu povolí, či nikoliv. V případě povolení služby se služba spustí. V případě zamítnutí se oznámí nevyhovění jeho požadavku a spojení se uzavře.

Záznamy v těchto souborech jsou rozděleny po řádcích a každý řádek je ještě oddělen znakem ":". První pole každého záznamu je seznam služeb, kterých se omezení týká. Hodnota `ALL` znamená všechny služby. Hodnota `ALL EXCEPT`, za kterou následuje název služby, znamená všechny služby s výjimka té specifikované. Druhé pole je seznam počítačů podle souboru `/etc/hosts`, služby DNS nebo samotná IP adresa. Nemusíme se omezovat pouze na jednotlivé počítače, ale lze zadávat celé domény jako například:

```
.cvut.cz
```

Jak již bylo řečeno, zadávat lze i IP adresy a rozsahy IP adres. Omezení adresního rozsahu třídy B může vypadat takto:

```
192.168.
```

nebo takto:

```
192.168.0.0/255.255.0.0
```

Opět lze použít hodnotu `ALL` pro libovolné hostitelské počítače. Také lze použít hodnotu `LOCAL`, která představuje lokální hostitelské počítače, jenž neobsahuje ve svém názvu tečku. Hodnotami `KNOWN` a `UNKNOWN` lze specifikovat počítače, jejichž názvy lze nebo nelze vyhledat pomocí DNS. Lze použít i zápis:

```
.cvut.cz EXCEPT lab1.cvut.cz
```

což znamená všechny počítače v doméně `cvut.cz` s výjimkou počítače `lab1` v té samé doméně. Do třetího, ovšem nepovinného, pole lze uvést příkaz nebo příkazy, které budou vykonány příkazovým interpretem. Za tímto polem příkazů musí následovat hodnota "allow" nebo "deny", podle toho, jestli chceme danou službu povolit nebo odepřít. Z toho také vyplývá, že můžeme soubory `/etc/hosts.allow` a `/etc/hosts.deny` sloučit do jednoho souboru. Pomocí této možnosti můžeme být upozorněny například e-mailem o neoprávněném pokusu použití dané služby.

Jestliže není nalezený záznam ani v jednom se souborů, služba ze nepovolí. Měli bychom upřednostňovat politiku "co není výslovně povoleno, je zakázáno." – tzn. že v souboru `/etc/hosts.deny` uvedeme hodnotu "`ALL: ALL`" a v souboru `/etc/hosts.allow` specifikujeme služby a počítače, které chceme povolit.

Pokud se někdo pokusí připojit na službu, ke které nemá povolení, zapíše to logovací systém (nejčastěji `syslogd`) do systémového protokolu.

4.2 Firewall

Pod výrazem firewall se může skrývat několik významů. Za prvé, že je to stroj oddělující nějakou síť či podsíť od ostatních nadřazených segmentů. Nebo aplikace, která může chránit byť jen pracovní stanici. V této souvislosti bych chtěl upozornit na to, že aplikování firewallu jako aplikace na stroje, které jsou chráněné už fyzickým firewallem, se velmi doporučuje. Značně to zvedne úroveň zabezpečení a vytvoří to další bariéru v případě, že by vetřelec získal kontrolu nad samotným firewallem.

Rozlišujeme dva základní druhy firewallů a to firewally filtrující pakety a aplikační proxy servery[9]. My se budeme zabývat především první skupinou a tím jsou firewally, které přijímají nebo odmítají pakety v závislosti na zdroji nebo cíli – na rozdíl od proxy serverů, které dokáží filtrovat i obsah. V GNU/Linuxu je filtrování paketu zabudováno do jádra. Pakety mohou projít skrze firewall jen tehdy pokud odpovídají sadě pravidel – mohou být filtrovány podle jejich typu, zdrojové a cílové IP adresy (možno filtrovat i podle MAC), zdrojového a cílového portu, atd. Pro zavedení paketového filtru se v dnešní době jader 2.4 a vyšších používá systém Netfilter[10], jehož stěžejní částí je program `iptables`, který nahradil starší program `ipchains`, používaný v jádrech řady 2.2. Ovšem z hlediska zpětné kompatibility lze používat `ipchains` i na jádrech 2.4 a vyšších. Abychom mohli využívat Netfilter v plné síle, je potřeba v jádru povolit tyto položky:

CONFIG_PACKET
CONFIG_NETFILTER
CONFIG_IP_NF_CONNTRACK
CONFIG_IP_NF_FTP
CONFIG_IP_NF_IPTABLES
CONFIG_IP_NF_MATCH_LIMIT
CONFIG_IP_NF_MATCH_MAC
CONFIG_IP_NF_MATCH_MARK
CONFIG_IP_NF_MATCH_MULTIPORT
CONFIG_IP_NF_MATCH_TOS
CONFIG_IP_NF_MATCH_STATE
CONFIG_IP_NF_MATCH_UNCLEAN
CONFIG_IP_NF_MATCH_OWNER
CONFIG_IP_NF_FILTER
CONFIG_IP_NF_TARGET_REJECT
CONFIG_IP_NF_TARGET_MIRROR
CONFIG_IP_NF_NAT
CONFIG_IP_NF_NAT_NEEDED
CONFIG_IP_NF_TARGET_MASQUERADE
CONFIG_IP_NF_TARGET_REDIRECT
CONFIG_IP_NF_NAT_FTP

Ve většině dominantních linuxových distribucí by však měla být tato podpora do kernelu zakompilována.

Mezi `ipchains` a `iptables` jsou však formální i neformální rozdíly. V `iptables` jsou jména řetězců psána velkými písmeny (například INPUT, OUTPUT, FORWARD, PREROUTING, apod.). Příznak `-i` označuje pouze vstupní rozhraní a `-o`

[9] Více o proxy v sekci "4.3 Aplikační proxy".
[10] http://www.netfilter.org/

výstupní rozhraní. Místo DENY se používá DROP. MASQ se změnil na MASQUERADE a má odlišnou syntaxi a našlo by se ještě pár dalších rozdílů. Mezi ty neformální rozdíly patři to, že `iptables` nekontroluje pouze adresy zdroje či cíle a jejich porty, ale ověřuje také, zdali komunikace splňuje správná pravidla komunikace – například u protokolu HTTP kontroluje, zdali jsou zasílány POST, GET, HEAD a další požadavky ve tvaru, který vyhovuje specifikaci HTTP. To nám zaručí, že komunikujeme na daném portu skutečně se službou, která na ní má běžet a ne s nějakou podvrženou službou, která může být spuštěna na stejném portu.

Důležité je říci, že pakety přijaté z neformátovaného soketu[11] (angl. *raw socket*) nejsou pomocí `iptables` ani `ipchains` filtrovány. Odchozí pakety však již filtrovány jsou.

4.2.1 NAT

NAT[12] umožňuje maskovat celou podsíť pod jednu IP – tzn. že počítače vnitřní sítě přistupují k vnější síti pod jednou IP adresou. Když počítač vnitřní sítě odešle požadavek počítači do vnější sítě, bude se vnějšímu systému jevit jakoby byl požadavek zaslán přímo z routeru[13]. Zdrojová IP adresa paketů z vnitřní sítě je na routeru přepisována a je jí přiřazeno číslo portu, které slouží jako dočasný zdroj pro router (dynamický NAT). Paket odpovědi zasílaný z vnější sítě do vnitřní sítě bude obsahovat paket s cílovou adresou routeru a číslo portu bude to, které bylo paketu přiděleno jako dočasný zdroj. Router podle toho dohledá systém ve vnitřní síti, kterému paket patří a nasměruje ho tam.

Pokud užijeme pro počítače ve vnitřní síti privátní rozsah IP adres, nedají se z vnější strany adresovat – toto se používá zejména v IP protokolu verze 4, kdy s masivním nárůstem počítačů a sítí připojených do Internetu, docházejí veřejné IP adresy. Tento problém opět souvisí se stářím tohoto protokolu. Problém řeší nový IP protokol verze 6, kde je adresní rozsah IP adres dostačující. Použití privátních adres přináší jednu obrovskou výhodu a tou je, že vetřelec nemůže zasílat pakety přímo počítačům ve vnitřní síti (za routerem) – pakety s privátními cílovými adresami se v Internetu nesměrují.

Privátní rozsahy síťových adres lze rozdělit podle RFC 1597 do třech tříd A, B a C. Třída A má rozsah 10.0.0.0-10.255.255.255. Třída B pak 172.16.0.0-172.31.255.255. A třída C pak 192.168.0.0-192.168.255.255. Podle rozsahu sítě zvolíme příslušnou třídu – pokud si nejsme jisti, jak veliká bude jednou naše síť, zvolíme třídu A.

[11] Neformátovaný soket je spojení mezi procesem, který jej otevřel a zařízením, k němuž je připojen.

[12] NAT upravuje norma RFC 1631

[13] Routerem neboli směrovačem se označuje stroj, který směruje pakety. Často je router a firewall jeden stroj.

Abychom mohli používat překlad adres v Linuxu, je potřeba zavést modul `ip_MASQUERADE` (příkaz `modprobe ip_MASQUERADE`).

Mezi nedostatky NATu patří především to, že ho nelze použít s některými protokoly, protože například vyžadují zpětné otevření kanálu, použití původní adresy IP, atd. Bez aplikování nějaké další úpravy (například přídavného modulu, který by daný protokol dělal skrze NAT funkční), nelze použít IPSec a PPTP, protože šifrované TCP hlavičky by musely být přístupné routeru. IPSec je založen na kontrole integrity přenesených dat, takže kdyby router přepsal hlavičku TCP paketu, mechanismus IPSec by tyto pakety na cílovém systému zahodil. Dalším problematickým protokolem může být FTP, které do TCP/IP paketů vkládá adresní informace, které dále využívají, ale po přepsání TCP hlavičky paketu už nejsou informace správné. Tento problém lze však vyřešit zavedením modulu `ip_conntrack_ftp` a `ip_nat_ftp`. Další problémy mohou být s protokolem H.323, který slouží pro videokonference, atd.

Samozřejmě zajištěním toho, že jsou počítače vnitřní sítě neviditelné, nám nezaručí úplnou bezpečnost těchto strojů vůči okolnímu světu. Takže nesmíme například zapomínat na to, že pokud je navázáno spojení, tak existuje zpětné připojení. Potencionálně zneužitelná chyba může spočívat ve vypršení lhůty ve stavové tabulce. Dokud časový limit nevyprší je spojení z počítače ve vnitřní síti přes router na počítač, se kterým probíhala komunikace, stále aktivní. Toto může být napadnutelné, pokud útočník zná IP adresu a port původního místa určení. Tyto informace mohl zjistit předchozím sledováním sítě. Samozřejmě se mohou vyskytnout další problémy a to i třeba přímo v implementaci NATu.

4.2.2 iptables

Nejdříve je nutné si vysvětlit, jak kernel třídí pakety a dělí je do tří základních řetězců (angl. *chains*) – INPUT, OUTPUT a FORWARD. Síťový adaptér přijme data ze sítě a kernel je zpracuje do formy paketu a určí jeho destinaci. Ta může být buď lokální, tj. pokud jsou pakety určeny pro tento počítač (řetězec INPUT), nebo aplikace odesílá pakety síťovým rozhraním směrem ven (řetězec OUTPUT), nebo pakety nejsou určeny pro tento počítač a náš počítač slouží pouze jako router, který směruje pakety dál (řetězec FORWARD). Pokud není zaplé směrování paketů a to implicitně není je potřeba ho zapnout, což pro IPv4 provedeme například příkazem:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Jinak budou pakety v řetězci FORWARD zahazovány. Samotný řetězec pak tvoří sekvenci pravidel, kterým paket prochází. Pokud paket vyhovuje určitému

pravidlu, je na něj aplikována akce uvedená u pravidla (například ACCEPT, DROP, REJECT či LOG). Některou z akcí však nemusí cesta paketu končit (například u LOG) a pokračovat v dalším procházení pravidel.

Cílem je vytvořit co nejtěsnější pravidla, aby nevznikaly žádné toky dat navíc. Proto aplikujeme metodu identickou s tou, kterou jsem použil u TCP wrappers a všechny řetězce zakážeme a povolíme pouze to, co potřebujeme ke správnému chodu. Samotná pravidla firewallu je výhodné napsat do skriptu, který se bude spouštět při startu systému. Restart systému totiž všechna pravidla `iptables` odstraní – to lze samozřejmě udělat i explicitně.

Nemá cenu, abych tu dopodrobna opisoval manuál k iptables. Proto nejprve řeknu několik obecných pravidel a následně rozeberu ukázkový příklad. Zdrojová nebo cílová adresa se zadává buď jako hostitelský název (angl. *hostname*) – například `localhost` či `poc01.fjfi.cvut.cz`. Druhým způsobem je jednotlivá IP adresa – například 127.0.0.1 či 212.21.2.1. Dalším způsobem je adresace s maskou – například 212.21.2.1/255.255.255.0 nebo adresace s počtem bitů masky 212.21.2.1/24. U specifikování pravidla, že jde o zdrojovou adresu, lze použít i negaci a to pomocí symbolu "!" – například příznak `-s ! 212.21.2.1` vyhovuje všem paketům, které nemají jako zdrojovou adresu 212.21.2.1. Pomocí příznaku `-p` nebo `-protocol` lze rozlišovat datagramy, které lze specifikovat buď číselně (podle `/etc/protocols` nebo slovně – například UDP, TCP, ICMP, atd.). Dále lze specifikovat vstupní (`-i` nebo `-in-interface`) a výstupní (`-o` nebo `-out-interface`) zařízení. Neexistující rozhraní budou ignorována do té doby, dokud se neobjeví.

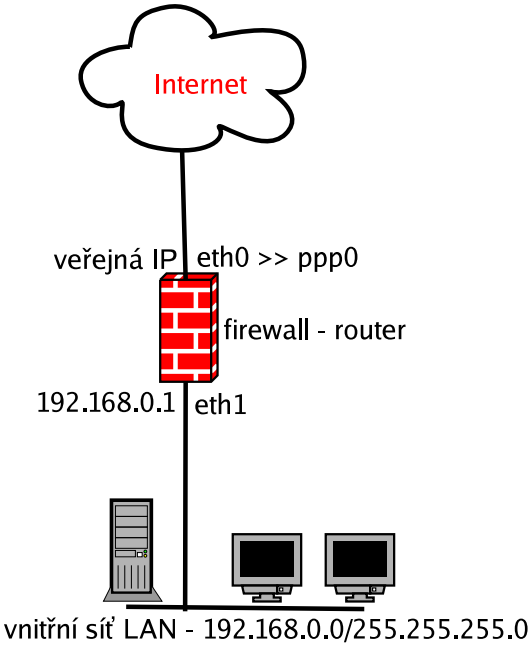
A teď proberu slíbený skript. Jedná se o skript, který chrání mojí síť připojenou technologií ADSL. Jedná se o síť kategorie SOHO[14]. Firewall má rozhraní `eth0`, které směřuje ven ze sítě (konkrétně do ADSL modemu, přes který se tuneluje spojení pomocí PPTP protokolu na stranu providera, takže firewall je vlastně přístupný pod rozhraním `ppp0`). Druhým rozhraním je `eth1`, který směřuje do privátní sítě. Topologii sítě znázorňuje obrázek 4.1.

Ve skriptu vynechávám příkazy `iptables`, které souvisí s dělením pásma a s určováním priority paketů (pomocí paketů s příznakem TOS) – toto téma je mimo rozsah této práce. Předpokládejme, že všechny potřebné moduly načítáme mimo tento skript (např. v souboru `/etc/rc.d/rc.local`).

```
#!/bin/sh
```

Rutina specifikující shellový interpret.

[14] small office/home office



Obrázek 4.1: Topologie sítě s tunelem PPTP

```
IPT="/usr/sbin/iptables"
```

Pro zjednodušení práce si vytvoříme proměnou IPT.

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Zapneme směrování (nebo přeposílání) paketů.

```
for interface in /proc/sys/net/ipv4/conf/*/rp_filter; do
echo "1" > $interface
done
```

Zapne rp_filter na všech dostupných rozhraních. Jedná se o ochranu před IP spoofingem[15] (podvržení zdrojové IP adresy). Jádro zablokuje všechny pakety se zdrojovou adresou, která by podle routovací tabulky měla přijít z jiného dostupného rozhraní. Na rozhraní `ppp0` (které směřuje ven do Internetu) by se

[15] Více o IP spoofingu v sekci 4.9.

neměly objevit adresy privátních rozsahů, které nejsou v Internetu směrovány. Totéž platí pro některé další rezervované adresy, které najdete na domovských stránkách koordinační skupiny IANA[16]. Tento seznam se však může měnit, proto dejte pozor, abyste zbytečně nezahazovali korektní pakety.

```
$IPT -F
```

Tímto příkazem vyresetujeme případná již zadaná pravidla ve všech řetězcích. Přepínačem `-t` lze specifikovat tabulku řetězců (například filter, nat či mangle), které chceme odstranit. Ještě dodám, že výpis současných pravidel se provádí přepínačem `-L`.

```
$IPT -P INPUT DROP
$IPT -P FORWARD DROP
$IPT -P OUTPUT ACCEPT
```

Základní politiku jsem si nastavil tak, že nejprve všechny pakety určené pro firewall (řetězec INPUT) a pakety, které jsou určeny pro počítače vnitřní sítě (řetězec FORWARD) zahazuji (akce DROP) a v dalších částech skriptu teprve povolím pouze ty pakety, které chci, aby došly na místo určení. Paketům, které generuje firewall (řetězec OUTPUT) důvěřuji – povoluji (akce ACCEPT) je všechny - pro zajištění vyšší bezpečnosti filtrujte i tento řetězec.

```
$IPT -A POSTROUTING -t nat -o ppp0 -j MASQUERADE
```

Akce MASQUERADE říká, že zdrojová adresa může být změněna a to pouze v řetězci POSTROUTING (směrované pakety i pakety, které odcházejí z počítače ven). Jinak také řečeno, tento příkaz zprovozňuje NAT. MASQUERADE se používá především tehdy, pokud máme k dispozici dynamickou adresu, což je můj případ, ale chtěl bych upozornit, že v případě ADSL je situace trochu složitější. I když je můj počítač přístupný přes veřejnou statickou IP adresu z Internetu, tak tato adresa není pro naše účely směrodatná. Musíme brát v úvahu IP adresu v rámci tunelovaného spojení přes PPTP, která je dynamická. V případě, že máme skutečně statickou IP adresu, můžeme použít akci SNAT.

```
$IPT -t nat -A PREROUTING -p tcp -i ppp0 -dport 2222 -j DNAT -to 192.168.0.3:22
```

Tento příkaz nám přesměruje port 2222 na port 22 na počítači s adresou

[16] http://www.iana.com/assignments/ipv4-address-space

192.168.0.4, který je součástí vnitřní sítě. Z /etc/services je zřejmé, že port 22 je SSH, o kterém se v této práci již zmiňoval. Tento příkaz nám tedy umožňuje přihlášení přes SSH na počítač s adresou 192.168.0.3. Stačí v klientu SSH nastavit adresu firewallu a port 2222. Výhoda je zřejmá. Pokud bychom se jinak chtěli dostat na počítač s adresou 192.168.0.3 z Internetu, museli bychom se přihlásit na firewall a až odsud bychom byli schopni se přihlásit na počítač vnitřní sítě – nehledě na to, že na firewallu by neměli mít konto žádní uživatelé. A firewall by mělo být synonymem pro bezpečnost se vším všudy – mít striktně nainstalováno opravdu jen to, co je potřeba, bezpečná hesla a další aspekty, které jsem již probíral. Nyní zpět k samotnému příkazu. Akce DNAT říká, že cílová adresa může být změněna. Tato akce je aplikovatelná na řetězec OUTPUT, který už známe a na řetězec PREROUTING (pakety určené pro tento stroj i pakety, které se mají směrovat). Další příznaky jsou více než zřejmé.

```
$IPT -N logdrop
$IPT -A logdrop -m limit -limit 5/h -limit-burst 3 -j LOG -log-prefix
"Rezervovana adresa: "
$IPT -A logdrop -j DROP
```

Zde vytvářím vlastní řetězec logdrop (příznak -N), který nám později ušetří práci, když budeme chtít nějaké pakety logovat. Řetězec logdrop definuji tak, aby se logovaly (akce LOG) pouze první 3 pakety a to maximálně pětkrát za hodinu. Tím zabráníme případnému zaplnění disku, který by mohl vést úmyslně, či neúmyslně k odepření služby. Příznakem -log-prefix definujeme textový řetězec, který se vloží před samotný záznam v paketu a tím ulehčuje práci při samotném hledání těchto záznamů v logách. Pakety se pak zahodí.

```
$IPT -N IN_FW
$IPT -A IN_FW -s 192.168.0.0/16 -j logdrop
$IPT -A IN_FW -s 10.0.0.0/8 -j logdrop
$IPT -A IN_FW -s 172.16.0.0/12 -j logdrop
```

Zde právě aplikujeme výše vytvořený řetězec logdrop, který nám zajistí bezpečné logování neplatných paketů v řetězci IN_FW, které používají zdrojovou adresu, která je z privátního rozsahu a v Internetu se nesměruje. Můžeme si zde nadefinovat i další rezervované adresy.

```
$IPT -N syn-flood
$IPT -A INPUT -i ppp0 -p tcp -syn -j syn-flood
$IPT -A syn-flood -m limit -limit 1/s -limit-burst 4 -j RETURN
```

```
$IPT -A syn-flood -j DROP
```

SYN flooding je jedna z metod útoků, která má za úkol vyvolat odepření služby tím, že posílá spoustu TCP segmentu s nastaveným SYN příznakem (angl. *SYN flag*). Nejdříve se vytvoří řetězec syn-flood, do kterého ukládáme pakety s příznakem SYN patřící do řetězce INPUT. Pokud bude takových paketů méně než čtyři za sekundu, vrátí se do řetězce INPUT.

```
$IPT -A FORWARD -p tcp ! -syn -m state -state NEW -j DROP
```

Nyní přejdeme k akcím s řetězcem FORWARD. Tento příkaz znamená, že se zahodí (akce DROP) TCP pakety patřící do řetězce FORWARD, které navazují nové spojení (datagram ze skupiny NEW) a nemají nastavený příznak SYN.

```
$IPT -A FORWARD -i ppp0 -j IN_FW
```

Logujeme a zahazujeme pakety, které patří do řetězce IN_FW, což jsou tedy rezervované adresy.

```
$IPT -A FORWARD -i ppp0 -o eth1 -p tcp -d 192.168.0.3 -dport ssh -j ACCEPT
```

Tímto povolíme přesměrování portu pro službu SSH, které jsem nadefinoval u řetězce INPUT.

```
$IPT -A FORWARD -i eth1 -j ACCEPT
```

Směrování z vnitřní sítě ven neomezuji.

```
$IPT -A FORWARD -i ppp0 -o eth1 -m state -state ESTABLISHED,RELATED
-j ACCEPT
```

Toto je velmi důležitá část, která povolí směrování paketů dovnitř sítě pro již vytvořená spojení. Když použijete nějakou aplikaci ve vnitřní síti, která komunikuje přes port, který je pro řetězec FORWARD zakázaný a neměli byste nastavené toto směrování paketů, nemohli byste takřka používat síťové aplikace z vnitřní sítě, protože odpovědi na požadavky vaší aplikace by byly zahozeny. ESTABLISHED a RELATED tedy charakterizuje datagramy, které jsou součástí již navázaného spojení nebo s ním nějak souvisí. INVALID pak značí datagram,

který není součástí žádného spojení nebo se jej nepodařilo identifikovat.

```
$IPT -A FORWARD -m limit -limit 12/h -limit-burst 3 -j LOG
-log-prefix "forward-drop: "
```

Ostatní pakety budou zahozeny. Budeme je logovat a to tři pakety, maximálně dvanáctkrát za hodinu.

```
$IPT -A INPUT -p tcp ! -syn -m state -state NEW -j DROP
```

Nyní se budu zabývat řetězcem INPUT. Příkaz je identický s tím, který jsem použil u řetězce FORWARD.

```
$IPT -A INPUT -i ppp0 -j IN_FW
```

Opět identické použití příkazu jako u řetězce FORWARD.

```
$IPT -A INPUT -i ppp0 -p tcp -syn -j syn-flood
```

Pomocí již výše vytvořeného řetězce odfiltrujeme pokusy o SYN flooding.

```
$IPT -A INPUT -i ppp0 -p icmp -j syn-flood
```

Zahlcení pakety ICMP je další ze způsobů, jak vyvolat DoS. Budeme se tímto útokem v této kapitole zabývat později. Tento příkaz opět využívá řetězec syn-flood, který ovšem nyní aplikujeme na pakety ICMP, které specifikujeme pomocí příznaku `-p`.

```
$IPT -A INPUT -i ppp0 -p TCP -dport 113 -m limit -limit 12/h -j LOG
$IPT -A INPUT -i ppp0 -p TCP -dport 113 -j REJECT -reject-with tcp-reset
```

Tímto způsobem filtruji službu AUTH (`ident`) – port 113. První řádek by měl být již jasný. Ve druhém řádku zahazuji pakety, které se pokouší použít tuto službu, akcí REJECT, která je identická s akcí DROP s tím rozdílem, že původce bude o tomto kroku informován zprávou ICMP. V našem případě se posílá paket s příznakem TCP RST, což zajišťuje hodnota tcp-reset u přepínače `-reject-with`. Službu `ident` právě není vhodné filtrovat pomocí DROP, protože tím může dojít k prodlevám při navazování některých spojení. Stává se především při zasílání e-mailů na vadné poštovní servery.

```
$IPT -A INPUT -i ppp0 -p ICMP -icmp-type echo-request -j ACCEPT
```

Tímto příkazem docílíme toho, že firewall z ICMP paketů propustí pouze ICMP s příznakem echo-request, což jsou pakety pro příkaz `ping`. Pokud bychom to neudělali, byl by firewall pro příkaz `ping` neviditelný. To by mohlo způsobit jisté administrační potíže, nebo nesprávný chod aplikací na něm závislých. Ale z hlediska bezpečnosti to má jistě výhodu. Mnoho skenerů pro hromadné testování bere právě odpověď na `ping`, jako příznak toho, že počítač je zapojen v síti a může ho podrobit hlubšímu testování. Pokud je tato informace pro útočníka směrodatná, nebude tento stroj podrobovat dalším testům, protože je pro něj obrazně řečeno neviditelný.

```
$IPT -A INPUT -i ppp0 -p TCP -dport 22 -j ACCEPT
```

Tímto příkazem povoluji připojování přes SSH na firewall. Změnou portu u příznaku `-dport` můžeme povolovat další služby. Seznam portů a k nim příslušné služby naleznete v souboru `/etc/services`.

```
$IPT -A INPUT -i lo -j ACCEPT
```

Loopback nijak neomezujeme. Může to vést k problémům.

```
$IPT -A INPUT -i eth1 -j ACCEPT
```

Toto je velmi volné pravidlo a znamená, že propustíme všechny pakety z vnitřní sítě a to i ty, které nejsou určeny pro firewall, což by bylo samozřejmě ideální, ale vzhledem k tomu, že mám dynamickou IP, musela by se tato situace řešit jinak. Pokud chceme tedy zavést pravidlo, aby se propouštěly jen ty pakety z vnitřní sítě, které jsou určeny pro firewall, musíme doplnit k tomuto pravidlu přepínač `-d` s hodnotou IP rozhraní směrujícího do vnitřní sítě a dále přidat ještě jedno identické pravidlo, kde bude mít přepínač `-d` hodnotu IP na rozhraní směřující do Internetu. Když doplníte ještě jeden identický příkaz přepínače `-d` s hodnotou adresace vnitřní sítě s počtem bitů masky, tak to znamená, že povolíme broadcasty z vnitřní sítě.

```
$IPT -A INPUT -m state -state ESTABLISHED,RELATED -j ACCEPT
```

Pakety od navázaných spojení povoluji – tímto jsem se již zabýval.

```
$IPT -A INPUT -m limit -limit 12/h -j LOG -log-prefix "INPUT drop: "
```

Všechny ostatní pakety zakazuji a loguji.

U řetězce OUTPUT nemám žádná pravidla a povoluji ho celý. To znamená, že plně důvěřuji paketům odeslaných z aplikací na firewallu. Toto není ovšem zcela správné. Správně bychom měli celý OUTPUT zakázat a povolit jen pakety s IP adresou rozhraní do vnitřní sítě, IP adresou rozhraní do Internetu a loopack zařízení (popř. povolit ještě DHCP broadcasty, pokud je potřebuje). Ostatní pakety by byly zahozeny, popř. ještě logovány. Já zvolil tuto volnější politiku opět z důvodu přidělování dynamické IP adresy.

Celý skript by ještě více získal na přehlednosti a zjednodušil manipulaci, pokud bychom rozhraní `ppp0` přiřadili například do proměnné `$INET_IFC` a `eth1` například do proměnné `$LAN_IFC`.

Více informací o `iptables` naleznete například na jeho manuálových stránkách.

Důležité je také ověřit, zdali námi nakonfigurovaný firewall pracuje tak, jak má a popřípadě ho doladit. K tomu nám mohou posloužit skenovací a odposlouchávací nástroje jako nmap[17], Nessus[18], tcpdump[19], Ethereal[20], dsniff[21], Snort[22] a mnoho dalších.

Abychom byli úplně důslední a to v případě bezpečnosti informačních systémů bychom být měli, neměli bychom dovolit to, že bude firewall v síti zapojen byť jen na malou chvíli bez kompletně zavedených firewallových pravidel. Tato situace může lehce nastat při samotné instalaci systému nebo i při restartu systému – zde by měla být pravidla zavedena dříve, než se aktivuje samotná síť.

Pokud byste si chtěli ulehčit práci s psaním skriptů pro `iptables`, existují frontendy, které se vám tuto práci snaží ulehčit. Já osobně je nepoužívám a moc

[17] http://www.insecure.org/nmap/
[18] http://www.nessus.org/
[19] http://www.tcpdump.org/
[20] http://www.ethereal.com/
[21] http://naughty.monkey.org/~dugsong/dsniff/
[22] http://www.snort.org/

jim nedůvěřuji (leckdy určitě neprávem). Jmenovitě se jedná například o MonMotha's Firewall[23], Ferm[24], AGT[25], Knetfilter[26], gShield[27], a další.

4.2.3 Adaptivní firewall

Myšlenka adaptivních firewallů je na světě asi od roku 1999. Adaptivní firewall se snaží na základě určitých pravidel jako například přístup na zakázané služby (porty) nebo u sofistikovanějších aplikací i pokusy o exploitování určitého síťového démonu, upravit pravidla paketového firewallu tak, aby znemožnila přístup útočníka na všechny služby stroje, který adaptivní firewall chrání a nakonec i samotného stroje, který slouží jako firewall. Jedná se vlastně o jakýsi prvek aktivní obrany. Takže například pokud se útočník bude snažit přistoupit na službu, na kterou nemá právo přistupovat, bude mu automaticky zakázán přístup i na všechny ostatní služby, ke kterým měl za normálních podmínek právo přístupu. Samozřejmě tento přístup má i své nevýhody a jednou z nich může být to, že zakážete přístup osobě, která je důvěryhodná a nestandardní situace vznikla pouze omylem. Ovšem praxe ukázala, že těchto případů je zanedbatelné procento ve srovnání s opravdovými případy napadení systému. Při studování problematiky adaptivních firewallů jsem zjistil, že tato oblast má co do činění s IDS (*Intrusion Detection System*), ze kterých se při jisté konfiguraci takový adaptivní firewall stává.

Takovýto firewall se skládá z části, která odliší pakety s nečistými úmysly a zjistí jejich zdrojovou adresu a druhá část, která modifikuje současná pravidla paketového filtru tak, aby útočník se zjištěnou zdrojovou adresou nemohl přistupovat k žádné ze služeb na strojích, které jsou pod naší ochranou. První část může tvořit samotná IDS aplikace jako například PortSentry nebo velice sofistikovaný Snort, která volá skript (jakožto druhou část), který se stará o modifikaci pravidel firewallu. Existuje implementace adaptivního firewallu s názvem Cracker Trap, kterou vytvořil sám Bob Toxen[28]. Tato aplikace dokáže spolupracovat s `iptables`, `ipchains` či TCP wrappers. Přes velké nasazení a úspěchy, které byly s tímto nástrojem dosaženy, již nedokáže rozpoznat mnoho útoků (jako například tzv. neviditelné skenování, které lze provést nástrojem nmap – tento způsob neodpovídá specifikaci TCP, protože je odesílán jen první paket TCP nebo paket s příznakem RST), které například Snort rozpoznat dokáže. Co se týče druhé části, která má za úkol modifikovat firewall a popř. nějakým způsobem upozornit na zjevně nekalou aktivitu na síti, je nástroj Cracker Trap vybaven víc než dobře. Modifikaci firewallu lze uskutečnit velmi jednoduše a to tak, že

[23] http://monmotha.mplug.org/firewall/index.php
[24] http://ferm.sourceforge.net/
[25] http://sourceforge.net/projects/agt
[26] http://expansa.sns.it/knetfilter/
[27] http://muse.linuxmafia.org/gshield/
[28] Mimo jiné jeden z vývojářů Berkeley Unixu a autor knihy [2].

v případě zachycení útoku, se přidá pravidlo, které zakazuje přístup paketů se zdrojovou IP adresou útočníka. To může vypadat například takto:

```
/sbin/iptables -I FORWARD 1 -s 212.23.32.23 -j DROP
/sbin/iptables -I FORWARD 1 -d 212.23.32.23 -j REJECT
/sbin/iptables -I INPUT 1 -s 212.23.32.23 -j DROP
/sbin/iptables -I OUTPUT 1 -d 212.23.32.23 -j REJECT
```

Z pohledu programátora na tom není vůbec nic složitého nebo komplikovaného. Jen poznamenám, že jsem použil přepínač `-I`, za kterým následuje jméno řetězce a za ním číslo pravidla, které udává pořadí vykonávání pravidel. V našem případě má číslo pravidla hodnotu 1, což znamená, že se vykoná v řetězci jako první. Cracker Trap (resp. skript `blockip`, který je součástí) dále odešle elektronickou poštou varování systémovým administrátorům a nebo také odešle varování na pager (v podmínkách České Republiky není příliš využitelné). Tyto zprávy jsou dále rozlišeny podle toho, zdali se jedná o nového útočníka nebo o někoho, kdo se již do našeho systému snažil dostat dříve. Dále dokáže na napadeném stroji nebo na jakémkoliv jiném spustit zvuk, který vás upozorní na právě probíhající útok. Také je uzpůsoben k využití technologie X10 FireCracker[29], pomocí které lze rozblikat světla nebo spustit houkačku. To zní sice velmi impozantně, ale v běžných podmínkách asi nebudete potřebovat spustit takovýto poplach, kvůli tomu, že někdo skenuje porty na vašem serveru.

Ideální by určitě bylo zkombinovat vlastnosti skriptu `blockip` a nástroj Snort, který teď blíže popíši. Jak už bylo řečeno, Snort je tzv. IDS (Intrusion Detection System). IDS můžeme rozdělit na HIDS (Host Intrusion Detection System), které odhalují nekalé aktivity přímo v systému – sem patří například již zmiňovaný LIDS či HostSentry. Druhou skupinou jsou NIDS (Network Intrusion Detection System), které odhalují průniky na základě aktivity na sítí – sem patří právě Snort, dále pak například Prelude[30]. Takový IDS funguje na základě určitých pravidel, které odrážejí jisté charakteristiky podezřelých aktivit – v případě Snortu je na jeho domovských stránkách[31] k dispozici databáze, která čítá přes 2200 pravidel aktualizovaná každou půlhodinu. Taková signatura pravidla, které by mělo zachytit pokus o zneužití chyby v DNS serveru BIND[32] ve verzi 8.2 a 8.2.1, může vypadat takto:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"DNS EXPLOIT named
8.2->8.2.1"; flow:to_server,established; content:"../../../";
reference:cve,CVE-1999-0833; reference:bugtraq,788; classtype:
attempted-admin; sid:258; rev:4;)
```

[29] http://www.x10.com
[30] http://www.prelude-ids.org/
[31] http://www.snort.org/
[32] Podrobnosti o chybě – http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0833 nebo http://www.securityfocus.com/bid/788

Toto pravidlo říká, že paket směřující z externí sítě (proměnná `$EXTERNAL_NET` se nastaví v konfiguračním souboru) z jakéhokoliv portu do naší sítě (proměnná `$HOME_NET`) na port 53, obsahující sekvenci znaků "`../../../`" bude klasifikován jako pokus o exploitování named serveru BIND a provede se akce alert. Signatura může obsahovat i další informace, které určují například příznaky paketů (příznak `flags`) nebo počet bajtů, kam se bude sekvence znaků prohledávat (příznak `depth`).

Pokud paket nebo sada paketů bude odpovídat nějakému z pravidel, provede se určitá akce – v naší tématice adaptivních firewallů to bude modifikace pravidel pro firewall (například pomocí skriptu `blockip` či skriptu `Guardian`, který je dodávaný se samotným Snortem). Ale obecně to mohou být i jiné akce jako zápis do syslogu, logování paketů ve formátu tcpdump, výstup do XML souboru, výstup do databáze nebo upozornění přes SNMP. Na tyto zprávy můžeme použít program logcheck, který prohledá logy z IDS a zašle zprávu elektronickou poštou na uvedenou e-mailovou adresu.

Mezi další, ovšem velmi mladý projekt stejného zaměření patří Magic Box[33], který si také rozumí se Snortem.

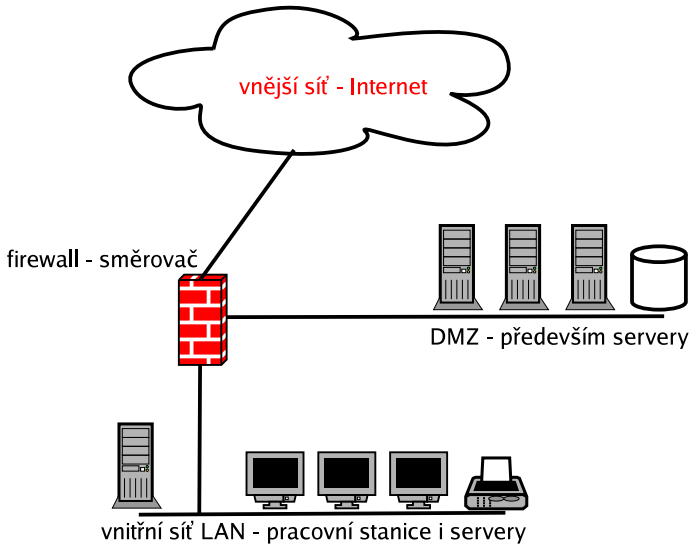
Vidíme, že i administrátor má k dispozici mocné nástroje na obranu proti crackerům a mohou proti nim aktivně bojovat a ne pouze přihlížet, jak jejich systém odolává nebo neodolává náporu vetřelců.

4.2.4 DMZ

DMZ je zkratkou pro "Demilitarized Zone" (česky pak "Demilitarizovaná zóna"). Jde o označení části sítě (i jiné informační části), která je oddělena od ostatních zařízení – v případě sítě se jedná o část, která je chráněna firewallem od vnější, tak vnitřní sítě. Můžeme si představit například firewall, který má tři síťová rozhraní – `ppp0`, které směřuje do vnější sítě (například do Internetu), `eth0`, které směřuje do vnitřní sítě a `eth1`, které směřuje právě do DMZ. Obecnou topologii takové sítě znázorňuje obrázek 4.2.

Toto schéma je v prostředí akademických sítí více než optimální, protože můžete efektivně chránit své servery od nebezpečí z vnější sítě, tak i vnitřní, kde jak už jsem řekl, z důvodu přívalu stále nových studentů, se skrývá mnoho potencionálních hrozeb. Pokud použijeme firewall na bázi paketového filtru (například `iptables`), vytvoříme si jednoduše tři skripty s pravidly, z nichž každý bude obhospodařovat svoje pakety podle zdrojového a cílového zařízení – v našem příkladě tedy jeden skript bude filtrovat pakety mezi rozhraním `ppp0` a `eth0`, druhý skript mezi `ppp0` a `eth1` a třetí mezi `eth0` a `eth1`. V DMZ by měly být umístěny

[33] http://security.tamu.edu/db.html



Obrázek 4.2: Topologie sítě s DMZ

servery, které mají být zpřístupněny z vnější sítě – na to se také váže přidělení veřejných IP adres, popř. přesměrování na routeru, které daný server vnější sítí zpřístupní.

Samozřejmě podobné ochrany můžeme docílit pomocí zapojení dvou či více firewallů. Možné je i vytvoření mnohem složitější varianty segmentace celé sítě.

4.3 Aplikační proxy

Proxy vznikla už v dávných dobách Internetu, kdy bylo webových stránek velmi málo. Byly především statické a propojení v sítích pomalé. Zde našla uplatnění právě proxy, která ukládala navštěvované stránky do vyrovnávací paměti a pokud někdo z vnitřní sítě požadoval danou stránku, mohla mu být načtena rovnou z proxy cache. Tímto se minimalizovalo zbytečné připojování na servery v Internetu a následné zbytečné stahování stránky, kterou již někdo požadoval. Ale jak se začal Internet dále široce rozšiřovat a vznikaly dynamické stránky (s dynamickým obsahem), vyplatila se tato funkce pouze ve velkých sítích (například u ISP (*Internet Service Provider*). Později se zjistilo, že použití proxy má další výhody a to především tu, že umožňuje skrýt všechny uživatele za jedno zařízení a následnou kontrolu obsahu nebo blokovat URL. Základní funkcí proxy je tedy založen na principu forwardování protokolů služeb – některé proxy servery zajistí i překlad adres nebo filtrování paketů, čímž už se ale spíše mění na firewall.

Příkladem takového firewallu může být *Checkpoint Firewall-1*, což je velmi mocný nástroj, ovšem také dost drahý. Nás budou zajímat především aplikace, které zajišťují výhradně funkci forwardování protokolů služeb a to zejména protokolu HTTP. Klientské počítače musí být pro využívání explicitní proxy serveru nakonfigurovány – v případě transparentních proxy si použití proxy zjistí samotné klientské aplikace, pokud tuto schopnost mají.

Jednou z klíčových funkcí HTTP proxy serverů je blokování URL. I když se moc nepřikláním k tomuto omezování uživatelů interní sítě, může se stát, že jako administrátoři budeme nuceni některé URL blokovat. Požadavky na danou URL jsou jednoduše porovnávány se seznamem zakázaných adres (tzv. *blacklist*) a daný požadavek směřující na zakázanou URL, nebude ani z proxy serveru vyslán. Ovšem toto lze někdy obejít zadáním URL ve formě IP adresy nebo celého čísla, které zjistíme z IP adresy podle vzorce:

$$A \times 2^{24} + B \times 2^{16} + C \times 2^{8} + D \tag{4.1}$$

IP adresa je zde reprezentována tvarem A.B.C.D. Takže například URL adresu portálu Seznam.cz lze do prohlížeče zadat ve tvarech:

```
http://www.seznam.cz
http://212.80.76.3
http://3562032131
```

Proxy servery také umožňují filtrovat samotný obsah. U HTTP proxy serverů je zajímavé především filtrování *ActiveX*, *Java apletů* nebo objemných obrázků a jiných multimediálních prvků. Kontrolovat lze i výskyt určitých znakových řetězců.

Proxy by měla také zajistit správnou formu posílaných dat a zamezit přeposílání zdeformovaných požadavků.

Další funkcí proxy serveru je to, že skrývá počítače ve vnitřní síti a všechna data protékají jedním bodem. U HTTP proxy serverů to neznamená nic jiného, než že počítač z vnitřní sítě vyšle požadavek a proxy server ho přijme, jakoby byl konečný webový server. Vygeneruje znovu svůj požadavek a pošle ho skutečnému webovému serveru, který odpoví zpátky proxy serveru. Ten už jen přepošle odpověď počítači ve vnitřní síti. S tímto také souvisí to, že proxy se hodí pro vytváření různých statistik o navštěvovaných stránkách nebo stažených datech.

Z předešlé výhody ovšem také vyplývá jedno riziko a to pokud z nějakých důvodů selže proxy server, znemožní to celé podsíti, kterou proxy zastřešovala, používání dané služby. Další nevýhodou je to, že pokud budeme chtít použít proxy na více služeb, budeme muset pro každou službu zavést svojí proxy. Pokud pro službu neexistuje proxy server, můžeme využít tzv. *SOCKS proxy*, kdy se připojení realizuje přes protokol TCP.

V akademických sítích najdou proxy servery (zejména HTTP proxy nebo FTP proxy) uplatnění při oddělování větších segmentů sítě a to především z důvodu kontroly nebo případného blokování určitých URL a šetření provozu pomocí proxy cache ve větších sítích. V menších sítích se vyplácí umisťovat proxy server přímo na hranici, kde se odděluje vnitřní síť od Internetu.

4.3.1 SQUID

Squid[34] je jedna z nejpoužívanějších proxy na bázi Open Source. Podporuje protokoly HTTP (i HTTPS), FTP, Gopher a další.

Konfigurační soubor Squidu `squid.conf` je standardně umístěn v `/etc/squid/`. Nejprve je potřeba modifikovat nepříjemné implicitní nastavení, které umožňuje komukoliv, aby se připojil na proxy server a využíval jeho služeb. Tohoto hojně využívají uživatelé s nekalými úmysly, kteří tímto mohou používat danou službu s adresou proxy serveru a tím skrývat svojí identitu před servery, na které se skrze proxy připojují. Tyto servery pak zpravidla vidí adresu proxy serveru. Správné chování je takové, aby se na proxy server mohli hlásit jen počítače z vnitřní sítě. Tohoto můžeme docílit samozřejmě zablokováním příslušného portu pomocí paketového firewallu. Toto omezení lze zprostředkovat i pomocí konfiguračního souboru `squid.conf`, ve kterém lze definovat sekvenci pravidel. Následující příklad povolí přístup ke Squid serveru pouze IP adresám 192.168.0.3 a 192.168.0.4.

```
acl OK 192.168.0.3 192.168.0.4
http_access allow OK
http_access deny all
```

Samozřejmě lze zadávat i celé podsítě – například 192.168.0.0/255.255.0.0 pro síť třídy B s adresami 192.168.*.*. Jednotlivá pravidla jsou procházena od shora dolů dokud se nenalezne první odpovídající pravidlo, které daný požadavek povoluje nebo zamítá.

Další implicitní položkou je port, na kterém Squid naslouchá. Je to port 3128. Pokud máte nastavená správná pravidla, která omezí přístup nežádoucím uživatelům, tak je zbytečné, abyste tento port měnili na nějaký jiný a tím se vyhnuli

[34] http://www.squid-cache.org/

skenerům, které hledají volné Squid proxy servery. Tento port se musí nastavit v aplikaci na klientské straně – například v internetovém prohlížeči v případě HTTP proxy. Samozřejmě musíme na firewallu zakázat HTTP port 80 (popřípadě ještě HTTPs port 443), aby uživatelé nemohli proxy server obejít přímým využitím protokolu HTTP nebo HTTPs a byli donuceni se připojit na port, na kterém naslouchá proxy server.

Dále je dobré upravit porty, ke kterým se bude Squid připojovat, protože jsou zde implicitně uvedeny kromě vybraných portů nižších jak 1023, také kompletně všechny porty nad 1024, což jsou dostatečně otevřené dveře pro různé typy trojských koní a dalším hrozbám. Upravená omezení v konfiguračním souboru by mohla vypadat takto:

```
acl Safe_ports port 80 21 443 563
http_access deny !Safe_ports
```

Tato konfigurace povolí Squidu připojovat se pouze na porty 80 (HTTP), 21 (FTP), 443 (HTTPs) a 563 (NNTP přes SSL).

4.4 VPN

Tato podkapitola by mohla být zařazena do kapitoly "3 Základní hardening GNU/Linuxu", kde se zabývám mimo jiné šifrováním, protože tématika VPN (Virtual private network) se šifrování úzce týká. Přesto se tato tématika týká především bezpečnosti na síti a proto je zařazena do této kapitoly.

VPN nechrání přímo síť (od toho tu jsou firewally a další prvky), ale chrání přenos po síti. VPN lze tedy přímo využít k bezpečnému propojení sítí LAN nebo jednotlivých počítačů, kde musí data procházet nedůvěryhodnými sítěmi jako například Internet. VPN je tedy levný způsob (oproti klasickým privátním sítím), jak rozšířit lokální sítě na vzdálené sítě a ke vzdáleným klientským počítačům přes Internet. Pakety jsou přenášeny v zašifrované podobě, takže je mezilehlé počítače nemohou číst – v některých implementacích (například IPSec) je zaručena i integrita dat. Samozřejmě lze touto cestou, byť nechtěně, přenášet viry a jiná zákeřná data (resp. pakety). Důležité je do naší sítě v rámci VPN nepovolit žádné spojení, které by mělo nižší zabezpečení než naše lokální síť.

Bezpečnost přenosu je zajištěna třemi základními prvky – zapouzdření IP, šifrovaná autentizace a šifrování datové části. Zapouzdření sítě stručně znamená, že skutečná IP je zapouzdřena do další IP adresy, protože tak lze zkontaktovat počítač v jiné síti, i když není možné realizovat přímé síťové spojení. Z pohledu síťových počítačů vzniká představa, že dvě vzdálené sítě jsou sítě oddělené pouze jedním směrovačem, přičemž ve skutečnosti takových směrovačů může být více.

Umožňuje také kontaktovat počítače v privátním adresním rozsahu, které se jinak v Internetu nesměrují. Popiši podrobněji na konkrétním příkladě, jak taková komunikace probíhá. Počítač s IP adresou 192.168.0.5 v síti 192.168.0.0/255.255.255.0 potřebuje poslat paket počítači 192.168.1.8 v síti 192.168.1.0/255.255.255.0. Nejprve vysílající počítač zjistí, že adresa sítě, ve které se cílový počítač nachází, se neshoduje s adresou jeho sítě. Vysílající počítač pošle paket na implicitní bránu jeho sítě, což je v našem případě počítač s adresou 192.168.0.1. Směrovač paket přečte a odešle do cílové sítě 192.168.1.0/255.255.255.0. Směrovač v cílové síti s adresou 192.168.1.1 paket přečte a odešle na cílový počítač 192.168.1.8, který paket přijme.

Šifrovanou autentizací se ověřuje totožnost vzdáleného uživatele, aby mohla VPN určit, zda je možno s uživatelem bezpečný tunel vytvořit. Tuto autentizaci je také možno využít k výměně privátních nebo veřejných klíčů pro šifrování datové části. Šifrovaná autentizace se provádí buď pomocí privátního klíče, kde se spoléhá na tajnou hodnotu, kterou obě strany znají. Informace, že uživatel tunelu tento klíč zná, dokládá důvěryhodnost tohoto uživatele druhému účastníkovi tunelu. Nevýhoda této komunikace spočívá v počáteční výměně klíčů. Neexistuje zcela bezpečný způsob, aby se obě strany dozvěděly najednou privátní klíč. Vždycky musí nejdříve proběhnout nějaká forma nešifrované komunikace. Výhodou je několika násobná rychlost oproti šifrování na principu veřejného klíče, kde se spoléhá na výměnu klíčů, kde veřejný klíč slouží jen na zašifrování dat a privátní klíč jen na rozšifrování dat.

Šifrování datové části se používá k prostému utajení obsahu vložených dat. Neutajuje ovšem informace z hlavičky, proto je možné zjistit podrobnosti o síti.

Ovšem pokud bychom srovnali klasické privátní sítě (například WAN) a VPN, můžeme říci, že VPN jsou sice levnější, dají se lehce implementovat, ale oproti WAN jsou pomalejší a méně spolehlivé.

V akademických sítích najdou VPN uplatnění především při propojování segmentů sítě, kde musí datový provoz procházet Internetem. Může se jednat o propojení různých pracovišť, fakult, apod. Samozřejmě lze pomocí VPN umožnit bezpečný přístup do interní sítě jednotlivým počítačům. Takže administrátor nebo jakýkoliv jiný zaměstnanec může s interní akademickou sítí bezpečně komunikovat.

4.4.1 IPSec

IPSec je rozšiřující protokol pro bezpečnou komunikaci přes IP a popsáno je především v RFC 2401[35]. Ostatní odkazy na RFC týkající se IPSec jsou k nalezení na URL:

`http://www.ietf.org/html.charters/ipsec-charter.html`

Šifrování v IPSec probíhá na úrovni IP protokolu, což znamená, že je pro aplikace a další síťové prvky zcela transparentní. To může být ovšem v některých případech nežádoucí – například pokud požadujeme navíc autentizaci uživatele v rámci nějaké aplikace. Toto však lze vyřešit jinými prostředky typu SSL.

Předem je nutné říci, že IPSec má cenu instalovat pouze na IPv4, protože v novějším IPv6 je již jeho povinnou součástí. Základními kameny IPSec jsou rozšiřující mechanismy AH (Authentication Header) popsané v RFC 2402[36] a ESP (Encapsulated Security Payload) popsané v RFC 2406[37]. První základní část AH poskytuje prostředky pro autentizaci a ESP k tomu včetně autentizace přidává ještě šifrování. Obě části jsou vzájemně provázány a mohou být použity společně nebo nezávisle na sobě. Tyto robustní protokoly jsou doplněny mechanismem IKE (Internet Key Exchange) popsaném v RFC 2409[38], který má za úkol zajistit výměnu klíčů. Prvním krokem je vytvoření SA (Security Association). Aby spolu mohly bezpečně komunikovat dvě strany, je zapotřebí dvou SA. SA obsahuje atributy jako cílovou IP adresu, IPSec protokol (AH nebo ESP), SPI[39], mód přenosu (tunel nebo transportní režim), a další. Již zmiňované SPI (Security Parametr Index) je číslo, které jednoznačně identifikuje daný SA. IPSec může fungovat v jednom ze dvou módů a to buď v transportním režimu, kde nedochází k zapouzdřování paketů a funguje tedy stejně jako běžný IP přenos s tím rozdílem, že se autentizují hlavičky pomocí AH a obsah se šifruje pomocí ESP. Druhým módem je tzv. tunel, kde již k zapouzdření paketů do AH nebo ESP dochází. Transportní mód tedy slouží k vytvoření autentizované komunikace přes veřejné intervaly IP mezi hostitelskými počítači. Tunel slouží především k vytvoření bezpečného spojení mezi koncovými body sítě (směrovači), čímž vznikne bezpečné propojení dvou sítí. Důležité je upozornit na to, že nativně nelze provozovat IPSec přes NAT, protože NAT mění informace v hlavičce, což je pro IPSec nepřípustné. Více o tomto problému se lze dočíst na URL:

`http://www.cisco.com/warp/public/759/ipj_3-4/ipj_3-4_nat.html`

[35] http://www.ietf.org/rfc/rfc2401.txt
[36] http://www.ietf.org/rfc/rfc2402.txt
[37] http://www.ietf.org/rfc/rfc2406.txt
[38] http://www.ietf.org/rfc/rfc2409.txt
[39] Podrobněji o SPI dále.

Nyní popíši na konkrétním příkladě propojení dvou sítí pomocí IPSec. Adresa první sítě je 192.168.0.0/255.255.255.0 s implicitní privátní IP adresou směrovače. Veřejná IP adresa tohoto směrovače je 212.123.22.4. Druhá síť má adresu 192.168.1.0/255.255.255.0 opět s implicitní IP adresou směrovače. Veřejná IP adresa tohoto směrovače je 147.34.32.5. Vysílající počítač zjistí, že cílová adresa se neshoduje s jeho síťovou adresou a pošle paket na implicitní adresu pro bránu sítě – v našem případě tedy 192.168.0.1. Směrovač IPSec v síti 192.168.0.0/255.255.255.0 paket přečte a určí, že paket se má odeslat do sítě 192.168.1.0/255.255.255.0, pro niž má nadefinovanou SA, která přes Internet odkazuje na směrovač 147.34.32.5. Směrovač IPSec pošle na vzdálený směrovač požadavek o domluvu IKE, autentizuje se a domluví se na sadě šifrovacích a autentizačních klíčů. Směrovač s veřejnou IP adresou 212.123.22.4 zašifruje paket, zapouzdří ho a pošle směrovači s veřejnou IP adresou 147.34.32.5. Cílový směrovač paket přijme a dešifruje. Úspěšné dešifrování paketu dokazuje platnost klíčů. Ten samý směrovač pošle paket na počítač v privátní síti, který ho přijme.

IPSec nemusí sloužit pouze k vytváření VPN, ale také například k náhradě šifrování WEP a jehož slabinách jsem se zmiňoval v úvodu kapitoly "4 Bezpečnost na úrovni sítě".

V GNU/Linuxu je k dispozici hned několik implementací IPSec. Jednou z nejkvalitnějších implementací je projekt FreeS/WAN[40], který ovšem na začátku března 2004 skončil[41] a zanedlouho se objevil projekt, který ve vývoji pokračuje a je jím projekt Openswan[42]. Další implementací je patch `patch-int`, který je součástí CryptoAPI[43]. CryptoAPI je již součástí jádra řady 2.6. V jádrech počínaje verzí 2.5.47 (takže i jádra řady 2.6) se nachází nativní implementace IPSec, která se inspirovala projektem KAME[44]. Ovládací nástroje pro tuto implementaci lze najít na URL:

`http://sourceforge.net/projects/ipsec-tools`

IPSec je složitý mechanismus, což zvyšuje pravděpodobnost výskytu chyb. Může se jednat například o nekompatibilitu různých implementací IPSec.

[40] http://www.freeswan.org/
[41] http://www.freeswan.org/ending_letter.html
[42] http://www.openswan.org/
[43] http://ftp.kerneli.org/pub/linux/kernel/crypto/
[44] http://www.kame.net/

4.4.2 OpenSSH, SSH

V prvé řadě je důležité si vyjasnit pojmy a vysvětlit rozdíl mezi SSH1, SSH2[45] a OpenSSH[46]. SSH či ssh je zkratkou pro "Secure Shell". Úplně původní SSH1 obsahuje programy `slogin`, `scp` a `ssh` jako náhrada utilit `rlogin`, `rcp` a `rsh`, které jsem probíral v kapitole "2 Základní hardening GNU/Linuxu". Licence této verze požaduje poplatek za používání pro zisk. SSH2 rozšiřuje verzi 1. Obsahuje další utilitu a to `sftp`, což je jakési bezpečné FTP, což je protokol sloužící k přenosu souborů. Programy SSH2 nejsou kompatibilní s SSH1, ale v případě potřeby jsou schopny zavolat starší verzi. Licence u této verze byla dost matoucí a ke změně licence došlo až po tom, co se objevilo OpenSSH. Nyní je tedy SSH2 volně dostupné. OpenSSH bylo vystavěno na starších verzích SSH1, než se licence této verze stala omezující. OpenSSH podporuje oba protokoly ssh. Pokud se podíváte ve vaší linuxové distribuci na verzi vašeho ssh klienta a serveru, například příkazem `ssh -V`, budete vlastnit s největší pravděpodobností právě OpenSSH.

Začnu obecněji. Jak probíhá takové ssh spojení. Ssh server naslouchá standardně na portu 22, takže se nejdříve naváže spojení na tento port. Server o sobě prozradí, kterou verzi SSH používá. To samé v vzápětí udělá klient. Pokud některá ze stran zjistí, že by si spolu nerozuměli, ukončí se spojení. Autentizace na serveru probíhá tak, že nejdříve server pošle svůj hostkey (veřejný klíč) a některé další informace. Dále server pošle svůj serverkey, což je klíč generovaný v pravidelných intervalech. Klient vygeneruje sessionkey a zašifruje ho oběma klíči na serveru. Klient vrátí zašifrovaný sessionkey s dalšími informacemi (například jaký bude použit šifrovací algoritmus). Poté už mohou obě strany používat šifrovanou komunikaci. Samotná autentizace se provádí pomocí RSA nebo v případě SSH2 pomocí DSA.

Existují i jiné metody autentizace jako například autentizace pomocí systému Kerberos[47], autentizace pomocí `~/.rhosts` či `~/.shosts` a další. Autentizace RSA probíhá na již zmiňované bázi veřejného a privátního klíče – privátní (soukromý) klíč má klient u sebe a veřejný klíč nahrává na servery, proti kterým se chce autentizovat. Klient tedy nabídne serveru formu autentizace a shodnou se například na RSA (DSA probíhá obdobně). Klient pošle svůj veřejný klíč a server ho porovná s klíčem, který již má a spojení buď povolí nebo zamítne. Server vygeneruje výzvu a zašifruje ji veřejným klíčem klienta, který ji rozšifruje a vytvoří tzv. fingerprint. Ten může být ve tvaru SHA-1, který vypadá například takto:

```
2f:b6:95:d5:s1:5f:r3:gd:7d:03:51:2d:18:b7:86:c3
```

[45] SSH1 a SSH2 se většinou obecně nazývá SSH (http://www.ssh.com/). Protokol je pak označován ssh.

[46] http://www.openssh.org/

[47] Přesněji Key Distribution Center KDC – tzv. třetí hlava psa Kerbera.

nebo ve tvaru Bubble Babble a vypadat například takto:

```
xulip-fihor-hijah-kevad-fyzyt-bulah-cyzuz-mezid-firek-luvyz-pixus.
```

OpenSSH používá první variantu SHA-1.

Klient pošle fingerprint serveru, který ho porovná a autentizuje klienta. Dále se může například spustit shell (následný přechod do interaktivního režimu), přesměrovat X11, vykonání nějakého příkazu, přesměrování portů, atd.

A nyní již praktičtěji. Pokud se poprvé přihlásíme na ssh server například takto:

```
ssh pepa@fjfi.cvut.cz
```

nebo

```
ssh -l pepa fjfi.cvut.cz
```

zeptá se vás server, jestli souhlasíte s fingerprintem serveru. Veřejný klíč si pak uloží a upozorňuje pouze v případě, když se fingerprint změní. Pokud se fingerprint změní, nemusí to hned znamenat ohrožení bezpečnosti, ale například přeinstalování serveru. Ovšem ssh vás nebude chtít pustit dál. Abyste mohli pokračovat, je nutné smazat příslušné řádky v souboru `~/.ssh/known_hosts` (pokud se jedná o OpenSSH) nebo v souboru `~/.ssh2/hostkeys` (pokud se jedná o SSH). Fingerprinty svého stroje můžete zjistit v případě, že klient i server používá OpenSSH těmito příkazy:

```
ssh-keygen -l -f /etc/ssh/ssh_host_key.pub
ssh-keygen -l -f /etc/ssh/ssh_host_dsa_key.pub
ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub
```

Jeden z těchto fingerprintů musí být s fingerprintem dotazujícího stroje shodný. V případě, že klient i server používá SSH, zjistíme fingerprint příkazem:

```
ssh-keygen2 -F /etc/ssh2/hostkey.pub
```

Pokud server používá OpenSSH a klient SSH, vypadá příkaz takto:

```
ssh-keygen -B -f /etc/ssh/ssh_host_dsa_key.pub
```

Pokud server používá SSH a klient OpenSSH je situace složitější, protože SSH neumí generovat SHA-1 fingerprinty. Tento problém jde obejít tak, že se budeme

chovat, jakoby jsme se přihlašovali z SSH, zadáme příkaz:

```
ssh-keygen2 -F /etc/ssh2/hostkey.pub
```

Toto nám vytvoří fingerprint, který budeme posléze porovnávat. Poté se připojíme na server, kde akceptujeme fingerprint, ale nezadáme heslo, nýbrž klávesy Ctrl+C. Poté zadáme:

```
ssh-keygen -B -l -f ~/ssh/known_hosts
```

V tomto výpisu bychom měli získat jeden nebo více fingerprintů, kde jeden by se měl shodovat s naším původně vygenerovaným fingerprintem.

K přihlašování pomocí klíče je nutné si takové klíče vytvořit a umístit je na cílový server. Tato metoda vůbec nepoužívá heslo, které se váže k systémovému kontu. Toto může mít však za následek, že vetřelec podstrčí svůj klíč se svým heslem a bude mu umožněn přístup. Tomuto lze zabránit buď vypnutím této formy přihlašování a to nastavením této konfigurace "`RSAAuthentication no`" v konfiguračním souboru `/etc/sshd_config` nebo kontrolou integrity u souboru `~/.ssh/authorized_keys`. V případě OpenSSH se provádí generování klíče takto:

```
ssh-keygen -t dsa
```

Můžeme použít i `-t rsa` pro vygenerování RSA klíče. Po tomto příkazu je nutné zadat soubor, do kterého bude klíč uložen (nebo nechat implicitní hodnotu) a dále je nutné zadat heslo. Implicitně se vám vytvoří dva soubory a to `~/.ssh/id_dsa`, kde je uložena vaše identifikace a `~/.ssh/id_dsa.pub`, což je samotný veřejný klíč, který umístíme na server a to do souboru `~/.ssh/authorized_keys` (u starších verzí OpenSSH se může jednat o soubor `~/.ssh/authorized_keys2`). K přenosu můžeme použít utilitu `scp`, kterou budu probírat podrobněji později. Při přihlašování pomocí příkazu `ssh` můžeme využít přepínačů `-1` a `-2`, kterými ovlivníme použití verze protokolu ssh – použitím jiné verze ssh protokolu se také změní fingerprinty.

U SSH je situace podobná. Ke generování se používá příkaz `ssh-keygen2` (použitím `-t rsa` si vynutíme použití RSA). V adresáři `~/.ssh2/` by se měl být soukromý i veřejný klíč. Implicitní názvy je dobré přejmenovat na něco smysluplnějšího. Dále je potřeba nastavit v souboru `~/.ssh2/identification` řádku `IdKey nazev_privatniho_klice`. Zkopírujeme veřejný klíč na server a v souboru `~/.ssh2/authorization` nastavíme konfiguraci "`Key nazev_verejneho_klice`".
Tímto je autorizace pomocí klíče zprovozněna.

Privátní klíč by měl mít taková práva, aby byl čitelný pouze pro vlastníka.

Problémy mohou nastat při použití NFS, kde je znemožněno číst z domovského adresáře před přihlášením.

Pomocí ssh se lze také přihlašovat bez hesla a to pomocí souboru `/etc/hosts.equiv` a `~/.rhosts` – lépe však přes zabezpečené podoby těchto souborů `/etc/shosts.equiv` a `~/.shosts`.

Nyní stručně na příkladech vysvětlím použití programu `scp`. Jak jsem již zmínil, tento program slouží k bezpečnému přesouvání souborů. Pokud chceme přesunout soubor `file.log` do adresáře `log` v domovském adresáři uživatele `pepa` na serveru `fjfi.cvut.cz`, provedeme to tímto zápisem:

```
scp file.log pepa@fjfi.cvut.cz:./log/
```

Pokud bychom chtěli do toho samého adresáře přenést místo souboru `file.log` adresář `dic` a všechny soubory, které obsahuje použili bychom zápis:

```
scp -r dic pepa@fjfi.cvut.cz:./log/
```

Funguje to i opačným směrem a to ze serveru na klienta:

```
scp pepa@fjfi.cvut.cz:./log/file.log ./
```

Trochu krkolomné je kopírování mezi dvěma vzdálenými servery. Zápis je takovýto:

```
scp pepa@fjfi.cvut.cz:file.log pepa@dc.fjfi.cvut.cz:
```

V praxi je však vhodnější se na server `fjfi.cvut.cz` nejprve přihlásit a použít `scp` klasicky.

Ale co když chceme procházet filesystémem, pracovat s adresáři, právy, apod. a to všechno v rámci jednoho spojení. Můžeme použít například klasické ssh spojení, které lze realizovat přes `mc`[48]. Také můžeme použít program `sftp`. Na straně klienta je situace jednoduchá. Stačí použít nějakého klienta, který `sftp` podporuje. Já používám například grafického klienta `gftp`[49]. Problém nastává na serverové straně. Je potřeba vytvořit chroot souborový systém. Nebo použít záplatu na OpenSSH z URL:

```
http://mail.incredimail.com/howto/openssh
```

[48] http://www.ibiblio.org/mc/
[49] http://gftp.seul.org/

Existují i další možnosti. V případě SSH stačí použít programy `ssh-chrootmgr` a `ssh-dummy-shell`, které by měly zmíněné akce zvládnout.

Pomocí ssh lze i přesměrovávat porty. K přesměrovávání portů ovšem preferuji utilitu stunnel[50], která je pro tento účel specializována – touto tématikou se budu zabývat v některé z dalších podkapitol.

Poslední věcí, kterou bych chtěl zmínit v souvislosti s ssh, je skript `authprogs`[51]. Tento program umožňuje definovat v jednom konfiguračním souboru seznam příkazů pro jednu autentizační identitu. Tyto příkazy lze spouštět z určitých počítačů na základě definovaných IP adres.

Konfigurační soubor `~/.ssh/authprogs.conf` může vypadat následovně:

```
[ 192.168.0.3 ]
uname -a; free
w
```

První řádek s IP adresou určuje, ze kterých počítačů budou moci být povolené příkazy zadávány. Na druhém řádku vidíme, že lze zadávat i seznamy příkazů. Třetí řádek povoluje užití samostatného příkazu `w`. K tomu, aby mohlo být toto realizováno, je potřeba vygenerovat příslušné klíče, což už jsem v souvislosti s ssh probíral. Samotné použití je jednoduché. Náš seznam dvou příkazů bychom spustili na serveru `fjfi.cvut.cz` takto:

```
ssh -l pepa -i id_dsa fjfi.cvut.cz 'uname -a; free'
```

Pokud bychom se pokusili spustit příkaz, který nemáme povolen, mělo by se vypsat varování "You're not allowed to run 'prikaz'". Akce jsou zaznamenávány do souboru `~/.ssh/authprogs.log`.

4.4.3 OpenSSL

OpenSSL[52] je tvořena knihovnou pro šifrování a autentizaci a dalšími podpůrnými programy. Tato knihovna se většinou používá k zabezpečování protokolů – například HTTPs (HTTP over SSL), což je šifrovaná verze protokolu HTTP, dále například IMAPs, POP3s, SMTPs nebo IRCs.

Spojení funguje tak, že se vytvoří klasické TCP spojení, dále tzv. handshake, při kterém se ověří certifikáty a vymění se šifrovací klíče. Poté už se začnou

[50] http://www.stunnel.org/
[51] http://www.hackinglinuxexposed.com/tools/authprogs/
[52] http://www.openssl.org/

posílat zašifrovaná data – tento postup je nám již znám.

K ověření identity se používají tzv. certifikáty. Mohou sloužit i pro ověřování osob. V našem smyslu ovšem certifikát slouží k ochraně klientů, kteří se přihlašují na servery. Certifikát klientům zaručí, že server, ke kterému se připojí, není nějak podvržený (tzn. ovládá ho vetřelec). K přesměrování provozu mohl útočník použít například ARP poisoning nebo DNS přesměrování – o těchto útocích bude ještě řeč. Pravost certifikátů je podložena podpisem. Může být podepsán sám sebou (tzv. *self-signed certificate*) nebo tzv. CA (Certificate Authority). Certifikát CA může být podepsán další certifikační autoritou[53]. Tyto certifikáty jsou umístěny v uživatelských aplikací (např. v internetových prohlížečích) a jsou to certifikáty, kterým uživatel důvěřuje. Podepsání certifikátu certifikační autoritou, které uživatel věří, mu garantuje to, že je daný certifikát pravý (autentický). Tato vlastnost se dále dědí, takže certifikát podepsaný tímto autentickým certifikátem je též pravý. Certifikát obsahuje CN (Common Name), což je jméno toho, koho certifikát identifikuje. Dále platnost – počáteční a koncové datum platnosti. Platnost může být předčasně ukončena zveřejněním v tzv. CRL (Certificate Revocation List). Dále také veřejný klíč, kterým se šifruje a některé další údaje pro samotné SSL nepodstatné jako například jméno, e-mail, adresa toho, komu byl certifikát vydán. Pro šifrování se používá metoda založená na výměně klíčů, se kterou sem vás již dostatečně obeznámil. Ovšem klíče zde slouží pouze pro navázání spojení. Poté se strany dohodnou na symetrickém klíči, kterým šifrují. Je to podstatně rychlejší, než kdyby se celý přenos šifroval asymetricky. Jenom bych chtěl upozornit na to, že zde je výhodnější používat soukromé klíče bez hesla, jinak bude nutná lidská asistence například při restartování HTTP serveru.

Nyní předvedu, jak můžete jednoduše vygenerovat x509 certifikát. Použijeme příkaz `openssl` v tomto tvaru:

```
openssl req -new -x509 -nodes -out cert.pem -keyout key.pem -days 1098
```

Příznak `req` značí, že se jedná o certifikační žádost. Příznak `-new` znamená, že vytváříme novou žádost. Příznak `-x509` znamená právě vytváření certifikátu podepsaného sám sebou (self-signed certificate). Příznakem `-nodes` říkáme, aby se soukromí klíč nešifroval. Certifikát bude zapsán do souboru `cert.pem` a klíč do souboru `key.pem`. Certifikát bude platný 1098 dní (3 roky). Budete dotázáni, abyste vyplnili zbývající informace jako e-mail, adresa a také CN, kam by jste měli vyplnit jméno serveru, pro který je certifikát vystaven. Soubory `cert.pem` a `key.pem` jsou při běžném čtení nečitelné (pokud neuvedeme příznak `-text`).

[53] Jako CA působí v ČR například CESNET (http://www.cesnet.cz/pki/). U této CA si mohou, po předložení dokladu totožnosti, nechat vytvořit certifikát zaměstnanci a studenti ČVUT zdarma. Bohužel CESNET není ověřenou certifikační autoritou, takže tento certifikát zatím například ve státní správě neuplatníte.

Do lidsky čitelné podoby je lze převést takto:

```
openssl x509 -in cert.pem -text
```

a

```
openssl rsa -in key.pem -text
```

Některé aplikace požadují certifikát a klíč v jednom souboru. Toho lze docílit jednoduchým spojením:

```
cat cert.pem key.pem > cert_key.pem
```

Konfigurační soubor OpenSSL se nachází většinou v `/etc/ssl/` a jmenuje se `openssl.cnf` nebo `openssl.conf`.

Pokud chceme, aby klienti nebyli otravováni hlášením o nedůvěryhodných certifikátech, je optimální si vytvořit vlastní CA. Po vygenerování našeho certifikátu je nutné jej naimportovat do počítačů, které hodlají s naším serverem komunikovat. My jako CA poté můžeme generovat certifikáty, které budou již pro klienty důvěryhodné. Pokud chceme vygenerovat certifikát CA, je nejprve nutné vytvořit adresář pro CA – direktiva `dir` (implicitně `demoCA`) v konfiguračním souboru pro OpenSSL. Dále vytvoříme další nutné podadresáře (implicitně `certs`, `crl`, `newcerts` a `private`) podle konfiguračního souboru pro OpenSSL. Do adresáře pro CA je potřeba vytvořit soubor `index.txt`, což je jakýsi "database index file". V tom samém adresáři je nutné vytvořit soubor `serial`, do kterého je nutné napsat hodnotu "01". Dále již můžeme vygenerovat certifikát a klíč s výstupy do souboru nastavených v konfiguračního souboru implicitně takto:

```
openssl req -new -x509 -nodes -out cacert.pem -keyout cakey.pem -days 1825
```

Jako CA můžeme certifikát podepsat příkazem:

```
openssl ca -in request.pem -out cert.pem
```

Tímto se změní některé soubory, které jsme vytvářeli při zakládání CA. Je to z toho důvodu, že CA si udržuje databázi podepsaných certifikátů.

Pokud chceme zabezpečit jednotlivé protokoly (IMAPs, HTTPs, atd.) je potřeba to samotnému serveru (v případě HTTP například Apache) sdělit. Server musí vědět odkud má certifikát a klíč číst. Tuto funkci musí samozřejmě server podporovat. V případě serveru Apache k této činnosti slouží mod_ssl[54]. Ověření

[54] http://www.modssl.org/

funkčnosti takového serveru můžeme provést příkazem:

```
openssl s_client -connect server.cz:https
```

4.4.4 stunnel

Stunnel[55] je program distribuovaný pod licencí GPL, ale je omezen licencí SSL knihovny, pod kterou je zkompilován. Tato aplikace umožňuje spojit dva počítače (tím i celé sítě) šifrovaným tunelem (toto ovšem vyžaduje dvě instance stunnelu), nebo zabezpečit nešifrované protokoly popř. daemony jako IRC, POP3, IMAP, MySQL apod.

Stunnel pracuje ve dvou módech. První mód vynutíme příznakem `-d` s povinným parametrem, který značí port, na kterém bude stunnel naslouchat a nepovinným parametrem, který značí adresu počítače, na které bude stunnel naslouchat. Druhý (serverový) mód se použije při zadání příznaku `-d` bez parametrů a pak se jedná o standardní vstup a výstup stunnelu. První mód umožňuje obsluhovat více spojení najednou. Mezi další důležité příznaky patří především příznak `-r` s nepovinným parametrem adresy počítače, kam se bude připojovat přes port, který je druhým povinným parametrem. Pokud se nezadá jméno počítače, bude se stunnel snažit připojit na `localhost`. Příznakem `-l` lze specifikovat program, který bude spuštěn a jeho standardní vstup, výstup (neměly by mít terminálové vlastnosti) budou jedním koncem stunnelu. Programu lze předat jeho parametry přes nepovinné parametry (u příznaku `-l`). Pro programy s terminálovými vlastnostmi slouží příznak `-L`. Velmi důležitý příznak je příznak `-c`, kterým určíme to, který konec se bude šifrovat. Pokud ho zadáme, bude první konec nešifrovaný. Poté dojde k zašifrování dat a jejich dopravení na druhý konec (druhým směrem funguje přesně opačným způsob). Pokud ovšem příznak `-c` zadáme, bude první konec šifrovaný – tzn. že na něm bude očekáváno spojení. Data se tedy rozšifrují a dopraví nešifrovaná na druhý konec (druhým směrem funguje přesně opačný způsob).

Serverový mód vyžaduje certifikát (jeho vytvoření jsem popisoval v podkapitole o SSL), který se čte implicitně ze souboru `/etc/ssl/certs/stunnel.pem` nebo ho lze specifikovat za příznak `-p`. Tento certifikát nesmí obsahovat klíč chráněný heslem.

Ať se stále neopakuji s HTTPs, předvedu, jak jednoduše pomocí stunnelu zprovoznit POP3s. Provedeme to pomocí příkazu `stunnel` v tomto tvaru:

```
stunnel -d pop3s -l /usr/sbin/pop3d - pop3d -p /data/certst/cert.pem
```

[55] http://www.stunnel.org/

Vidíme, že se spustí POP3 daemon, který pak bude běžet na pozadí. Certifikát se načte ze souboru `/data/certst/cert.pem`. Parametr `pop3s` lze nahradit numerickou reprezentací portu. Poté je vhodné přesměrovat POP3s na POP3, abychom nemuseli spouštět novou instanci POP3 daemona pokaždé, když se někdo připojí. Provedeme to příkazem:

```
stunnel -d pop3s -r pop3
```

Všechny parametry by nám měli být již jasné. Pokud náš klient, se kterým přistupujeme na POP3 nepodporuje šifrování, můžeme si opět pomoci pomocí stunnelu a to tímto příkazem, který spustíme na počítači, ze kterého budeme na POP3s server přistupovat:

```
stunnel -c -d localhost:pop3 -r pop3s.server:pop3s
```

Po zadání tohoto příkazu stačí použít normálního POP3 klienta a připojit se na port POP3 serveru na `localhostu`. Mezi klientem a serverem, kde je provozován POP3s, se vytvoří šifrovaný tunel.

Pro vytváření podobných tunelů lze použít, kromě již zmíněného ssh, také protokol GRE[56] (General Routing Encapsulation). Další variantou pro vytváření šifrovaných spojení je PPTP (Point-to-Point Tunneling Protocol), který umožňuje vytvářet šifrovanou relaci PPP (Point to Point Protocol) mezi dvěma počítači. Toto je v současné době aktuální především ve spojení s ADSL, kde je právě PPTP využíváno ke spojení klientů s ISP. Další mechanismus, který lze využít k vytváření VPN, je CIPE[57] (Crypto IP Encapsulation). Tento mechanismus se také osvědčil při zabezpečování bezdrátových sítí. Ještě zmíním projekt Zebedee[58], který se snaží především o snadnou konfiguraci VPN.

4.5 GnuPG

GnuPG[59] (The GNU Privacy Guard – zkráceně označováno také jako GPG) vzniklo jako kompletní, volně šiřitelná náhrada PGP[60] (Pretty Good Privacy). Oba tyto programy slouží k šifrování dat, elektronickému podepisování, ověřování elektronických podpisů, správě klíčů, apod. GPG je tedy na rozdíl od PGP bezplatný software, který lze používat za podmínek licence GNU GPL. PGP používá patenty IDEA a RSA, které tedy v současné době již neomezují (jejich platnost vypršela v září roku 2000), takže je volně k dispozici i mimo USA a Kanadu. Protože

[56] http://www.ietf.org/rfc/rfc1701.txt
[57] http://sites.inka.de/~W1011/devel/cipe.html
[58] http://www.winton.org.uk/zebedee/
[59] http://www.gnupg.org/
[60] http://www.pgpi.org/

omezení vývoje a používání šifrovacího softwaru, který je například v USA řazen na stejnou úroveň jako zbraně, jsou značně nejasná a v dnešní době se týká politik jednotlivých států, zmínil bych dokument "*Crypto Law Survey*"[61], který veškerá omezení, týkající se šifrování, shrnuje. Ovšem PGP jako freeware obsahuje pouze omezené množství funkcí a to pouze k nekomerčnímu použití.

GPG využívá metod asymetrického šifrování – tzn. že veřejným klíčem data zašifrujeme a soukromým (privátním) naopak rozšifrujeme. Pokud tedy chceme, aby byla data čitelná pouze pro Josefa, zašifrujeme je jeho veřejným klíčem. Privátní klíč by správně měl vlastnit jen on, proto je jediný, který může data rozšifrovat. Lze ale data zašifrovat soukromým klíčem a veřejným klíčem je rozšifrovat. Tohoto se využívá při podepisování dokumentů i jiných dat. Podepisování slouží k ověřování autenticity (pravosti) zprávy. Obecně je totiž veřejný klíč Josefa veřejně k dispozici, takže zašifrovanou zprávu mu může poslat každý. Podepisování funguje tak, že ještě před zašifrováním zprávy se vytvoří hash zprávy pomocí symetrického šifrovacího algoritmu (například MD5 či kvalitnější SHA1). Pokud by se ve zprávě cokoliv změnilo, hash by nesouhlasila. Hash je poté zašifrována privátním klíčem odesílatele. Do zprávy je doplněno datum a identifikátor uživatele. Pak už jen odesílatel zašifruje výslednou zprávu veřejným klíčem adresáta. Pokud chce odesílatel zajistit pouze integritu dat (nejde mu o utajení dat), může poslední operaci (zašifrování veřejným klíčem adresáta) vynechat. Příjemce (adresát) po dešifrování zprávy, rozpoznání podpisu pomocí veřejného klíče odesílatele a vytvoření hashe ze zprávy, porovná jím vytvořený hash s hashem od odesílatele – shoda garantuje pravost zprávy. Všechny tyto operace jsou samozřejmě programem GPG automatizovány.

Nyní popíši proces vytvoření vlastních klíčů. Začneme příkazem:

```
gpg -gen key
```

Pokud je toto vaše první práce s GPG na daném počítači, vytvoří se adresář `~/.gnupg`, kde se budou nacházet potřebné konfigurační soubory. Po zadání příkazu potvrďte implicitní volbu "DSA a ElGamal (implicitní)", kterou vytvoříte pár klíčů. Dále vyberte velikost klíče. Optimální bude zvolit 1024 nebo 2048 bitů. Více už by bylo zbytečné a nepohodlné – velikost 2048 je ještě v současné době prakticky neprolomitelná a větší velikost klíče by šifrování a rozšifrování značně prodlužovala. Dále vybere dobu platnosti klíče – nedoporučuje se nechat dobu platnosti klíče neomezenou. Dále je nutné vyplnit identifikátory klíče ve formě jména, e-mailové adresy a komentáře. Při následné generaci klíče potřebuje program náhodné hodnoty – ke zvýšení entropie můžeme přispět například pohybem myši. Nyní máme k dispozici pár klíčů.

[61] http://rechten.kub.nl/koops/cryptolaw/

Lze vytvořit i tzv. odvolací klíč, kterým lze zrušit platnost klíče, ke kterému se odvolací klíč vztahuje:

```
gpg -gen-revoke e-mailova_adresa_klice
```

Místo e-mailové adresy lze použít i jméno vztahující se ke klíči.

Pokud chceme předat veřejný klíč druhé straně, která s ním bude šifrovat data pro nás, je nutné ho nejdříve vyexportovat – nejlépe do souboru:

```
gpg -export -armor e-mailova_adresa_klice > verejny_klic.pub
```

Po bezpečném předání tohoto souboru druhé straně[62], může druhá strana tento soubor (veřejný klíč) importovat:

```
gpg -import verejny_klic.pub
```

Měli bychom také vyloučit možnost, že někdo klíč změnil po cestě a to pomocí již zmiňovaných fingerprintů (otisků). Pro zobrazení fingerprintu daného klíče zadají obě strany:

```
gpg -fingerprint e-mailova_adresa_klice
```

Místo e-mailové adresy lze opět použít i jméno vztahující se ke klíči. Poté vhodným způsobem komunikace otisky porovnají. Pokud se shodují, je vše v pořádku a druhá strana (jakožto příjemce veřejného klíče) by měla klíč uznat pravým a podepsat ho příkazem `sign`, který zadá v interaktivním režimu, který se spustí takto:

```
gpg -edit-key e-mailova_adresa_klice
```

Veřejné klíče lze stahovat i ze serveru, pokud jsou na něm k dispozici:

```
gpg -keyserver server_s_klici -recv-keys ciselna_hodnota
```

Klíč nelze stáhnout přímo přes e-mail či jméno vztahující se ke klíči, ale pomocí jakési číselné hodnoty. Tato služba využívá TCP port 11 371. K těmto účelům lze využít například servery `www.keyserver.net`, `wwwkeys.pgp.net` či `pgp.ai.mit.edu`.

[62] Uživatelé často vystavují své veřejné klíče na webových stránkách, což nemusí být z hlediska bezpečnosti optimální. Je proto nutné důkladně kontrolovat fingerprinty (viz. dále) takto stahovaných klíčů.

Pokud chceme umístit naše veřejné klíče na takovýto server, uskutečníme to příkazem:

```
gpg -send-keys e-mailova_adresa_klice -keyserver server_s_klici
```

Nyní k samotnému šifrování a dešifrování dat. Prosté zašifrování souboru provedeme takto:

```
gpg -e -r jmeno_klice soubor
```

Příznak `-e` znamená, že soubor bude zašifrován - příznakem `-s` si lze vynutit i podepsání šifrovaného souboru. Příznak `-r` definuje klíč, kterým bude soubor zašifrován. Lze zadat jméno nebo e-mail vztahující se ke klíči. Po zašifrování se v aktuálním adresáři vytvoří soubor s příponou `.gpg`, což je výsledný zašifrovaný soubor. Rozšifrování takového souboru provedeme takto:

```
gpg -d soubor.gpg
```

Pokud nechceme mít výsledný rozšifrovaný soubor vypsaný na standardním výstupu, lze výstup specifikovat i pomocí příznaku `-o`, za kterým následuje název souboru, kam chceme výsledná data uložit.

Pokud chceme ověřit pravost souboru (například program stažený z Internetu), ke kterému je k dispozici podpis, provedeme to takto:

```
gpg -verify program.asc program.tgz
```

kde `program.asc` je soubor s podpisem (často má také koncovku `.sig`, což je většinou binární podoba podpisu) a `program.tgz` samotný program.

Pokud chceme tedy vytvořit soubor s ASCII podobou podpisu bez originálních dat s koncovkou `.asc`, provedeme to takto:

```
gpg -b -armor soubor
```

Pokud neuvedeme příznak `-armor`, vytvoří se binární podoba podpisu s koncovkou `.sig`.

Program GPG nám samozřejmě nabízí nesčetně dalších voleb, které nalezneme v dokumentaci[63].

[63] http://www.gnupg.org/(en)/documentation/index.html

Šifrovat elektronickou korespondenci bychom jistě měli v případech, pokud například uživatelům rozesíláme citlivé informace jako hesla, apod. a existuje nebezpečí, že by se k e-mailu mohla dostat třetí osoba. E-mailové klienty lze nakonfigurovat tak, aby úkony jako zašifrování, rozšifrování, podepisování, atd. prováděly automaticky. Podepisování e-mailů a například na webu vystavovaných dokumentů jistě dodá jistou dávku důvěryhodnosti lidem, kteří je čtou.

4.6 Síťové hrozby

Tato podkapitola se zabývá popisem možných síťových hrozeb a jejich eliminací. Zabývám se ovšem nativními problémy, protože rozbor subsystémů není náplní této práce.

ARP cache poisoning a ARP spoofing

Jestliže potřebuje jedno zařízení komunikovat s jiným (přes sít Ethernet), normální IP adresa mu k tomu nestačí. K tomu, aby mohl být paket doručen je potřeba použít MAC (Media Acces Control) adresu, kterou lze zjistit pomocí protokolu ARP (Address Resolution Protocol). MAC adresa je tvořena šesticí dvouciferných hexadecimálních čísel – například 00:4F:4E:16:DA:68. Zařízení, které chce komunikovat s jiným tedy odešle ARP požadavek, kde se dotazuje komu patří IP adresa, se kterou chce komunikovat. Zařízení s touto IP adresou odpoví zasláním své MAC adresy a přesnost se může uskutečnit. Kvůli úspoře provozu je ovšem na každém systému vytvořena tzv. ARP cache, kde se ukládají převody IP adres na adresy MAC. Její obsah lze prohlédnout zobrazením souboru `/proc/net/arp` nebo použít příkaz `arp -a`. Tato tabulka je stále obměňována (netýká se pevných záznamů) – na GNU/Linuxu je tato doba implicitně 60 sekund. Tuto hodnotu lze změnit přepsáním hodnoty v souboru `/proc/sys/net/ipv4/neigh/eth0/gc_stale_time`, což však není obecně doporučováno. Se záznamy lze manipulovat. Můžeme je mazat nebo přidávat již zmiňované pevné (permanentní) záznamy pomocí souboru `/etc/ethers`. Více o protokolu ARP lze nalézt v RFC 826[64]. Protokol ARP má také rozšíření "proxy ARP", s jehož pomocí může zařízení odpovídat na ARP požadavky jménem jiného zařízení.

Tohoto může ovšem zneužít vetřelec, který získal kontrolu nad systémem uvnitř sítě. Pomocí proxy ARP může ovládat směrování paketů, takže může směrovat ARP pakety na kterýkoliv jiný systém v Internetu. Tomuto se lze vyhnout použitím pevných záznamů. Vetřelec toto může obejít zfalšováním vlastní MAC adresy (pomocí důvěrně známého příkazu `ifconfig`), pokud to daná karta podporuje. Tomuto se lze opět vyhnout zvýšením bezpečnosti pomocí nástroje Arpwatch, což je sofistikovaná aplikace, která nám bude hlásit nejen nové počí-

[64] http://www.ietf.org/rfc/rfc826.txt

tače komunikující s naším systémem, ale i jiné podezřelé aktivity. Mnoho bezpečnostních rizik na této vrstvě lze také eliminovat použitím šifrovaného přenosu, kterému jsem se věnoval v předešlých sekcích nebo pomocí firewallu (například iptables) – kontrolou nesmyslných IP adres a také kontrolou samotných MAC adres, což v iptables lze.

IP spoofing

IP spoofing (označovaný také jako IP faking, packet spoofing či packet faking) je metoda, kdy útočník skrývá svoji identitu předstíráním, že je někdo jiný změnou zdrojové adresy. Toto je způsobeno jistými nedokonalostmi protokolů TCP, UDP, ICMP a dalšími. Zapsání zdrojové adresy se děje na straně odesílajícího systému a je tedy schopen takovou adresu podvrhnout. V současné době jsou tyto techniky již těžko použitelné – s výjimkou bezdrátových sítí, kde je možno si dovolit to, co v době linuxových jader řady 2.0 a nižších, kde byla lehce předvídatelná čísla sekvence TCP (od jader 2.0.36 je problém vyřešen). Další závažný problém je v protokolu UDP, který způsobuje snadné zneužití tohoto protokolu k falšování zdrojové adresy.

Eliminovat riziko podvržení zdrojové adresy lze především pomocí dobře nakonfigurovaného firewallu (například aktivováním funkce rp_filter) a opět použitím šifrovacích a autentizačních mechanismů jako SSH či IPSec probírané v předešlých sekcích této práce. Také bychom neměli důvěřovat paketům UDP. Ještě bych zmínil nástroj ipsentinel[65], který dokáže zamezit neautorizovanému používání IP adres na lokálním Ethernetu.

Odposlech přenosu

Odposlech přenosu (sniffing) je metoda, kdy někdo zachytává (čte) přenášená data (pakety) mezi místem, odkud byl paket vyslán a místem, kam paket směřuje. Tato situace může nastat například přímo na jednom systému, kdy superuživatel zachytává pakety proudící přes síťová rozhraní lokálního stroje. Data mohou být zachytávána také na směrovačích mezi zdrojovým a cílovým počítačem. V případě bezdrátového přenosu je možnost odposlechu doslova snadná, což samozřejmě vychází z vlastností přenosového média. Další situace možnosti odposlechu nastává v případě použití rozbočovačů (angl. *hubs*), který přijatý paket rozešle všem systémům s předpokladem, že daný systém přijme jen ten provoz, který mu skutečně patří. Inteligentněji se chovají přepínače (angl. *switches*), které směrují pakety jen tomu systému, kterému skutečně patří. Ovšem není zas tak velký problém přinutit přepínač, aby se choval jako rozbočovač. Toto lze provést záplavou paketů s podvrženou cílovou a zdrojovou adresou (MAC nebo IP) a současně odposlouchávat provoz na síti. Toto riziko je potřeba eliminovat, protože na sítí mohou být přenášena citlivá data jako uživatelské přístupy s hesly,

[65] http://www-user.tu-chemnitz.de/~ensc/ip-sentinel/

e-maily, atd.

Existuje mnoho programů, které dokáží provoz odposlouchávat – například dsniff[66], Ethereal[67], tcpdump[68], Snort[69], Sniffit[70] a spousta dalších.

Obranou je opět použití šifrovaných verzí služeb a vůbec šifrovaného přenosu. Vidíme, že šifrování je opravdu silný prostředek v posilování bezpečnosti na úrovni sítě.

MITM

MITM (man in the middle attack[71]) útok spočívá v tom, že pakety nedorazí přímo na místo určení, ale k někomu jinému, kdo tvoří jakéhosi nechtěného prostředníka mezi odesílatelem a právoplatným příjemcem. Vetřelec, který takto narušuje komunikaci, se pak chová vzhledem k odesílateli jako příjemce a vzhledem k právoplatnému příjemci jako odesílatel (v opačném směru komunikace zase naopak). Útočník pak může takovouto komunikaci modifikovat, aniž by si toho obě komunikující strany všimly.

Prostředkem eliminace tohoto rizika je ovšem autentizace (popř. i šifrování), která nám zaručí pravost přijatých dat. V případě e-mailové komunikace můžeme využít GPG. V jiných případech komunikace například již několikrát zmiňované IPSec či SSH.

Detekce síťových karet v promiskuitním módu

Síťová karta v normálním režimu ignoruje rámce, které jsou určené pro jiné rozhraní. Zatímco v promiskuitním režimu se chová síťové rozhraní tak, že přijímá všechny rámce, čehož lze využít k hledání chyb na síti, ovšem z hlediska bezpečnosti by tento mód neměl být používán. Co víc, měli bychom pravidelně kontrolovat, zdali někdo či něco síťové rozhraní do tohoto módu nepřepnul.

To jestli je karta v promiskuitním režimu zjistíte podle toho, zdali ve výpisu `ifconfig` obsahuje řetězec `PROMISC`. Vše můžete zautomatizovat skriptem, který budete pravidelně spouštět například každých 30 minut, což provedete pomocí démonu `cron`.

[66] http://naughty.monkey.org/~dugsong/dsniff/
[67] http://www.ethereal.com/
[68] http://www.tcpdump.org/
[69] http://www.snort.org/
[70] http://reptile.rug.ac.be/~coder/sniffit/sniffit.html
[71] Dozvěděl jsem se, že korektní název tohoto útoku by měl být "human in the middle attack", aby tento útok nebyl mylně spojen pouze s mužským pohlavím.

DoS, DDoS

Útoky DoS (Denial of Service) slouží většinou k blokování síťového provozu (síťových služeb). Toto je realizováno zasláním velkého množství požadavků, které není cílový systém schopen zvládnout. S těmito útoky se potýkala síť (Internet) již od samého počátku a v dnešní době, kdy máme k dispozici nejmodernější prostředky obrany, ho pořád nelze úplně eliminovat. K tomuto útoku lze využít prakticky všechny druhy paketů – některé sofistikovanější nástroje pakety dokonce kombinují, aby zvýšily efektivnost samotného útoku. Útok je samozřejmě efektivnější, pokud útočník disponuje širším internetovým pásmem než oběť.

Prvním typem jsou tzv. ICMP záplavy, které lze vyvolat obyčejným příkazem `ping` například takto:

```
ping -f -s 1024 obet.cz
```

Příznak `-f` zajistí, že se pakety budou zasílat tak rychle, jak to jen půjde. Příznakem `-s` lze specifikovat velikost paketu, která je v uvedeném případě 1KB. Proti tomuto druhu útoku se lze efektivně bránit správnou konfigurací firewallu, kde omezíme nebo úplně zakážeme množství propouštěných ICMP paketů (ICMP ECHO REQUEST pakety).

Dalším typem jsou tzv. UDP záplavy. Ty jsou realizovány pomocí služeb jako `echo` nebo `chargen`. Tomu zamezíme zakázáním těchto služeb. Nemají již v této době význam.

Smurf útokem[72] se označuje útok, kdy je zasíláno velké množství ICMP ECHO REQUEST paketů, které mají podvrženou IP adresu oběti. Systémy, kterým je paket určen, na tento paket odpoví a tím se celá efektivnost útoku znásobí. Stejného mechanismu využívá program fraggle, který používá UDP pakety.

TCP/IP protokolu využívá tzv. SYN záplava. Počítač je zahlcen množstvím polootevřených spojení, což po zaplnění fronty u oběti způsobí to, že nebude schopen přijímat další TCP spojení. Vzhledem k tomu, že v nových jádrech je fronta zvětšena a prodleva, kdy počítač čeká, jestli od komunikujícího počítače neobdrží druhý inicializační paket, zkrácena, je realizace těchto útoků obtížnější.

Sofistikovanějšími útoky DoS jsou tzv. DDoS (Distributed Denial of Services), které využívají množství koordinovaných počítačů, které souběžně zahlcují váš počítač. Útočník musí mít samozřejmě nad útočícími systémy kontrolu. Na tyto počítače nainstaluje klienty, které jsou řízeny z jednoho místa.

[72] Název je odvozen podle stejnojmenné aplikace.

Viry a GNU/Linux

Viry jsou programy, které se nějakým způsobem infiltrují do systému a přinutí ho k vykonávání vlastních instrukcí. Ovšem zatímco operační systém MS Windows je viry doslova ubíjen, GNU/Linux je proti tomuto typu útoku prakticky imunní. Proč tomu tak ale je? Prvním faktorem je především to, že GNU/Linux je mnohem rozmanitější ve smyslu konfigurace systému. Existuje velké množství distribucí, kde každá obsahuje jinou verzi softwaru či jádra. Zatímco například MS Windows XP jsou takřka na všech počítačích identické, takže pokud vytvořím vir pro jednu instalaci, bude fungovat stejně na ostatních počítačích, čímž vzniká úrodná půda pro viry. Vše je umocněno tím, že MS Windows se nacházejí mnohem častěji na pracovních stanicích, kam mohou být infiltrovány například skrze elektronickou poštu. I přesto se na Internetu pár takových linuxových virů objevilo a také dokumenty, jak takové viry pod GNU/Linux psát[73].

Jedním z virů je Linux/Bliss, který napadá spustitelné soubory ELF. Druhým virem je Linux/Vit, který infikuje spustitelné soubory. Oba viry jsou parazitické, což znamená, že soubor infikují bez toho, aniž by ho poškodily. Také jsou nerezidentní, což znamená, že infikování proběhne po spuštění souboru s virem. Podobných virů se postupem času objevilo více – například Linux.Winter, Linux.Rike.1627, Linux.Diesel, Linux.Kagob a další. Ovšem účinnost těchto virů je, oproti jejich kolegům z rodiny windowsových virů, mizivá.

Ovšem akademické sítě jsou sítě heterogenní, takže virové nebezpečí zde hrozí především pro počítače s operačními systémy MS Windows, kam se viry mohou dostat například skrze elektronickou poštu. Toto ohrožení lze eliminovat, i když používaný poštovní server (angl. *mail server*) běží na GNU/Linuxu. Existují antiviry, které lze nainstalovat na poštovní server a kontrolovat veškerou poštu, která přes tento server projde. Jedním takovým antivirem, který je vydávaný pod licencí GNU GPL, je ClamAV[74]. Tato aplikace je napsána podle norem POSIX, takže stejně dobře poběží na různých hardwarových platformách v GNU/Linuxu, *BSD, Solarisu či MacOS. Antivirová databáze tohoto programu je založena na databázi OpenAntivirus[75], který vyvíjí ClamAV jako jeden ze svých projektů. OpenAntivirus vyvíjí mnoho dalších projektů zaměřených na problematiku virů – příkladem může být další poštovní virový skenr AMaViS[76], se kterým dokáže ClamAV spolupracovat. Pokud bychom chtěli využít nějaký komerční projekt, můžeme sáhnout po antiviru NOD32[77], který má opravdu velmi kvalitní heuristiku odhalování virů.

[73] Například http://www.lwfug.org/abartoli/virus-writing-HOWTO/_html/
[74] http://www.clamav.net/
[75] http://www.openantivirus.org/
[76] http://www.amavis.org/
[77] http://www.nod32.com/home/home.htm

I když je tedy GNU/Linux proti virům prakticky imunní, najdou antiviry na tomto systému své využití a to při eliminaci virů především na platformě MS Windows. Do důsledku může virová infekce na těchto počítačích ohrozit i linuxové stroje nebo samotnou síť. Jednoduchým příkladem může být virus, který po usazení se na hostitelském počítači vysílá záplavu paketů na určité stroje v síti, čímž může celý provoz na sítí zpomalit nebo dané počítače úplně znepřístupnit. Přímé riziko virového ohrožení linuxových strojů ovšem postupem času roste a to z důvodu rostoucí používanosti GNU/Linuxu i na pracovních stanicích a také díky fenoménu, který se objevil v nedávné době a tím je objevení několika kernelových chyb, které umožňovali přímé získaní práv superuživatele. Jednalo se zejména o chybu v systémovém volání (angl. *syscall*) ptrace z března 2003[78], která ohrožovala jádra řady 2.2 až 2.4. Následovala chyba v do_brk[79]. A pak se objevily dvě chyby v syscallu mremap[80]. Pomocí těchto lokálně zneužitelných chyb se již nějaký takový linuxový virus dal využít a určitě se o to i někdo pokusil. Pořád to ovšem není optimální prostředí k masivnímu rozšíření nákazy.

[78] http://marc.theaimsgroup.com/?l=linux-kernel&m=104791735604202&w=2

[79] http://isec.pl/vulnerabilities/isec-0012-do_brk.txt

[80] http://isec.pl/vulnerabilities/isec-0013-mremap.txt, http://www.isec.pl/vulnerabilities/isec-0014-mremap-unmap.txt

Kapitola 5

Fyzická bezpečnost

Existují způsoby ohrožení bezpečnosti, které vycházejí pouze s fyzického působení osoby nebo jiných faktorů na počítač či systém.

Samozřejmě pokud má vetřelec přístup k počítačům, může narušit jejich provoz přímým odpojením z datové nebo elektrické sítě, poškozením, apod. Toto riziko lze jednoduše eliminovat omezením přístupu k daným počítačům. Tímto zamezíme i případné ztrátě dat odcizením fyzických médií. Čtení samotných dat lze znemožnit šifrováním na disku například pomocí CryptoAPI – ovšem to nám nezaručí, že my sami data ještě někdy uvidíme.

Pak jsou fyzická ohrožení, která nejsou způsobena úmyslně – například výpadek proudu. Tomuto lze předejít využitím UPS (Uninterruptible Power Supply), což je zařízení, které dokáže prodloužit přísun elektrické energie na dobu, po kterou mu stačí jeho vlastní baterie. Tato doba není většinou příliš dlouhá (u běžných UPS od 20 do 50 minut v závislosti na příkonu počítače a kapacitě baterie, které nejsou vůbec levnou záležitostí) a je určena ke korektnímu ukončení systému. Občas se také stane, že dojde k poruše disku, což může znemožnit čtení z disku. Tomuto zabráníme zálohováním na jiná média – například DVD média, CD média, magnetické pásky, apod.

Další ohrožení úzce navazuje na předchozí odstavec. Pokud už média se zálohami přerůstají únosné meze, je potřeba je vhodným způsobem odstranit. Toto se týká i případných výstupů z tiskáren, i když toto se běžně v akademických sítích nepraktikuje. Tyto materiály je nutné vhodným způsobem znehodnotit, aby z nich nebylo možné data získat (o tzv. trashingu jsem se již zmiňoval). Rozhodně by se tato činnost měla objevit v případné bezpečnostní politice.

Pomalu se dostáváme od těch hrubých technik k těm jemnějším. Možná se to na první pohled zdá absurdní, ale zneužít lze i diod LED, které indikují odesílání a příjem dat, které najdete na ethernetových přepínačích. V současné době ak-

tuální bezdrátové myši a klávesnice, popř. infra porty, které jsou používány u notebooků, PDA a jiných přenosných zařízení, vysílají signály prostřednictvím infračervených diod v nijak nechráněné podobě – tzn. nešifrované. Jsou popsány metody, jak efektivně i na velké vzdálenosti tyto signály snímat – popsané v dokumentu "*Information Leakage from Optical Emanations*"[1]. Snímat lze i záření monitoru (tato metoda se nazývá "van Eck Phreaking" nebo také "Tempest") – popsané v dokumentu "*Compromising emanations: eavesdropping risks of computer displays*"[2] a "*Optical Time-Domain Eavesdropping Risks of CRT Displays*"[3]. Protiopatřením proti těmto útokům je dobré skrytí vyzařujícího signálu, což nemusí být vždy snadné. Existuje projekt eckbox[4], který představuje software a k němu příslušný hardware k zamezení snímání záření monitoru. Nic nekončí u záření monitoru, cenné informace lze získat i vedlejším zvukovým kanálem. Adi Shamir a Eran Tromer popisují ve svém dokumentu[5] tzv. akustickou kryptoanalýzu.

Pokud bude mít vetřelec fyzický přístup k počítači a bude chtít nad ním získat kontrolu (a nebude zrovna moci ho úplně odcizit či rozebrat) je nejjednodušším způsobem zavedení vlastního systému z CD-ROM, využitím disketové mechaniky, nebo jiných zařízení. Tomuto lze zabránit vícero způsoby. Prvním může být změna nastavení paměti CMOS tak, aby stroj bootoval z disku, kde máme systém a následné nastavení hesla CMOS. Pokud vetřelec nemůže uskutečnit fyzický zásah do útrob počítače a heslo CMOS vyřadit z činnosti vyjmutím baterie z CMOS nebo speciálním přepínačem na desce, je toto řešení bezpečné. Druhým způsobem obrany proti zavedení vetřelcova systému je vyjmutí samotných mechanik či úschovou počítače do uzamčené skříně (případy, kdy je server zavřen v trezoru a obsluha má k dispozici jen vstupní, výstupní zařízení, už nejsou tak neojedinělé). Tímto se chráníme i proti úmyslnému restartování stroje, ovšem je potřeba také ošetřit možnost restartování pomocí kombinace kláves `Ctrl+Alt+Del`, která může být zneužita běžným uživatelem.

Zablokování kombinace kláves `Ctrl+Alt+Del` se děje přes soubor `/etc/inittab`. Stačí zakomentovat řádek:

```
ca::ctrlaltdel:/sbin/shutdown -t5 -r now
```

Nebo změnit akci, která se při stisku této kombinace vykoná (např. zaznamenání do syslogu nebo jiná forma upozornění administrátorovi).

Další nebezpečí fyzického přístupu číhá v zneužití jednouživatelského módu.

[1] http://www.applied-math.org/optical_tempest.pdf"
[2] http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-577.pdf
[3] http://www.cl.cam.ac.uk/~mgk25/ieee02-optical.pdf
[4] http://eckbox.sourceforge.net/
[5] http://www.wisdom.weizmann.ac.il/ tromer/acoustic/

To je mód, kdy je umožněno využívat systém s právy superuživatele bez toho, aniž by bylo nutné zadat heslo. Tento mód se používá k opravným nebo ladícím akcím. V zavaděči LILO[6] si tento mód vynutíte zadáním `linux 1` nebo `linux single`. Tomuto lze zabránit jednoduše tak, že nikdo nebude moci v zavaděči zadat žádný příkaz. U zavaděče LILO toto provedeme doplněním řádku:

```
timeout=0
```

což nastaví nulový časový limit a řádek:

```
prompt
```

což zamezí automatickému zavedení systému po restartu. Druhou možností je chránit zavaděč heslem, což v zavaděči LILO provedeme přidáním řádků:

```
restricted
password=maminka
```

kde `maminka` bude přístupové heslo zavaděče.

Poslední situací týkající se fyzické bezpečnosti, kterou zde zmíním je ta, pokud se uživatel lokálně přihlásí na linuxovém stroji a opustí počítač, aniž by ho jakkoliv zajistil. Zajistit takový počítač lze univerzálně například použitím programu `xlock`, který umožní další práci se systémem až po zadání hesla, které patří ke kontu přihlášeného uživatele. Použít lze také aplikaci `xscreensaver`[7] s příznakem `-lock-mode`, což bude mít stejný efekt. Aplikace `xscreensaver` slouží jinak jako spořič obrazovky. Ovšem je potřeba si také dávat pozor na hrozbu simulovaného spořiče obrazovky, kdy záškodník upraví například použitý program `xlock`, kde doplní programový kód, který mu zajistí odeslání hesla po jeho zadání pravým uživatelem. Takto lze získat heslo i například pomocí simulovaného přihlášení.

V prostředí akademických sítí dochází k fyzickému kontaktu s počítači deno denně. Ať už jsou to správci, studenti nebo jiní zaměstnanci, všichni vytvářejí riziko, které je potřeba výše popsanými prostředky eliminovat.

[6] http://lilo.go.dyndns.org/
[7] http://www.jwz.org/xscreensaver/

Kapitola 6

Ukázkový příklad akademické sítě

V této kapitole budu demonstrovat to, jak může konkrétně vypadat taková malá akademická síť. Síť by měla mít svůj webový server, mail server a server pro databáze, který bude také sloužit jako úložiště dat. V síti by se dále měla nacházet počítačová laboratoř, dále místo, kde by se dal ethernetovým kabelem připojit notebook a také AP (Access Point) šířící bezdrátový signál po určité části budovy. Tato síť by měla mít vytvoření bezpečný komunikační kanál s jinou akademickou sítí. A také by měla umožnit především administrátorovi, bezpečně se připojovat na firewall a servery za ním za účelem správy. K dispozici je pouze jedna veřejná IP adresa.

A nyní již k přesnému popisu sítě, který vidíme na obrázku 6.1. Začnu firewallem, který odděluje naší síť od vnějšího prostředí. Tento stroj slouží zároveň jako firewall, router a proxy server. Má tři rozhraní. Prvním je `eth0`, které směřuje ven ze sítě. Druhým rozhraním je `eth1`, které uvozuje demilitarizovanou zónu (DMZ) a třetí rozhraní `eth2`, které směřuje do druhé části vnitřní sítě, odkud se připojují pracovní stanice. Tento stroj bude z Internetu viditelný pod přidělenou veřejnou IP adresou – v našem případě 195.168.3.87. Z vnitřní sítě bude přístupný také pod IP adresou 172.20.20.20. Vidíme, že ve vnitřní síti používám adresní rozsah privátních adres třídy B, která má rozsah 172.16.0.0-172.31.255.255, což pro naší síť bude dostatečné nadimenzování. Paketový filtr je založen na `iptables`, jak jsem popsal v sekci "4.2.2 iptables". Kromě filtrování běžných protokolů by měl obsahovat ochranu proti IP spoofingu, SYN floodingu, ICMP floodingu, apod. Můžeme odfiltrovat ICMP ECHO REQUEST pakety a tím udělat firewall částečně neviditelný. Ve skriptu s firewallem mohou být ještě některé další prvky, které zde ještě zmíním. Na tomto systému také necháme běžet HTTP proxy server Squid, kterým budeme moci kontrolovat HTTP přenos a popřípadě ho blokovat. Je potřeba přesměrovat odchozí HTTP požadavky s výjimkou lokálního stroje na port 3128, kde naslouchá Squid. Příkaz pomocí `iptables` by vy-

padal takto:

```
/usr/sbin/iptables -t nat -A PREROUTING -p tcp -i ! eth0 -d ! 195.168.3.87
-dport 80 -j REDIRECT -to-port 3128
```

Funkci proxy cache bych zde nezaváděl, protože se to u takovéto relativně malé sítě nevyplatí. Na tomto serveru by jinak neměly běžet žádné síťové služby a měly by zde platit přísná bezpečnostní pravidla utvrzená prostředky popsanými v kapitolách "2 Základní hardening GNU/Linuxu" a "3 Pokročilý hardening GNU/Linuxu". Firewall je také vyzbrojen aplikací Snort, která je nakonfigurována tak, aby se chovala jako adaptivní firewall. Snort dokáže podchytit i útoky červů (angl. *worms*), což ochrání windowsové systémy ve vnitřní síti, které jsou k tomu zvláště náchylné. V případě potřeby lze se Snortem sledovat ostatní (běžný) provoz.

Na firewall se půjde přihlásit pouze ze stroje s IP adresou 172.20.20.31, který bude sloužit především jako server pro sdílení souborů NIS, print server nebo file server. Budou zde umístěny domovské adresáře uživatelů, důležité konfigurační soubory a další potřebné služby a soubory pro správný běh vnitřní sítě. To že se na server dá připojit jen z tohoto počítače, zajistí `iptables` a TCP wrappers, které ostatním počítačům zakáží přístup. Přístup k tomuto počítači (a vlastně i ostatním serverům) z vnější sítě bude zajištěn přesměrováním portu 2222 z firewallu na port 22 tohoto stroje:

```
/usr/sbin/iptables -t nat -A PREROUTING -p tcp -d 195.168.3.87 -dport 2222
-j DNAT -to 172.20.20.31:22
```

Což umožňuje vnějším počítačům použít ssh ke spojení se s tímto strojem. Odtud už se lze přihlásit na všechny servery včetně firewallu. Tento počítač je vybaven systémem grsecurity, který velkou měrou podpoří bezpečnost na úrovni OS (lokální bezpečnost). Nedůvěryhodné uživatele zamkneme do chroot prostředí, zakážeme jim kompilování a spouštění vlastních binárních souborů, neuvidí běžící procesy ostatních uživatelů atd.

Dále si všimněme, že mezi naší a nějakou další akademickou sítí je vytvořena VPN pomocí IPSec, což zabezpečí čtení a pozměnění dat, ke kterému by mohlo dojít při přenosu dat v Internetu. Protože se stále pohybujeme v sítích, kde je používána IPv4, použili bychom k vytvoření IPSec tunelu projekt Openswan.

Dalším počítačem v DMZ je server sloužící jako webový a poštovní server. Coby webový server bude přístupný opět přes přesměrování na firewallu, což provedeme přes `iptables` (konkrétně přes DNAT aplikovaný na řetězec PREROUTING), abychom měli veškerou konfiguraci konzistentní. Webový server by

zajišťoval Apache[1] s příslušnými bezpečnostními moduly jako `mod_dosevasive`[2] a `mod_security`[3]. Poštovní server by měl zajišťovat elektronickou poštu skrze protokoly SMTP, POP3, IMAP a jejich šifrované varianty, k čemuž bych použil OpenSSL. Aby byl umožněn výběr pošty i uživatelům vně sítě, je potřeba opět tyto služby přesměrovat. V případě POP3s a IMAPs musíme dát navíc k dispozici použitý certifikát. Jako poštovní server bych doporučoval použít Postfix[4] v kombinaci s antivirem ClamAV nebo NOD32.

Na všech těchto serverech je samozřejmostí kontrola integrity systémových souborů pomocí projektu Tripwire[5] (lze použít i projekty Aide[6] nebo Afick[7]). Systémové logy[8] pouze pravidelně sledujeme a mažeme (tuto práci nám zjednoduší program logrotate). Nejideálnější by ovšem bylo směrovat logy na jiný počítač v síti po jednosměrném spojení, kde by je nemohl nikdo modifikovat. My můžeme směrovat logy na server s IP adresou 172.20.20.31, ale toto spojení je obousměrné, takže pokud někdo nad tímto počítačem získá kontrolu, může logy upravit nebo smazat. Vytvoříme tím určitou bariéru, takže smysl to určitě neztrácí.

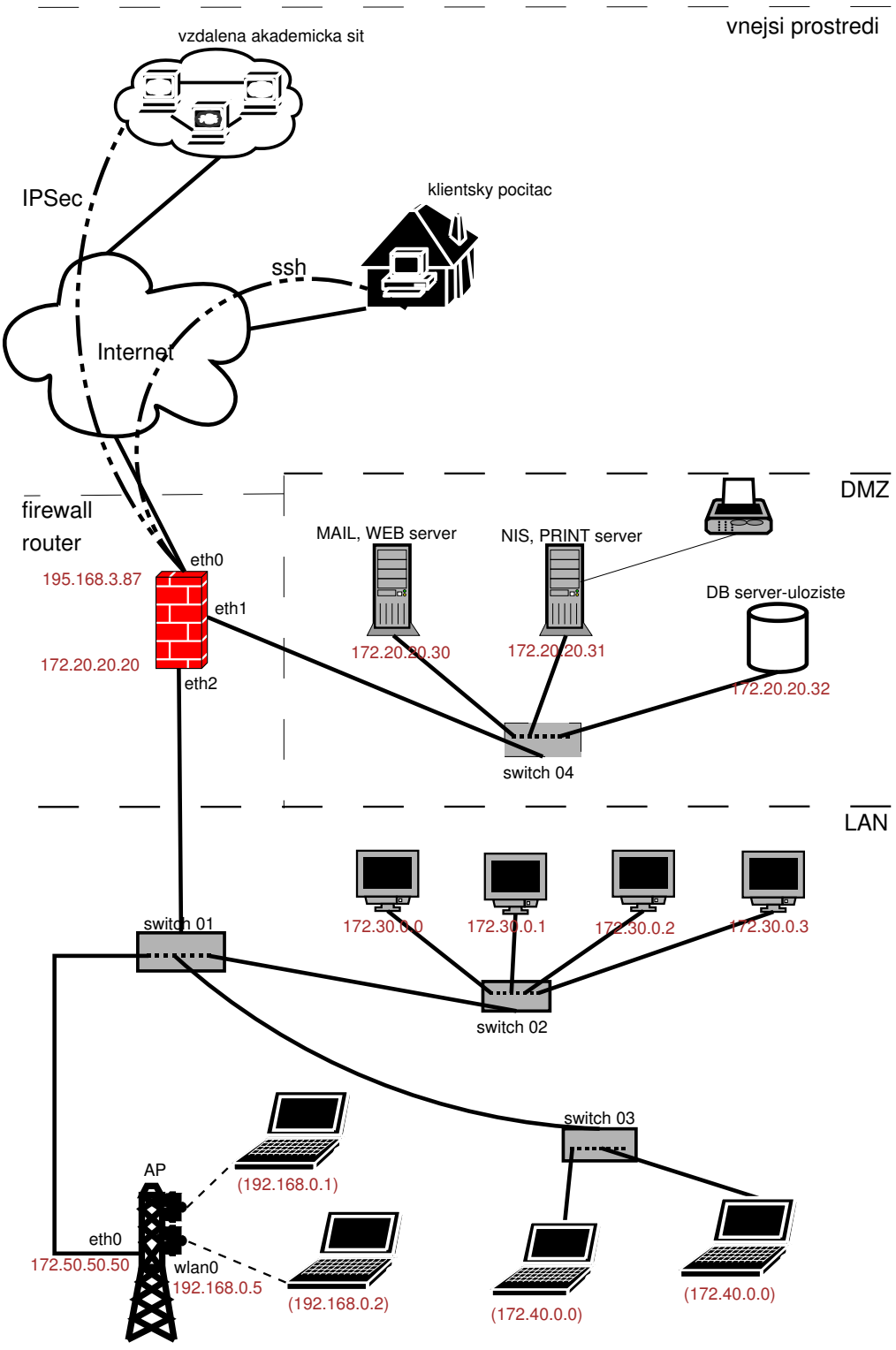
Na serveru 172.20.20.31 tedy zajistíme především integritu dat a samozřejmě ostatní zabezpečení lokální bezpečnosti.

Na přepínač s číslem dvě jsou napojeny pracovní stanice. Tyto počítače mají stejně jako servery pevně nastavenou IP adresu. O zabezpečení pracovních stanic sem v této práci již psal.

Přepínač s číslem tři je volný pro připojování přenosných počítačů. Těmto počítačům bude umožněno získat IP adresu dynamicky přes DHCP.

K přepínači číslo 1 je také připojen AP (Access Point), kterým rozvádíme bezdrátový signál. Zde bychom ovšem DHCP povolovat neměli. A měli bychom dodržet další zásady, které jsem probíral v úvodu kapitoly "4 Bezpečnost na úrovni sítě".

[1] http://www.apache.org/
[2] http://www.nuclearelephant.com/projects/dosevasive/
[3] www.modsecurity.org
[4] http://www.postfix.org/
[5] http://www.tripwire.org/
[6] http://www.cs.tut.fi/~rammer/aide.html
[7] http://prdownloads.sourceforge.net/afick/
[8] Problematikou linuxových logů sem se v této práci nezabýval.



Obrázek 6.1: Ukázkový příklad akademické sítě

Kapitola 7

Závěr

Rešeršní práce nás provedla možnostmi ohrožení OS GNU/Linux a samozřejmě také prostředky k jejich eliminaci. Práce se věnuje výhradně operačnímu systému GNU/Linux, ale mnoho z těchto informací má obecnou platnost. Vidíme, že i administrátor má prostředky, které může účinně využít v boji proti hrozbám, kterým jsou jím spravované počítače, denodenně vystavovány. Zjistili jsme také, že pomocí programového vybavení typu Open Source, lze vytvořit bezpečnostní bariéry velmi levně (nebo úplně zadarmo), což se v akademickém prostředí jistě hodí. Díky otevřenosti zdrojového kódu je navíc možno tyto prostředky snadno rozšiřovat a vylepšovat. V běžných sítích je zvykem vytvořit pevnou bezpečností politiku, která by měla být striktně dodržována. Ovšem, jak jsem již řekl, v prostředí akademických sítí dochází ke zvýšené fluktuaci zaměstnanců, což nevytváří optimální podmínky k jejímu dodržování.

Práce tvoří informačně velmi rozsáhlého průvodce problematiky bezpečnosti od nativních bezpečnostních prostředků systému GNU/Linux až po komplexní softwarová řešení. Tuto práci dále rozšiřují odkazy na další zdroje informací, které provázejí celou práci a zvyšují tím její informační hodnotu. Popisovány jsou prostředky, ke kterým často nejsou k dispozici kvalitní informační zdroje v češtině – například šifrované souborové systémy či popis bezpečnostních problémů Wi-Fi, o kterých se v práci několikrat zmiňuji. Práci doplňuje mnoho příkladů, které pomáhají k pochopení vysvětlované problematiky. Mezi stěžejní příklady patří kompletní vysvětlení skriptu, který zajisťuje firewall pomocí iptables a také ukázkový příklad školní sítě, který završuje celou práci.

Je tu ovšem několik věcí, které se již do této práce nevešly a je potřeba je alespoň zmínit. V první řadě zde nebyla rozebírána bezpečnost subsystémů (FTP server, web server, name server, samba server, atd.), jejíž popis by vydal minimálně na práci stejně obsáhlou, jako je tato.

Jen lehce sem se v této práci zmínil o logování. Logování je silný prostředek nejen pro ladění konfigurace například síťových démonů, ale sledovány by měli být především proto, abychom se ujistili o tom, zdali je na našem systému vše v pořádku (myšleno především z pohledu bezpečnosti). Logy může útočník při získání potřebných privilegií měnit – může je jednoduše odstranit a nebo modifikovat tak, že budou vypovídat o nepravdivých aktivitách. Proto je nejlepší směrovat logy na centrální log server. K tomuto serveru by měl být znemožněn přístup útočníkovi pomocí speciální hardwarové úpravy. Tohoto lze dosáhnout pomocí speciálních hardwarových řešení, která zde nebudu popisovat. Logy mají mnoho podob – binární nebo textové. V GNU/Linuxu většinou zajišťuje správu logů démon `syslogd`, který je již velmi zastaralý a jeho úlohu by měl nahradit novější Socklog[1]. Pro filtrování logu lze použít utilitu logcheck[2].

Dalším preventivním prvkem, kterým lze předejít mnoha problémům spojených z bezpečností je zálohování. Zálohovat by se měly především potřebně systémové soubory nutné k chodu systému. O tato data lze přijít mnohými způsoby a jedním z nich může být i činnost vetřelce. Ten nemusí ani data přímo mazat, ale pouze modifikovat – například umisťovat konfigurační backdoory. Návrat do původního stavu může být takový, že administrátor přepíše soubory zálohovanými soubory z doby, o které ví, že bylo vše v pořádku, místo toho, aby přepisoval soubory ručně. Vydařený popis toho, jak optimálně navrhnout politiku zálohování je v knize [8].

Ale jak zjistit, že někdo určité konfigurační nebo jiné systémové soubory modifikoval? Samozřejmě pravidelnou kontrolou integrity těchto souborů. K těmto účelům slouží již zmiňované projekty Tripwire, Aide či Afick.

Všimněme si, že toto jsou všechno obrané mechanismy, které nám ukazují na to, že útočník už jistá (klidně ta nejvyšší) privilegia na námi spravovaném systému má. O co se vlastně snaží útočník, pokud získá kontrolu nad systémem? Především o to, aby si zajistil pohodlný přístup a byl skryt před očima administrátora – tzn. aby nebyly vidět jeho soubory, procesy, atd. Zajištění přístupu lze provést opravdu mnoha způsoby a záleží jen na ostražitosti administrátora, jestli si jich všimne. Všimli byste si například toho, že nejste jediný, kdo má v souboru `/etc/passwd` hodnotu UID rovnu 0, nebo že vám na portu 5002 naslouchá rootovský shell? Pokud kontrolujete integritu souborů, tak jistě ano. Ovšem existují mnohem sofistikovanější metody a zde, jak se říká, se fantazii meze nekladou. Za zmínku jistě stojí LKM (Loadble Kernel Modules), s jejichž pomocí lze převést přesměrování systémových volání jádra a proto se zvláště hodí pro tvorbu kernelových rootkitů[3]. Jmenovitě například adore, knark, kis

[1] http://smarden.org/socklog/

[2] http://www.linux-sxs.org/files/psionic/logcheck-1.1.1.tar.gz

[3] Programové vybavení pro skryté ovládání operačního systému.

či specifický suckit. Obranou proti zavedení těchto rootkitů, je používání jádra, ve kterém není podpora zavádění modulů, což nás může ovšem velmi omezovat. Navíc tímto způsobem nezabráníme v zavedení rootkitu suckit, který využívá `/dev/kmem`. Účinnou metodou proti těmto známým rootkitům je použití programů, které je dokáží odhalit – například chkrootkit[4] či kstat[5].

Nejen odstraňování následků útoků by mělo být součástí dané bezpečnostní politiky sítě. Je ovšem potřeba, aby někdo dohlížel na její dodržování, což v prostředí akademických sítích je značně obtížné z důvodu fluktuace zaměstnanců.

V práci také nezmiňuji techniky, které se používají pro zkušební audit sítě nebo techniky forensní analýzy, která se používá pokud k nějakému bezpečnostnímu incidentu dojde. Tyto techniky už se netýkají samotného zebezpečování systémů, jako spíše analýz, které se provádějí po procesu zabezpečení systému.

S normami v oblasti ochrany informací se setkáme především v mezinárodní podobě. Snaží se o jistou normalizaci v oblasti ochrany informací. Největší institucí zabývající se normami v oblasti informační bezpečnosti je ANSI (Americký národní úřad pro normalizaci), která se zabývá především sběrem a tříděním amerických i zahraničních norem. Mezi další instituce patří jmenovitě CCITT (Mezinárodní poradní sbor pro telefonii a telegrafii), IEEE (Společnost pracovníků v elektronice a elektrotechnice), ISO (Mezinárodní organizace pro normalizaci), ECMA (Sdružení evropských výrobců počítačů), IFIP (Mezinárodní federace pro zpracování informací), NCSC (Národní centrum pro počítačovou bezpečnost) a další.

Do budoucna lze očekávat zvýšený zájem o počítačovou bezpečnost. Jistě velký rozmach v oblasti bezpečnosti (nejen v té počítačové) nastal po událostech z devátého září 2001. Je však zřejmé, že hackeři (jakkoliv je nazýváme) jsou vždy o krok před správci. Jak jinak by bylo možné, aby se seznamu tajných agentů rozmístěných na letištích v souvislosti s teroristickými útoky na mrakodrapy Twins, zmocnil pár mladých kluků, kteří se přezdívali "The Deceptive Duo"[6]. Tím ukázali, že celá informační bezpečnost Amerických vládních organizací je pouze fraška. Tato skupina měla na svědomí webové stránky mnoha amerických vládních institucí a jasně upozorňovala na to, že počítačová bezpečnost rozhodně není na úrovni, na které by být měla. Jeden mladík z této dvojky byl zatčen FBI v květnu 2004[7]. Běžným jevem je to, že chyby se objevují s příchodem nových technologií a produktů. Proto je potřeba dbát na bezpečnost již v době jejich vývoje.

[4] http://www.chkrootkit.org/
[5] http://www.s0ftpj.org/en/site.html
[6] http://www.zone-h.org/defaced/2002/04/24/extra-cas.faa.gov/
[7] http://www.securityfocus.com/news/8559

Literatura

[1] kolektiv autorů: Linux Dokumentační Projekt 3. aktualizované vydání, Computer Press, Praha 2003, ISBN 80-7226-761-2

[2] Bob Toxen: Bezpečnost v Linuxu, Computer Press, Brno 2003, ISBN 80-7226-716-7

[3] Libor Dostálek, Alena Kabelová: Velký průvodce protokoly TCP/IP a systémem DNS, Computer Press, Praha 2000, ISBN 80-7226-323-4

[4] Libor Dostálek a kolektiv: Velký průvodce TCP/IP Bezpečnost, Computer Press, Praha 2003, ISBN 80-7226-849-X

[5] Jiří Přibyl, Jindřich Kodl: Ochrana dat v informatice, Vydavatelství ČVUT, Praha 1996, ISBN 80-01-01664-1

[6] Brian Hatch, James Lee, George Kurtz: Linux Hackerské Útoky, SoftPress, Praha 2002, ISBN 80-86497-17-8

[7] Matthew Strebe, Charles Perkins: Firewally a proxy-servery Praktický průvodce, Computer Press, Brno 2003, ISBN 80-7226-983-6

[8] Vicki Stanfield, Roderick W. Smith: Správa operačního systému Linux, SoftPress, Praha 2002, ISBN 80-86497-25-9

[9] Michal Maťejka: Počítačová kriminalita, Computer Press, Brno 2002, ISBN 80-7226-419-2

[10] Simson Garfinkel: PGP: Pretty Good Privacy, Computer Press, Praha 1998, ISBN 80-7226-054-5

[11] SecurityFocus, online na Internetu: http://www.securityfocus.com/

[12] ROOT, online na Internetu: http://www.root.cz/

[13] AbcLinuxu, online na Internetu: http://www.abclinuxu.cz/

[14] LinuxZone, online na Internetu: http://www.linuxzone.cz/

[15] BlackHole, online na Internetu: http://www.blackhole.sk/

[16] Penguin, online na Internetu: http://www.penguin.cz/

[17] Linux security, online na Internetu: http://www.linuxsecurity.com/

[18] SecuriTeam, online na Internetu: http://www.securiteam.com/

[19] Packet Storm, online na Internetu: http://packetstormsecurity.nl/

[20] Security-protocols, online na Internetu: http://security-protocols.com/

[21] Hackwire, online na Internetu: http://www.hackwire.com/

[22] Underground, online na Internetu: http://www.underground.cz/

[23] Hysteria, online na Internetu: http://www.hysteria.sk/

[24] Phrack, online na Internetu: http://www.phrack.org/

[25] WiFi ONLINE, online na Internetu: http://www.wifionline.net/