

Implementace UNIX systémů do bezpečnostních technologií

Implementation of UNIX systems in to security technologies

Richard Bachánek

Bakalářská práce
2012



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2011/2012

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Richard BACHÁNEK**
Osobní číslo: **A09203**
Studijní program: **B 3902 Inženýrská informatika**
Studijní obor: **Bezpečnostní technologie, systémy a management**

Téma práce: **Implementace UNIX systémů do bezpečnostních technologií**

Zásady pro vypracování:

1. Popište problematiku bezpečnosti operačních systémů.
2. Zhodnoťte bezpečnost síťových protokolů.
3. Povedte návrh připojení detektorů k PC.
4. Navrhněte komunikační část bezpečnostního systému.
5. Uvedený návrh softwarově realizujte.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. Komunita Linuxu, Linux dokumentační projekt 4. vyd. Praha: Computer Press, 2008. 1336 s. ISBN 978-80-251-1525-1
2. KRČMÁŘ, P.: Linux-postavte si počítačovou síť 1. vyd. Praha: Computer Press, 2008. 184 s. ISBN 978-80-247-1290-1
3. CONWREL, P., ZONG, L.: Linux, Thomb. Diath. Haemorrh., USA, 2009; 19:p. 186-19, ISBN 978-80-7318-707-1
4. ECKEL, Bruce. Myslíme v jazyku C. Vyd. 1. Praha: Grada, 2000, 554 s. ISBN 80-247-9009-2
5. ECKEL, Bruce. Myslíme v jazyku C. Vyd. 2. Praha: Grada Publishing, 2006, 608 s. ISBN 80-247-1015-3

Vedoucí bakalářské práce:

Ing. Ján Ivanka

Ústav bezpečnostního inženýrství

Datum zadání bakalářské práce:

24. února 2012

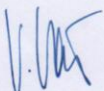
Termín odevzdání bakalářské práce:

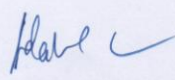
25. května 2012

Ve Zlíně dne 24. února 2012



L.S.


prof. Ing. Vladimír Vašek, CSc.
děkan


doc. Mgr. Milan Adámek, Ph.D.
ředitel ústavu

ABSTRAKT

Bakalářská práce seznamuje odbornou veřejnost s problematikou softwarového zabezpečení operačních systémů. Objasňuje problematiku licencí v systému Windows a Linux. Předložená práce popisuje výhody nasazení UNIX systémů oproti konkurenčním operačním systémům, bezpečnostní rizika spojená s používáním operačních systémů a nasazení UNIX-ových protokolů při propojování počítačů. Součástí práce je charakteristika protokolů SSH, NFS, FTP a jejich využití v bezpečnostních systémech. Praktická část bakalářské práce řeší nový způsob propojení budov s mobilními zařízeními, jako jsou notebooky nebo mobilní telefony, za účelem neustálého sledování aktuálního stavu senzorů v chráněném objektu.

Klíčová slova: UNIX, Windows, Linux, SSH, NFS, FTP.

ABSTRACT

Bachelor's dissertation introduces professional public with issues of software security in operating systems. It clarifies issues of licenses in operating systems Windows and Linux. Theese dissertation describes benefits of usig a UNIX systems against competitive solution, security risks in operating systems and using UNIX protocols to connect between computers. Dissertation characterizes protocols SSH, NFS, FTP a describes their use in security systems. Practical part of dissertation describes new way in connection a buidings with mobile devices such as notebooks or mobile phones for constant monitoring current status of sensors in protected object.

Keywords: UNIX, Windows, Linux, SSH, NFS, FTP.

Rád bych tímto poděkoval svému vedoucímu bakalářské práce Ing. Jánů Ivankovi za ochotu, pomoc, cenné rady, připomínky, za věnovaný čas k úpravě a návrhům formy zpracování bakalářské práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému, dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....

podpis diplomanta

OBSAH

| | |
|-----------------------------------------------------------|-----------|
| ÚVOD | 9 |
| I TEORETICKÁ ČÁST | 11 |
| 1 LINUX V BEZPEČNOSTNÍCH TECHNOLOGIÍCH | 12 |
| 1.1 BEZPEČNOST LINUXU | 13 |
| 1.2 JÁDRO..... | 16 |
| 1.2.1 Tuning jádra..... | 17 |
| 2 BEZPEČNOST LINUXU NA JEDNOTLIVÝCH ÚROVNÍCH | 20 |
| 2.1 UŽIVATELSKÁ ÚROVEŇ..... | 20 |
| 2.2 APLIKAČNÍ ÚROVEŇ | 22 |
| 2.3 ÚROVEŇ JÁDRA..... | 22 |
| 3 SECURE SHELL | 23 |
| 3.1 BEZPEČNOST POČÍTAČOVÝCH SÍTÍ | 23 |
| 3.2 SECURE SHELL | 24 |
| 3.2.1 SSH a RSA autentizace..... | 25 |
| 3.2.2 SSH a bezpečnost | 27 |
| 3.2.2.1 Vnější útok | 27 |
| 3.2.2.2 Dodatečné a speciální zabezpečení | 29 |
| 3.2.2.3 Řízení přístupu..... | 29 |
| 4 FILE TRANSFER PROTOCOL | 31 |
| 4.1 BEZPEČNOST FTP..... | 31 |
| 4.2 SSH FILE TRANSFER PROTOCOL | 31 |
| 4.3 KONFIGURACE FTP SERVERU | 32 |
| 4.3.1 Konfigurace Pure-ftpd..... | 32 |
| 5 NETWORK FILE SYSTEM | 33 |
| 5.1 KONFIGURACE SERVERU..... | 33 |
| 5.2 KONFIGURACE KLIENTSKÉHO POČÍTAČE | 34 |
| 5.3 NFS A BEZPEČNOST..... | 35 |
| 6 SAMBA..... | 36 |
| 6.1 KONFIGURACE SAMBA SERVERU | 36 |
| II PRAKTICKÁ ČÁST | 38 |
| 7 LINUX JAKO ŘÍDÍCÍ SYSTÉM | 39 |

| | | |
|----------|-------------------------------------------------|-----------|
| 7.1 | PROPOJENÍ SENZORU S POČÍTAČEM | 39 |
| 7.2 | SENZORY JAKO MINIPOČÍTAČE | 42 |
| 7.3 | ARDUINO | 42 |
| 7.3.1 | IDE Processing | 47 |
| 8 | DEMONSTRAČNÍ PROGRAM | 48 |
| 8.1 | POPIS DEMONSTRAČNÍCH PROGRAMŮ | 50 |
| | ZÁVĚR | 53 |
| | ZÁVĚR V ANGLIČTINĚ | 54 |
| | SEZNAM POUŽITÉ LITERATURY | 55 |
| | SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK | 57 |
| | SEZNAM PŘÍKAZŮ | 59 |
| | SEZNAM OBRÁZKŮ | 60 |
| | SEZNAM TABULEK | 61 |
| | SEZNAM PŘÍLOH | 62 |

ÚVOD

V dnešní době se počítače staly nepostradatelnou součástí běžného života. Setkáváme se s nimi denně a doslova všude. Počítače jsou v televizích, mobilních telefonech, dokonce i autech. Těžko už si představíme bankovní systém bez této technologie. Vědecké výpočty, správa dat, evidence, nákupy – to všechno se bez počítačů již neobejde, respektive – i kdyby ano, byly by běžné činnosti daleko složitější a proč je provádět a ztrácet tak čas, když za nás vyřeší běžné úkony výpočetní technika. Nicméně s velkým potenciálem, který nám bezesporu technika nabízí, přichází také velká zodpovědnost. Tím je myšleno zabezpečení, které se stává zas nedílnou součástí softwarového vybavení našich počítačů. Minulost ukázala, že ulehčování života těmito přístroji může mít katastrofální důsledky. Bez šifrování dat by nemohlo fungovat žádné internetové bankovníctví. Bez firewallů by byly počítačové sítě zranitelné a dostal by se do nich každý amatérský útočník, následně by pak mohl snadno převzít kontrolu nad všemi počítači zapojenými ve společné síti. Problematice softwarového zabezpečení se práce bude věnovat v celé teoretické části. Jde přece o jednu z nejdůležitějších zásad v každém počítači. Jak již bylo zmíněno na začátku práce, počítače představují obrovský potenciál a s rychlostí, jakou průmysl informačních technologií (dále jen IT) roste, se tento potenciál exponenciálně zvyšuje. Současná technologie na bázi křemíku nám umožňuje taktovat procesor asi do 4 GHz, ale i tento, relativně slabý výkon nám umožňuje provádět i velmi náročné výpočty. S objevem grafenu, za který byla minulý rok udělena Nobelova cena, se výkon procesorů razantně zvyšuje, předpokládá se pracovní frekvence až 1 THz. Umělá inteligence, superrychlé „chytré“ telefony, to všechno se v budoucnu může stát naprosto běžným faktem.

Od samotného začátku směřovaly osobní počítače k určité otevřenosti, jednoduchosti, k co největším schopnostem rozšíření a univerzálnosti. Na bezpečnost se pohlíželo jako na druhořadou. Nikdo si jistě před 30. lety nedokázal představit, jaké množství dat se bude v budoucnu na počítačích zpracovávat, jaký zlom bude mít příchod internetu a různých sítí. Neuvědomovali jsme si, že data znamenají moc, peníze a vědomosti. Z této problematiky se nakonec vyvinul obor počítačové bezpečnosti. Jde o obor, který se zabývá ochranou informací před neoprávněným přístupem, zahrnuje obranu před počítačovými viry, neodbornou obsluhou a samozřejmě zloději – třeba špionáž, vedená konkurenční firmou.

Cílem bakalářské práce je poukázat v první řadě na bezpečnost počítačových systémů a potenciál, který nám osobní počítače nabízí k ochraně našeho majetku. Se správným softwarovým a hardwarovým vybavením můžeme mít trvale pod kontrolou náš majetek, nemovitosti, naše blízké. Konkrétně se práce zabývá propojením bezpečnostního systému chráněné budovy s mobilním zařízením, které může představovat notebook, tablet či chytrý mobilní telefon. Pro tyto účely jsem se rozhodl využít platformu Linux a síťový protokol SSH. Jeho hlavní výhodou je vysoká míra bezpečnosti a relativně jednoduchá konfigurovatelnost. Ukázkové programy budou programovány v jazyce C++ a GNU/Bash shell. Práce dále poukazuje na bezpečnost celého spojení a naznačuje potenciál, který by nám tento systém mohl dát ve spojení s inteligentními budovami.[1]

I. TEORETICKÁ ČÁST

1 LINUX V BEZPEČNOSTNÍCH TECHNOLOGIÍCH

Důvod, proč kombinovat operační systém (dále jen OS) Linux s bezpečnostními technologiemi, je především vysoká míra jeho bezpečnosti a v neposlední řadě i licence, která umožňuje šířit některé distribuce zdarma. Tento OS nám dává samozřejmě další výhody. Ve většině případů jsou distribuce Linuxu šířeny volně se zdrojovým kódem, což umožňuje jeho uživatelům zasahovat do samotného systému. Vývojáři tímto dávají uživatelům naprostou volnost. Kdokoliv tedy můžeme systém pozměnit přesně dle svých potřeb. Není problém přizpůsobit jádro systému na konkrétní hardware, následně ho zkompileovat a zavést s ním systém. Výsledkem pak bude OS přesně optimalizovaný pro konkrétní hardware a určité specifické potřeby.

Linux je tedy šířen pod určitými licencemi. Nejčastěji je to GNU General Public License (dále jen GPL), kde GNU je projekt inspirovaný unixovým systémem, který se zaměřuje na vývoj svobodného OS. Licence říká, že kdokoli bude chtít, má právo Linux modifikovat a změněnou verzi může dále distribuovat – s jedinou podmínkou: že dá k dispozici změněný zdrojový kód. V praxi to znamená, že si můžeme stáhnout zdrojový kód jádra, doplnit do něj třeba nástroj pro cestování časem. Změněný kód pak můžeme prodávat, ale musíme zákazníkovi poskytnout upravený zdrojový kód. Další nespornou výhodou je, že Linux dokáže fungovat trvale, bez nutnosti restartu. Proto se může řada úloh provádět v noci, nebo v automaticky plánovanou dobu. Poslední velkou výhodou, kterou stojí za to zmínit, je škálovatelnost. Můžeme mít mobilní telefon se 2 MB paměti nebo server. Stačí přidat nebo odebrat určité balíčky a Linux bude fungovat na obou strojích. Není potřeba superpočítačů. Náročné úkoly může Linux plnit s použitím stavebních kamenů, které standardně obsahuje. Jakožto v každém operačním systému, i v Linuxu se samozřejmě vyskytují chyby. Ovšem z důvodů, že tento systém vyvíjejí a testují tisíce lidí, nebývá neobvyklé, že od ohlášení chyby k jejímu opravení uplynou pouze hodiny.

Jsou zde ovšem i nevýhody. Tou první může být velké množství distribucí. Na první pohled může uživatele velké množství distribucí zmást, ovšem zde záleží na úhlu pohledu. Každý může na druhou stranu najít přesně to, co mu vyhovuje. Důležité je, že se různé distribuce odlišují pouze v drobnostech, jako grafické rozhraní nebo určité specifické příkazy. Dále jsou zde například změny v balíčkových systémech, ale uživatel si rychle zvykne a udělá si obrázek, co mu více vyhovuje. Co se týče grafického prostředí (ve většině případů, jsou to

změny, které uvidíme na první pohled), jde jen o drobné změny například v menu, ve správě hardwaru. Hlavní ale je, že většina příkazů, které zadáváme do terminálu, je ve všech distribucích naprosto totožná.

Každý si teď jistě klade otázku – je možné Linuxu důvěřovat? Jde přece o projekt s otevřeným kódem, který je zdarma. Uživatelé Linuxu měli možnost se rozhodnout, jestli ho budou používat, nebo ne. Po delší době testování došla většina uživatelů k názoru, že Linux je nejen dobrý, ale v řadě případů dokonce lepší než tradiční řešení. Pro začínající uživatele je možná méně přehledný než Microsoft Windows a určitě složitější, než konkurenční Mac OS, ale vzhledem k jeho rostoucí popularitě se zvyšuje snaha ho co nejvíce přizpůsobit běžným uživatelům. Pokud by tedy Linux nebyl důvěryhodný, zmizel by již dávno a nikdy by nedosáhl své současné popularity. Každý uživatel může svůj systém ovlivnit. Sdílet své poznatky s komunitou, která systém denně vylepšuje. Linux je nikdy nekončící projekt, ale také projekt, který usiluje o dokonalost.[1]

1.1 Bezpečnost Linuxu

Vysoká míra bezpečnosti tohoto OS je bezesporu hlavní triumf oproti konkurenčním řešením. Na druhou stranu umožňuje nekonečnou konfigurovatelnou, a to dělá systém také zranitelným. Například nevhodné zásahy do systému se superuživatelským oprávněním znehodnocují původní bezpečné nastavení. Bez rootovských práv však nelze systém nijak poškodit, ani instalací balíků z repozitářů. Naopak se tak vyhneme instalaci malware (škodlivý software), protože v repozitářích jsou vždy prověřené aplikace komunitou. Díky tomuto nemůže systém poškodit ani začínající uživatel.

V Linuxu platí přísloví: „všechno je soubor nebo proces“. Na rozdíl od příbuzných operačních systémů jsou v Linuxu i jednotlivá vlákna normální procesy. Ty pak sdílí společné prostředky, s kterými manipuluje pouze jádro. Při nedostatku oprávnění, nebo systémových prostředků se požadavek samozřejmě neprovede. Pro jádro je směrodatné pouze to, který proces provedl systémové volání s požadavkem, což znamená, že se nezabývá tím, kdo u počítače sedí, ani kdo požadavek inicioval. Právě pro tyto účely si jádro uchovává v datových strukturách číselně reprezentovaná uživatelská a skupinová oprávnění, která daný proces požívá. I když uživatelské účty mají samozřejmě svá jména,

například „jan_novak“, převádějí se podle konfiguračních souborů na čísla – pro vnitřní potřeby systémů. To, že pracujeme (resp. jsme přihlášení) jako daný uživatel, znamená pouze, že shell (desktop apod.) běží s právy příslušného uživatele a nejedná se tedy o žádný „globální stav“ systému. V každé Linuxové distribuci vždy existuje nejvyšší uživatelské oprávnění – root, které je reprezentováno číslem 0. I když uživatel root nutně nemusí mít svůj uživatelský účet, některá oprávnění jsou přidělena pouze rootovy, což je natvrdo implementováno v jádře. Opakem uživatele root je obvykle uživatel nobody, který má naopak nejvyšší číslo. Skupiny jsou pak množiny uživatelů. Členství ve skupině přináší uživateli oprávnění přidělená většímu množství uživatelů. Takový systém je velmi výhodný například ve firmách, kde přidělujeme stejná oprávnění velkému množství uživatelů. Uživatelé v kancelářích musí mít jiná oprávnění, jako uživatelé pracující například ve skladu. Každý soubor v systému má svého vlastníka – tedy uživatele, který jej vytvořil, dále svou skupinu a práva. Přehledněji to lze vidět z následujících tabulek. První tabulka popisuje význam práv souborů.

Tab. 1. Význam práv souborů

| práva | význam | pro soubor | pro adresář |
|-------|---------|------------------|-----------------------|
| r | Read | Číst soubor | Vypsát obsah adresáře |
| w | Write | Zápis do souboru | Vytvářet nebo mazat |
| x | Execute | Spouštět soubor | Procházet adresářem |

Změna práv se provádí příkazem **chmod**. Prvním znakem se určuje, či práva budeme měnit:

Tab. 2. Změna práv

| | | |
|---|--------|----------|
| u | user | vlastník |
| g | group | skupina |
| o | others | ostatní |
| a | all | všichni |

Pro názorný příklad si nechme vypsát práva nějakého adresáře příkazem *ls*, například:

ls -laF (1)

Parametry za příkazem upřesňují filtrování výsledků. V tomto případě *-l* znamená podrobný popis včetně typu, práv, počtu pevných odkazů, jména vlastníka a skupiny, velikost a času založení. *-a* zobrazí i soubory a adresáře začínající tečkou, tedy skryté. Parametrem *-F* přidáváme ukazatel k zobrazeným záznamům. Následující obrázek jasně ukazuje práva a ostatní informace zobrazené příkazem (1).

```

Soubor  Upravit  Zobrazit  Hledat  Terminál  nápověda
drwxr-xr-x  3 root   root     4096 2011-03-03 15:16 ./
drwx-----  3 richie richie  4096 2011-03-03 15:22 .adobe/
-rw-----  1 richie richie 23515 2011-07-31 12:03 .bash_history
-rw-r--r--  1 richie richie   220 2011-03-03 14:30 .bash_logout
-rw-r--r--  1 richie richie  3353 2011-03-03 14:30 .bashrc
drwx----- 16 richie richie  4096 2011-08-02 09:35 .cache/
drwxr-xr-x  3 richie richie  4096 2011-07-10 20:20 .codeblocks/
drwx-----  3 richie richie  4096 2011-03-03 16:07 .compiz/
drwxr-xr-x 29 richie richie  4096 2011-06-26 18:58 .config/
-rw-----  1 richie richie  3932 2011-03-03 17:47 .conkyrc
drwx-----  3 richie richie  4096 2011-03-03 14:41 .dbus/
-rw-r--r--  1 richie richie    55 2011-08-02 09:35 .dmrc
drwxr-xr-x  2 richie richie  4096 2011-07-07 20:06 Dokumenty/
drwxr-x--  2 richie richie  4096 2011-04-08 22:57 .easytag/
-rw-r--r--  1 richie richie   889 2011-03-03 22:37 .eeschema
-rw-----  1 richie richie    16 2011-03-03 14:42 .esd_auth
drwx-----  7 richie richie  4096 2011-04-26 20:56 .evolution/
-rw-r--r--  1 richie richie   179 2011-03-03 14:30 examples.desktop
drwxr-xr-x  2 richie richie  4096 2011-03-06 12:43 .fontconfig/
drwx-----  5 richie richie  4096 2011-08-02 09:35 .gconf/
drwx-----  2 richie richie  4096 2011-08-02 10:12 .gconfd/
drwx-----  4 richie richie  4096 2011-03-06 12:43 .gegl-0.0/
drwxr-xr-x 22 richie richie  4096 2011-06-08 18:27 .gimp-2.6/
-rw-r-----  1 richie richie     0 2011-07-21 22:41 .gksu.lock
drwxr-xr-x  9 richie richie  4096 2011-08-02 00:00 .gnome2/
drwx-----  2 richie richie  4096 2011-03-03 15:29 .gnome2_private/
drwx-----  3 richie richie  4096 2011-03-03 22:21 .gnupg/
drwxr-xr-x  2 richie richie  4096 2011-06-09 14:27 .gststreamer-0.10/
-rw-r--r--  1 richie richie   695 2011-08-02 09:35 .gtk-bookmarks
-rw-r--r--  1 richie richie    50 2011-03-03 17:12 .gtkrc-2.0-kde4
dr-x-----  3 richie richie     0 2011-08-02 09:35 .gvfs/
-rw-r--r--  1 richie richie 88887 2011-04-29 18:41 hs_err_pid25371.log
drwxr-xr-x  2 richie richie  4096 2011-03-03 14:41 Hudba/
-rw-----  1 richie richie 60100 2011-08-02 09:35 .ICEauthority
drwxr-xr-x  3 richie richie  4096 2011-06-10 22:05 .icedteaplugin/
drwxr-xr-x  5 richie richie  4096 2011-03-03 15:33 .icons/

```

Obr. 1. Výpis práv souborů

Rozložení písmen může být trochu nepřehledné. Je nutné si představit práva jako trojice znaků, mimo první znak. Tab. 3 popisuje přehledné zobrazení po trojicích znaků.

Tab 3. Přehledné zobrazení práv souborů a složek

| Typ | Práva vlastníka | Práva skupiny | Práva ostatních |
|-----|-----------------|---------------|-----------------|
| d | rwX | r-X | r-X |

První sloupec je jednoznakový a určuje, o jaký typ souboru se jedná.

- je soubor

c je znakové zařízení (tiskárna...)

b je blokové zařízení (disky...)

d je adresář

l je link

Další tři sloupce již označují skupiny, kterým můžeme práva přiřazovat. Práva vlastníka určují, co se souborem může dělat vlastník souboru. Práva skupiny udávají, co se souborem mohou provádět ostatní členové skupiny, které soubor náleží. A nakonec jsou práva ostatních, kteří nepatří do skupiny, již soubor přísluší.[3][6][7]

1.2 Jádro

Jádro systému Linux je šířeno s otevřeným kódem a často vznikají jeho doplňky jako bezpečnostní modul SELinux, Apparmor. Uvedené moduly původní bezpečnostní model rozšiřují na MAC model. Pro běžného uživatele nepřináší takový model výhody, stejně jako například instalace antiviru nebo firewallu, neboť na Linuxu neexistují žádné životaschopné viry. Verze antivirů se nasazují většinou pouze na servery, kde slouží k detekci virů například v emailech. Firewall je zase integrovaný v jádru, přesněji v jeho síťovém subsystému, ale lze jej také samozřejmě konfigurovat.

Jádro systému je pravděpodobně nejsledovanější kód vůbec a většina chyb je objevena právě hackry při jejich studiu jádra. Závažné chyby mají většinou charakter, že přídatný software (známé případy jsou se SELinuxem a Wine) o jádře něco nepravdivě předpokládá; zneužití

vždy předpokládá splnění opomenutých velmi hypotetických podmínek. Chyby komunita velmi rychle záplatuje.[4]

1.2.1 Tuning jádra

Vývojáři jádra v čele s Linusem Torvaldsem jsou často velmi konzervativní. Jejich projekt se jmenuje „Vanilla“ – znamená to v podstatě, že tyto verze jsou velmi stabilní, ale často nepřinášejí žádné velké novinky. Je to ale logické. Jádro systému je velmi citlivá záležitost a do oficiální distribuce by se mělo dostat vždy stabilní a otestované jádro a pokud chceme něco navíc, pak na vlastní riziko. Zajímavou alternativou ke standardnímu jádru je projekt Zen kernel.

„Zen Kernel je výsledkem společné práce jaderných hackerů, kteří se snaží vytvořit nejlepší možné linuxové jádro. Zahrnujeme do něj kód, který není součástí hlavní větve jádra a snažíme se vytvořit po všech stránkách lepší jádro pro desktop. Přidáváme nové vlastnosti, podporujeme nejnovější hardware a zahrnujeme různý kód a optimalizace. Zen je sto procentně komunitní projekt, takže se každý může přidat“.¹

Jednou ze zajímavých funkcí, kterou nám Zen přidává, je Boost. Ten zvyšuje nice u prioritních procesů na 10. Nice je program, který mění prioritu při spuštění programu. Z programů se v Linuxu po spuštění stávají procesy. O běh každého procesu se pak stará jádro, to přidělí na základě jeho priority část paměťového prostoru. Linux je plně multistaskingový operační systém. V podstatě to znamená, že sice v systému běží spousta procesů najednou, ale procesor v daný okamžik zpracovává pouze jeden. Procesy tedy bojují o získání nejvyššího procesorového času. Zen dále vylepšuje standardní plánovač jádra a navíc k němu nabízí alternativu BFS, což je plánovač, který vyniká mnohem rychlejší odezvou. Zen plánovač, který se stará o zařízení, se nazývá BFQ I/O. Oproti standardnímu CFQ přiděluje zpracovatelné sektory, navíc jde výborně ladit hlavně díky spoustě nastavitelných parametrů. Výsledkem ladění by měla být samozřejmě rychlejší odezva a hladší běh. Je to poměrně elegantní řešení, které nám zajistí rychlejší odezvu při velké diskové I/O zátěži. Další funkcí, která stojí za zmínku, je Compsache. Tato vlastnost vytváří v paměti nové blokové zařízení, které se nazývá ramzswap. Do tohoto zařízení potom systém swapuje. Je to dobré k tomu, že data ramzswap automaticky komprimuje.

¹<http://www.root.cz/clanky/zen-kernel-vytunene-linuxove-jadro/>

Logicky je swapování do paměti podstatně rychlejší, než swapování na disk a navíc pak máme k dispozici více paměti. Zen kernel dále vylepšuje práci se senzory. Program, který se stará o zobrazení senzorů základní desky, se jmenuje lm-sensors. Jeho standardní výstup lze vidět na následujícím obrázku.

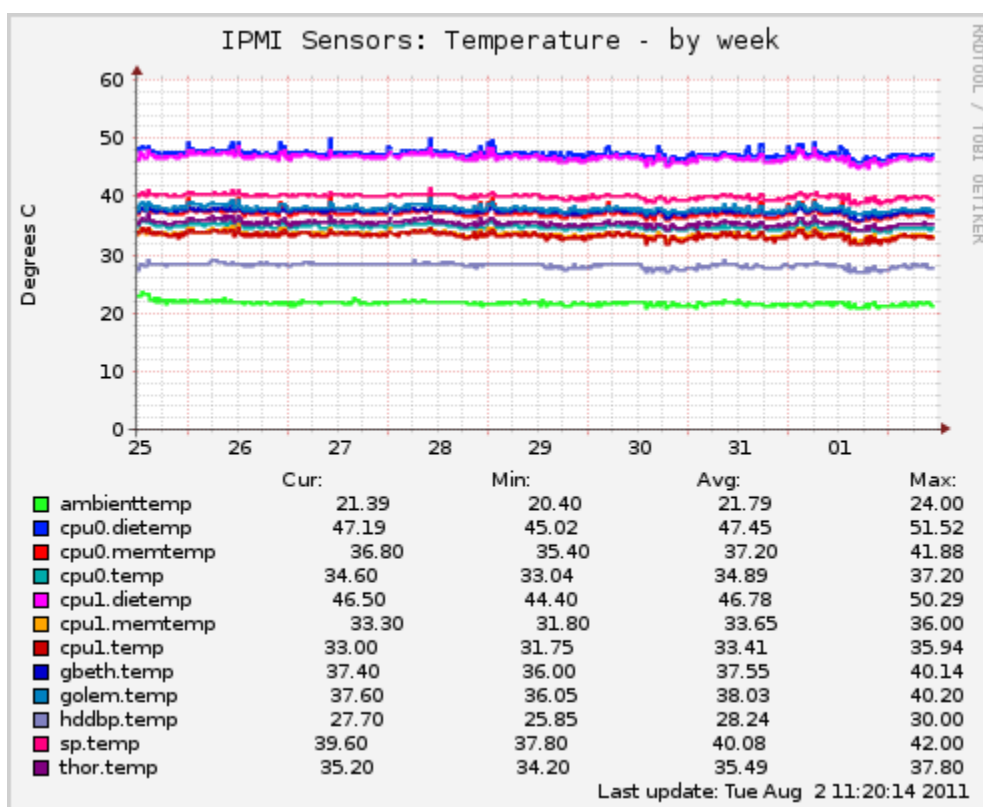
```

richie@ntbubuntu: ~
_Soubor _Upravit _Zobrazit _Hledat _Terminál _Nápověda
richie@ntbubuntu:~$ sensors
acpitz-virtual-0
Adapter: Virtual device
temp1:          +56.0°C (crit = +90.0°C)
richie@ntbubuntu:~$

```

Obr. 2. Výpis z lm-sensors

Zen přidává technologii IPMI. Uvedenou technologii pomáhali vyvíjet firmy Intel a HP a je poměrně hodně rozšířena. Informace které lze ze základních desek získat jsou potom výrazně detailnější.



Obr. 3. IPMI Sensors

Technologie linux-PHC umožňuje snížení napětí na procesoru. Tuto funkci využijeme převážně u notebooků, jelikož nám umožní snížení příkonu a tím i teploty, což povede k

výraznému prodloužení životnosti baterie – pochopitelně také ke snížení výkonu. Další technologií velmi užitečnou pro notebooky je TuxOnce. Je to vlastně vylepšená hibernace, která nám umožní hibernovat do běžného souboru.

Technologií, které nám toto jádro přináší je obrovské množství a nemá smysl je všechny jmenovat. Na závěr dodejme podporu velkého množství nových filesystemů a co se sítí týče, jde o rozšíření IPtables o podporu vrstvy Layer7-filter. Je pro nás výhodný hlavně proto, že rozšiřuje pravidla o aplikační vrstvu, tím pádem je schopen identifikovat protokoly podle obsahu paketů.[8]

2 BEZPEČNOST LINUXU NA JEDNOTLIVÝCH ÚROVNÍCH

2.1 Uživatelská úroveň

Nejslabším článkem celého systému je uživatel. Zde hraje významnou roli lidský faktor. Na nás, jakožto správcích systému, je zajistit, aby nedošlo k závažnému poškození systému nebo ztrátě citlivých dat v případě, že uživatelský účet útočník napadne. Toto můžeme zajistit správným přidělením uživatelských práv a pravidelným zálohováním. Neměli bychom uživateli přidělovat přístup na místa, která nejsou nezbytná pro jeho práci. Linux je víceuživatelský operační systém, proto musíme dbát na správné vytvoření bezpečnostní politiky. V případě většího množství uživatelů nám výrazně usnadní práci vytvoření uživatelských skupin a následné zařazení uživatelů do těchto skupin. V případě, že budeme chtít omezit práva většímu množství uživatelů, patřících do jedné skupiny, upravíme práva jen této skupině. Nemusíme pak měnit práva jednotlivým uživatelům. Extrémním případem by bylo ponechání super- uživatelských práv všem uživatelům. Předpokládejme, že všichni tito uživatelé by byli správci systému a perfektně mu rozuměli. Mohla by pak nastat situace, kdy by uživatel poškodil vědomě či nevědomě systém? V případě přihlášení do systému by takový uživatel spouštěl celý X server se super-uživatelskými oprávněními. Jedná se o tisíce malých či velkých programů, ve kterých se zcela jistě objevují drobné chyby a útočník, který by jich využil, by pak snadno získal veškerá práva potřebná k naprosté vládě nad systémem. Největší chybou začínajících uživatelů je lehkomyšlnost. Představa, že na počítači nemáme žádná důležitá data a kdokoli se na ně může podívat, může mít fatální důsledky. Útočníkovi pravděpodobně ani o naše data nejde. Bude se snažit použít naši pracovní stanici jako základnu pro páchání další trestné činnosti. Pochopitelně všechny stopy, které za sebou útočník zanechá, budou ukazovat právě na nás. To je ale pouze jedna z možností. Napadený počítač může sloužit k rozesílání spamu, k šíření virů, úložiště nelegálního softwaru a mnoho dalšího. Proto musíme dbát na to, aby byl systém co nejbezpečnější, i když podle nás neobsahuje žádná cenná data. Řekněme si nyní alespoň základní zásady práce s hesly a uživatelskými jmény.

Heslo by mělo být voleno optimálně. Nemělo by být moc krátké, jednoduché, ale ani moc dlouhé. V případě krátkého hesla lze výborně využít automatizované nástroje, které zadávají všechny možné kombinace znaků a pokud je heslo krátké, velmi rychle se jim ho podaří prolomit. Stejná situace platí i v případě jednoduchého hesla. Příkladem jsou jména

domácích mazlíčků, našich blízkých nebo data narození. Útočník, který nás může znát, pak heslo snadno odhadne. V případě příliš dlouhého hesla hrozí jeho zapomenutí. Obecně ale platí, že pokud se útočník fyzicky u počítače nachází, získá nad ním kontrolu. Stejný postup ale můžeme využít i v případě zapomenutého hesla. Uvedu tedy příklad, jak se nabourat do kteréhokoli linuxového systému. Nyní opomeňme možnost použití některého z LiveCD, zde je situace jasná. Při zapnutí počítače nám zavaděč systému (v posledních letech převážně GRUB), nabízí možnost nabotovat systém v recovery módu. Tento mód nám umožňuje opravit systém, pokud se s ním něco stane. Pokud tedy tuto možnost využijeme, nezastaví nás už nic, abychom s počítačem provedly naprosto cokoliv. Získáváme totiž v tomto módu veškerá superuživatelská oprávnění. A konečně onen „zázračný“ příkaz, který umožňuje změnit heslo:

```
passwd user_name (2)
```

Za user_name pochopitelně dosadíme požadované uživatelské jméno a už jen zadáme nové heslo. Po restartu systému získáme plný přístup do systému. Existují stovky způsobů, jak tohoto dosáhnout a to nejen u Linuxu. Podobně lze měnit heslo i u operačního systému Windows XP, či novější Vista. Každého jistě napadne, zda toto není obrovská bezpečnostní díra, ale není. Jednak se i tomuto dá zabránit, například zakázáním recovery módu, čímž se ale připravíme o možnost opravy v případě pádu operačního systému a navíc má útočník další možnosti – již zmiňované LiveCD. Stačí ho vložit do mechaniky, nabotovat operační systém a rázem má přístup ke všem našim datům. Řekněme tedy, že pokud se útočník fyzicky k počítači dostane, nemáme šanci mu nijak zabránit v převzetí kontroly nad celým systémem. Jak již bylo řečeno dříve, podobná situace je i v případě ostatních operačních systémů. Například mechanismus obejití hesla konkurenčního operačního systému Windows XP je také velmi jednoduchý. Obdoba recovery mode ve Windows se nazývá „nouzový režim“. Můžeme ho vyvolat stiskem klávesy F8 při bootování operačního systému. Zobrazí se nám nabídka s několika možnostmi. Zde zvolíme právě onen nouzový režim a potvrdíme stiskem klávesy enter. Po načtení systému vybereme uživatelský účet s názvem administrator. Systém nás upozorní a na jeho výzvu odpovíme kladně. Načte se systém s maximálními možnými oprávněními a můžeme provádět prakticky cokoli nás napadne. Změna hesla se provádí z ovládacích panelů, kde vybereme možnost uživatelské účty a přidáme nový účet. Zbývající účty můžeme dle libosti odstranit tak, aby se původní majitel k účtu už nepřihlásil. Proti této možnosti se jde samozřejmě účinně bránit. Každá ochrana jde

ale obejít a v tomto případě nám opět pomůže operační systém Linux. Můžeme totiž nabootovat operační systém z již zmiňovaného LiveCD kterékoliv Linuxové distribuce. Práci nám může značně ulehčit distribuce **Öpncrâck**, která je určená právě k prolamování hesel u MS Windows. Je založená na distribuci Linux SLAX6. Hesla jsou ve Windows XP uložena v souboru:

```
C:\Windows\System32\config
```

Ten je pochopitelně šifrován 128 bitovým klíčem, s kterým si ovšem zmiňovaná distribuce hravě poradí a veškerá hesla přehledně a jednoduše zobrazí.[1][2][9]

2.2 Aplikační úroveň

V obrovském množství programů se objevují bezpečnostní chyby. I když se zdá toto tvrzení pesimistické, je pravdivé. Na bezpečnost aplikace se musíme dívat ze dvou pohledů – z jejího chování, které má vliv na operační systém, což může být například odkládání dat do souboru a jejího interního chování – chybně ošetřený vstup a výstup uživatele. To může vést k zápisu mimo přidělenou paměť, což způsobí pád programu.

Častými chybami jsou opomenutí odstranit testovací procedury vývojové fáze programu, například přeskočení ověření identity uživatele.[9]

2.3 Úroveň jádra

Z hlediska bezpečnosti se jedná o nejdůležitější část celého systému. Souvisí se stabilitou běhu operačního systému. Chyby zde mohou způsobit kolaps procesů a tím zapříčinit nový start systému, v horších případech i chyby ve filesystému a dojde opravdu k nevratné ztrátě dat. Z programů se po spuštění stávají procesy a jádro vytváří bezpečné prostředí pro jejich běh. Pro každý jednotlivý proces jádro provádí ochranu paměťového prostoru, který je mu přidělen. Dále umožňuje procesům přístup k perifériím přes virtuální souborový systém a jednotlivým procesům přiděluje procesor.

Jádro se stará o komunikaci periférií s ovladači a kontrolu oprávnění jednotlivých procesů. Procesy jádra běží v režimu jádra a uživatelské procesy v uživatelském režimu. Oba tyto segmenty jsou oddělené, aby jádro mohlo omezit uživatelským procesům používání privilegovaných instrukcí. Pokud tedy chceme komunikovat s nějakým hardwarem, musíme provést systémové volání, což nám poskytuje právě jádro systému.[9]

3 SECURE SHELL

3.1 Bezpečnost počítačových sítí

Počítačové sítě založené na Internet Protocol (IP) neobsahují žádné zabezpečení. Naše přenosy mohou být velmi snadno zachyceny a dokonce modifikovány. I když se domníváme, že data, která přes naši síť posíláme, nejsou důvěrná, neměli bychom zapomínat, že už pouhé přihlášení znamená, že síť přenáší naše heslo v čistě textové podobě – a heslo důvěrná informace je. Každé přihlášení, například k emailu, k bankovnímu účtu nebo sociální síti pak znamená velmi vážné bezpečnostní riziko. V posledních letech se míra bezpečnosti razantně zvýšila a ukázalo se, že je to krok správným směrem. Doby, kdy byly v síti pouze malé skupiny účastníků, jsou dávno pryč, stejně jako využívání nezabezpečených protokolů, jako Telnet, Remote Shell (dále jen RSH). Způsobů, jak v dnešní době realizovat zabezpečení těchto protokolů, je mnoho. Využívá se například SSL, nebo pokročilejší TLS. SSL spojení funguje na principu asymetrické šifry, kdy každá z komunikujících stran má dvojici šifrovacích klíčů – veřejný a soukromý. Veřejný klíč je možné zveřejnit, a pokud tímto klíčem kdokoliv zašifruje nějakou zprávu, je zajištěno, že ji bude moci rozšifrovat jen majitel použitého veřejného klíče svým soukromým klíčem.

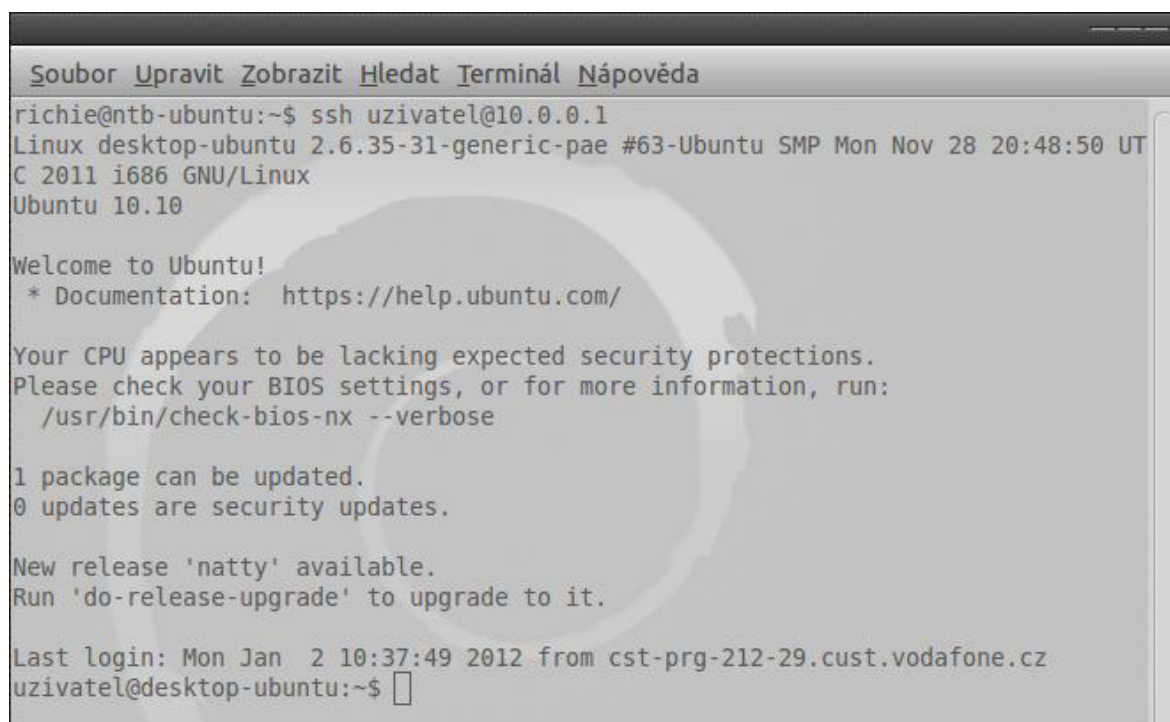
- *Klient pošle serveru požadavek na SSL spojení, spolu s různými doplňujícími informacemi (verze SSL, nastavení šifrování atd.).*
- *Server pošle klientovi odpověď na jeho požadavek, která obsahuje stejný typ informací a hlavně certifikát serveru.*
- *Podle přijatého certifikátu si klient ověří autentičnost serveru. Certifikát také obsahuje veřejný klíč serveru.*
- *Na základě dosud obdržených informací vygeneruje klient základ šifrovacího klíče, kterým se bude šifrovat následná komunikace. Ten zašifruje veřejným klíčem serveru a pošle mu ho.*
- *Server použije svůj soukromý klíč k rozšifrování základu šifrovacího klíče. Z tohoto základu vygenerují jak server, tak klient hlavní šifrovací klíč.*
- *Klient a server si navzájem potvrdí, že od teď bude jejich komunikace šifrovaná tímto klíčem. Fáze handshake tímto končí.*
- *Je ustaveno zabezpečené spojení šifrované vygenerovaným šifrovacím klíčem.*
- *Aplikace od teď dál komunikují přes šifrované spojení. Například POST požadavek na server se do té doby neodešle²*

²http://cs.wikipedia.org/wiki/Secure_Sockets_Layer

3.2 Secure Shell

Secure Shell (dále jen SSH) vzniklo ze svého poněkud méně bezpečného předchůdce RSH. Pokud rozebereme název podrobně, zjistíme, že žádný šel neposkytuje. Toto je záležitost operačního systému. Shell je v podstatě program, který vytváří v počítači rozhraní pro uživatele. Umožňuje uživateli využívat funkce jádra operačního systému, zejména spouštět programy, zajišťovat pro ně vstupy, zobrazovat, uchovávat a přeměřovat jejich výstupy, spojovat jednotlivé programy do kolon a podobně. SSH tedy vytváří nějaký transparentně šifrovaný bezpečnostní kanál mezi dvěma místy (počítači) v síti a právě díky transparentnímu šifrování se nemusí programy v síti o přítomnost SSH vůbec starat.

Využití SSH protokolu je obrovské. Tento protokol se využívá hlavně pro vzdálenou správu počítačů a serverů. Výhodou správy přes tento protokol je právě bezpečnost. Útočník nezjistí ani jaké programy přes síť spouštíme, natož jejich výstup. Programy, které spouštíme, navíc nemusí být pouze konzolové (textové). Stačí se při přihlášení přihlásit s parametrem `-X`, což nám spustí xserver. S SSH se tedy můžeme i bezpečně přihlásit ke vzdálenému PC a pracovat zde jako normální uživatel – jako bychom u počítače přímo seděli. Omezení jsme pouze rychlostí našeho síťového připojení. Dále se tento protokol využívá na bezpečné směrování portů a v neposlední řadě při kopírování souborů. Bohužel je v tomto ohledu daleko pomalejší než protokol FTP. Pro názornost se podívejme na příklad přihlášení k již předem nakonfigurovanému SSH serveru, zobrazeném na obr. 4.[10]



```
Soubor Upravit Zobrazit Hledat Terminál Nápověda
richie@ntb-ubuntu:~$ ssh uzivatel@10.0.0.1
Linux desktop-ubuntu 2.6.35-31-generic-pae #63-Ubuntu SMP Mon Nov 28 20:48:50 UTC
2011 i686 GNU/Linux
Ubuntu 10.10

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

Your CPU appears to be lacking expected security protections.
Please check your BIOS settings, or for more information, run:
  /usr/bin/check-bios-nx --verbose

1 package can be updated.
0 updates are security updates.

New release 'natty' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Jan  2 10:37:49 2012 from cst-prg-212-29.cust.vodafone.cz
uzivatel@desktop-ubuntu:~$
```

Obr. 4. Přihlášení k SSH

3.2.1 SSH a RSA autentizace

Dříve se k přihlašování neboli k prokázání identity SSH serveru využívalo standardní uživatelské jméno a heslo. Používá se to samozřejmě dodnes, ale systém nám nabízí i další možnosti. Totiž – i kdybychom si zvolili velmi bezpečné heslo, u kterého využijeme velká, malá písmena, číslice a speciální znaky, ztížíme tím sice útočnickovi dešifrování hesla, problém ale je, že naše heslo může někdo například odposlechnout nebo jinak zjistit. Na jakékoliv zabezpečení můžeme v takovém případě zapomenout. Protokol SSH nám nabízí možnosti, jak řešit i tento bezpečnostní problém – a to RSA autentizací, neboli „ověřením veřejného klíče“. Zde se využívá privátních a veřejných klíčů místo „jednoduchých“ hesel. Příkazy 3 až 8 ukazují, jak lze tyto klíče vytvořit, a celé spojení tímto systémem zabezpečit. První, co musíme udělat, je vytvoření privátního klíče. To provedeme následujícím příkazem:

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

(3)

Nyní nám systém nabídne, kam má být klíč uložen. Můžeme ho uložit samozřejmě, kam budeme chtít, nicméně hlavně kvůli bezpečnosti se doporučuje zvolit standardní umístění navržené SSH. Dále musíme určit vstupní frázi. Ta zašifruje onen privátní klíč. Pokud bychom tuto frázi vynechali, klíč by zůstal nezabezpečený a každý, kdo by se k němu dostal, by získal naše práva na jiných počítačích. Zde se doporučuje využít opravdu silné heslo. Výhodou je, že ho už nikdy nebudeme muset zadávat. Společně s privátním klíčem se nám vytvořil i veřejný klíč a obvykle ho lze nalézt v adresáři:

```
~/.ssh/id_dsa.pub
```

Pro získání přístupu do vzdálených počítačů musí tyto počítače privátnímu klíči důvěřovat, tudíž musíme nakopírovat obsah tohoto souboru do souboru `authorized_keys` na vzdáleném počítači. Jedná se o řetězec ASCII znaků a zpravidla bývá umístěn v následujícím adresáři:

```
~/.ssh/authorized_keys
```

nebo můžeme jednoduše soubor překopírovat:

```
ssh-copy-id -i ~/.ssh/id_dsa.pub uzivatel@server (4)
```

Budeme dotázáni na heslo na cílovém počítači. Pokud je ověřování pomocí hesla vypnuté, pak musíme zkopírovat a vložit klíč za použití jiného média. Poté, co bude veřejný klíč přidán, stane se tento počítač důvěryhodný.

Pro použití *ssh-copy-id* je nutné mít vstupní frázi uloženou v ssh-agentovi. To lze provést příkazem:

```
ssh-add (5)
```

```
ssh uzivatel@server (6)
```

Následující dotaz již nebude na heslo, ale na vstupní frázi. Heslo a vstupní frázi musíme rozlišovat. Jde o dva rozdílné pojmy. Zatímco heslo je primárně uloženo v souboru:

```
/etc/passwd
```

(v cílovém systému), výstupní fráze je použita výhradně pro dešifrování našeho privátního klíče v lokálním systému. Lepší ochrana při použití ověřování veřejného klíče oproti ověřování pomocí hesla je tedy v tom, že pro získání přístupu potřebujeme dvě věci:

- zašifrovaný privátní klíč

- vstupní frázi (která je potřebná pro dešifrování privátního klíče) [9]

3.2.2 SSH a bezpečnost

Útoky na SSH server se zpravidla dělí do dvou kategorií, a to vnitřní a vnější. Vnitřní útok musí být proveden uživatelem, který má na serveru účet a má k tomuto účtu přístup přes SSH. Jedná se tedy o oprávněného uživatele a našim cílem je, aby na serveru prováděl pouze to, co mu povolíme. Tento útočník tedy nesmí získat žádná práva, která mu přidělit nechceme. V tomto případě se převážně jedná o správné nastavení práv. Cílem obrany proti vnějšímu útoku pochopitelně je, aby útočník nebyl schopen získat přístup na server. Tento útočník na serveru nemá žádný účet, jeho možnosti jsou buď využití nějaké slabiny SSH protokolu nebo zjištění hesla a uživatelského jména již existujícího uživatele.[13]

3.2.2.1 Vnější útok

Nejčastějším typem tohoto útoku je zpravidla ten nejhlupejší a pokud si budeme při zařizování serveru dávat alespoň trochu pozor, jaká hesla a uživatelská jména používáme, jednoduše se mu vyhneme. Nejjednodušší způsob, jak napadnout SSH server, je využití automatizovaných nástrojů. Ty prohledají určitý rozsah IP adres a hledají, jestli se na nich nachází nějaký SSH server. Práce s těmito programy je velmi jednoduchá. Zvolíme požadovaný rozsah (u pokročilejších nástrojů lze také zvolit rozsah portů – standardně se totiž prohledá pouze port 22, na kterém SSH normálně komunikuje) a program spustíme. V případě, že program najde SSH server, začne zkoušet nejtypičtější, takzvané „slovníkové“ kombinace uživatelských jmen a hesel. V případě, že program žádný SSH server nenajde, zkouší jinou adresu. Úspěšnost těchto útoků je minimální. Navíc lze tyto aktivity objevit stejně snadno, jak útokům zabránit. Velmi užitečné je kontrolovat následující logovací soubory.

`/var/log/auth.log`

`/var/log/auth.log.0`

`/var/log/auth.log.1`

Zde se dovíme všechno o tom, kdo se přihlašoval - jak, odkud a kdy. Jedná se pochopitelně o velmi obsáhlé soubory. Zobrazení je možné filtrovat pomocí parametrů, například příkazem:

```
awk '/Invalid user/ {print $8}' /var/log/auth.log{,.0} | sort | uniq -c
```

 (7)

Tento příkaz nám zobrazí, kdo se snažil přihlásit s neplatným uživatelským jménem a počet pokusů o přihlášení s tímto jménem. Seznam uživatelů, a kam se přihlásili, zobrazíme příkazem:

```
last
```

 (8)

Rychlejší varianta je použití příkazu *lastb* (last bad – tedy poslední špatný).

Výpis z logovacího souboru můžeme charakterizovat následujícím způsobem například takto:

```
auth.log:May 10 14:14:22 hyperion sshd[8182]: Invalid user test from 218.56.61.114
auth.log:May 10 14:14:27 hyperion sshd[8216]: Invalid user guest from 218.56.61.114
auth.log:May 10 14:14:31 hyperion sshd[8254]: Invalid user admin from 218.56.61.114
auth.log:May 10 14:14:40 hyperion sshd[8328]: Invalid user admin from 218.56.61.114
auth.log:May 10 14:14:44 hyperion sshd[8362]: Invalid user user from 218.56.61.114
auth.log:May 10 14:15:04 hyperion sshd[8530]: Invalid user test from 218.56.61.114
```

Přesně takto vypadá automatizovaný útok. Na začátku řádku vidíme datum a čas, kdy bylo neoprávněné přihlášení prováděno a na konci souboru adresu, z které útok probíhal. Podle této adresy lze zjistit totožnost jejího majitele, což je pravděpodobně adresa útočnicka. Obrana proti tomuto útoku je také jednoduchá, neboť stačí krátký skript, který útočnicka velmi rychle odstříhne. Existují na to i různé nástroje a metody, které jsou popsány níže.

Existují ovšem i metody, jak tyto typy útoků provést podstatně chytřejším způsobem. Například použít různou dobu mezi jednotlivými útoky a hlavně použít různé IP adresy – nejčastěji bývá útok proveden z nějakého botnetu. Ještě chytřejší řešení je hledat pouze starší, neaktualizované SSH servery, u kterých je možné využít dostupné exploity.

Asi nejhorším typem útočnicka je takový, který vlastní velký seznam SSH serverů, a vyčkává si na takzvaný **0-day exploit**. Je to nejnovější (poslední) objevená díra v SSH protokolu. Útočnick pak projede svůj seznam a pravděpodobně se nám na server dostane. Toto riziko je ovšem minimální. Navíc pokud pravidelně systém aktualizujeme, toto riziko ještě značně

snížíme. Pro účely programu, který je prezentován v praktické části, by bylo například dále vhodné cíleně omezit na přístup pouze určitého zařízení, ze kterého hodláme přístup provádět. Tímto minimalizujeme téměř veškerá rizika.[10][11][12][13]

3.2.2.2 *Dodatečné a speciální zabezpečení*

V OpenSSH (což je používáno na většině Linuxových a BSD distribucích) se nachází konfigurační soubory SSH protokolu ve složce:

```
/etc/ssh/sshd_config
```

Prakticky veškeré elementární zabezpečení provedeme úpravami v tomto souboru. První věc, kterou bychom měli zkontrolovat, je verze protokolu, který využíváme. Zde se doporučuje ponechat pouze protokol 2. Protokol 1 má spoustu děr a prakticky se již nepoužívá. O kontrolu práv v domovském adresáři uživatele se stará takzvaný striktní režim. Tato procedura nám zkontroluje, jestli není umožněn přístup uživatelů do kritických míst – pokud ano, přihlášení odmítne. Dalším slabým místem každého serveru je grafické rozhraní. Na většině serverů se nepoužívá, proto ho můžeme v konfiguračním souboru zakázat. Výpis z konfiguračního souboru

by měl obsahovat tyto položky:

```
Protocol 2
```

```
StrictModes yes
```

```
X11Forwarding no
```

```
PermitTunnel yes
```

Podrobný výpis souboru je zobrazen v příloze PII.

[13]

3.2.2.3 *Řízení přístupu*

Tento krok bychom měli začít malou analýzou. Důkladně zvážit, kdo opravdu přístup přes SSH potřebuje. Například, když už povolíme přístup konkrétnímu uživateli, je nutné, aby měl přístup k shellu? Pokud má uživatel na serveru pouze poštovní schránku, je nutné mu povolit přístup k SSH? A tak dále. I když se rozhodneme přístup udělit, můžeme například omezit přístup jen z konkrétní IP adresy, z místa, kterému důvěřujeme. Po zvážení

této analýzy se můžeme pustit do bezpečnostní politiky. Začneme skupinami. Pokud máme více uživatelů, vytvoříme třeba skupinu netusers, nebo sshusers do které začleníme všechny uživatele, kterým hodláme přístup povolit. Tuto skupinu pak přiřadíme jako parametr k volbě *AllowGroups*. Tuto volbu je ale výhodné provést pouze v případě, že uživatelů je velké množství. Pokud uživatelů není mnoho, bude přehlednější jednotlivé uživatele vypsat a ke každému přiřadit parametr *AllowUsers*. Pokud tyto volby použijeme v konfiguraci SSH, nikdo jiný, než povolení uživatelé, se k serveru nepřihlásí.

Pokud se naopak rozhodneme některému uživateli přístup zakázat, můžeme použít volby *DenyUsers* a *DenyGroups*.

Dalším krokem, který bychom měli zvážit, je povolení přístupu uživatele *Root*. Uživatel *Root* je, co se práv týče, všemocný. Obvykle tento problém řešíme tak, že vytvoříme jiného neprivilegovaného uživatele, který může použít příkazy „su“, nebo „sudo“ pro provádění administrativních úkonů. Představme si například situaci, kdy se nám útočník nabourá do našeho SSH účtu. Pokud jsme zakázali přístup rootovi, útočník bude muset navíc zjišťovat heslo k administrátorskému účtu. S klasickými uživatelskými oprávněními sice může procházet určité složky, ale rozhodně nezpůsobí pád celého serveru. Tato volba se nazývá:

PermitRootLogin

Výše uvedená volba má tři parametry – yes, no, without-password. Poslední jmenovaná nám způsobí situaci, že se root nebude moct přihlásit pod heslem, ale pouze jinou metodou, například SSH klíčem (RSA).

Dalším zajímavým souborem, který stojí za zmínění je */etc/security/access.conf*.

Představme si situaci, že máme skupinu uživatelů, které důvěřujeme a další skupinu, které ne. To lze vyřešit editací souboru *access.conf*. Můžeme zde přesně definovat, kteří uživatelé se budou moci přihlásit a odkud.[13]

4 FILE TRANSFER PROTOCOL

File Transfer Protocol (dále jen FTP) je protokol pro přenos souborů mezi počítači zapojených do počítačové sítě. Lze jej použít jak pro vzdálenou síť, tedy Internet, tak pro přenos souborů po lokální síti. Ke své funkci využívá Transmission Control Protocol (dále jen TCP), což je jeden ze základních protokolů Internetu. FTP je jeden z nejstarších protokolů. Byl definován v roce 1985, ale svou nynější podobu získal až v roce 1997. V posledních letech se jeho podpora stala běžnou součástí většiny webových prohlížečů. Tento multiplatformní protokol normálně pracuje na portech 20 a 21. Port 20 slouží pro běžný přenos souborů, pomocí portu 21 lze přenášet příkazy. Nevýhodou je, že při přenosu souborů je blokován 21. port, což znemožňuje zadávání příkazů. Zvláště při kopírování velkých souborů to může způsobit zásadní problémy. Výhodou FTP protokolu je jeho rychlost. Při správném nastavení je omezena pouze rychlostí síťového připojení, nebo rychlostí zápisu na harddisk.[14]

4.1 Bezpečnost FTP

FTP protokol není nijak šifrovaný. Veškerá data, počínaje heslem, které posíláme z klientského počítače na server, ale i opačně, se přenáší v takzvaném „plain-textu“, což v překladu znamená prostý text – znaky se převádí na čísla odpovídající kódům použité znakové sady, například ASCII. Data může číst hlavně poskytovatel internetového připojení, ale i útočník, který by se připojil na vnitřní síť, ze které bychom se k FTP serveru připojovali. V posledních letech vznikla různá řešení na zvýšení míry bezpečnosti tohoto poměrně starého protokolu.[2]

4.2 SSH file transfer protocol

Jedním ze způsobů, jak FTP přenos zabezpečit, je využití SSL šifrování. Zkráceně se tato metoda nazývá SFTP. Je to vlastně FTP protokol doplněný o SSL vrstvu. Tento multiplatformní protokol byl navržen skupinou Internet Engineering Task Force (dále jen IETF). Protokol nezajišťuje autentizaci ani šifrování dat. K zajištění těchto služeb využívá protokol SSH. SFTP bývá také označováno jako jednoduchý vzdálený souborový systém, protože nabízí široké možnosti při práci se vzdálenými soubory, například pokračovat v přerušovaných přenosech, výpis adresářů nebo mazání souborů.[14]

4.3 Konfigurace FTP serveru

V Linuxu máme na výběr z několika FTP serverů, například Pure-ftpd, ProFTP, nebo bezpečnější Very Secure File Transfer Protocol Daemon (dále jen vsftpd) jehož konfigurace se provádí pomocí souboru:

```
/etc/vsftpd/vsftpd.conf
```

Příklad bezpečné konfigurace serveru je uveden v příloze PI. Některé servery poskytují i grafické konfigurační rozhraní, ale je zde nutné chápat toto grafické rozhraní pouze jako jakousi nástavbu. Konfigurace se zpravidla provádí přes konfigurační soubory. Vsftpd byl napsán Chrisem Evansem. Jeho hlavní zaměření je především bezpečnost. Server je lehce konfigurovatelný, ale nepodporuje některé důležité funkce jako kvótování nebo virtuální uživatele.

4.3.1 Konfigurace Pure-ftpd

Zde se provádí konfigurace přes soubor:

```
/etc/pureftpd/pure-ftpd.conf
```

V tomto souboru jsou uloženy informace o FTP serveru. Při konfiguraci je možné vycházet z konfiguračních voleb uvedených v příloze PI, nicméně jednotlivé volby se mohou měnit. Pure-ftpd podporuje virtuální uživatele, což jsou uživatelé, kteří neexistují v operačním systému, ale pouze v FTP serveru. V konfiguračním souboru toto nastavení představuje položka:

```
PureDB /etc/pure-ftpd/pureftpd.pdb
```

Data jsou na FTP serveru standardně umístěna v adresáři:

```
/srv/ftp
```

Do adresáře kopírujeme soubory, ke kterým chceme umožnit přístup ostatním uživatelům. Dále je nutné při konfiguraci nastavit firewall, kde je nutné povolit výše uvedené porty 20 a 21 pro TCP. V případě vzdáleného přístupu je pak nutné vytvořit Network address translation (dále jen NAT).[21]

5 NETWORK FILE SYSTEM

Network File system (dále jen NFS) byl vyvinut firmou Sun Microsystems v roce 1984. V současné době ho vlastní firma IETF. Jedná se o internetový protokol sloužící pro přenos souborů mezi serverem a klientem. Na Linuxu se jedná pravděpodobně o nejrozšířenější protokol používaný pro tyto účely. Protokol umožňuje připojit vzdálený disk, či přímo složku umístěnou na něm do adresáře na cílovém počítači. Z toho plyne velká výhoda tohoto spojení. Se vzdálenou složkou lze pracovat stejně, jako bysme se na vzdáleném serveru nacházeli. Komunikace zde probíhá přes 2049 port. Dále je patrné, že nepotřebujeme žádného specializovaného klienta pro zobrazení vzdálených složek. Podpora je u většiny linuxových distribucí implementována přímo v jádře systému. Pokud ne, lze NFS server snadno doinstalovat těmito balíčky:

nfs-kernel-server

portmap

nfs-common

[2][16][17]

5.1 Konfigurace serveru

NFS server je možné nastavovat pomocí konfiguračních souborů. Nastavení uživatelů provádíme v souboru `/etc/exports`. Každý řádek znamená jedno sdílení. Výpis souboru může vypadat například takto:

`/home 192.168.1.2(rw,no_root_squash,sync)`

- `/home` – obecně adresář který chceme sdílet
- `192.168.1.2` – IP adresa počítače, kam chceme sdílet

(192.0.0.0/255.0.0.0 - povolí sdílení pro celou síť 192)

- `rw` - čtení a zápis povolen
- `no_root_squash` – pokud se přihlásí root přes klienta bude root i na serveru
- `sync` - bude pracovat synchronně

[16][17]

5.2 Konfigurace klientského počítače

Připojení vzdáleného disku na klientském PC je už víceméně intuitivní záležitost. Stačí klasicky vytvořit v kterémkoli adresáři přípojný bod, například:

```
/media/nfs_server
```

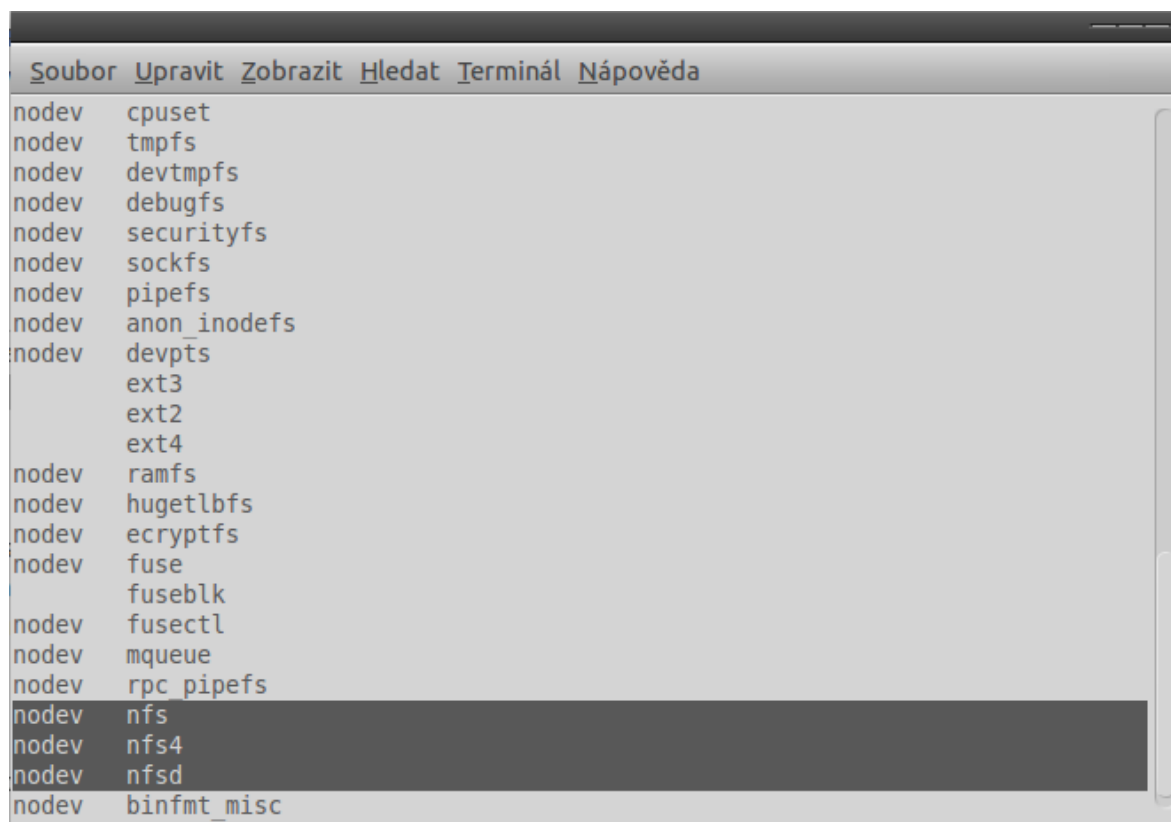
Samotné připojení se provede příkazem:

```
mount 192.168.1.2:/home /media/nfs_server (9)
```

Dále je možno přidat záznam přímo do souboru:

```
/etc/fstab
```

což nám umožní automatické připojení vzdáleného disku hned po startu OS. Podpora klientské části musí být také implementována v jádře. Aktuální stav lze zjistit výpisem adresáře `/proc/filesystems`, který je uveden na obr. 5. Zvýrazněné řádky ukazují, jak by měl soubor vypadat, pokud chceme provozovat NFS na našich počítačích. U většiny linuxových distribucí je podpora implementována přímo v jádře.



Obr. 5. Seznam podporovaných souborových systémů

5.3 NFS a bezpečnost

NFS je bezstavový protokol, což znamená, že zde neexistuje žádné přihlašování, nebo navazování spojení. Klient pouze posílá požadavky na server a ten mu odpoví. Komunikace není šifrovaná a tak ji lze odposlechnout. Nedoporučuje se používání tohoto protokolu mimo lokální síť. Pokud důvěřujeme uživatelům na lokální síti, je výhodné tento protokol využívat. Kvůli jeho jednoduchosti, ale hlavně kvůli jeho rychlosti.

6 SAMBA

Samba je implementace Server Message Block protokolu (dále jen SMB), který se využívá rovněž pro vzdálený přístup souborů. SMB protokol je také někdy nazýván NetBIOS, nebo LanManager. Výhodou tohoto protokolu je různorodost souborových systémů, mezi kterými poskytuje sdílení. Především se tento protokol využívá pro sdílení mezi operačními systémy Linux a Windows. Tento protokol byl vyvinut v roce 1992 studentem Andrew Tridgellem. Je složen z následujících částí:

Deamon soubory:

smbd – souborové a tiskárnové služby

nmbd – nameserver služby

Tyto soubory se spouští po startu systému, resp. Samby a vykonávají svou činnost na pozadí operačního systému.

Smbclient – SMB klient pro UNIX, slouží i pro vzdálený tisk

testparm a **smbstatus** – nástroje pro testování

Protokol samba tedy umožňuje sdílení souborů a tiskáren mezi operačními systémy Windows a Linux. Dále umožňuje autentizaci a autorizaci, vyhledávání jmen a oznamování služeb (prohledávání souborových a tiskových serverů). Pro její chod jsou nutné balíčky:

samba

smbfs

Lze je získat z repozitářů, případně z oficiálních webových stránek projektu. Pro grafickou konfiguraci serveru je dostupný nástroj system-config-samba. Poskytuje pouze základní konfiguraci. Sambu lze konfigurovat také pomocí konfiguračního souboru:

```
/etc/samba/smb.conf
```

Další užitečné nástroje vhodné pro konfiguraci samby jsou SWAT a Webadmin.

[18][19][20]

6.1 Konfigurace Samba serveru

Prvním krokem po instalaci Samby je přidání uživatele/uživatelů. Uvedenou operaci provedeme zadáním následujících příkazů do terminálu. Je nutné provést příkazy jako root.

```
smbpasswd -a jmeno_uzivatele (10)
```

Jméno uživatele představuje jméno, pod kterým se bude možné připojit do sítě. Odstranění uživatele se provede obdobným způsobem:

```
smbpasswd -x jmeno_uzivatele (11)
```

Samba ukládá přehled uživatelů do souboru:

```
/etc/samba/smbusers
```

Je možné uživatele z tohoto souboru také editovat – čili přidávat, nebo odebírat. Dále je nezbytné přidat uživatele Samby i do samotného systému. Musí být také zařazen do vhodné skupiny. Uživatel musí existovat na třech místech. První je samotný Samba server, dále počítač, na kterém je Samba server spuštěn a pochopitelně i v systému, kam se budeme přihlašovat.

Samotné sdílení pak můžeme nastavovat s několika pravidly, například s oprávněním jen pro čtení a s autentizací, nebo pro čtení a zápis s autentizací, ale také s oprávněními pro skupiny uživatelů. Například oprávnění pro čtení a s autentizací by se realizovalo následovně:

```
gedit /etc/samba/smb.conf (12)
```

Editací tohoto měníme veškerá oprávnění. Nejprve nalezneme řádek:

```
; security = user
```

který změníme na:

```
security = user
```

```
username map = /etc/samba/smbusers
```

Dále je nutné restartovat Sambu příkazem:

```
/etc/init.d/samba restart (13)
```

[19]

II. PRAKTICKÁ ČÁST

7 LINUX JAKO ŘÍDÍCÍ SYSTÉM

V poslední době zaznamenáváme obrovský boom jak v oblasti bezpečnostních technologií, tak IT průmyslu. Mým cílem bylo tyto technologie sjednotit do jednoho celku a vytvořit tak systém ke sledování stavu senzorů odkudkoli na naší planetě. Stále více se uplatňují nové technologie k ovládání našich obydlí, sledování blízkých, střežení majetku a informací. Postupně se z našich domů stanou inteligentní budovy, které nám budou pomáhat řešit běžné úkony, se kterými se teď musíme manuálně vypořádat sami. Otevírat okna kvůli větrání, zatahovat závěsy, když v domě rozsvítíme světla, udržovat teplotu v různých místnostech a mnoho dalšího. Přitom by to náš dům mohl udělat za nás. Technologie, která by nám usnadnila život, je řadu let dostupná, ale v kombinaci s řízením našeho obydlí o ní slycháme jen posledních pár let. Jistě, existují inteligentní budovy, ale i když je tato technika na trhu, proč ji nevyužíváme všichni? Odpověď je prostá – vysoká cena. Pokud budeme chtít do našeho příbytku zavést určitou inteligenci, nebude to zadarmo, a to nemluvím o rozsáhlé kabeláži, která je nutná k propojení jednotlivých periférií. Jednoduše řečeno, pokud toto nevyřešíme již při stavbě budovy, nevyplatí se nám posléze tyto technologie zavádět, přesto, že bychom ušetřili nemalé částky například za energii.

Srdcem každé inteligentní budovy je řídicí systém. Je to vlastně jednoduchý počítač, který má plnit pouze funkce spojené s ovládáním budovy. Nabízí se tedy otázka, proč kupovat tento drahý systém, když už ho máme víceméně všichni doma? Mluvíme tady o obyčejném stolním počítači, notebooku, netbooku nebo dokonce chytrém mobilním telefonu. Dovolím si tvrdit, že k účelu řízení budovy bude stolní počítač nejen naprosto dostatečný, ale dokonce lepší. S využitím síťových topologií můžeme z průměrného notebooku řídit takový komplex jako je škola, nebo mrakodrap. Navíc pokud vezmeme v potaz, jak rychle jde dopředu rozvoj IT průmyslu, bylo by téměř hloupé nevyužít počítače k těmto účelům.

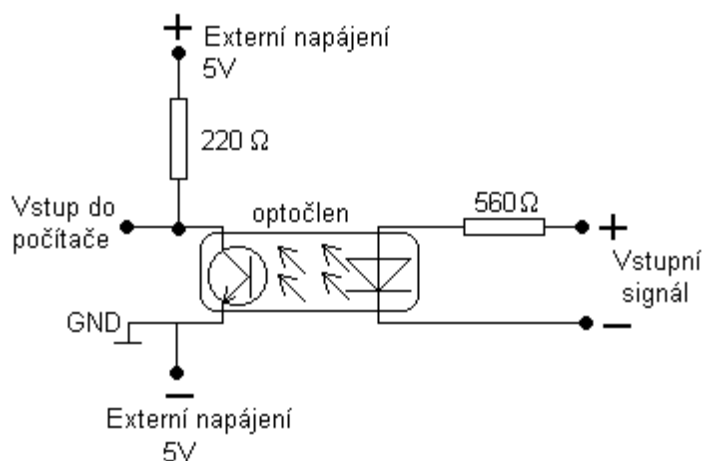
Níže je popsán program, který simuluje výstup senzorů fiktivního domu a bezpečně je za pomoci SSH protokolu posílá na námi určená zařízení (takzvaná koncová zařízení). Z důvodu zvýšení míry bezpečnosti je použito SSH tunelování a směrování portů.

7.1 Propojení senzoru s počítačem

Abychom dali tomuto systému reálnou podobu, musíme se v první řadě zaměřit na propojení jednotlivých senzorů s počítačem. Existuje spousta možností. Nejjednodušší

řešení nám poskytuje zapojení přes sériový port. Následující schéma ukazuje člen, pomocí kterého můžeme senzory připojovat k sériovému portu počítače.

Vstup do počítače pomocí optočlenu:



Obr. 6. Vstup do počítače pomocí optočlenu

Optočlen na výše uvedeném obrázku slouží jako oddělovač mezi vstupním signálem a vstupem počítače. Je zde z důvodu galvanického oddělení vstupního portu počítače a vstupního signálu. Díky tomuto prvku se v případě, že se na svorkách objeví špičkové napětí, zničí pouze optočlen. Pokud budeme chtít připojit více zařízení, stačí vyrobít více těchto obvodů a připojovat je k dalším vstupům. Externí napájení by mělo být dobře stabilizované. Slouží k tomu, že když do optočlenu neteče dostatečný proud, je na vstupním pinu logická jednička (5V). Logická nula se na vstupu objeví v případě, že optočlenem začne protékat proud. Na výstupu se objeví logická nula, protože napětí je nulové. V tab. 2. je popsáno zapojení sériového portu. Z tabulky je patrné, že k jednomu sériovému portu lze připojit až 6 zařízení, a to tři vstupní a tři výstupní.

Tab 4. Zapojení sériového portu

| | | | |
|--------|-------------------------------------|--------|------------------------------------|
| 1.pin | Zem určená pro stínění | 14.pin | Nezapojeno |
| 2.pin | Asynchronní výstup dat | 15.pin | Nezapojeno |
| 3.pin | Asynchronní vstup dat | 15.pin | Nezapojeno |
| 4.pin | Výstup z počítače (1. bit t.j. 2) | 16.pin | Nezapojeno |
| 5.pin | Vstup do počítače (4. bit t.j. 16) | 17.pin | Nezapojeno |
| 6.pin | Vstup do počítače (5. bit t.j. 32) | 18.pin | Nezapojeno |
| 7.pin | GND - zem | 19.pin | Nezapojeno |
| 8.pin | Vstup do počítače (7. bit t.j. 128) | 20.pin | Výstup z počítače (0. bit t.j. 1) |
| 9.pin | Nezapojeno | 21.pin | Nezapojeno |
| 10.pin | Nezapojeno | 22.pin | Vstup do počítače (6. bit t.j. 64) |
| 11.pin | Nezapojeno | 23.pin | Nezapojeno |
| 12.pin | Nezapojeno | 24.pin | Nezapojeno |
| 13.pin | Nezapojeno | 25.pin | Nezapojeno |

Tato universální sběrnice umožňuje jednoduše připojit externí zařízení k počítači. Přesto, že je v poslední době port RS-232 postupně vytlačován USB a FireWire technologií, k testování prototypů periferních zařízení je to ideální varianta.

Zařízení je v systému identifikováno jako:

```
/dev/ttyS0
```

Představuje první sériový port. Jedná se o běžný textový soubor. To, co do něho zapíšeme, se objeví na výstupu sériového portu. Následující příkaz názorně ukazuje, jak lze poslat informaci na sériový port.

```
echo Hello world! > /dev/ttyS0
```

(14)

Externí zařízení by takový řetězec mohlo přečíst a následně odpovědět. Podobně lze ze sériového portu také číst. Zadáním následujícího příkazu přečteme obsah souboru „ttyS0“.

```
cat /dev/ttyS0
```

 (15)

7.2 Senzory jako minipočítače

Zapojení přes sériový port slouží pouze k demonstraci, jak pracuje počítač s externím zařízením. Takové řešení by bylo nepraktické a počet zařízení, které můžeme k tomuto portu připojit, je také omezen. Pro větší počet periférií se nabízí využití ethernetového rozhraní, respektive některé z bezdrátových technologií bluetooth, či wifi.

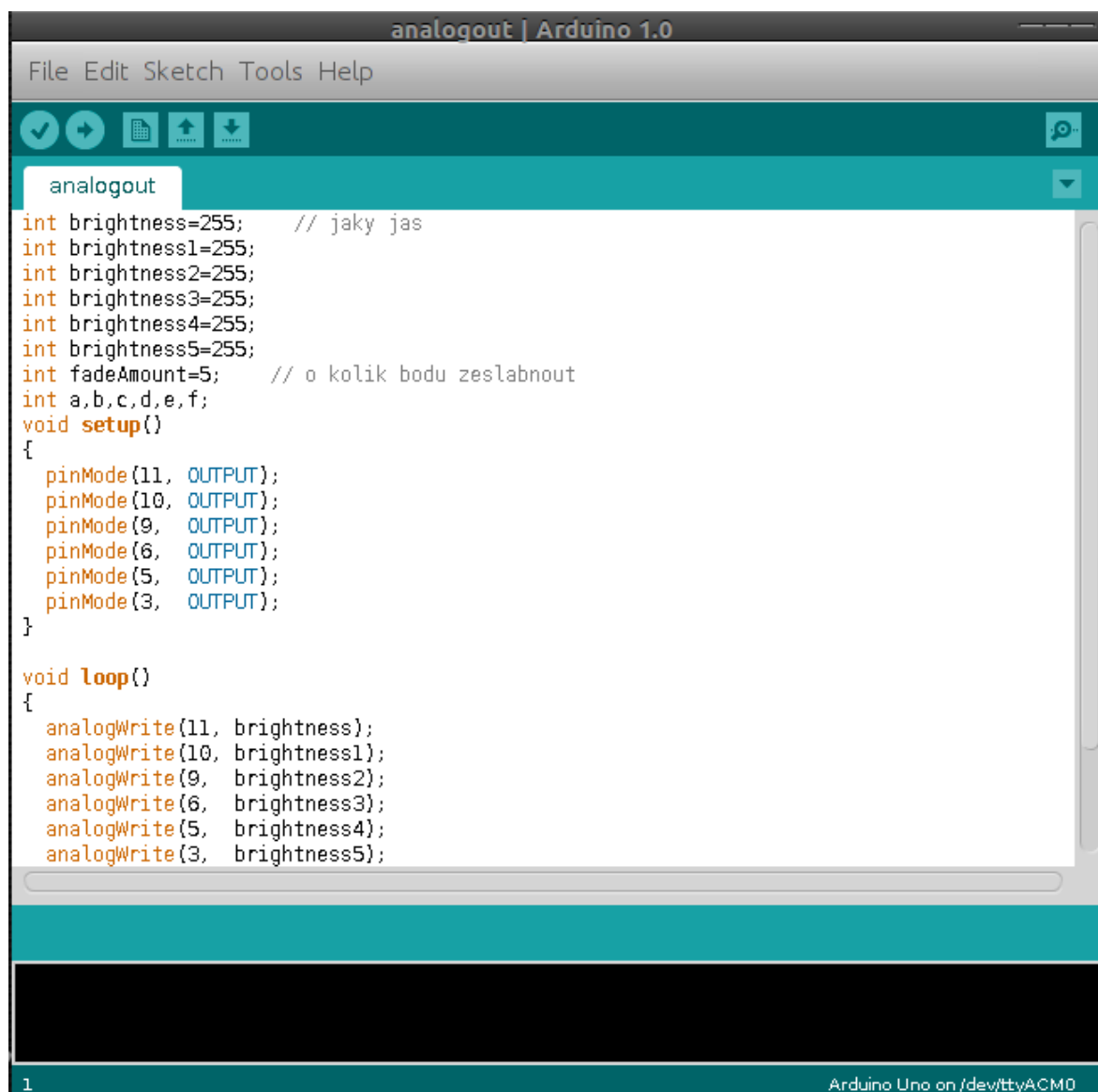
7.3 Arduino

Arduino je malý, levný vývojový kit, vhodný pro návrhy amatérských nebo poloprofesionálních aplikací. Tento projekt je také open-source, což v elektronice znamená, že je k němu volně dostupné schéma. Uživatel si tak může sestavit tento vývojový kit úplně sám, nebo přizpůsobit aplikaci konkrétnímu zapojení. Obrovskou výhodou je škálovatelnost tohoto kitu. K Arduino lze dokoupit takzvané „shields“, které podstatně rozšíří schopnost samotného zařízení, například o možnost ethernetové komunikace, ovládání servomotorů či displejů. Srdcem Arduina je procesor ATmega328, který pracuje na frekvenci 20MHz, což umožňuje provádět až 20MIPS. Dále obsahuje 32kB flash paměť a 2kB statické RAM.



Obr. 7. Arduino Duemilanove

K Arduinu lze zdarma stáhnout vývojové prostředí, které se jmenuje Wiring. Je to multiplatformní Integrated Development Environment (dále jen IDE) a pro jeho chod je nutné nainstalovat Javu. Jedná se o velmi jednoduchý nástroj sloužící k programování Arduina. V podstatě se jedná o zjednodušené C. Arduino lze programovat samozřejmě i v klasickém C/C++, což je ale podstatně komplikovanější. Následující obrázek ukazuje, jak Wiring IDE ve skutečnosti vypadá.



```
analogout | Arduino 1.0
File Edit Sketch Tools Help
analogout
int brightness=255; // jaky jas
int brightness1=255;
int brightness2=255;
int brightness3=255;
int brightness4=255;
int brightness5=255;
int fadeAmount=5; // o kolik bodu zeslabnout
int a,b,c,d,e,f;
void setup()
{
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(3, OUTPUT);
}

void loop()
{
  analogWrite(11, brightness);
  analogWrite(10, brightness1);
  analogWrite(9, brightness2);
  analogWrite(6, brightness3);
  analogWrite(5, brightness4);
  analogWrite(3, brightness5);
}
1 Arduino Uno on /dev/ttyACM0
```

Obr. 8. Vývojové prostředí Wiring

Pracovat ve Wringu jako v C/C++ umožňuje určité výhody. Pokud například budeme chtít programovat procesor, pro který ve Wringu není podpora. Následující program po nahrání do procesoru rozblíká LED na desce Arduino, připojenou k 13. výstupu.

```

1. void setup() { pinMode(13, OUTPUT); //definuje pin 13 jako výstup
2.           }
3. void loop() { digitalWrite(13, HIGH); // nastaví LED ON
4.           delay(1000);           // počká 1sekundu
5.           digitalWrite(13, LOW); // nastaví LED OFF
6.           delay(1000);
7.           }

```

V čistém C/C++ by takový kód vypadal následovně:

```

1. #include <avr/io.h>           // definice pinů Pbn (registrů DDRB, PORTB)
2. #define F_CPU 16.0E6         // nastaví frekvenci procesoru
3. #include <util/delay.h>       // práce s časováním - funkce _delay_ms

```

Na rozdíl od varianty ve Wringu je zde nutné zahrnout knihovnu avr, která definuje registry. Pro funkci čekání knihovnu delay.h.

```

1. int main (void) {
2.   DDRB |= (1 << PB5);        // pinMode(13, OUTPUT)
3.   while (1) {                // nekonečný cyklus
4.     PORTB |= (1 << PB5);      // digitalWrite(13, HIGH)
5.     delay_ms(1000);          // počká jednu vteřinu
6.     PORTB &= ~(1 << PB5);    // digitalWrite(13, LOW)
7.     delay_ms(1000);          // počká jednu vteřinu
8.   } return 1;}

```

Rozdíl mezi oběma kódy je patrný. Kód v C/C++ pracuje přímo s registry procesoru, které mohou být na různých čípech odlišné. Oba kódy vyžadují knihovnu avr-libc, která je základní volně šiřitelná knihovna pro mikroprocesory AVR (procesor, na kterém je založeno například Arduino).

Poměrně užitečná nástavba, neboli „Shield“ ,je ethernetové rozhraní, které lze k Arduino připojit a komunikovat tak se serverem/clientem na lokální či globální síti. V případě povolení přístupu Arduino na Internet je nutné nastavit MAC adresu zařízení. V posledních verzích toho shieldu je již adresa dodávána se zařízením, ale lze vložit i vlastní MAC adresu. Může se ale stát, že tuto adresu již používá jiné zařízení a může dojít k nefunkčnosti obou zařízení. Již připojený Ethernet Shield je ukázán na následujícím obrázku. Komunikace probíhá oboustranně a lze tak Arduino ovládat jak z vnitřní, tak vnější sítě, což je výhodné u úkolů zabývajících se regulací.



Obr. 9. Arduino + Ethernet Shield

Jednoduchý příklad ukazuje jak lze Ethernet Shield začlenit do domácí sítě. Je nutné nastavit MAC adresu a IP adresu zařízení.

```
1#include <Ethernet.h>
```

```
2 byte mac[] = { 0xAA, 0xAB, 0xAB, 0xAB, 0xAB, 0xAA };
```

```
3 byte ip[] = { 10, 0, 0, 20 };
```

```
4 void setup()
```

```
5 {
```

```
6   Ethernet.begin(mac, ip);
```

```
7 }
```

```
8 void loop ()
```

```
9 {;}
```

Příkazem `Ethernet.begin` sdělujeme řadiči W5100, jakou MAC a IP adresu má použít. Pokud ze vzdáleného počítače zadáme příkaz:

`ping 10.0.0.20` (16),

dostaneme od serveru odpověď a Ethernet Shield nás upozorní bliknutím LED. `Ethernet.begin` umožňuje nastavit nejen MAC a IP, ale i gateway a masku podsítě. Jednoduchou modifikací pak získáme jednoduchý telnet server. Hlavička zůstane v obou případech totožná, jenom je nutné definovat novou proměnou `server`.

```
Server server = Server(23);
```

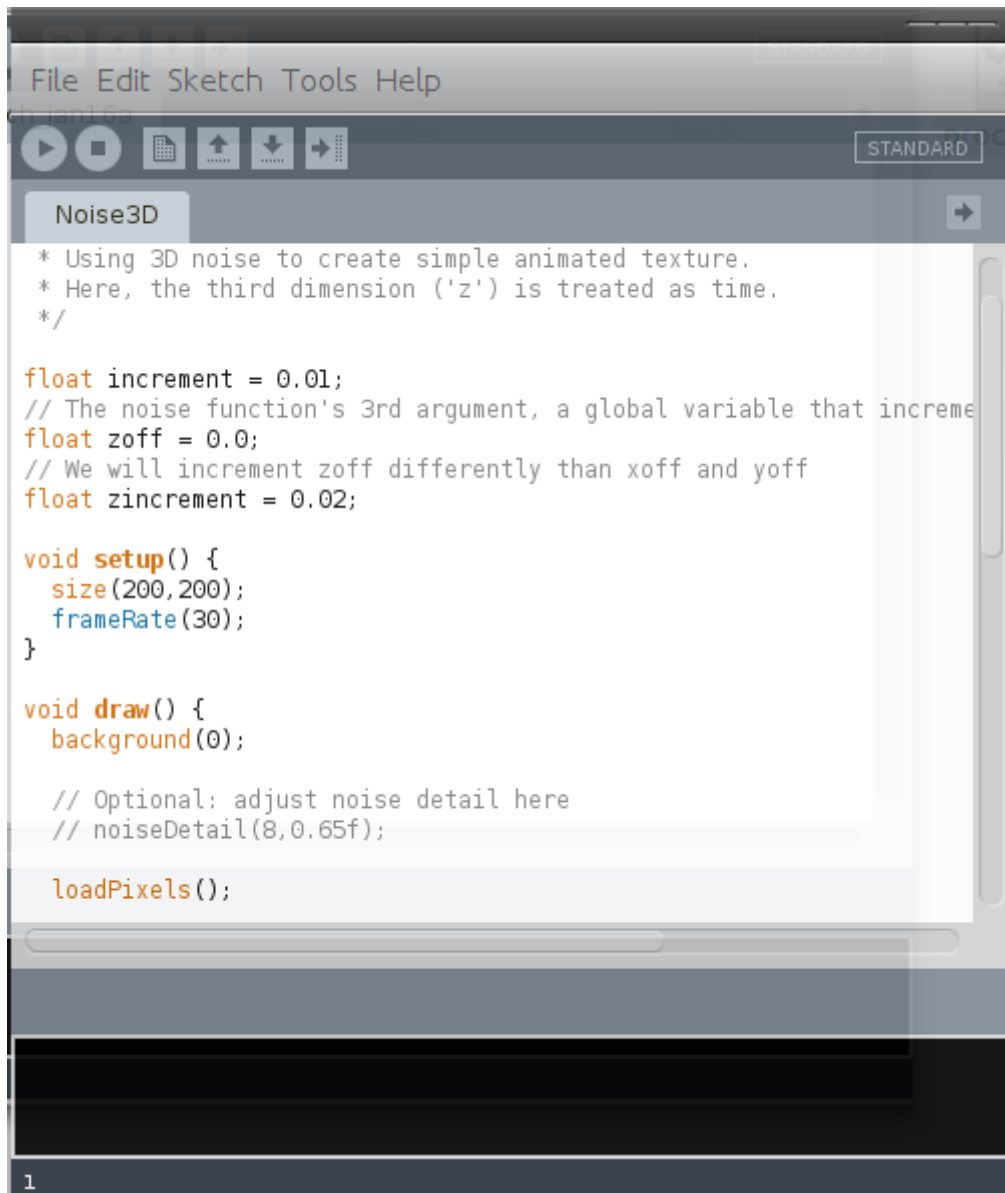
Parametr udává číslo TCP portu, na kterém bude server přístupný.

```
4. void setup()
5. {
6. Ethernet.begin(mac, ip);
7. server.begin();
8. }
9. void loop ()
10. {
11. Client client = server.available();
12. if (client) {
13. server.write(client.read());
14. }}
```

Metoda `begin()` na řádce 7 nainicializuje server. Metoda `available()` vrací objekt třídy `client`, který slouží ke čtení dat z předchozího spojení. Z řádku 13 je patrné, že server vrátí hodnotu, kterou jsme napsali do `client` programu. Vykoná tak proceduru `echo`.

7.3.1 IDE Processing

Na první pohled vypadá Processing stejně jako Wiring. Rozdíl mezi těmito programovacími jazyky spočívá v jejich účelu. Wiring je určen k nahrávání řídicích programů do mikroprocesoru, naproti tomu Processing je určen ke komunikaci s Arduinem, například, když chceme odečíst nějaká data, která Arduino shromažďuje.



```
File Edit Sketch Tools Help
STANDARD
Noise3D
* Using 3D noise to create simple animated texture.
* Here, the third dimension ('z') is treated as time.
*/

float increment = 0.01;
// The noise function's 3rd argument, a global variable that increments
float zoff = 0.0;
// We will increment zoff differently than xoff and yoff
float zincrement = 0.02;

void setup() {
  size(200,200);
  frameRate(30);
}

void draw() {
  background(0);

  // Optional: adjust noise detail here
  // noiseDetail(8,0.65f);

  loadPixels();
}
1
```

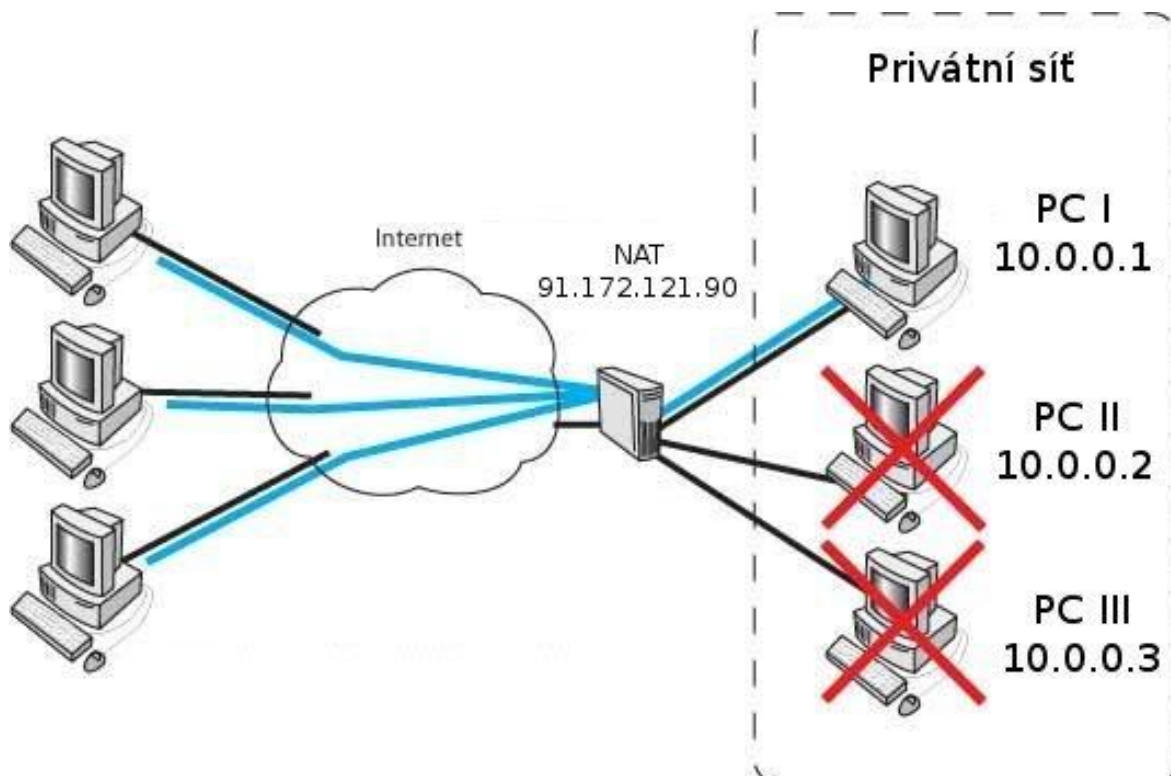
Obr. 10. Vývojové prostředí Processing

8 DEMONSTRAČNÍ PROGRAM

Základní funkce celého systému demonstrují dva programy, které jsou předmětem praktické části práce, jeden na straně serveru a druhý na straně klienta. Programy si mezi sebou vyměňují informace, čímž nám zobrazí na klientském počítači aktuální stav programu, který ve stejném čase pracuje na serveru. Toto představuje analogii, kdy v chráněném objektu vznikne poplach aktivací nějakého senzoru a tento stav je okamžitě přenesen na klientské zařízení. Základní příkaz pro vytvoření SSH spojení mezi serverem a klientem

```
ssh uzivatel@server (17)
```

kde *uzivatel* je jméno uživatele, na kterého se chceme přihlásit a server je v tomto případě 10.0.0.1, což je vnitřní IP adresa serveru. Na této adrese je server dostupný pouze z lokální sítě. Pokud bychom se chtěli k serveru připojit z Internetu, budeme muset vytvořit na routeru NAT, který přesměruje veškerá spojení z vnější adresy právě na lokální IP adresu serveru, tedy 10.0.0.1. Názorně to prezentuje obr. 7.



Obr. 11. Příklad NAT spojení

V případě, že bychom se připojovali z vnější sítě, museli bychom zadat příkaz:

```
ssh uzivatel@91.172.121.90
```

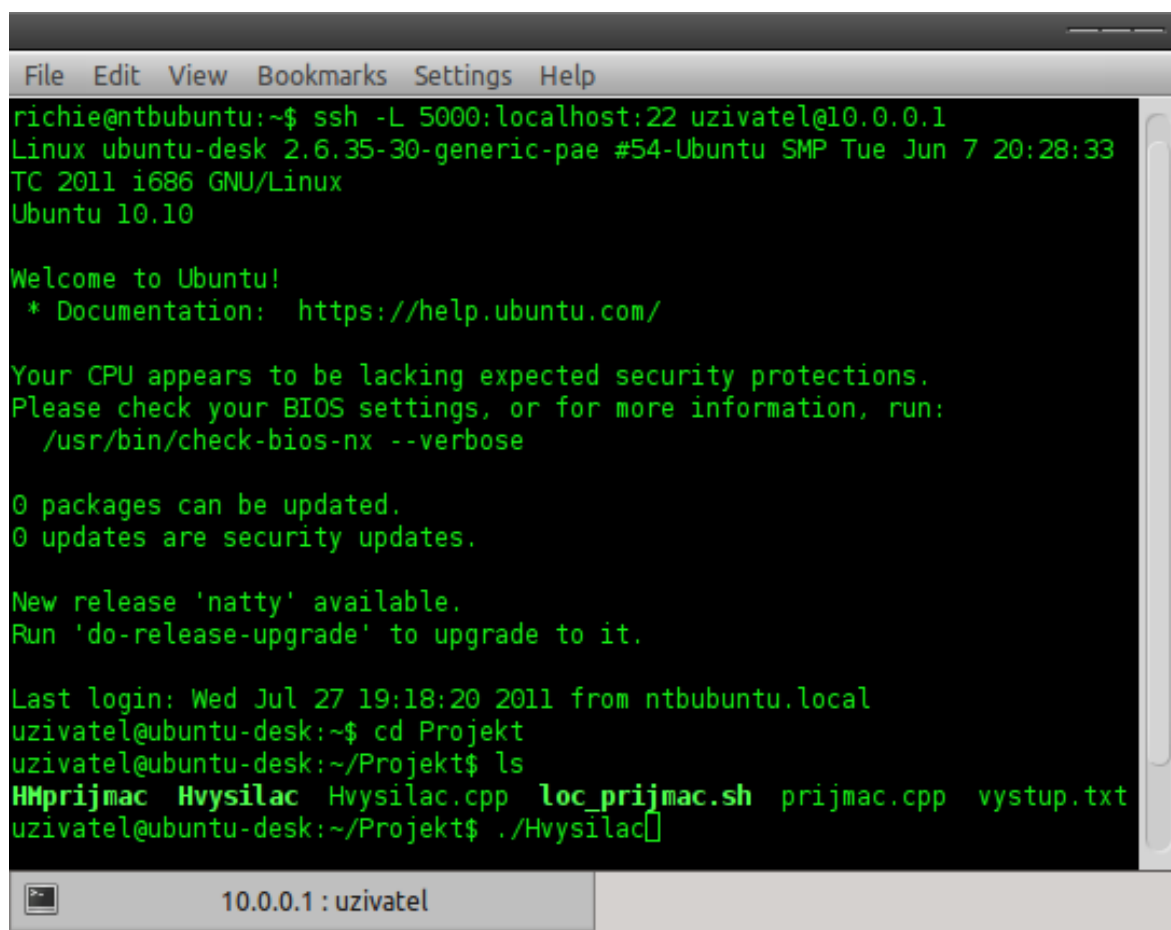
 (18)

91.172.121.90 značí veřejnou IP adresu. V podstatě se jedná o adresu modemu (routeru), který se jí prezentuje na Internetu. Po připojení na ni budeme automaticky přesměrováni na server s lokální adresou 10.0.0.1. Pro zvýšení bezpečnosti je využito SSH tunelu mezi porty 22 na straně serveru a 5000 na straně vzdáleného zařízení. Příkaz (16) značí, jak lze toto spojení s tunelem vytvořit

```
ssh -L 5000:localhost:22 uzivatel@server
```

 (19)

Parametr `-L` udává, že se má uvnitř SSH spojení `uzivatel@server` vytvořit tunel od určitého portu vzdáleného zařízení k určitému portu serveru. Dále je možné použít `ssh` s parametrem `-R`, což značí, že se má realizovat tunel od určitého portu serveru k určitému portu vzdáleného zařízení. Z obr 7. je patrné, že není potřeba zadávat žádné heslo, právě díky RSA autentizaci, která je popsána v teoretické části. Po zadání příkazu (16) se ocitneme na serveru, kde spustíme testovací program s názvem vysílač.



```
File Edit View Bookmarks Settings Help
richie@ntubuntu:~$ ssh -L 5000:localhost:22 uzivatel@10.0.0.1
Linux ubuntu-desk 2.6.35-30-generic-pae #54-Ubuntu SMP Tue Jun 7 20:28:33
TC 2011 i686 GNU/Linux
Ubuntu 10.10

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

Your CPU appears to be lacking expected security protections.
Please check your BIOS settings, or for more information, run:
  /usr/bin/check-bios-nx --verbose

0 packages can be updated.
0 updates are security updates.

New release 'natty' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed Jul 27 19:18:20 2011 from ntubuntu.local
uzivatel@ubuntu-desk:~$ cd Projekt
uzivatel@ubuntu-desk:~/Projekt$ ls
HMprijmac  Hvysilac  Hvysilac.cpp  loc_prijmac.sh  prijmaccpp  vystup.txt
uzivatel@ubuntu-desk:~/Projekt$ ./Hvysilac[]
```

Obr. 12. Ukázka SSH připojení

8.1 Popis demonstračních programů

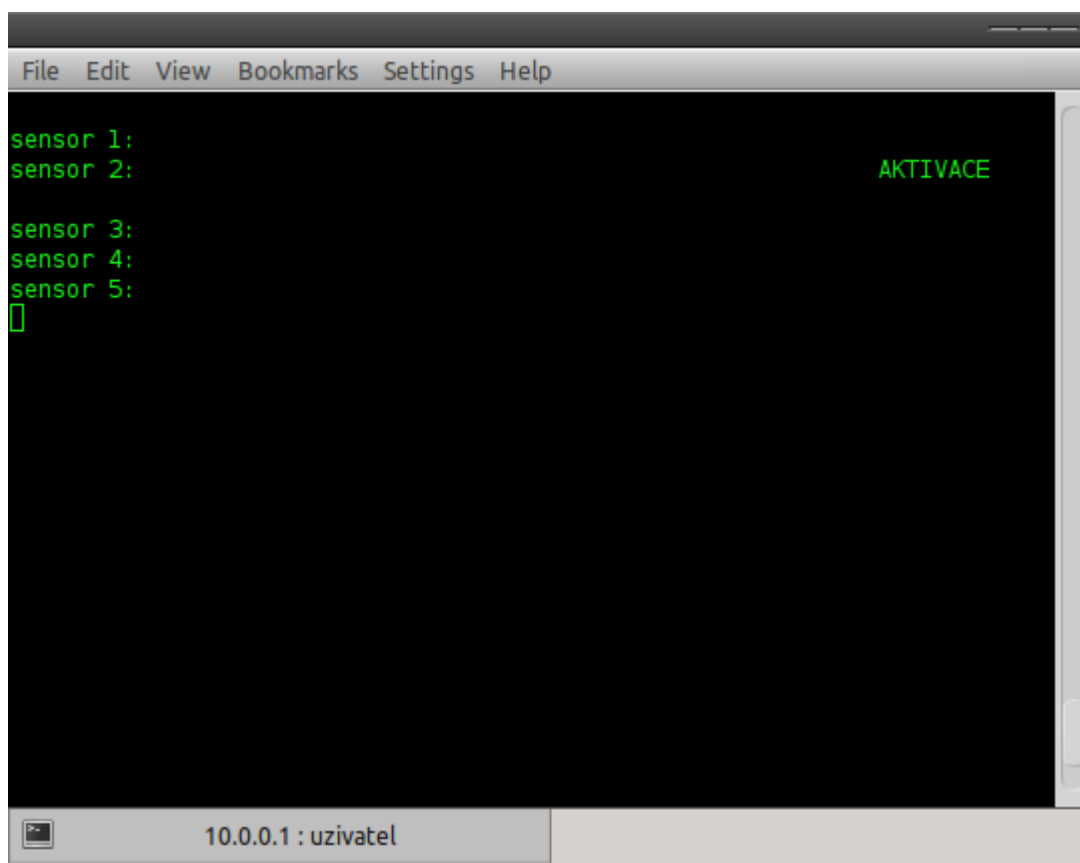
O náhodnou aktivaci se stará funkce rand.

1. f=rand() % 11;
2. sleep(f);
3. system("clear");
4. s=rand() % 5;

Náhodné číslo od 1 do 5, které reprezentuje jednotlivý senzor, je generováno v náhodných intervalech. Po vygenerování čísla program otestuje, kterému senzoru číslo odpovídá a podle toho vyhodnotí aktivaci. O této proceduře nás informuje výpisem „AKTIVACE“ u aktuálně aktivovaných senzorů. Výstup programu do souboru zajišťují následující procedury.

5. vystup = fopen("/media/virtual/vystup.txt","wt");
6. fprintf(vystup,"1");fclose(vystup);

Příkaz na pátém řádku otevírá soubor vystup.txt pro zápis. Šestý příkaz ukládá aktuálně aktivovaný senzor do výše zmíněného souboru.



```
File Edit View Bookmarks Settings Help

sensor 1:
sensor 2:          AKTIVACE

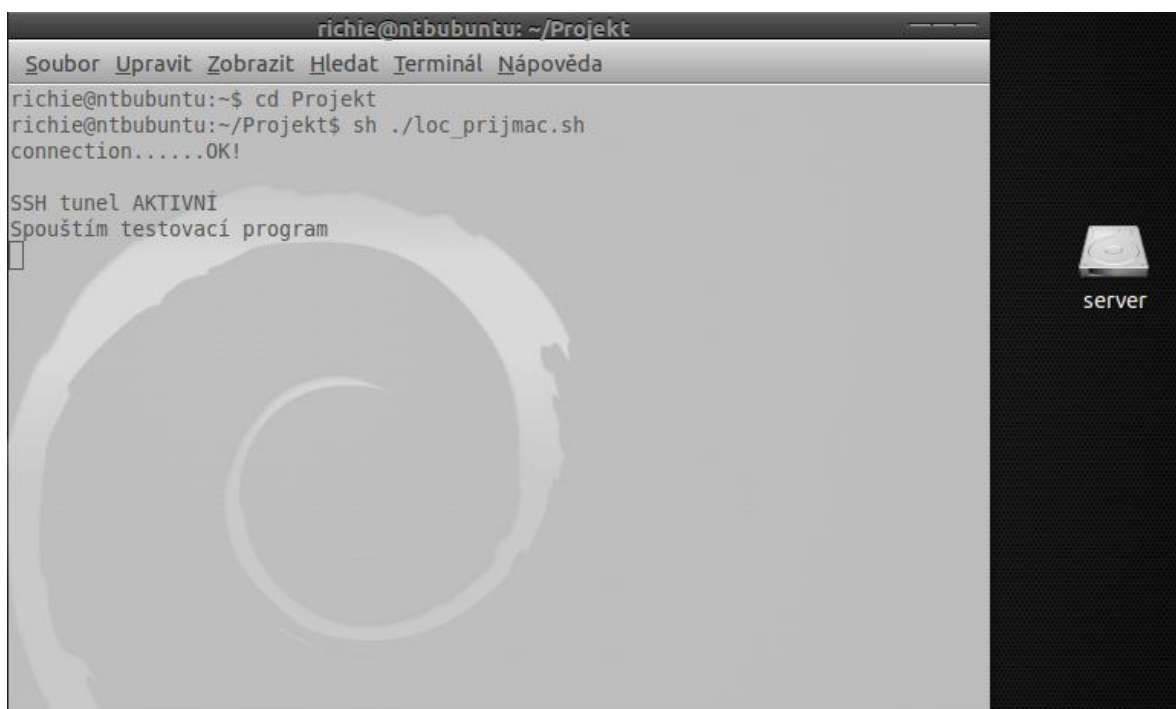
sensor 3:
sensor 4:
sensor 5:
█

10.0.0.1 : uzivatel
```

Obr. 13. Demonstrační program na straně serveru

Program na straně klienta je spouštěn skriptem, který propojí klientský program s daným portem tunelu SSH spojení.

1. `#!/bin/sh`
2. `sshfs -p 5000 localhost:/home/uzivatel/Projekt/ /media/server &&`
3. `if [-d /media/server]`
4. `then`
5. `echo OK! && printf '\nSSH tunel AKTIVNÍ'`
6. `else echo Not OK! && exit`
7. `fi &&`
8. `printf '\nSpouštím testovací program\n' &&`
9. `./Prijmac`



```
richie@ntbubuntu: ~/Projekt
Soubor Upravit Zobrazit Hledat Terminál Nápověda
richie@ntbubuntu:~$ cd Projekt
richie@ntbubuntu:~/Projekt$ sh ./loc_prijmac.sh
connection.....OK!

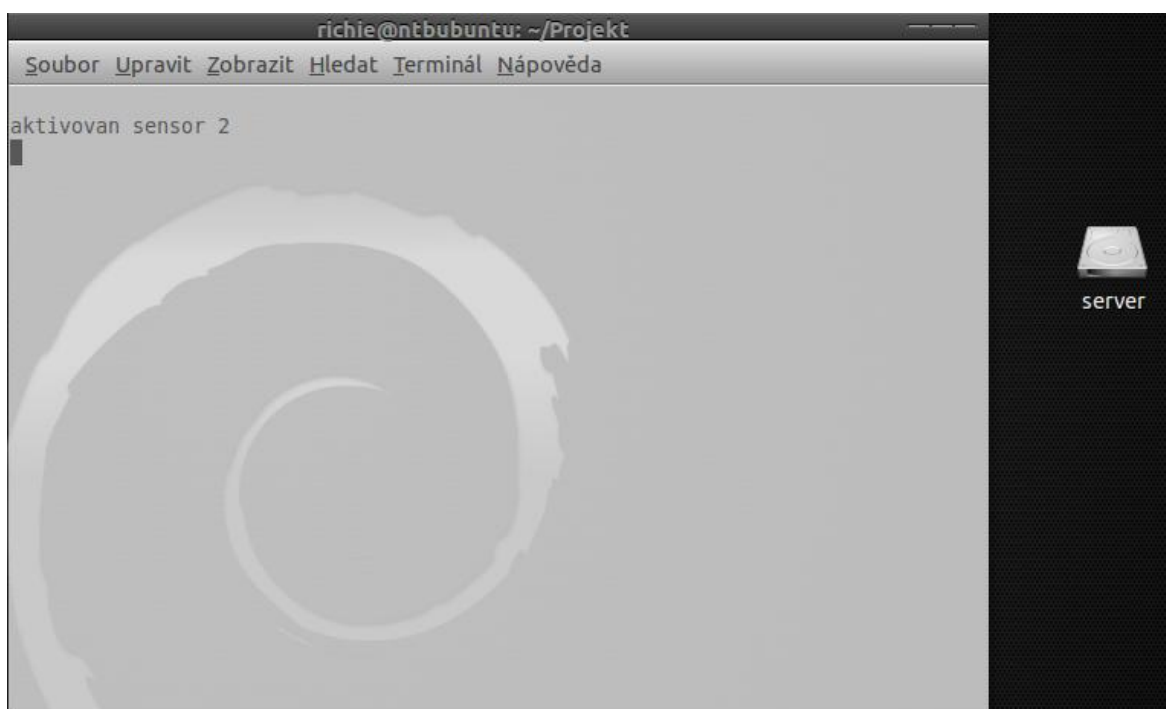
SSH tunel AKTIVNÍ
Spouštím testovací program

```

Obr. 14. Ukázka funkce spouštěcího skriptu

Pomocí tohoto skriptu se připojíme k SSH tunelu přímo k požadovanému portu, v tomto případě 5000. Program SSH File systém (dále jen SSHFS) zajistí přímé spojení mezi

požadovanými adresáři, podobně jako například NFS protokol, ale na rozdíl od tohoto protokolu je celé spojení šifrované. Pokud je spojení úspěšné, dojde k vytvoření adresáře /media/server, do kterého se automaticky připojí adresář z serveru. Můžeme tak zároveň testovat, zda proběhlo spojení úspěšně. Ve spouštěcím skriptu toto zajišťuje podmínka na 3. řádku. Zároveň skript ukončí v případě, že SSH spojení neproběhlo v pořádku. V případě splnění podmínky dojde ke spuštění klientského programu, který se již automaticky připojí



Obr. 15. Demonstrační program na straně klienta

na server.

Příkazem 1 otevíráme soubor s aktuální konfigurací senzorů pouze pro čtení (na rozdíl od serverového programu). Program zde pouze aktuální konfiguraci odčítá, ale nepřepisuje. Druhý příkaz ukládá do své paměti hodnotu aktuálně aktivovaného senzoru, aby program mohl uživateli tento senzor zobrazit.

1. {vstup = fopen("/media/server/vystup.txt", "rt");
2. fscanf (vstup, "%u", &r);

[4][5]

ZÁVĚR

Cílem práce bylo poukázat na alternativní řešení v oblasti zabezpečení objektů s využitím standardní ethernetové komunikace a informování správce objektu o stavu senzorů v chráněném objektu bezpečnou cestou přes Internet a hlavně zdarma. Je zde popsána bezpečnost spojení mezi serverem a klientem na vzdáleném zařízení a návrhy, jak dosáhnout co nejbezpečnějšího přenosu informací přes Internet. Součástí práce je charakteristika alternativních protokolů, přes které komunikace mezi serverem a klientem může probíhat, jako jsou FTP či NFS a jejich nevýhody.

Bezpečnost se v posledních letech stává stále více aktuálním tématem, a to jak bezpečnost informací, tak zabezpečení objektů. Současný trend dnešní společnosti je mít pod kontrolou veškeré dění v objektech, které vlastníme, i když jsme v práci nebo na služební cestě. Technologie, která se momentálně k těmto účelům využívá, je založena na principech používaných již desítky let. Přestože prochází určitou modernizací, základní koncepty se prakticky nemění. Nabízí se tedy otázka, proč se k těmto účelům stále využívá zastaralých GSM sítí, když můžeme k doručení poplachové informace využít Internet? Výhody jsou zřejmé – nízká cena, vyšší míra bezpečnosti a především rychlost. Stejně jako se rozvíjí průmysl komerční bezpečnosti, rozvíjí se i průmysl informačních technologií. Dovolují si tvrdit, že tyto dvě odvětví mohou spolu nejen spolupracovat, ale jejich kombinací můžeme dosáhnout podstatně efektivnějšího zabezpečení.

Využití takto řešeného bezpečnostního systému v průmyslu komerční bezpečnosti by znamenalo zásadní milník, jelikož nepotřebuje zmiňované ústředny ani další periferie využívané v běžných bezpečnostních systémech. Důvodem, proč se bezpečnostní systémy nerozšiřují do běžných domácností, je především vysoká cena a vysoké náklady na jejich instalaci. Řešení, které zde popisují, tyto náklady snižuje na minimum a to hlavně proto, že můžeme využít zařízení, které již běžně vlastníme téměř všichni, což jsou osobní počítače, routery a access-pointy. Systém nepotřebuje žádné ústředny EZS ani kabeláž. Spolehlivě využije bezdrátovou síť, ke které se běžně připojujeme doma k Internetu. Ústřednu nahradí osobní počítač. Vysoká míra variability umožňuje přizpůsobit řešení na míru požadavkům zákazníka a snížit cenu systému opravdu na naprosté minimum – pouhou cenu za jednotlivé detektory. V konečném důsledku může vnést takový systém bezpečnost do každého rodinného domu na světě a v důsledku toho snížit kriminalitu na celém světě.

ZÁVĚR V ANGLIČTINĚ

The aim of the dissertation has been pointed to alternative solutions for building security using standard Ethernet communication and inform the administrator of the building about the status of the sensors in a protected object via the Internet safely and most importantly for free. The dissertation points to security of the connection between server and client on the distant device and also suggests how to achieve the safest possible transfer of information via the Internet. In addition, there are alternative protocols described described which communication between the server and the client can take place, such as FTP or NFS, and their disadvantages.

Security has recently become an increasingly hot topic, both information security and building security. The current trend in today's society is to control everything that happens in the buildings that we own, even when we are at work or on business. Technology currently used for these purposes is based on the principles used for decades. While it is going through some upgrades, the basic concepts are practically unchanged. The question is, why we are still using outdated GSM networks when we can send information from the alarm via Internet. The advantages are obvious - low cost, higher level of safety and especially high speed. Just as the industry of commercial security is developing, as the industry of information technology is developed too. Let me say that these two sectors cannot only coexist together, but with the combination of these sectors we can achieve the security to be much more effective.

Using of this security system in the industry of commercial security would mean a great breakthrough. The reason why the security systems are not extended to ordinary households is especially the high costs and high price of installation. The solution described here, reduces these costs to a minimum, mainly because we can use devices which almost all of us currently own such as personal computers, routers and access points. The system requires no wiring or control panel. The system is using the same wireless network as we are normally connecting to the Internet at home. The personal computer is then replacing the control panel. A high amount of variability allows us to adapt the solutions to customer requirements and reduce system cost to a minimum - just the price for individual detectors. Ultimately, this system can secure every house in the world and reduce crime in the world.

SEZNAM POUŽITÉ LITERATURY

Monografie:

- [1] Komunita Linuxu, Linux dokumentační projekt 4. vyd. Praha: Computer Press, 2008. 1336 s. ISBN 978-80-251-1525-1
- [2] KRČMÁŘ, P.: Linux-postavte si počítačovou síť 1. vyd. Praha: Computer Press, 2008. 184 s. ISBN 978-80-247-1290-1
- [3] CONWREL, P., ZONG, L.: Linux, Thormb. Diath. Haemorrh., USA, 2009; 19:p. 186-19, ISBN 978-80-7318-707-1
- [4] ECKEL, Bruce. *Myslíme v jazyku C*. Vyd. 1. Praha: Grada, 2000, 554 s. ISBN 80-247-9009-2.
- [5] ECKEL, Bruce. *Myslíme v jazyku C*. Vyd. 2. Praha: Grada Publishing, 2006, 608 s. ISBN 80-247-1015-3.

Internetové zdroje:

- [6] Práva souborů. *Wiki.ubuntu.cz* [online]. [cit. 2012-01-16]. Dostupné z: <http://wiki.ubuntu.cz/Pr%C3%A1va%20soubor%C5%AF>
- [7] Linux z blízka: Bezpečnost Linuxu [online]. [cit. 2011-10-16]. Dostupný z WWW: <http://linuxzblizka.blog.zive.cz/2009/11/bezpecnost-linuxu/>.
- [8] Root: Zen Kernel [online]. [cit. 2011-10-16]. Dostupný z WWW: <http://www.root.cz/clanky/zen-kernel-vytunene-linuxove-jadro/>.
- [9] Linuxsoft: Linuxová bezpečnost z pohledu uživatele [online]. [cit. 2011-10-16]. Dostupný z WWW: http://www.linuxsoft.cz/article.php?id_article=115.
- [10] Linuxexpres: SSH receptem na bezpečnost I [online]. [cit. 2011-10-16]. Dostupný z WWW: <http://www.linuxexpres.cz/praxe/ssh-receptem-na-bezpecnost-1>.
- [11] Linuxexpres: SSH receptem na bezpečnost II [online]. [cit. 2011-10-16]. Dostupný z WWW:

<<http://www.linuxexpres.cz/praxe/ssh-receptem-na-bezpecnost-2>>.

[12] Wiki Ubuntu: SSH [online]. [cit. 2011-10-16]. Dostupný z WWW:

<<http://wiki.ubuntu.cz/SSH>>.

[13] Linuxexpres: Správa linuxového serveru: Praktické rady pro zabezpečení SSH

[online]. [cit. 2011-10-16]. Dostupný z WWW:

<<http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-prakticke-rady-pro-zabezpeceni-ssh>>.

[14] Wikipedia: FTP [online]. [cit. 2011-10-16]. Dostupný z WWW:

<http://cs.wikipedia.org/wiki/File_Transfer_Protocol>.

[15] Wikipedia: SSH file transfer protocol [online]. [cit. 2011-10-16].

Dostupný z WWW: <http://cs.wikipedia.org/wiki/SSH_file_transfer_protocol>.

[16] Wikipedia: NFS [online]. [cit. 2011-10-16].

Dostupný z WWW: http://cs.wikipedia.org/wiki/Network_File_System

[17] Wiki Ubuntu: NFS [online]. [cit. 2011-10-16].

Dostupný z WWW: <http://wiki.ubuntu.cz/NFS>

[18] Wikipedia: Samba [online]. [cit. 2011-10-16].

Dostupný z WWW: [http://cs.wikipedia.org/wiki/Samba_\(software\)](http://cs.wikipedia.org/wiki/Samba_(software))

[19] Fakulta informatiky Masarykovy univerzity: Samba [online]. [cit. 2011-10-16].

Dostupný z WWW: <http://www.fi.muni.cz/~xbatko/samba/>

[20] Wiki Ubuntu: Samba [online]. [cit. 2011-10-16].

Dostupný z WWW: <http://wiki.ubuntu.cz/Samba>

[21] Linuxexpres: Základní konfigurace bezpečného FTP serveru

[online]. [cit. 2011-10-16].

Dostupný z WWW:

<http://www.linuxexpres.cz/praxe/zakladni-konfigurace-bezpecneho-ftp-serveru>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

| | |
|-------|----------------------------------------------------|
| UNIX | Uniplexed Information Computing System |
| SSH | Secure Shell |
| NFS | Network File System |
| FTP | File Transfer Protocol |
| IT | Information technology |
| GNU | GNU's Not Unix! |
| OS | Operační systém |
| GPL | General Public License |
| MAC | Media Access Control |
| RSH | Remote shell |
| TLS | Transport Layer Security |
| SSL | Secure Sockets Layer |
| RSA | Rivest, Shamir, Adleman |
| BSD | Berkeley Software Distribution |
| TCP | Transmission Control Protocol |
| ASCII | American Standard Code for Information Interchange |
| IETF | Internet Engineering Task Force |
| SFTP | Secure File Transfer Protocol |
| SMB | Server Message Block |
| SSHFS | Secure Shell File System |
| UNIX | Uniplexed Information Computing System |
| SSH | Secure Shell |
| NFS | Network File System |
| FTP | File Transfer Protocol |

IT Information technology

SEZNAM PŘÍKAZŮ

1. `ls -laF`
2. `passwd user_name`
3. `ssh-keygen -t dsa`
4. `ssh-add`
5. `ssh uzivatel@server`
6. `awk 'Invalid user/ {print $8}' /var/log/auth.log{,0} | sort | uniq -c`
7. `last`
8. `mount 192.168.1.2:/home /media/nfs_server`
9. `smbpasswd -a jmeno_uzivatele`
10. `smbpasswd -x jmeno_uzivatele`
11. `gedit /etc/samba/smb.conf`
12. `/etc/init.d/samba restart`
13. `echo Hello world! > /dev/ttyS0`
14. `cat /dev/ttyS0`
15. `ping 10.0.0.20`
16. `ssh uzivatel@server`
17. `ssh uzivatel@91.172.121.90`
18. `ssh -L 5000:localhost:22 uzivatel@server`

SEZNAM OBRÁZKŮ

| | |
|-------------------------------------------------------|----|
| Obr. 1: Výpis práv souborů..... | 15 |
| Obr. 2: Výpis z lm-sensors..... | 18 |
| Obr. 3: IPMI Sensors..... | 18 |
| Obr. 4: Přihlášení k SSH..... | 25 |
| Obr. 5: Seznam podporovaných souborových systémů..... | 34 |
| Obr. 6: Vstup do počítače pomocí optočlenu..... | 40 |
| Obr. 7: Arduino Duemilanove..... | 42 |
| Obr. 8: Vývojové prostředí Wiring..... | 43 |
| Obr. 9: Arduino + Ethernet Shield..... | 45 |
| Obr. 10 Vývojové prostředí Processing..... | 47 |
| Obr. 11: Příklad NAT spojení..... | 48 |
| Obr. 12: Ukázka SSH připojení..... | 49 |
| Obr. 13: Demonstrační program na straně serveru..... | 50 |
| Obr. 14: Ukázka funkce spouštěcího skriptu..... | 51 |
| Obr. 15: Demonstrační program na straně klienta..... | 52 |

SEZNAM TABULEK

| | |
|-----------------------------------------------------------|----|
| Tabulka 1: Význam práv souborů..... | 14 |
| Tabulka 2: Změna práv..... | 14 |
| Tabulka 3: Přehledné zobrazení práv souborů a složek..... | 16 |
| Tabulka 4: Zapojení sériového portu..... | 41 |

SEZNAM PŘÍLOH

Příloha PI: výpis souboru /etc/vsftpd/vsftpd.conf

Příloha PII: výpis souboru /etc/ssh/sshd_config

Příloha PIII: CD s ukázkovými programy

PŘÍLOHA P I: výpis souboru /etc/vsftpd/vsftpd.conf

`anonymous_enable=YES/NO` - Anonymní přihlášení. Uživatelská jména `anonymous` a `ftp` jsou definována jako anonymní.

`anon_upload_enable=YES/NO` - Zapnutím povolíte anonymním uživatelům za určitých podmínek provádět upload souborů. Aby tato volba fungovala, musí být zapnuta volba `write_enable` a anonymní ftp uživatel musí mít právo zápisu do požadovaného adresáře.

`anon_mkdir_write_enable=YES/NO` - Povolí vytváření adresářů anonymními uživateli. Opět aby tato volba fungovala, musí být zapnuta volba `write_enable` a anonymní ftp uživatel musí mít právo zápisu do nadřazeného adresáře.

`local_enable=YES/NO` - Povolí lokální přihlášení. Používá se běžný uživatelský účet uvedený v `/etc/passwd`.

`write_enable=YES/NO` - Povolení zápisu. U distribucí se zapnutým SELinuxem je potřeba povolit upload souborů. Popis nastavení SELinuxu pro FTP server lze najít v manuálové stránce.

`local_umask=` - Volba určuje implicitní formát masky pro přidělování přístupových práv, standardně 022.

`anon_root=` - Domovský adresář pro anonymní uživatele.

`dirmessage_enable=YES/NO` - Zapnutím mohou uživatelé obdržet zprávu při prvním vstupu do nového adresáře. Standardně se v adresáři hledá soubor `.message`, nastavení je možné změnit volbou `message_file`.

`xferlog_enable=YES/NO` - Aktivuje detailní záznam nahrávání a stahování souborů do log souboru. Standardně je použit soubor `/var/log/vsftpd.log`, toto umístění lze změnit konfigurací volby `vsftpd_log_file`.

`xferlog_std_format=YES/NO` - Pokud je volba aktivována, soubor s logováním přenosů bude zapsán v `xferlog` formátu, který používají některé statistiky pro přenos souborů. Vypnutí volby aktivuje přehlednější zápis.

`connect_from_port_20=YES/NO` - Tato povoluje port pro datovou komunikaci serveru (`ftp-data`). Z bezpečnostních důvodů někteří klienti mohou klást důraz na tuto volbu.

`chown_uploads=YES/NO` - Aktivací se všem anonymně uploadnutým souborům přidělí vlastnická práva definovaná v nastavení `chown_username`.

`idle_session_timeout=` - Časový limit (ve vteřinách), který je maximální možnou prodlevou mezi jednotlivými FTP příkazy. Po uplynutí je uživatel odpojen.

`data_connection_timeout=` - Časový limit (ve vteřinách), po který se čeká před odpojením uživatele, pokud byl přerušen datový přenos.

`nopriv_user=` - Jméno uživatele, který pro práci se soubory nemá žádná práva.

`async_abor_enable=YES/NO` - Aktivací povolíte speciální FTP příkaz `async ABOR`. Tuto volbu vyžadují některé FTP klienty, je však nebezpečná a není doporučeno ji používat.

`ascii_upload_enable=YES/NO` - Zapnutí ASCII módu pro upload.

`ascii_download_enable=YES/NO` - Zapnutí ASCII módu pro download.

`ftp_banner=` - Zpráva, která se zobrazí uživateli po přihlášení k serveru. Lze nastavit i soubor hodnotou `banner_file`.

`deny_email_enable=YES/NO` - Aktivací lze vytvořit seznam anonymních hesel - e-mailů, kterým bude odmítnuto přihlášení. Standardně je soubor obsahující seznam umístěn v `/etc/vsftpd/banned_emails`, jeho umístění lze změnit nastavením `banned_email_file`.

`chroot_list_enable=YES/NO` - Aktivací lze vytvořit seznam lokálních uživatelů, kteří budou omezeni pouze na svůj domovský adresář. (Opačně tedy vytváří seznam uživatelů, kteří nejsou omezeni jen na domovský adresář, funguje volba `chroot_local_user`). Soubor obsahující seznam umístěn v `/etc/vsftpd/chroot_list`, jeho umístění lze změnit nastavením `chroot_list_file`.

`ls_recurse_enable=YES/NO` - Zapnutím umožníte použití `ls -R`. Pokud server obsahuje velké množství souborů, může operace zabrat část paměti a snížit výkonnost serveru.

`pam_service_name=vsftpd` - Řetězec pro identifikaci v PAM vrstvě.

`userlist_enable=YES/NO` - Jestliže je tato volba zapnuta, pak se zakázaného uživatele server neptá na heslo a jeho připojení okamžitě odmítne.

`userlist_file=` - Soubor se zakázanými uživateli (standardně `/etc/vsftpd/ftpusers`). Je vhodné zakázat uživatele `root`, `nobody`.

listen=YES/NO - Standardně běží vsftpd v samostatném režimu (standalone mode). Vsftpd je tedy spustitelný pouze přímo.

tcp_wrappers=YES/NO - Tato volba umožňuje nastavení kontroly příchozích spojení přes tcp_wrapper, tedy nastavení přístupů podle IP adres (podobného nastavení je možné dosáhnout i přes ověřovací vrstvu PAM).

anon_max_rate= - Nastavení omezení datového toku [b/s].

local_max_rate= - Nastavení omezení datového toku lokálními uživateli [b/s].

Příloha PII: výpis souboru /etc/ssh/sshd_config

Package generated configuration file

See the sshd_config(5) manpage for details

What ports, IPs and protocols we listen for

Port 22

Use these options to restrict which interfaces/protocols sshd will bind to

#ListenAddress ::

#ListenAddress 0.0.0.0

Protocol 2

HostKeys for protocol version 2

HostKey /etc/ssh/ssh_host_rsa_key

HostKey /etc/ssh/ssh_host_dsa_key

#Privilege Separation is turned on for security

UsePrivilegeSeparation yes

Lifetime and size of ephemeral version 1 server key

KeyRegenerationInterval 3600

ServerKeyBits 768

Logging

SyslogFacility AUTH

LogLevel INFO

Authentication:

LoginGraceTime 120

PermitRootLogin yes

StrictModes yes

RSAAuthentication yes

PubkeyAuthentication yes

#AuthorizedKeysFile %h/.ssh/authorized_keys

Don't read the user's ~/.rhosts and ~/.shosts files

IgnoreRhosts yes

For this to work you will also need host keys in /etc/ssh_known_hosts

RhostsRSAAuthentication no

similar for protocol version 2

HostbasedAuthentication no

Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication

#IgnoreUserKnownHosts yes

To enable empty passwords, change to yes (NOT RECOMMENDED)

PermitEmptyPasswords no

Change to yes to enable challenge-response passwords (beware issues with

some PAM modules and threads)

ChallengeResponseAuthentication no

Change to no to disable tunnelled clear text passwords

#PasswordAuthentication yes

Kerberos options

#KerberosAuthentication no

#KerberosGetAFSToken no

#KerberosOrLocalPasswd yes

#KerberosTicketCleanup yes

GSSAPI options

#GSSAPIAuthentication no

```
#GSSAPICleanupCredentials yes

X11Forwarding yes

X11DisplayOffset 10

PrintMotd no

PrintLastLog yes

TCPKeepAlive yes

#UseLogin no

#MaxStartups 10:30:60

#Banner /etc/issue.net

# Allow client to pass locale environment variables

AcceptEnv LANG LC_*

Subsystem sftp /usr/lib/openssh/sftp-server

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication and
# PasswordAuthentication. Depending on your PAM configuration,
# PAM authentication via ChallengeResponseAuthentication may bypass
# the setting of "PermitRootLogin without-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and ChallengeResponseAuthentication to 'no'.

UsePAM yes
```