Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Bakalářská práce

2010 Matěj Trakal

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Aplikace pro profilování přístupových seznamů

Matěj Trakal

Bakalářská práce
2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Matěj TRAKAL**
Osobní číslo:	**I07819**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Aplikace pro profilování přístupových seznamů**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Student nastuduje možnosti vzdálené konfigurace směrovačů Cisco z hlediska možností ovlivňování přístupových seznamů, provede výběr způsobu vzdálené administrace těchto seznamů. V praktické části práce student vytvoří systém pro volbu přístupových seznamů podle předpřipravených profilů. Po konzultaci s učiteli připraví profily, které by mohli sloužit k řízení přístupu k síťovým zdrojům z počítačových učeben (minimálně 4 profily). Vytvoří jednoduché rozhraní, ve kterém bude možné profily zapínat (učebna + profil, aktivace/deaktivace, volba doby aktivace). při řešení musí být brán zřetel na bezpečnost systému z hlediska možností napadení (včetně popisů rizik a způsobů jakými se jim systém brání).

Rozsah grafických prací:
Rozsah pracovní zprávy:
Forma zpracování bakalářské práce: **tištěná/elektronická**
Seznam odborné literatury:

http://www.cisco.com/en/US/docs/ios/fundamentals/configuration/guide/12_4/c

Vedoucí bakalářské práce: **Ing. Tomáš Fidler**
Katedra softwarových technologií

Datum zadání bakalářské práce: **15. ledna 2010**
Termín odevzdání bakalářské práce: **14. května 2010**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 20. srpna 2010

Matěj Trakal

Poděkování

Tímto bych chtěl poděkovat Ing. Tomáši Fidlerovi, vedoucímu mé bakalářské práce, za jeho velmi cenné rady, pomoc při tvorbě tohoto textu a čas, po který se mi věnoval.

Souhrn

Tato práce je zaměřena na otázku zabezpečení učeben před neoprávněným používáním zdrojů sítě v určité době. Dále popisuje jednotlivé možnosti nastavení směrovače, abychom tohoto nastavení dosáhli.

Výsledkem práce je aplikační část pro ovládání směrovače pomocí předpřipravených přístupových seznamů, které mění jeho chování.

Klíčová slova

Přístupové seznamy, acl, Cisco IOS, tcl, telnet, ssh, sdm

Title

Application for profiling access control lists

Annotation

This work is aimed to question of classroom security. It handle question of using sources of network in specified time. Work described ways to set a router.

Result of the work of application part, which control router with predefined access lists. This lists change behaviour of router.

Keywords

Access Lists, acl, Cisco IOS, Tool Command Language (tcl), telnet, Secure Shell (ssh), sdm

Seznam zkratek

AAA Authentication, Authorization, and Accounting

ACL Access Control List

CAS Central Authentication Service

CDT Central Daylight Time

CLI Command Line Interface

CST Central Standard Time

DNS Domain Name System

DVD Digital Versatile Disc

ftp File Transfer Protocol

GUI Graphical User Interface

http Hypertext Transfer Protocol

https Hypertext Transfer Protocol Secure

IOS Internetwork Operating System

IP Internet Protocol

IPS Intrusion prevention system

LAN Local Area Network

MD5 Message-Digest algorithm 5

MITM Man in the midle

NTP Network Time Protocol

OOP Objektově orientované programování

PHP Hypertext Processor

QoS Quality of Service

RSA iniciály autorů Rivest, Shamir, Adleman

SDM Security Device Manager

SHA1 Secure Hash Algorithm

SSH Secure Shell

SSL Secure Sockets Layer

TCL Tool Command Language

TCLsh Tool Command Language shell

telnet Telecommunication Network

tftp Trivial File Transfer Protocol

TTY Teletype — asynchronní linka

UDP User Datagram Protocol

UPa Univerzita Pardubice

VLAN Virtual Local Area Network

VPN Virtual Private Network

VTY virtual Teletype — virtuální linka

WAN Wide Area Network

WLAN Wireless Local Area Network

Seznam obrázků

Obsah

1 Úvod

Jelikož v poslední době stoupá počet prohřešků při vypracovávání zkouškových testů, písemek a prací, bylo třeba navrhnout nový systém zabezpečení učeben proti neoprávněnému využívání služeb sítě Internet. Tato bakalářská práce se zabývá tím, jak vhodně umožnit, pro určitou učebnu, zamezení přístupu k nepovoleným službám, nebo naopak povolit omezený přístup pouze na příslušné servery.

Práce se stručně zabývá návrhem topologie sítě pro vhodné a snadné udržování přístupových seznamů a jejich aplikaci na jednotlivé učebny. Dále řeší otázku zabezpečení přístupu z vytvářené aplikace k centrálnímu prvku sítě — hlavnímu routeru, na kterém jsou všechny přístupové seznamy (ACLs) aplikovány.

Hlavním cílem práce je navrhnout zabezpečený systém správy přístupu do sítě Internet a k dalším službám poskytovaným v rámci univerzitní sítě UPa.

1.1 Cíl práce

Chceme omezovat provoz na síti. Toho budeme dosahovat změnou nastavení centrálního směrovače univerzity/fakulty. Změna konfigurace bude probíhat vzdáleně z klientské aplikace, která bude běžet na serveru[1]. Úprava nastavení by měla ovlivňovat přístupové seznamy aplikované na jednotlivých rozhraních směrovače. Tím, že rozhraním je myšlen pouze virtuální okruh lokální sítě, měl by umožnit měnit omezování provozu na síti v rámci jednotlivých učeben (virtuálních sítí).

Samotnému zabránění přístupu do sítě by mělo probíhat právě aplikací určitého přístupového seznamu na rozhraní, ke kterému je uživatel připojen. V přístupovém seznamu by měly být definovány a povoleny všechny služby využívané v rámci sítě UPa (přihlašování stanic, služba DNS, DHCP a další) a nadále podle požadovaných pravidel odepřeny služby, které by uživatel neměl mít právo v určité době využívat.

Aplikační část by bylo vhodné navrhnout tak, aby šla do budoucna začlenit do systému CAS, byla připravena na centrální způsob ověřování uživatelů a byla modulární — šla případně jednoduše rozšiřovat o další funkcionalitu. Její vhodné umístění by bylo na aplikačním serveru, tedy by měla umožňovat být spouštěna vzdáleně (webové rozhraní, aplikace typu klient/server).

[1] Aplikační server bude veřejný a všichni členové univerzity se k němu mohou připojit, tedy aplikace by měla předpokládat dostatečné zabezpečení proti neautorizovanému přístupu.

2 Cisco IOS

Cisco IOS je operační systém, navržený pro řízení provozu v síťových prvcích (routery a switche) z dílny Cisco Systems, Inc.

2.1 Možnosti ovládání

Cisco IOS je ovládaný pomocí textových příkazů psaných do konzole zařízení (CLI). Pro připojení k IOS máme několik variant:

Konzolový port na zařízení je dobře možné použít, pokud máme fyzický přístup k zařízení. Připojuje se pomocí rollover kabelu. Toto řešení je jediné možné při pořízení nového zařízení, které není předkonfigurované, jelikož nemá povoleno připojení přes telnet ani ssh.

Telnet nebo SSH je nejspíš nejpohodlnější varianta připojení k zařízení, jelikož umožňuje po správné konfiguraci připojení i ze vzdálených míst. Dále není třeba při správě více zařízení přepojovat kabely mezi fyzickými konzolovými porty. Pro připojení přes telnet a ssh je požadováno ověření uživatele pomocí hesla, případně i uživatelského jména.

Webové rozhraní není vyloženě připojení k IOS konzoli a neposkytuje plné ovládání zařízení. Tento způsob je vhodný spíše jen pro dohled nad funkcí zařízení, jelikož je názorné a nevyžaduje znalost konfiguračních příkazů.

SNMP poskytuje správu nejen Cisco zařízení, které tento protokol podporují. Pro komunikaci musí být v tomto případě dvě strany. Jedna monitorovaná (agent), která odesílá odpovědi, a druhá klientská. Ta zasílá speciální standardizované příkazy. Po příjmu dat od agenta data vyhodnocuje a zpracovává (např. monitoring stavu sítě).

TCL shell je jedno z dalších možných řešení. Využívá se odlišně, než výše zmíněné, a to tak, že se na zařízení připojí pomocí jiného výše zmíněného způsobu. Poté se z TCL shellu zažádá o konfigurační skript ze vzdáleného serveru (tftp, ftp, http, nebo dalších) a ten se nadále na zařízení vykoná. Druhá možnost je vstoupit do této konfigurační části přímo na zařízení a potřebné nastavení zadat jako řádkový příkaz, oddělený uvozovkami.

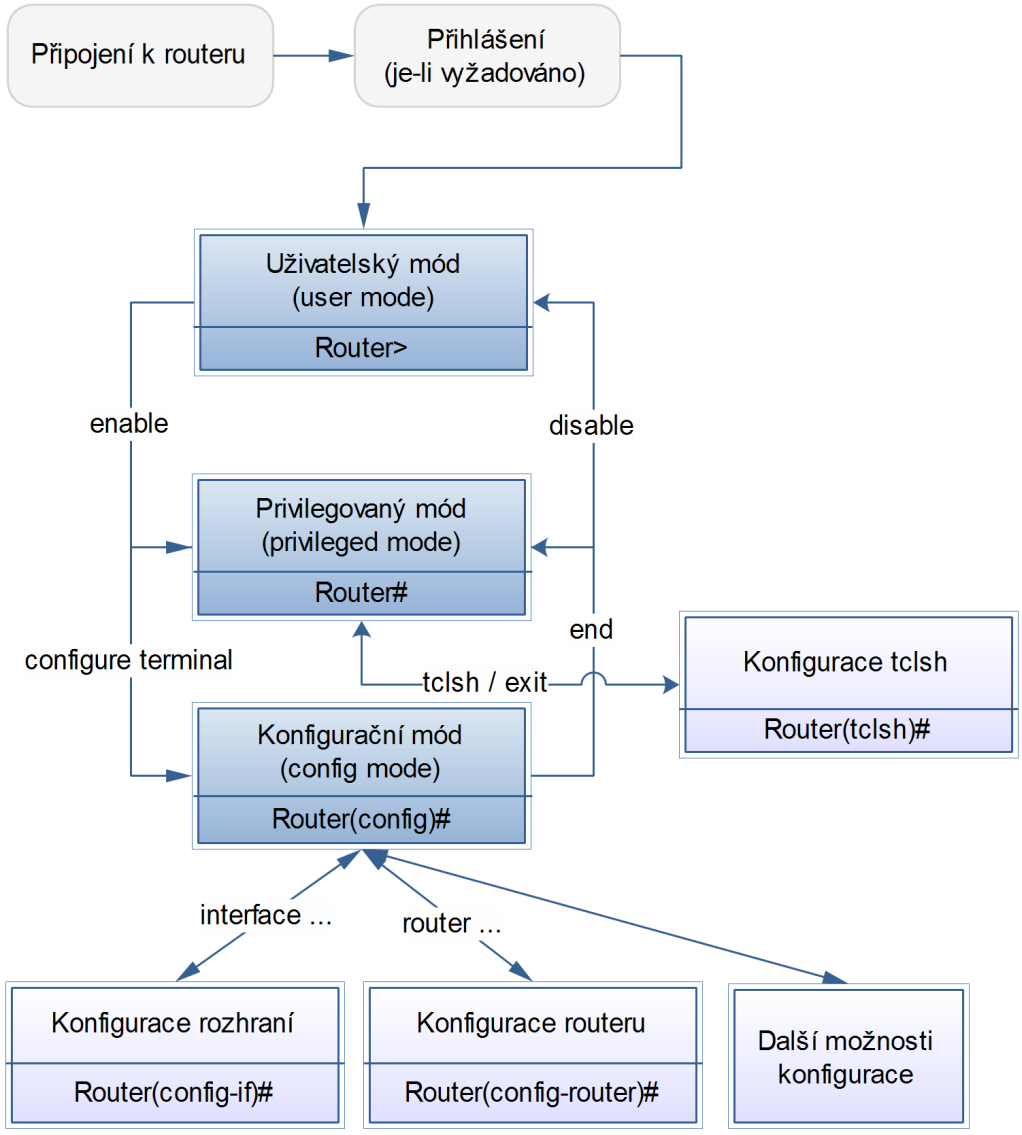
2.2 Módy IOS

Cisco IOS používá více úrovní zabezpečení přístupu. Oprávnění, které uživatel nad zařízením může provádět, určuje privilege level číslo 0–15. Díky tomuto číslu se určí pravomoci. Uživatel bez dostatečných práv může třeba na zařízení jen sledovat chod, ale nemá možnost ho konfigurovat.

Uživatelský mód má privilege level hodnotu 0. Uživatel v tomto režimu může používat velice omezené množství příkazů pro připojení na další zařízení, zasílání příkazů ping, zapnutí vyššího (privilegovaného) módu a některé další.

Privilegovaný mód umožňuje již nahlížení do aktuální konfigurace zařízení, zásahy do jeho běhu a spoustu dalších možností. Privilege level tohoto módu je nastaven na hodnotu 15, tedy jeho nejvyšší. Každému příkazu jde nastavit vlastní level v rozmezí 1–15, dle potřeby.

Konfigurační mód poskytuje možnost nastavování zařízení. Umožňuje zapínat další služby běžící na zařízení, konfiguraci rozhraní, směrování, nastavování přístupových práv a spoustu dalších.



Obrázek 1: Cisco IOS módy

3 Přístupové seznamy — Access lists

V Cisco zařízeních se přístupové seznamy používají pro určování pravidel směrování paketů na jednotlivé rozhraní zařízení, případně jednotlivé VLANy. Pomocí těchto seznamů jde zamezit přístupu dle konfigurace např.: z některých IP adres, na některé služby a jiné další možnosti. Kromě omezování provozu příchozích i odchozích dat přístupové seznamy šetří i síťové zdroje, jelikož zamezí průchodu paketu dále do sítě, kde by byly následně nejspíš zablokovány nějakým koncovým zařízením a jeho firewallem.

Seznamy se skládají z jednotlivých „vět“, kde každá věta (řádek) značí právě jedno pravidlo, buď povolení nebo zakázání, pro směrovač. Postup zpracování je ten, že se postupuje od prvního pravidla k poslednímu, kdy poslední pravidlo je vždy zakázání veškerého provozu. Pokud si některou požadovanou službu nepovolíme, bude následně právě posledním pravidlem zablokována.

3.1 Dělení přístupových seznamů

Standardní (Standard) přístupové seznamy filtrují provoz pouze na základě zdrojové IP adresy. Nedokáží filtrovat provoz na jednotlivých protokolech a jejich poskytovaných službách. Filtrují jednotlivé rozhraní zařízení jako celek, ne jednotlivé služby poskytované v rámci TCP/IP a UDP protokolů. Jsou většinou nastavovány co nejblíže koncovému zařízení.

Důvod, proč se stále používají, je jejich výpočetní nenáročnost. Zpracování je velice rychlé, zpracovává se jen na základě zdrojové IP adresy a wildcard masky[2].

Rozšířené (Extended) přístupové seznamy dokáží rozlišovat jednotlivé protokoly a služby na nich poskytované. Dokáží tedy filtrovat například webový provoz, emailové služby a další.

Jejich nevýhodou ovšem oproti standardním ACL je, že musí paket více rozebrat pro analýzu, aby dokázaly správně porovnat paket s pravidlem v rozšířeném seznamu. Tím spotřebují mnohem více systémových zdrojů a výpočetního výkonu potřebného pro jejich zpracování — zatěžují směrovač úkony, na který není primárně určen.

[2]Wildcard maska je negovanou variantou masky sítě. Udává počet zařízení, nikoli počet podsítí. Pro přepočet na wildcard masku stačí od čísla 255.255.255.255 odečíst po kvadrantech masku sítě.

3.2 Konfigurace v Cisco IOS

Základní ACL se po první konfiguraci již nedá změnit. Dá se pouze přidávat další záznam na konec. Jediná možnost opravy je zkopírování do textového souboru a oprava ručně s tím, že se upravený kód následně nakopíruje do konfigurace routeru.

Pojmenované ACL každému příkazu přiřadí jeho pořadí, pomocí kterého jdou následně jednotlivé řádky ACL mazat, nebo mezi již nakonfigurované vkládat nové. Další možnost konfigurace je, že se příkazům nejdříve specifikuje priorita (pořadí).

Vzhledem k jednodušší konfiguraci a správě pojmenovaných ACL se zde základními dále zabývat nebudu. Práce je s nimi obdobná, jen se místo názvů používají čísla.

3.2.1 Standardní pojmenované ACL

- Přidělení názvu ACL *Router(config)# ip access-list standard NÁZEV-ACL*
- Nastavení jednotlivých pravidel přístupového seznamu *Router(config-std-nacl)#[deny | permit | remark] source [source-wildcard] [log]*

3.2.2 Rozšířené pojmenované ACL

- Přidělení názvu ACL *Router(config)#ip access-list extended NÁZEV-ACL*
- Nastavení jednotlivých rozšířených pravidel *Router(config-ext-nacl)#deny | permit | remark protocol source source-wildcard [operator operand] [port port-number or name] destination [destination-wildcard] [operaotr operand] [port port-number or name][established]*

3.2.3 Aplikace na rozhraní

- Přidělení přístupového seznamu jednotlivému rozhraní *Router(config-if)#ip access-group NÁZEV-ACL [in | out]*

4 Metody vzdálené administrace

Na tomto místě popisuji, jak je možné se na směrovač připojit a zasílat na něj příkazy pro ovlivňování konfigurace.

4.1 Telnet

Telnet je relativně jednoduchá služba poskytující spojení serverové a klientské strany. Pro připojení se využívá IP adresy serveru a portu, na kterém služba standardně běží, 23.

4.1.1 Konfigurace na zařízeních Cisco

V případě routerů Cisco se před samotným připojením musí zajistit zprovoznění VTY[3] linky s ověřením přihlašovaného uživatele. Toho se dosáhne pomocí *Router (config)#enable secret HESLO*. Tím se zajistí zabezpečení privilegovaného módu pomocí hesla. Dále je třeba vykonat tyto příkazy:

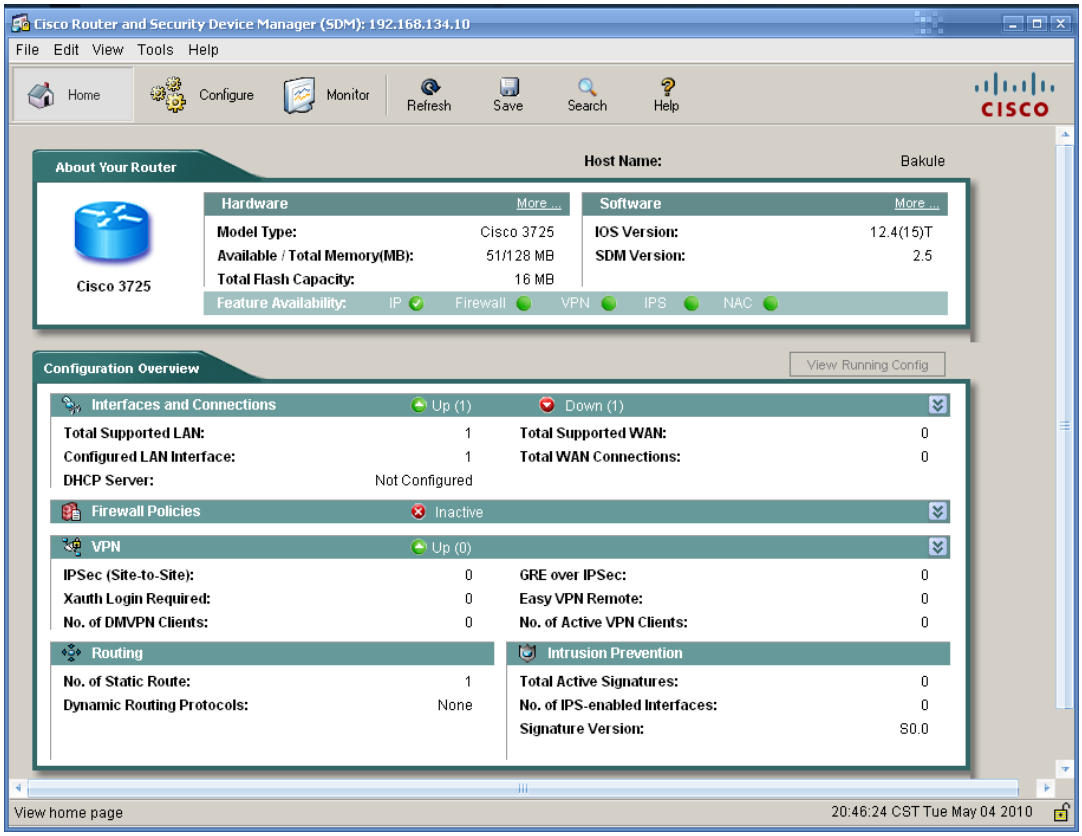
- *Router(config)#line vty 0 15* pro zapnutí 16 virtuálních linek umožňujících připojení pomocí telnetu
- *Router(config-line)#password JINE-HESLO* pro nastavení základního hesla pro přihlášení pomocí VTY linky
- *Router(config-line)#login*, čímž se povolí přihlášení k virtuální lince

Nyní máme směrovač připraven na připojení pomocí služby telnet. Tento protokol ovšem není nijak zabezpečen a je zde velká pravděpodobnost odchycení hesla případným útočníkem. Heslo a veškerá data jsou zasílána v plain–textu (nešifrovaně), tedy při odchycení paketu jdou surová data číst a zobrazovat si obsah jednotlivých paketů. MITM je bohužel relativně jednoduchá a funkční metoda, jak tyto pakety odchytit.

[3] Virtual Teletype

4.2 SDM

Služba SDM[4] je primárně určena pro správu a administraci zařízení pomocí GUI aplikační části. Usnadňuje ovládání zařízení začátečníkům, ovšem neposkytuje kompletní možnosti nastavení. Hlavní výhodou je, že dokáže detekovat zranitelnosti zařízení pomocí automatické konfigurace (zadáme, na co je zařízení v síti určeno a kterých služeb využíváme). Obsahuje různé průvodce, jak co nastavit, tedy i začátečník bez znalosti příkazů do CLI je schopen zařízení připravit k chodu. Tento nástroj poskytuje nastavení dynamického směrování, WAN přístupu, WLAN, firewall, VTPN, SSL VPN, IPS, a QoS.



Obrázek 2: Administrační část SDM, jak se reálně používá

[4]SDM: Security Device Manager

4.2.1 Zpřístupnění služby SDM na Cisco zařízeních

Pro povolení služby SDM musíme na Cisco zařízeních zapnout HTTP server a nastavit pomocí jaké autentizace se bude uživatel přihlašovat.

- Zapnutí http serveru *Router(config)#ip http server*
- Nastavení ověřování uživatele:
  - pomocí EXEC módu *Router(config)#ip http authentication enable*
  - pomocí lokálního ověření *Router(config)#ip http authentication local*

4.2.2 Využití SDM paketů

Díky tomu, že SDM umožňuje konfiguraci zařízení, musí také dokázat se zařízením komunikovat a konfigurovat ho. Po odchycení dostatečného počtu paketů pomocí nástroje Wireshark jde rozpoznat, jak jsou zprávy pro zařízení a od něj sestavovány, jak jsou zabezpečeny a hlavně, jak je využít pro vlastní zasílání konfiguračních příkazů.

```
POST /ios_web_exec/commandset HTTP/1.1
User-Agent: Mozilla/4.0 (Windows XP 5.1) Java/1.6.0_17
Host: 192.168.134.10
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
Content-type: application/x-www-form-urlencoded
Content-Length: 131

! COMMANDSET VERSION="1.0"
! OPTIONS BEGIN
! MODE="0"
! OPTIONS END
show interface FastEthernet0/0
! END
! COMMANDSET END
```

Jak je vidět z kódu, pošle se POST dotaz s hlavičkou. V té je zakódované uživatelské jméno a heslo *Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=*, pomocí *Base64* algoritmu. Po rozpoznání metody šifrování, a jistých dalších drobnostech, již není problém zasílat na router vlastní příkazy. *Mode="0"* v konfiguračních příkazech značí EXEC mód, *Mode="1"* pak režim konfigurace.

4.3 Secure Shel — SSH

Secure Shell je již zabezpečený protokol, nyní je ve dvou verzích, kdy každá poskytuje jinou úroveň zabezpečení. Už samotný protokol poskytuje autentizaci. Před samotným zahájením přenosu dat odešle server dva klíče ke klientovi. Klient následně pošle zašifrovaný klíč serveru a tím se vytvoří zabezpečený tunel mezi serverem a klientem. Až následně se zasílá samotná komunikace. Uvnitř tunelu jsou veškerá data chráněna proti odposlechnutí díky použitým klíčům. Jeden z klíčů, kterým se data šifrují, se každou hodinu mění, čímž zvyšuje bezpečnost. V průběhu cesty mezi zdrojem a cílem není možné si data přečíst, pokud útočník nevlastní privátní klíč cílové strany, nebo ještě před zahájením komunikace nepoužil metodu MITM. Zde je ale další metoda odhalení a to ta, že si klientská strana pamatuje jeden z otisků klíče serveru. Případně se tento klíč dá ověřit z více zdrojů a při jeho neplatnosti je zřejmé, že server je podvržen.

4.3.1 Konfigurace na zařízeních Cisco

Zde uvádím jak připravit směrovač pro použití protokolu SSH, aby s ním bylo možné pomocí tohoto protokolu komunikovat. U jednotlivých příkazů je stručně popsáno co způsobují, tedy by měl každý být schopen SSH na směrovači zprovoznit a začít ho využívat, namísto nezabezpečeného protokolu telnet.

- Nastavení názvu routeru: *Router(config)#hostname Central*
- Nastavení hesla pro privilegovaný mód: *Central(config)#enable secret HESLO*
- Zapnutí AAA[5]: *Central(config)#aaa new-model*
- Vytvoření uživatele: *Central(config)#username UŽIVATEL secret HESLO*
- Nastavení doménového jména routeru: *Central(config)#ip domain-name upce.cz*
- Vygenerování RSA klíče: *Central(config)#crypto key generate rsa*
- Počet pokusů o přihlášení: *Central(config)#ip ssh authentication-retries 2*
- Verze SSH: *Central(config)#ip ssh version 2*
- Konfigurace VTY linky (pro ID 0 – 15): *Central(config)#line vty 0 15*
- Nastavení přenosu na SSH: *Central(config-line)#transport input ssh*

[5]Authentication, Authorization, and Accounting

5 Tool Command Language shell

Tool Command Language je jednoduchý skriptovací jazyk. Je navržený tak, aby byl srozumitelný při psaní a čtení výsledného kódu. Poskytuje pouze základní prvky programovacích jazyků, a to proměnné, procedury a kontrolní struktury. Tento jazyk je implementován do Cisco IOS, od verze 12.3(2)T a 12.2(25)S.

5.1 Využití

Tohoto skriptovacího jazyka lze využít pro konfiguraci Cisco zařízení. Implementace ve směrovačích a přepínačích umožňuje zadávání příkazů řádek po řádku do konzole IOS. Po zadání TCL příkazu je ten poslán na interpret. Pokud je příkaz validní a projde, zobrazí se výsledek na TTY[6]. Jestliže příkaz není shledán jako TCL, je nadále zaslán do parseru IOS, kde se opět zkusí vykonat. Pokud ani zde nevyhoví, zobrazí se ve finále dvě chybové zprávy. První chybová zpráva je vygenerována TCL parserem, druhá následně Cisco IOS parserem. Jejich informační hodnota pro uživatele může být rozdílná, jelikož obě syntaktické analýzy mají rozdílnou chybovou zprávu.

Skripty lze vytvářet i mimo zařízení a následně je spouštět buď z externího zařízení (tftp, http, ...), nebo je z úložiště nahrát do zařízení a spustit je lokálně.

Jelikož je možné být na router připojen z mnoha míst, musí dokázat TCL běžet ve více vláknech najednou, aby se příkazy neovlivňovaly.

5.2 Ukázka použití na Cisco zařízení

Pro použití TCL shellu musí být uživatel přihlášen v privilegovaném módu, ze kterého se dá vstoupit do TCL režimu. Tento režim má více způsobů, jak načítat a vykonávat spouštění skriptů.

5.2.1 TCL shell režim

V tomto režimu již lze zadávat jednotlivé TCL příkazy. Zadávané příkazy se musí zapisovat vždy celé na jeden řádek. Pro spouštění příkazů z privilegovaného režimu se využívá příkazu *Router(tcl)#exec „PŘÍKAZ“* a pro konfiguraci zařízení *Router(tcl)#ios_config „PŘÍKAZ“ „PODPŘÍKAZ“*.

[6] Teletype — asynchronní linka

5.2.2 Spuštění skriptu z externího zdroje

Tato možnost byla přidána až v pozdějších verzích Cisco IOS. Dobře lze uplatnit, pokud máme úložiště, kde veškeré konfigurační soubory uchováváme a budeme je z něj volat. Nevýhoda je, že takto volané skripty nelze upravovat pomocí parametrů, jelikož se spouští přímo. Nelze využívat procedur.
Pokud soubor *vypistext.tcl* obsahuje *puts „Vypisovaný text“*, příkaz *Router#tclsh tftp://192.168.134.1/vypistext.tcl* vypíše do konzole hlášku *Vypisovaný text*.

5.2.3 Načtení souboru s procedurami

Výhodou tohoto způsobu je, že lze volat procedury, které jsou načteny v souboru. Tento soubor může být umístěn na libovolném z mnoha dostupných úložišť (nvram, flash, sloty, tftp, http, ...). Zpřístupnění souboru se provádí pomocí příkazu *Router(config)#scripting tcl init tftp://192.168.134.1/TclSkript.tcl*.

Samotné spuštění skriptu již probíhá v TCL shell režimu*Router(tcl)#Název-Procedury „parametr1“ „parametr2“ ...*

6 Další využívané možnosti zařízení

V této části uvádím další využívané služby, které Cisco směrovače poskytují. Jedná se o nastavení časových událostí, plánování událostí, nastavení aktuálního data na směrovači a k němu potřebnému nastavení DNS záznamů.

6.1 Časový interval platnosti událostí

Cisco zařízení umožňují aplikovat jisté druhy nastavení na omezenou dobu, buď na interval, nebo na opakovanou událost. Toho se dá využít, pokud požadujeme změnit nastavení v jisté době. Požadujeme například omezení možnosti připojení na službu SSH na všechní dny, kdy se připojují správci sítě, aby mohli měnit konfiguraci zařízení. Můžeme toho pomocí časového nastavení a přístupového seznamu docílit celkem snadno.

6.1.1 Aplikování na Cisco zařízení

Periodické jdou nastavit, aby měnili nastavení opakovaně v jistém intervalu

- Nastavení časového kontejneru *Router(config)#time-range time-range-name*
- Samotný interval *Router(config-time-range)#periodic days-of-the-week hh:mm to [days-of-the-week] hh:mm*

Absolutní se nastavují na dobu, od kdy do kdy mají být aplikovány

- Nastavení časového kontejneru *Router(config)#time-range time-range-name*
- Vlastní rozsah *Router(config-time-range)#absolute [start time date] [end time date]*

Aplikování na ACL pro aktivování časového plánu

- Nastavení časového kontejneru *Router(config-acl)#ip access-list name| number <extended_definition>time-rangename_of_time-range*

V této práci využívám časových událostí pouze jednorázových, bez opakování. Pro každý nastavovaný přístupový seznam udělám časové omezení jeho platnosti, tím aktivuji profil jen na určitou dobu a následně na jeho pozici navrátím původní profil, který na portu byl nastaven dříve.

6.2 Plánování událostí — Kron

Kron, neboli plánování událostí, se využívá, pokud chceme dopředu stanovit akci, která se v daný čas má provést. Můžeme tím aplikovat změnu celé konfigurace, vypnutí rozhraní, aktivaci přístupového seznamu nebo provést různé další změny.

6.2.1 Ukázka využití kronu pro opakovaný ping

- Nastaví kron kontejner: *kron occurrence UdalostPing in 10 recurring*
- Co se má vykonat za kontejner událostí: *policy-list StartPing*
- Seznam událostí (jejich název): *kron policy-list StartPing*
- Samotné příkazy kronu: *cli ping ip 192.168.134.2*

6.3 Network Time Protocol

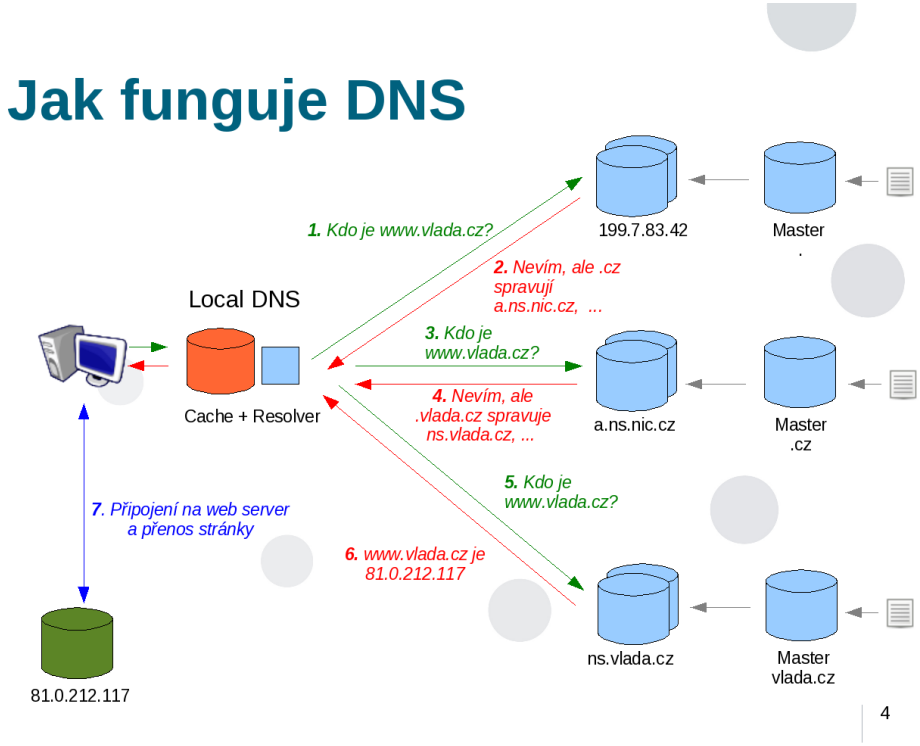
Network Time Protocol je jeden z nejdůležitějších protokolů sítě, který slouží pro synchronizaci času jednotlivých zařízení. Veškeré zálohy, plánované události a další jsou řízeny pomocí času. Správné nastavení času je tedy klíčové, a proto se hodí využít přesného času řízeného z atomových hodin. Pro synchronizaci s nimi a dalšími servery se využívá právě protokolu NTP. Bez synchronizace serveru a routeru by nebylo možné správně nastavit časové rozsahy platnosti přístupových seznamů, proto ho považuji za jeden z klíčových. Vzhledem k důležitosti správně nastaveného času je vhodné uvést více NTP zdrojů, se kterými se bude zařízení synchronizovat.

6.3.1 Zapnutí služby na Cisco zařízení

- Nastavení časové zóny *Router(config)#clock timezone CST +1*
- Nastavení letního času *Router(config)#clock summer-time CDT recurring*
- Přidání serveru *Router(config)#ntp server ntp.nic.cz*
- Přidání serveru *Router(config)#ntp server cz.pool.ntp.org*
- Přidání serveru *Router(config)#ntp server 0.cz.pool.ntp.org*
- Přidání serveru *Router(config)#ntp server 1.cz.pool.ntp.org*
- Přidání serveru *Router(config)#ntp server 2.cz.pool.ntp.org*

6.4 Domain Name System

Služba DNS umožňuje překlad IP adres na doménová jména, která se využívají všude v Internetu. Díky této službě obsažené v zařízení je umožněno získávat data nejen ze serveru podle jejich IP adresy, ale i jejich jmen na síti.



Obrázek 3: Jak funguje DNS[9]

6.4.1 Konfigurace na Cisco zařízeních

- Přidání primárního DNS serveru (OpenDNS) *Router(config)#ip name-server 208.67.222.222*
- Přidání sekundárního DNS serveru (OpenDNS) *Router(config)#ip name-server 208.67.220.220*

7 Praktická část

Aplikace má za cíl umožnit vyučujícím Univerzity Pardubice nastavovat přístupová práva k jednotlivým učebnám a na stanovenou dobu. Samozřejmostí je ověření vyučujícího pomocí jeho unikátních přístupových údajů (předpokládá se provázání aplikace s ověřovacím databázovým systémem CAS[7] a WEBLOGIN).

7.1 Výběr vhodných nástrojů

Zabýval jsem se výběrem metod, jak se na Cisco zařízení připojit a hlavně jaký zvolit postup pro samotnou konfiguraci. Nakonec byla jako nejvhodnější zvolena kombinace telnetu, SDM a skriptovacího jazyka TCL. Tato kombinace umožňuje vzdáleně kompletní ovládání zařízení. Nebyl by proto problém rozšířit aplikaci o další moduly, které by poskytovaly funkcionalitu nejen pro přístupové seznamy, nebo vyšší zabezpečení pomocí SSH.

Vzhledem k předpokládanému začlenění do systému Portal Univerzity Pardubice jsem zvolil jazyk PHP [8], díky kterému je snadné umístění aplikace do již zaběhlého systému. Další výhodou výběru této platformy je, že umožňuje ověřování uživatelů pomocí centralizovaného autentizačního systému CAS. Tento princip umožňuje uchovávat jedinou databázi uživatelů. Vzdáleně se pak z každé aplikace dotazuje do této databáze, zda-li je již uživatel přihlášen. Díky tomu se nemusí do každé aplikace přihlašovat znovu. Systém je praktický z důvodu udržování všech účtů na jednom místě a zároveň usnadňuje uživateli práci s častým přihlašováním do různorodých aplikací, které v rámci univerzitní sítě využívá.

[7] Central Authentication Service
[8] PHP: Hypertext Processor

7.2 Topologie sítě

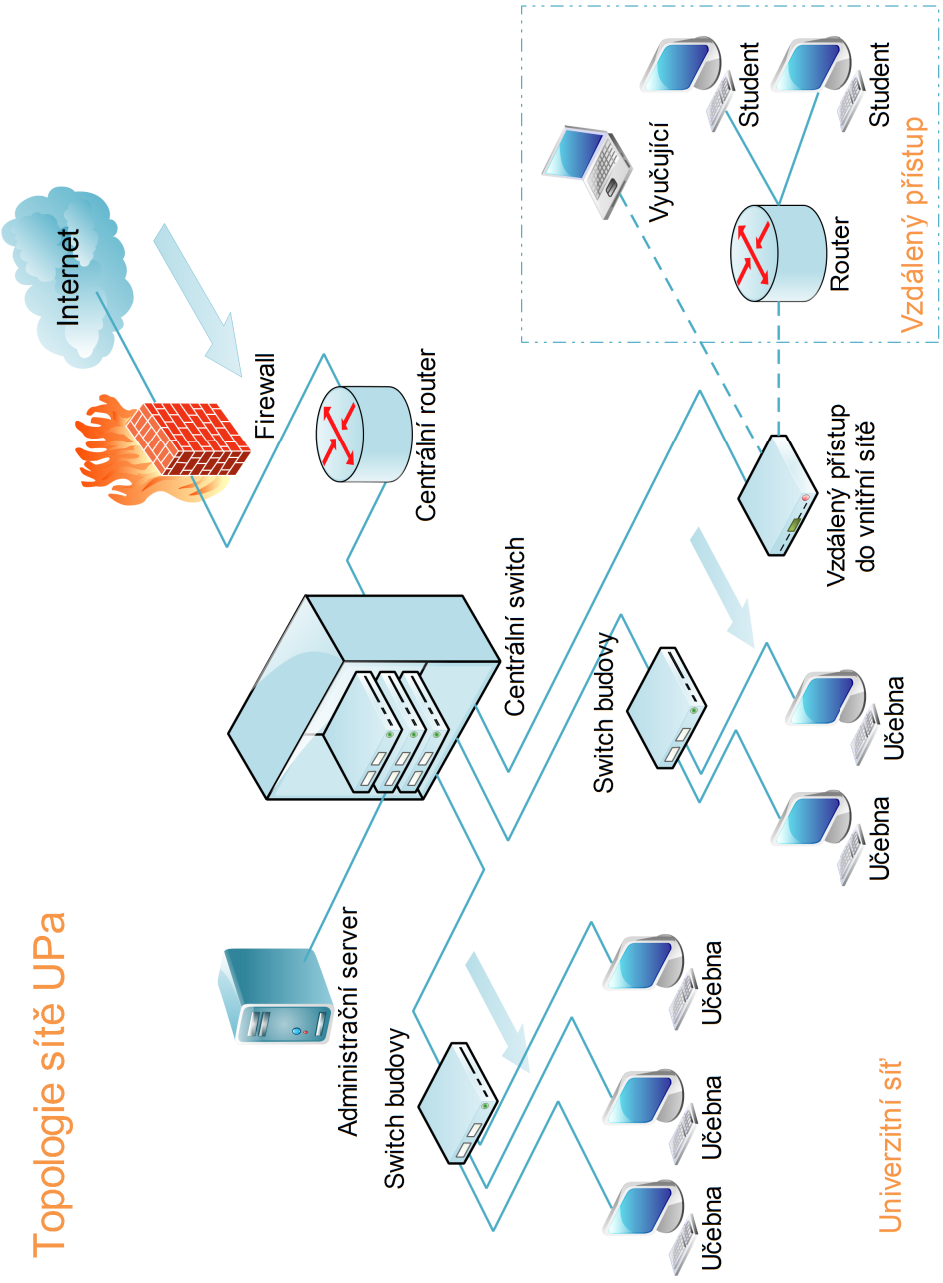
Vzhledem k rozlehlosti celé Univerzitní sítě zde uvádím topologii při využití vhodně navržených VLANů a přístupových seznamů.

Každá učebna má vlastní VLAN, díky čemuž lze snadno identifikovat, kde se nachází. Díky tomu, že má každá učebna vlastní VLAN, je možné je na směrovači dle nich samostatně konfigurovat. Pro každou učebnu může naráz běžet jiný přístupový seznam. Můžeme tím tedy omezit provoz jedné učebny, aniž bychom omezili provoz a výuku v učebnách dalších.

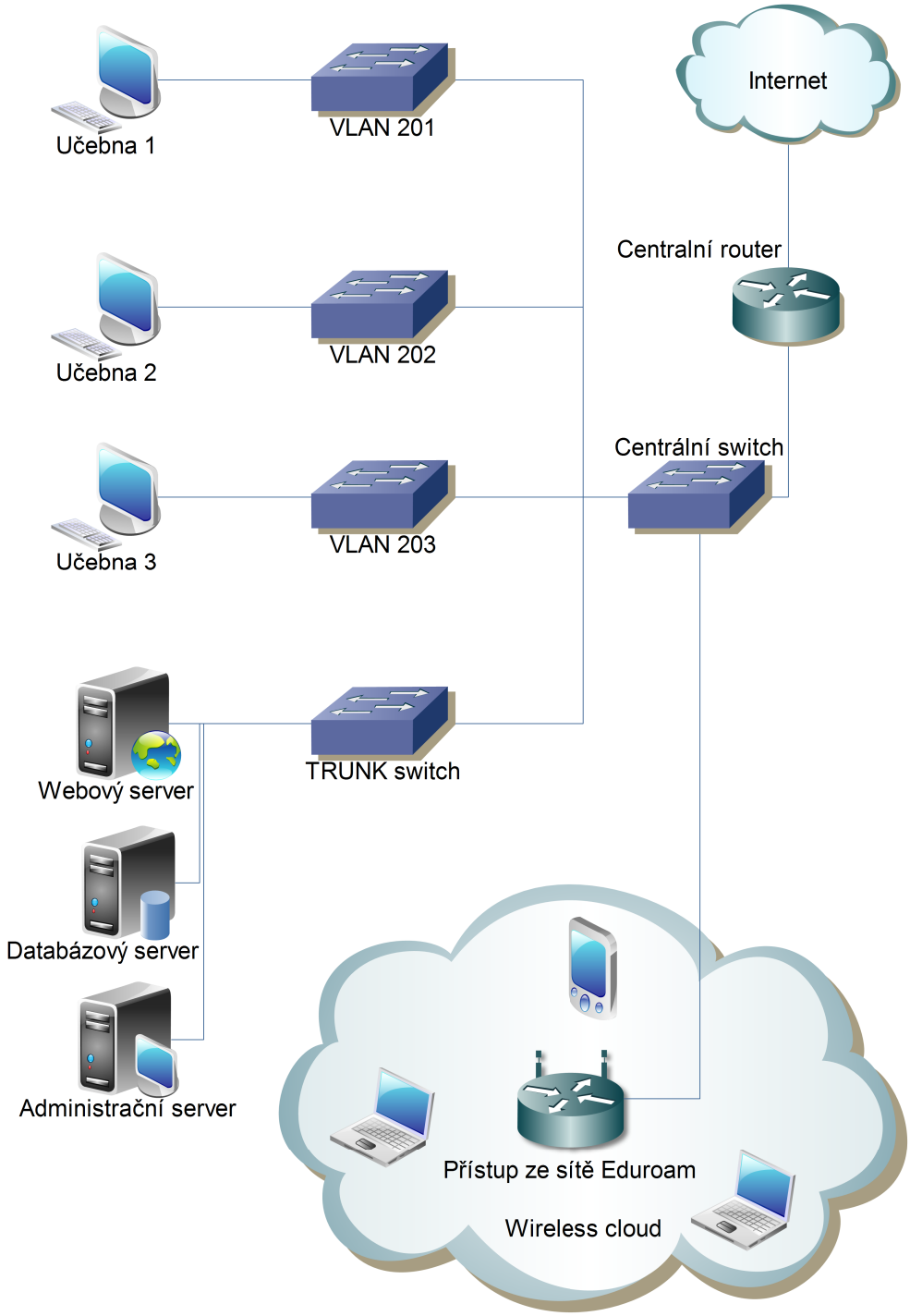
Pro administrativní účely byl zvolen rozsah VLAN0002 – VLAN0099, kde mezi nimi je možné zvolit i samostatnou VLAN pro univerzitní knihovnu, posluchárny, kanceláře a i koleje.

7.2.1 Ukázka zapojení učeben

Pro ukázku je zde obrázek na další straně. Každá učebna je svedena do vlastního switche (VLANy v případě použití většího přepínače). Z něj vychází TRUNK linka spojující VLANy do jedné s tím, že se přenáší i s informací, z jaké subsítě data jsou a v dalším přepínači se zase rozdělují do správných VLANů. Všechny přepínače jsou svedeny do centrálního prvku, který je připojen pomocí vysokorychlostní linky (optická vlákna) k centrálnímu směrovači. Ten umožňuje směrování dat mezi jednotlivými VLAN okruhy a právě na tomto místě se dá filtrovat veškerý datový provoz.



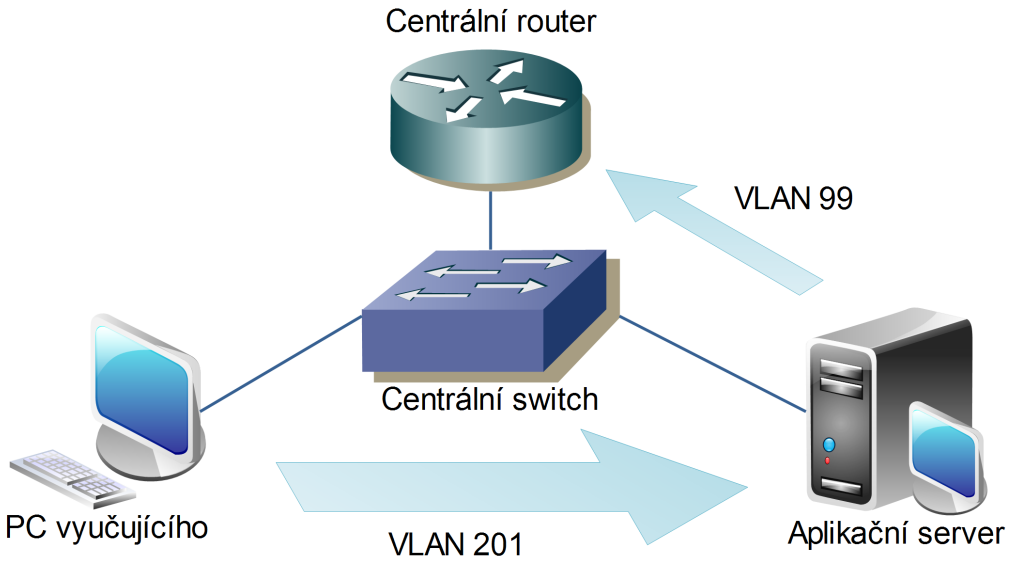
Obrázek 4: Topologie sítě — rozlehlá



Obrázek 5: Topologie sítě — upřesněná

7.3 Připojení aplikace k routeru

Jelikož aplikování přístupových seznamů není možné, aniž bychom byli připojeni na směrovač, začnu praktickou část popisem právě připojení na toto zařízení. Z již popsaných metod jsem zvolil jako první tu nejsnazší s tím, že případně je možné aplikaci dále snadno rozšířit o některý z pokročilejších způsobů připojení. Aplikoval jsem tedy metodu připojení pomocí protokolu telnet. Ač není zabezpečen, díky vlastní VLAN lince mezi směrovačem a aplikačním serverem, řešení je plně dostačující. Data směrující k routeru jsou chráněny právě samostatnou VLAN linkou.



Obrázek 6: Průchod skrz router a rozdílné VLAN

7.3.1 Připojení a odeslání dat pomocí služby telnet

Nejjednodušší připojení PHP poskytuje pomocí metody *fsockopen(adresa, port, error-number, error-string, timeout);*, čímž se vytvoří spojení mezi aplikací a vzdáleným serverem. Dále se pomocí metod *fputs(fsock-spojení, data)* a *fgets(fsock-spojení, počet-bytů)* zasílají a vyzvedávají data ke vzdálenému hostiteli.

```
    private function pripojitTelnet() {
        $this->pripojeni =
            fsockopen($this->host, $this->port, $errno, $errstr, 2);
        if ($this->pripojeni) {
            $this->prihlasit();
            return true;
        } else {
            return false;
        }
    }
    private function prihlasitTelnet() {
        if (isset($this->uzivatel)) {
            $this->posliPrikaz($this->uzivatel);
        }
        $this->posliPrikaz($this->heslo_routeru);
        $this->posliPrikaz('en');
        $this->posliPrikaz($this->heslo_privilege);
        return TRUE;
    }
    private function posliPrikazTelnet($prikaz) {
        time_nanosleep(0, 500000000); // pozdržení, před zasláním
        fputs($this->pripojeni, $prikaz . "\r\n");
    }
```

Z kódu je patrné, že po připojení na zařízení se automaticky provede přihlášení aplikace, která nadále již umožňuje zadávat příkazy potřebné k ovlivňování funkcí směrovače.

Před každým zasílaným příkazem je volána metoda *time_nanosleep(0, 500000000);*, což zapříčiní pozdržení půl vteřiny před zasláním příkazu. Toho je vhodné využít, jelikož ne všechny příkazy směrovač vykonává okamžitě a tak se na něm tvoří fronta nevykonaných příkazů. Tímto způsobem tuto „frontu“ částečně eliminujeme. Na směrovač se v aplikaci nezasílá větší množství příkazů, tedy toto drobné zdržení ničemu výrazně nevadí.

7.3.2 Příjem dat ze služby telnet

Pro ověření správně vykonané sekvence zasílaných dat je zde možnost si z routeru data stáhnout a zobrazit je. Pokud v sekvenci příkazů zaslaných pomocí TCL nenastane chyba, zobrazí se pouze zopakovaný řádek z TCL, aniž by zobrazil cokoliv dalšího. Naopak, pokud k chybě dojde, zobrazuje se chybová hláška, na kterém řádku nastala chyba. Tohoto lze snadno využít pro ověřování funkčnosti aplikace.

```
/**
 * Přijme data ze směrovače pomocí protokolu telnet
 * @return <string> příchozí data
 */
private function prijmiDataTelnet() {
    $time = time() + 5;
    $result = "";

    stream_set_timeout($this->pripojeni, 1);

    $radek = "";

    fgets($this->pripojeni, 128);
    while (!feof($this->pripojeni)) {
        $radek = fgets($this->pripojeni, 128);

        if (preg_replace("/\-\-More\-\-/", "\r\n", $radek)) {
            fputs($this->pripojeni, " ");
        }

        $result .= $radek . "\r\n";
        if (time() > $time) {
            return $result;
        }
    }
    return $result;
}
```

7.3.3 Připojení a odeslání dat pomocí služby SDM

Složitější variantou přenosu dat mezi serverovou aplikací a směrovačem je využití metody SDM. Tato metoda má již alespoň drobné šifrování zasílaného hesla.

```
private function posliPrikazSDM($prikaz) {
    $prikazy = array('! COMMANDSET VERSION="1.0"' . "\r\n",
        '! OPTIONS BEGIN' . "\r\n",
        '! MODE="0"' . "\r\n",
        '! OPTIONS END"' . "\r\n",
        $prikaz . "\r\n",
        '! END' . "\r\n",
        '! COMMANDSET END' . "\r\n");
    $velikost_dat = $this->array_size($prikazy) + 2;

    fputs($this->pripojeni, 'POST /ios_web_exec/commandset HTTP/1.1'
      . "\r\n");
    fputs($this->pripojeni, 'User-Agent: Mozilla/4.0 (Windows XP 5.1)
      Java/1.6.0_17' . "\r\n");
    fputs($this->pripojeni, 'Host: ' . $_SERVER['SERVER_ADDR']
      . "\r\n");
    fputs($this->pripojeni, 'Accept: text/html, image/gif, image/jpeg,
      *; q=.2, */*; q=.2' . "\r\n");
    fputs($this->pripojeni, 'Connection: keep-alive' . "\r\n");

    if (isset($this->uzivatel)) {
        fputs($this->pripojeni, 'Authorization: Basic '
        . base64_encode($this->uzivatel . ':' . $this->heslo_routeru)
        . "\r\n");
    } else {
        fputs($this->pripojeni, 'Authorization: Basic '
          . base64_encode(':' . $this->heslo_routeru) . "\r\n");
    }
    fputs($this->pripojeni,
      'Content-type: application/x-www-form-urlencoded' . "\r\n");
    fputs($this->pripojeni, 'Content-Length: '
      . $velikost_dat . "\r\n");
    fputs($this->pripojeni, "\r\n");

    foreach ($prikazy as $value) {
        fputs($this->pripojeni, $value);
    }
    fputs($this->pripojeni, "\r\n");
}
```

7.3.4 Příjem dat ze služby SDM a ověření zpracování

Samotný příjem dat je shodný s verzí, která přijímá data z telnetu. Komunikuje pouze na jiném portu. Pro kontrolu přijímaných dat je pak využito metody *provedlSePrikazSDM()*, která ověří pomocí regulárního výrazu, zda-li nedošlo při zpracování skriptu na routeru k chybě.

```
/**
 * Zjištění zda-li se vykonal příkaz zaslaný pomocí protokolu SDM
 * @return <boolean> true: vykonal se, false: nevykonal se
 */
private function provedlSePrikazSDM() {
    if (preg_match('/^PARSE_ERROR="[^0]"|while executing/m',
      $this->prichozi_data) == 0) {
        return true;
    }
    return false;
}
```

7.3.5 Připojení a odeslání dat pomocí služby SSH

Vzhledem k volbě PHP a spoustě dostupných hotových řešení v něm, je dostupné i řešení pomocí metody SSH. Tato metoda je již plně šifrována a odbourává tedy případnou potřebu samostatné virtuální linky směrem od aplikačního serveru k směrovači. Nevýhodou je, že tato metoda připojení je umožněna pouze za předpokladu použití operačního systému GNU/Linux na aplikačním serveru, jelikož jinde knihovna poskytující SSH v PHP není dostupná.

Jak nainstalovat a zprovoznit SSH pod Linuxem v PHP je dostupné na webových stránkách kevin.vanzonneveld.net, kde autor popisuje instalaci, připojení i samotné zasílání příkazů na vzdálený systém, ač ne zrovna Cisco směrovač.

7.4 Ověření uživatele aplikací

Pro ověření uživatele je využit jednoduchý skript, který zjistí, zda-li je uživatel v seznamu a zda-li se přihlašuje pomocí jeho hesla. Pokud k ověření nedojde, je na vyšší aplikační úrovni incident zaznamenán s údaji, podle kterých by šel podezřelý uživatel dohledat na síti.

Pro uchování hesel v bezpečí je využito dvojité šifry. Dříve se využíval hojně algoritmus MD5, který má ovšem bezpečnostní rizika. Pro některé řetězce existuje více hashů. Díky tomu se daly považovat dva různé řetězce za shodné, což je z bezpečnostního hlediska nepřípustné. Proto je zde využito novějšího a silnějšího algoritmu SHA1, který je pro jistotu doplněn o část, kde je za jeho konec připojen řetězec z šifrovacího algoritmu MD5, ve kterém je uložena délka vlastního hesla.

7.4.1 Ukázka ověření uživatele

Z kódu je patrné, že momentální verze se dotazuje na uživatele a heslo z konfiguračního souboru pomocí konstant. Tuto část kódu jsem převzal z dřívějšího školního projektu, kde byl napojen na Oracle 10g databázi. Není problém tuto část dále přizpůsobit tak, aby se mohla dotazovat do výše zmiňovaného centralizovaného autentizačního systému CAS, či jiného databázového systému.

```
/**
 * Přihlášení uživatele a skrz konfigurační soubor (bývalo skrz DB, zbytečné
 * pro ukázku - třída je použita z dřívějšího školního projektu, kde byla
 * celá třída napojena na Oracle DB pomocí DiBi layeru, tedy je jednoduché
 * ji opět na DB připojit)
 * @param <string> $username uživatelské jméno přihlašovaného uživatele
 * @param <string> $password heslo přihlašovaného uživatele
 * @return <boolean> true: ověřen, false: neověřen
 */
public function login_user($username, $password) {
    $this->username = $username;
    $this->password = sha1($password) . md5(strlen($password));
    if($this->username == APPL_USERNAME &&
      $this->password == APPL_PASSWORD) {
        $_SESSION['login']=true;
        $_SESSION['username']=$this->username;
        return true;
    } else {
        return false;
    }
}
```

Zde pro ukázku uvádím, jak vypadala původní verze metody s ověřením skrz databázi a DiBi layer.

```
/**
 * Přihlášení uživatele a ověření skrz databázi
 * @param <string> $user
 * @param <string> $password
 * @return <boolean> přihlášen úspěšně?
 */
public function login_user($username, $password) {
    $this->username = $username;
    $this->password = sha1($password) . md5(strlen($password));
    try {
        $result = null;
        $db = new Oci8();
        $result = $db->UserAuth($this->username, $this->password);
        unset($db);
        if(($result != null) && ($this->username == $result[0]['USERNAME']
          && ($this->password == $result[0]['PASSWORD']))) {
            $_SESSION['login']=true;
            $_SESSION['username']=$result[0]['USERNAME'];
            $_SESSION['role'] = $result[0]['ROLE'];
            return true;
        } else {
            return false;
        }
    } catch (DibiException $e) {
        echo get_class($e), ': ', $e->getMessage(), "\n";
        return false;
    }
}
```

7.5 Generování doby platnosti

Jako požadavek na aplikaci bylo zadáno, že musí jednotlivé přístupové seznamy nastavovat na předem stanovenou dobu. Pokud by zde toto časové omezení nebylo, zajisté by se stávalo, že by někdo přístupový seznam po skončení hodiny/písemky nevypnul a omezil by tím případné další vyučovací hodiny.

V aplikační části tedy každý, kdo aplikuje přístupový seznam **musí** nastavit časový interval, na který seznam aplikuje. Čas se nastavuje v minutách, což je dostatečně přesná jednotka pro určení doby pro běžné používání, jelikož mezi hodinami jsou přestávky, během kterých dojde k deaktivaci přístupového seznamu. Pár minut z přestávky, kdy by stále nebyla dostupná část sítě nikomu vadit nebude, nehledě na to, že by nikdo přesnější časový údaj nechtěl nastavovat.

7.5.1 Implementace

Uživatel zadá počet minut, po které chce seznam aplikovat. Nelze plánovat seznam na dobu předem, vždy se aplikuje seznam od aktuálního data na danou dobu. Zobrazený kód ukazuje zpracování časových údajů. Pokud by se mělo dodělat plánování času, jednalo by se o posunutí hodnoty `zacatek` o čas, ve který se má událost zahájit. Problém by ovšem mohl nastat na směrovači, kde by na tuto změnu nebyly připraveny kontrolní mechanismy této aplikace. Ta by se musela částečně opravit.

7.5.2 Ukázka generování časového razítka

Z kódu je patrné, že název této časové nálepky je vázaný na čas, kdy byl vytvářen, a na samotný přístupový seznam, na který je aplikován.

```
private function generujTimeRange() {
    $this->timerange['zacatek'] = date($this->timeformat);
    $this->timerange['konec'] = date($this->timeformat,
      $this->timerange['zacatek'] + strtotime($this->timerange['doba']
      . " minute"));
    $this->finalname = 'TR_'.$this->aclnazev.'_'.date("YmdHi");
    $this->finalnameshort = $this->aclnazev.date("Hi");
    $this->finaltimerange = array(
            'time-range '.$this->finalname,
            'absolute start '.$this->timerange['zacatek'].' end '
              . $this->timerange['konec']
    );
}
```

7.5.3 Ukázka statického vrácení zkráceného názvu časového razítka

Této metody se využívá pro umístění zálohy původního přístupového seznamu, který byl nastaven na rozhraní, na které se bude nové ACL aplikovat. Jelikož Cisco směrovače mají omezenou délku souboru, musel jsem použít zkráceného názvu, kde není celé časové razítko, jak bych si přál.

```
/**
 * Statické metoda, která při zavolání s parametry
 * vrací zkrácený název souboru.
 * @param <number> $doba Doba platnosti ACL
 * @param <string> $aclnazev Název přístupového seznamu
 * @return <string> zkrácený název ACL pro generování souborů ve směrovači
 */
public static function getTimeRangeNameStaticShort(
  $doba, $aclnazev = null) {
    $timerange = new TimeRange();
    $timerange->setTimeRange($doba, $aclnazev);
    return $timerange->getTimeRangeNameShort();
}
```

7.6 Generování ACL příkazů

Pro jednoduchou správu seznamů je každý z nich udržován ve vlastním souboru, kde jeho název je zároveň i název samotného přístupového seznamu. Jejich zpracování spočívá v načtení souboru a doplnění příkazů o časový interval, ve kterém budou platné.

7.6.1 Ukázka přidání časového razítka

Pro zpracování je podmínkou, aby v přístupovém seznamu (na řádku s požadovaným filtrem) byl požadavek na nastavení časového razítka vždy na konci řádku, což je z principu tvorby ACL vlastně i téměř nutností. Řádek tedy nekončí přímo časovou hodnotou, ale pouze slovy `time-range`.

```
    /**
     * Zpracování přístupového seznamu a doplnění o časové razítko
     */
    private function parseACL() {
        foreach ($this->acllist as $row) {
            if (strpos($row, 'time-range')) {
                $row .= ' ' . $this->timerangenazev;
            }
            $this->finalacl[] = $row;
        }
    }
```

7.6.2 Ukázka navrácení již připraveného ACL

Po zpracování jsou všechny přístupové seznamy udržovány v lokální paměti třídy, odkud je možné později seznamy z pole vyzvednout a předat je pro zpracování.

```
/**
 * Vrací obsah definovaného přístupového seznamu.
 * @param <string> $aclnazev Název přístupového seznamu (název souboru)
 * @return <array> Pole obsahující přístupový seznam (ACL), pokud je zadán
 * neznámý název, vrací null.
 */
public function getACL($aclnazev) {
    if (isset($this->acllists[$aclnazev])) {
        return $this->acllists[$aclnazev];
    }
    return null;
}
```

7.6.3 Ukázka navrácení pole seznamů ACL

Po zpracování ACL do pole (načtením souborů) je nutné zajistit předání jejich názvů do aplikační části, kde se pro uživatele vygeneruje seznam dostupných přístupových seznamů.

```
/**
 * Vrací seznam všech přístupových seznamů, které jsou dostupné.
 * @return <array> pole obsahující seznam přístupových seznamů,
 * pokud žádný seznam není, vrací prázdné pole.
 */
public function getACLNames() {
    $nazvy = array();

    if ($this->acllists != null) {
        foreach ($this->acllists as $key => $value) {
            $nazvy[] = $key;
        }
        return $nazvy;
    } else {
        return array(null);
    }
}
```

7.7 Generování TCL příkazů

Pokud již máme vygenerován přístupový seznam, časový interval platnosti seznamu a rozhraní (VLAN), na kterém bude seznam aplikován, je potřebné vytvořit výsledné příkazy, pomocí kterých se směrovač nastaví. Pro jeho nastavení se využívá právě jazyka TCL.

7.7.1 Tvorba výsledného seznamu TCL příkazů

```
/**
 * Vygeneruje výsledné TCL, které uskladní v lokální proměnné
 */
private function generujVysledneTCL() {
    foreach ($this->tcls as $tcl) {
        foreach ($tcl as $prikazy) {
            $this->finalTCL .= 'ios_config';
            foreach ($prikazy as $prikaz) {
                $this->finalTCL .= ' "'.$prikaz.'"';
            }
            $this->finalTCL .= "\r\n";
        }
    }
}
```

V testovacích režimech při vývoji aplikace jsem měl v ACL pouze pár příkazů, které nastavovaly přístupové seznamy. S tím problém nebyl. Bohužel při řešení finálních přístupových seznamů, které díky tomu, že musí být zachovány poskytované služby jako DNS, DHCP, NTP a další, celý přístupový seznam rázem nabobtnává a stává se delším. Zde jsem narazil na další omezení Cisco zařízení. Délka řádku, který obsahuje TCL příkaz, je omezena na nějakou velmi malou hodnotu (odhaduji hodnotu 255 znaků), což je nemyslitelně málo, vzhledem k tomu, že se celý příkaz musí zaslat na jednom řádku. Tímto omezením mi je vlastně znemožněno zadávání celého ACL, dle mnou předpokládaného systému. Do této malé hodnoty se se všemi pravidly přístupového seznamu nemohu vejít.

Řešení tohoto problému by bylo nejspíše jediné. Daly by se zasílat přímo konfigurační příkazy, jako by je psal sám uživatel do Cisco IOS konzole. Problém tohoto řešení je, pokud se někde v průběhu objeví chyba, směrovač zůstane nastaven částečně, což způsobí nekonzistenci dat v konfiguraci zařízení. Může dojít k „zastavení“ běhu v nespecifikovaném stavu, což je nežádoucí.

7.8 Zabezpečení chodu samotného směrovače

Z důvodu, že by mohl aplikační server vypadnout, musí směrovač obsahovat samoobnovovací funkce. Tyto funkce zajistí v případě, že aplikace aktivuje nějaký přístupový profil, zálohu původního profilu do souboru umístěného přímo na směrovači. Dále zajistí, že po uplynutí doby platnosti profilu obnoví původní nastavení daného portu (z dříve vytvořeného souboru).

Jak je vidět, pokud by router neobsahoval tyto samoobnovovací funkce, mohla by nastat situace, kdy by při výpadku aplikačního serveru již nedošlo k navrácení směrovače do původní podoby. Tím by mohlo dojít k zablokování některé z učeben v nespecifikovaném stavu (dle pravidel nastavených v aktivovaném přístupovém seznamu a pravidel definovaných jako standardní na směrovači).

Zde mi začal vyvstávat problém a začal jsem narážet již na samotné hranice použití, do té doby velice vhodného protokolu SDM. Následně jsem byl nucen ho změnit. Všechny již napsané kusy kódu jsou zachovány, jelikož jsou dobře použitelné, ovšem na omezenější programové využití. Protokol SDM totiž neumožňuje zavolání režimu TCLsh na směrovači, čímž mi neumožňuje spuštění kontrolního skriptu. Tento režim je jediný, který umožňuje zapnout lokální proceduru nahranou přímo v zařízení s požadovanými parametry.

Volání funkce přímo ve směrovači je nedílnou součástí zachování konzistence nastavení, jak jsem již popsal výše, směrovač musí dokázat fungovat samostatně, proto není možné umístit tuto TCL proceduru jinam. Jako parametry jsou předávány doba, ve které se má aktivovat obnovení nastavení původních parametrů (původní ACL) portu, VLAN (port na kterém je profil aktivován), a název souboru, do kterého se původní konfigurace portu ukládá.

Zde pro ukázku uvádím zjednodušenou verzi procedury:

```
proc final {interface filename doba} {
  set x "show run interface $interface | include interface|access-group|end"
  lappend res [exec $x]
  set myfileid [open "$filename" w+]
  foreach outs $res {
      puts $myfileid $outs
  }
  close $myfileid
  ios_config "kron policy-list $filename"
    "cli copy $filename running-config" "cli delete $filename"
  ios_config "kron occurrence $filename at $doba oneshot"
    "policy-list $filename"
}
```

Jak je vidět, jedná se o pár jednoduchých příkazů, jejichž zprovoznění přesto zabralo nemalou část doby. Každý řádek může na směrovači zahlásit chybu, aniž by zobrazil řádek, na kterém skript zkolaboval. Celkové ladění TCL je v podstatě věc téměř nemožná, proto napsat i nejjednodušší skript zabere řádově několik hodin. Toto již nepovažuji za triviální TCL skript. Z tohoto ukázkového kódu teprve vychází výsledný s ošetřením některých výjimek, které za běhu mohou vyvstávat.

Z důvodu omezení SDM protokolu jsem se ve vývoji vrátil zpět k protokolu telnet a následně jsem začal s implementací protokolu SSH. Zabezpečený protokol je pro mne bohužel nepoužitelný, jelikož jediný funkční hotový PHP modul je dostupný pouze pod operačním systémem Linux, který neovládám na dostatečné úrovni, abych zprovoznil vše potřebné.

V rámci této práce jsem si nainstaloval čistý systém Ubuntu 10.04 LTS, do něj dle návodu nainstaloval Apache, PHP a ostatní nutné moduly a v rámci několika hodin i zprovoznil SSH v PHP. Celkové ladění a práce pod OS Linux pro mne byla zcela nová zkušenost. Následně jsem byl nucen zprovoznění SSH pro náročnost a případné nedostatky v mé implementaci zavrhnout. SSH spojení by mělo být částečně implementované, ovšem ne tak použitelné, jako již dříve připravené metody spojení SDM a telnet.

Pod OS Windows jsem dlouho marně sháněl a snažil SSH oživit, všechny pokusy však byly ne zcela zdařilé, nebo bez očekávaného výsledku. Aktuální verze této práce tedy funguje ve finále pouze s protokolem telnet a SDM až do části, kdy by se měl spustit mechanismus pro navracení původní konfigurace na rozhraní.

8 Závěr

V rámci zvýšení bezpečnosti by bylo vhodné využít možnosti funkce routeru nastavit pro každý příkaz zvláštní privilege level, kde by se nastavila nižší hodnota funkcím pro tvorbu ACL a konfiguraci VLANů. Vytvořením uživatele s tímto privilege levelem by se pak dosáhlo toho, že by téměř nikdo nemohl zneužít pravomocí uživatele k jiným, než daným účelům. Nebezpečí by mohlo nastat, pokud by útočník narazil na slabinu v PHP kódu, či odchytil pakety jdoucí z aplikace k routeru, nesoucí přihlašovací údaje. V tomto případě by při získání přístupu do routeru útočník převzal veškerou kontrolu nad celým zařízením, což je nežádoucí. Díky tomu by měl přístup do centrálního prvku sítě a mohl tak v podstatě převzít nad sítí kontrolu.

Řešením tohoto problému by bylo umístit aplikační server a vlastní směrovač na takovou část sítě, kam by neměl útočník možnost se dostat. Jedním z řešení by bylo oddělit směrovač pomocí dvou rozhraní, s jedním konfiguračním na samostatné VLANě k routeru a druhým rozhraním, přes které by se připojovali uživatelé na aplikační server. Tím by se zamezilo možnosti, že by se kdokoliv mohl dostat k datům směřujícím na router. Toto řešení je naznačeno i v průběhu práce.

V práci jsem nakonec neimplementoval vše, co jsem měl v plánu. Bohužel její náročnost, jak časová, tak na znalost programovacích a síťových technik je značná a moje dovednosti, i přes veškerou snahu, omezené.

V aktuální fázi je možné práci využít pro aktivaci přístupových seznamů a za pomoci protokolu telnet (omezeně i SDM protokolu) vzdáleně nakonfigurovat směrovač. Ten si následně sám ohlídá všechny náležitosti pro případ, že by došlo k výpadku aplikačního serveru, nebo části sítě (směrovač zůstává v konzistentním stavu pro funkčnost sítě).

Zpracováním této práce jsem si odnesl nesčetné množství poznatků o možnostech pokročilých technik konfigurace směrovačů, implementace různých kódů v PHP o kterých jsem do této práce neměl ponětí, že PHP vůbec umožňuje. Celkově jsem si rozšířil znalost programovacích technik (OOP programování v PHP), naučil se podrobněji jak fungují některé protokoly (telnet, SDM, SSH), které jsem potřeboval nastudovat pro implementaci a také získal nějaké nové zkušenosti z prostředí systému Linux.

Reference

[1] BOUŠKA, P.: Cisco IOS 5 - komunikace se switchem — SAMURAJ-cz.com. 16. 05. 2007, [Online; cit. 2010-04-12].
URL <http://www.samuraj-cz.com/clanek/cisco-ios-5-komunikace-se-switchem/>

[2] Cisco Systems, I.: Cisco IOS Scripting with Tcl — Cisco Systems. July 2003, [Online; cit. 2010-04-10].
URL <http://www.cisco.com/en/US/products/sw/secursw/ps1018/products_tech_note09186a00800a5b9a.shtml>

[3] Cisco Systems, I.: Cisco IOS Configuration Fundamentals Configuration Guide, Release 12.4. c1992-2010, [Online; cit. 2010-02-10].
URL <http://www.cisco.com/en/US/docs/ios/fundamentals/configuration/guide/12_4/cf_12_4s_book.html>

[4] Cisco Systems, I.: Cisco IOS Scripting with Tcl — Cisco Systems. c1992-2010, [Online; cit. 2010-04-12].
URL <http://www.cisco.com/en/US/tech/tk583/tk617/technologies_tech_note09186a00800949e2.shtml>

[5] Cisco Systems, I.: Command Scheduler [Cisco IOS Software Releases 12.3 Mainline] — Cisco Systems. c1992-2010, [Online; cit. 2010-05-01].
URL <http://www.cisco.com/en/US/docs/ios/12_3/feature/guide/g_kron.html>

[6] Cisco Systems, I.: Configuring DNS on Cisco Routers [IP Application Services] — Cisco Systems. c1992-2010, [Online; cit. 2010-05-01].
URL <http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a00800c525f.shtml>

[7] Cisco Systems, I.: Configuring IP Access Lists — Cisco Systems. c1992-2010, [Online; cit. 2010-04-09].
URL <http://www.cisco.com/en/US/products/sw/secursw/ps1018/products_tech_note09186a00800a5b9a.shtml>

[8] Hubicka, J.: TCL — Skolicka. c1995, [Online; cit. 2010-04-10].
URL <http://www.ucw.cz/~hubicka/skolicky/skolicka11.html>

[9] SURÝ, O.: DNSSEC část první aneb je potřeba začít od píky. 8. 12. 2008, [Online; cit. 2010-05-06].
URL <http://www.root.cz/clanky/dnssec-cast-prvni-aneb-je-potreba-zacit-od-
>

[10] Wikipedia: Cisco IOS — Wikipedia, The Free Encyclopedia. 2010, [Online; accessed 8-April-2010].
URL <http://en.wikipedia.org/w/index.php?title=Cisco_IOS&oldid=348969365>

[11] Wikipedia: Simple Network Management Protocol — Wikipedia, The Free Encyclopedia. 2010, [Online; accessed 8-April-2010].
URL <http://en.wikipedia.org/w/index.php?title=Simple_Network_Management_Protocol&oldid=354058750>

[12] Wikipedie: Cisco IOS — Wikipedie: Otevřená encyklopedie. 2010, [Online; navštíveno 7. 04. 2010].
URL <http://cs.wikipedia.org/w/index.php?title=Cisco_IOS&oldid=4960137>

Přílohy

Přílohou k této práci je DVD médium, které obsahuje praktickou část této práce, dokumentaci a tuto textovou část v elektronické podobě.