VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zajištění bezpečnosti a implementace nových prvků řídícího systému virtuální laboratoře

Diplomová práce

2006 Roman Kubín

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 12. května 2006 .....

Abstrakt

Smyslem této práce je rozšířit stávající systém virtuální laboratoře počítačových sítí o nové možnosti a zajistit bezpečnost provozu stávajícího řešení při přístupu z Internetu. Dále se práce zaměřuje na vylepšení vzhledu aplikace v nejčastěji používaných prohlížečích. Analyzuje a poté upravuje vybrané části systému, čímž se snaží o zefektivnění jak komunikace s uživateli, tak i komunikace mezi jednotlivými částmi systému.

Klíčová slova: virtuální síťová laboratoř, Virtlab, bezpečnost, uživatelské prostředí

Abstract

The aim of this work is to extend the system of virtual network laboratory with new possibilities and ensure security of current function of Internet access. This work is also focused at visual improvements in commonly used browsers. It analyses and modifies selected parts of system, that results in more effective communication with users and also other system parts.

Keywords: virtual network laboratory, Virtlab, security, user interface

Seznam použitých zkratek a symbolů

ASP	–	Active Server Pages
CDP	–	Cisco Discovery Protocol
HTML	–	HyperText Markup Language
HTTP	–	Hyper Text Transfer Protocol
HTTPS	–	Secure Hyper Text Transfer Protocol
JSP	–	Java Server Pages
JSSE	–	Java Secure Socket Extension
LDAP	–	Lightweight Directory Access Protocol
PHP	–	PHP Hypertext Preprocessor / Professional Home Pages
SQL	–	Structured Query Language
SSL	–	Secure Sockets Layer
STP	–	Spanning Tree Protocol
TCP	–	Transmission Control Protocol
URL	–	Uniform Resource Locator
WWW	–	World Wide Web

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Virtuální síťová laboratoř (Virtlab) je projekt, který vznikl na základě požadavků o zpřístupnění zařízení, která se nacházejí ve školní síťové laboratoři pro více studentů a umožnil jim pracovat s těmito zařízeními i mimo vyučovací hodiny, nejen ze školní sítě, ale i z domova a v podstatě odkudkoliv ze světa. Návrh a hlavní jádro systému po softwarové stránce vytvořil Ing. Pavel Němec jako součást své diplomové práce Virtuální síťová laboratoř, kterou obhajoval v roce 2005. O hardwarovou stránku se postaral Ing. David Seidl v rámci své diplomové práce Automatizovaný systém správy síťových konfigurací, která byla obhájena v roce 2005. Celý systém virtuální síťové laboratoře umožňuje nejenom ovládání a konfiguraci jednotlivých síťových zařízení z pohodlí domova, ale umožňuje i jejich automatické vzájemné propojování a tudíž i změny topologie celé konfigurované sítě.

1.1 Popis systému

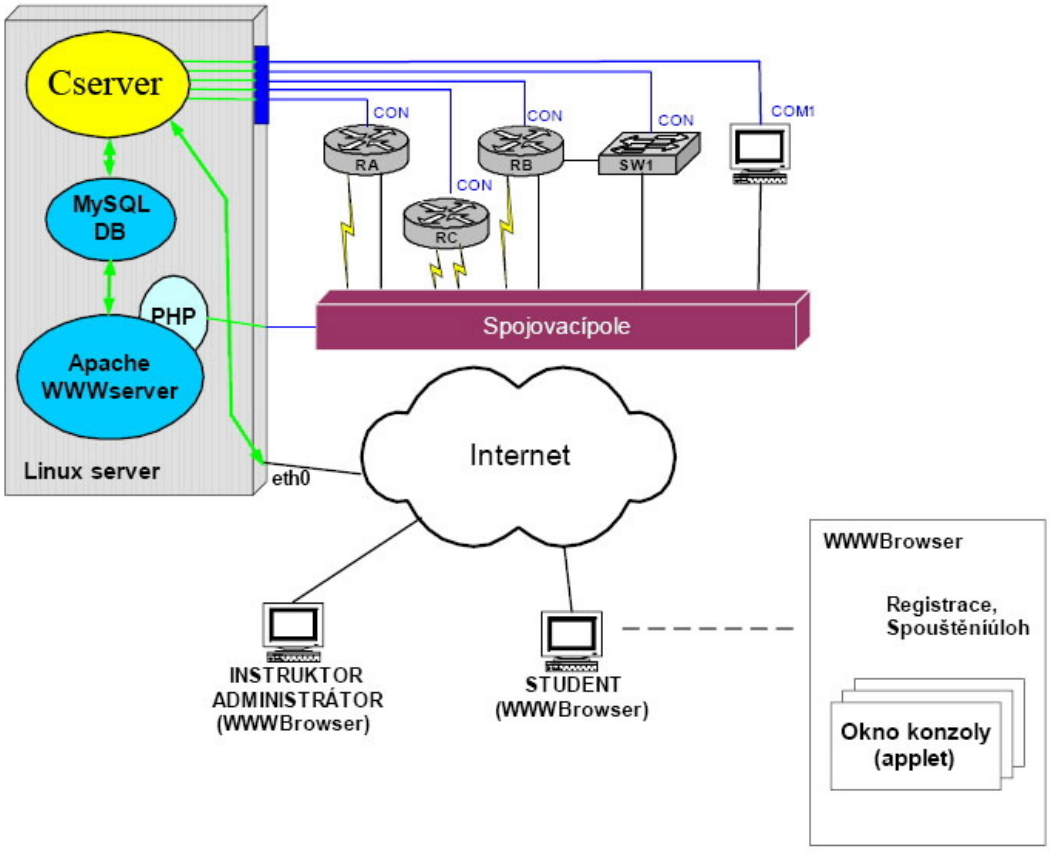
Virtlab se skládá z několika základních komponent, které mezi sebou spolupracují a zajišťují tím funkci celého systému jak je patrné z obrázku 1. Mimo těchto komponent je potřeba do Virtlabu zahrnout i zařízení, které pomocí těchto základních komponent budeme ovládat. Nejčastěji jimi jsou přepínače a routery, případně HUBy. V budoucnu se počítá i s použitím pracovních stanic. Základními komponenty Virtlabu jsou:

- HW zařízení sloužící pro propojení topologie sítě
- Webové rozhraní pro ovládání a konfiguraci Virtlabu
- Databáze sloužící k uložení všech dat Virtlabu
- Cserver, obstarávající přenos dat od uživatele k vybranému zařízení a naopak
- Java applet, fungující jako vzdálený terminál
- Pomocné programy doplňující Virtlab

V této kapitole si je všechny zevrubně popíšeme, aby i čtenář dosud neseznámený s Virtlabem získal přehled o funkci celého systému.

1.1.1 Propojení zařízení

K propojování jednotlivých zařízení slouží automatické spojovací pole, do kterého jsou připojena jednotlivá rozhraní (at'již se jedná o ethernetová nebo sériová rozhraní) všech zařízení, se kterými lze v systému pracovat. Toto spojovací pole je připojeno pomocí sériového portu k počítači, na kterém běží celý systém Virtlabu. Pomocí tohoto portu jsou mu posílány příkazy, jejichž vykonáním dochází ke změně topologie sítě. Lze propojovat různá sériová rozhraní jednoho zařízení (typicky routeru) s rozhraními jiného routeru. Totéž lze provádět s ethernetovými rozhraními.



Obrázek 1: Komponenty Virtlabu

Počítač, na kterém celý systém Virtlabu běží je založen na operačním systému unixového typu. Na němž je nainstalován Apache server pro ovládání systému pomocí webového rozhraní, MySQL databáze starající se o uložení potřebných dat a dále na něm běží speciální program (Cserver) napsaný v programovacím jazyce C, jehož funkce je popsána v kapitole 1.1.4. V počítači je také nainstalována speciální multiportová karta, umožňující připojit konzoly jednotlivých zařízení a ovládat je tak všechna z tohoto počítače.

1.1.2 Webové rozhraní a databáze

Webové rozhraní je vytvořeno pomocí dynamických webových stránek založených na technologii PHP a spolupracuje s dabatázi typu MySQL. Je rozděleno na několik důležitých částí, z nichž ne všechny jsou přístupné všem uživatelům.

Hlavní část zobrazuje důležité informace pro uživatele, mezi něž patří popis zacházení s jednotlivými zařízeními, příkazy pro jejich základní konfiguraci, informace o úlohách, které má uživatel rezervovány, případně které může právě spustit. Je zde taky zobrazena kvóta určená pro rezervaci jednotlivých úloh, kterou může uživatel ještě vyčerpat.

V části pro zařízení lze nalézt seznam zařízení, informace o jejich typu a jednotlivých rozhraní, kterými disponují. Administrátor v této sekci také nalezne seznam rozhraní, které mohou jednotlivá zařízení obsahovat. Tento seznam může samozřejmě upravovat. Taktéž může vytvářet či mazat jednotlivá zařízení, případně upravovat jejich parametry.

Část pro práci s úlohami umožňuje uživateli vyhledávat úlohy zavedené v systému pomocí skupin, do kterých se řadí. Po vyhledání úlohy je možno zjistit, o co se v úloze jedná, jaká zařízení se k úloze využijí, propojení jejich rozhraní, ale také ukázku správné konfigurace těchto zařízení. Uživatel si může úlohy rezervovat na speciální nástěnce k tomu určené. Na této nástěnce má přehled o dostupných úlohách k rezervaci, ale i úlohách, které již jsou rezervovány. Svou rezervaci zde může samozřejmě i zrušit. Nachází se zde i položka pro spuštění úlohy, která právě běží a uživatel ji má rezervovanou. V takovém případě uživatel uvidí potřebné informace o úloze a seznam zařízení, k nimž se po kliknutí na tlačítko může připojit a začít s nimi pracovat. Tato práce se děje pomocí Java appletu, jehož funkce je popsána v kapitole 1.1.5. Administrátor má v této části oproti běžnému uživateli mnoho možností navíc. Zejména může editovat již vytvořené úlohy (ty ale nesmí být umístěny na nástěnce v nejbližší době) a vytvářet nové. Nové úlohy lze vytvářet buď zapsáním do webového formuláře, nebo pomocí načtení souboru v XML formátu, který danou úlohu popisuje. Administrátor může taktéž měnit, přidávat a mazat kategorie úloh. Poslední možností administrátora v této části je přidávání úloh na nástěnku v libovolném čase po určených časových úsecích. Uživatel si pak může tyto úlohy rezervovat pro práci.

Uživatelům je taktéž k nahlédnutí obsáhlá nápověda, jak pracovat s celým systémem.

Poslední část je určená jen pro administrátory a slouží ke správě uživatelů, kteří mají přístup do Virtlabu. Lze jednotlivé uživatele přidávat, mazat, měnit jejich údaje, nastavovat jim kvótu pro rezervaci úloh a podobně.

1.1.3 Struktura databáze

Celá databáze se skládá z několika tabulek, zajišťujících vhodné uložení datových struktur. Postupně si je všechny v krátkosti popíšeme.
Category_of_task - Obsahuje informace o kategoriích úloh.
Class_category_list - Obsahuje seznam kategorií úloh úrovně 1.
Equipment - Obsahuje seznam zařízení, vyskytujících se ve Virtlabu.
Interface_equipment - Obsahuje seznam rozhraní jednotlivých zařízení.
Interface_list - Obsahuje seznam existujících rozhraní.
Link - Obsahuje propojení rozhraní jednotlivých zařízení.
Notice_board - Obsahuje seznam úloh, umístěných na nástěnce.
Rezervation - Obsahuje informace o provedených rezervacích.
Sessions - Obsahuje informace o uživatelích, jenž právě v systému pracují.
Task - Obsahuje úlohy vložené do systému.
Task_category - Obsahuje seznam kategorií úloh.
Task_equipment - Obsahuje seznam kategorií úloh úrovně 2.
Task_link - Obsahuje seznam propojení zařízení pro jednotlivé úlohy.
User - Obsahuje informace o uživatelích Virtlabu.

1.1.4 Cserver

Cserver je základním stavebním kamenem celého Virtlabu. Hlavní úlohou Cserveru je přijímat požadavky od klientů (uživatelů, kteří pracují se systémem) o spojení s konzolí určitého zařízení. Cserver musí v databázi ověřit, zda má klient právo k určenému zařízení v danou dobu přistupovat. V případě oprávněného požadavku zajistí Cserver komunikaci klienta s určeným zařízením tím, že předává znaky přicházející od klienta na určené zařízení (na rozhraní počítače, ke kterému je připojena jeho konzole) a naopak, znaky přicházející od zařízení předává klientovi. Dále sleduje aktivitu uživatelů a pokud zjistí, že uživatel nejeví na nějakém zařízení aktivitu, po určité době ho od tohoto zařízení odpojí. Taktéž v případě, že vyprší čas na úlohu, jsou o této skutečnosti uživatelé informování a poté odpojení od zařízení.

1.1.5 Java applet

Java applet simuluje konzoli zvoleného zařízení. Uživateli se práce jeví, jako by pracoval se zařízením na počítači, jehož sériový port je přímo připojený na konzoli tohoto zařízení. Applet dostává z WWW stránky parametry, pomocí nichž zjistí, který uživatel se chce připojit ke kterému zařízení. Kontaktuje Cserver s žádostí o připojení ke konkrétnímu zařízení. Nijak neověřuje, zda má uživatel právo na přístup k tomuto zařízení. Tuto kontrolu provádí jen Cserver a applet informuje o výsledku pomocí definovaného komunikačního protokolu 8.1. Applet nekomunikuje s databází a veškerá data potřebná pro svůj běh získá buď z WWW stránky nebo od Cserveru. V případě, že uživateli není z nějakého důvodu umožněn přístup k zařízení, informuje o tomto uživatele vhodným hlášením. Uživatel si může nastavit font i jeho velikost, který se bude v appletu zobrazovat.

1.1.6 Pomocné programy

Pro správnou funkčnost Virtlabu je zapotřebí dvou pomocných programů (napsaných v jazyce C), které se starají o podpůrné funkce, bez nichž by Virtlab nefungoval správně.

Prvním z nich je *Renew Quota*, který se stará o obnovování uživatelských kvót. Tyto kvóty slouží uživatelům k rezervaci úloh. Pokud uživatel svou kvótu vyčerpá, nemůže si již další úlohy rezervovat. Tento systém je zaveden z důvodu, aby si nemohl jeden uživatel rezervovat příliš úloh najednou a zbyl prostor i pro ostatní uživatele. *Renew Quota* se obvykle volá pomocí crontabu (například každé pondělí minutu po půlnoci) a tímto zajišťuje uživatelům každotýdenní obnovení jejich kvóty na maximální hodnotu.

Druhý program se jmenuje *Check Timeslot* a slouží k mazání zařízení před začátky úloh, jejich úvodnímu nastavení a taktéž k poslání konfiguračních příkazů do automatizovaného spojovacího pole, jenž propojí rozhraní zařízení se kterými se v úloze pracuje. Program se opět pouští automaticky pomocí crontabu a to vždy pět minut před začátkem nového časového úseku (timeslotu). Tento je standardně nastaven na délku 45-ti minut, stejně jako trvá běžná vyučovací hodina. Program zjišťuje, zda je v za chvíli začínajícím timeslotu připravena úloha pro spuštění. Pokud není, nic neprovádí. Ale pokud zjistí, že bude probíhat nějaká úloha, provede test, zda v právě končícím timeslotu neprobíhá stejná úloha, rezervována stejným uživatelem. Pokud ano, úloha pokračuje dále a nesmí se uživatelům přemazat zařízení, na kterých právě pracují. V opačném případě jsou vymazána všechna zařízení, která budou v úloze využívána. V případě, že je potřeba automatického nakonfigurování topologie sítě, jsou poslány příslušné konfigurační příkazy do automatizovaného spojovacího pole.

1.2 Role uživatelů

Uživatelé v systému mohou figurovat v nejrůznějších funkcích, od obyčejného studenta až po systémového správce, který má veškerá práva k systému. Proto je každému uživateli přiřazena role, kterou zastává v systému. Jeden uživatel může mít i více rolí. Role uživatelů jsou rozděleny do dvou základních kategorií a to na administrátory a na studenty.

Role studenta je nejdůležitější, protože právě studentům je celý systém určen a bez nich by neměl smysl. Studenti si mohou prohlížet úlohy zavedené v systému, mohou si úlohy, jenž jsou dány na nástěnku rezervovat a následně v příslušném čase je spustit. Další možností studenta je prohlížení zařízení, která jsou v systému dostupná. Poslední možností je změna emailu, případně hesla pro přihlášení.

Role administrátora se dělí na několik samostatných částí, přičemž každý uživatel může mít nastavena práva k jakékoliv z nich, nezávisle na sobě:

- Systémový správce
  má mimo všechna zde uvedená práva na starosti i funkčnost celého systému.
- Správa úloh
  umožňuje vytvářet, editovat a mazat úlohy. Dále umožňuje spravovat kategorie úloh.

- Správa zařízení
  umožňuje vytvářet, editovat a odstraňovat zařízení. Dále umožňuje spravovat typy rozhraní, která mohou zařízení obsahovat.
- Správa uživatelů
  umožňuje spravovat uživatele, měnit jejich údaje, kvótu, heslo a přiřazovat jim role.
- Nástěnkář
  umožnuje uživateli přidávat a odebírat úlohy z nástěnky.
- Zapojovač úloh
  má na starosti zapojování úloh, v případě, že nejsou zapojeny automaticky.

1.3 Cíle práce

Cílem práce je rozšířit stávající systém virtuální laboratoře počítačových sítí o nové možnosti a zajistit bezpečnost provozu stávajícího řešení při přístupu z Internetu. Dále je zapotřebí upravit některé části původního kódu s ohledem na zlepšení funkčnosti, či stability celého systému.

Hlavní část práce je zaměřena na analýzu bezpečnosti Virtlabu, navržení vhodného řešení a především na jeho implementaci. Součástí bezpečnosti je vylepšení a doplnění logovacích zpráv, které je potřeba zaznamenávat s ohledem na odhalování případných nekalých aktivit uživatelů a možnosti odhalit, lépe identifikovat a odstranit vyskytující se chyby v systému.

Celý systém je nyní navržen pro samostatnou práci uživatelů, ale ukázalo se, že by bylo vhodné, mít možnost zasahovat a řídit práci uživatelů. Tuto možnost budou mít pouze k tomu určení uživatelé, tzv. tutoři. Ti si budou moci zvolit, zda uživatelé uvidí jejich práci na zařízení, či nikoliv.

Dále je nutno vhodnou formou navrhnout komunikaci systému s uživateli v případě výskytu významných stavů. Je zapotřebí definovat tyto významné stavy a učinit potřebné kroky pro implementaci.

Při používání Virtlabu se ukázalo, že jeho WWW část není dobře zobrazována v některých prohlížečích a tak znesnadňuje, či dokonce znemožňuje použití určitých částí pro skupiny uživatelů, používající tyto webové prohlížeče. Proto je zapotřebí provést nápravu a zajistit dobrý vzhled a hlavně funkčnost v běžně dostupných prohlížečích. U některých akcí není GUI navrženo příliš šťastně, a proto se zaměřím i na upravení, či úplné přepracování GUI u těchto akcí.

Poslední část této diplomové práce je zaměřena na prozkoumání možností automatizovaného spojování ethernetových rozhraní laboratorních zařízení s využitím přepínače, podporujícího technologii *VLAN Tunneling* (dot1QinQ). Tato technologie nám umožní vytvořit propojení mezi dvěmi zařízeními přes přepínač tak, že se uživateli Virtlabu bude jevit ve všech směrech, jako by byly ony zařízení propojeny napřímo. Docílíme toho pomocí nastavování jednotlivých rozhraní do odlišných VLAN sítí a jejich tunelování na správné rozhraní dalšího zařízení.

2 Bezpečnost komponent virtuální síťové laboratoře

Bezpečnost Virtlabu je velice komplexní otázka a proto si v této kapitole popíšeme bezpečnost jednotlivých komponent, ze kterých se Virtlab skládá a v další kapitole si povíme o zabezpečení komunikace mezi nimi.

Velmi významným bezpečnostním rizikem, které je třeba mít na paměti, je samostatný PHP kód. Ten, pokud při jeho vytváření nejsou dodržovány doporučené bezpečnostní postupy, může obsahovat mnoho potenciálních děr v systému, které mohou útočníkovi dovolit manipulovat s daty na serveru, v nejhorším případě kompletně smazat jeho obsah. S ohledem na toto riziko je zapotřebí analyzovat možné způsoby útoku a v případě potřeby provést potřebné kroky k nápravě.

Bezpečnostním rizikem nemusí být jen útočník, který se snaží vědomě poškodit náš systém, ale mohou jím být i obyčejní uživatelé s mírumilovnými úmysly. Každé zařízení, respektive jeho operační systém (IOS), obsahuje jisté množství více, či méně nebezpečných příkazů, pomocí nichž mohou uživatelé, byť nevědomky toto zařízení zablokovat a znemožnit na něm práci jak sobě, tak ostatním uživatelům. Proto je zapotřebí vymyslet a implementovat nový systém, zabraňující zadávání těchto potenciálně nebezpečných příkazů. Některé příkazy, jako například nastavování hesla pro přístup k zařízení, chceme nechat povoleny, avšak jen s námi definovanou hodnotou tohoto hesla. Toto je potřeba zohlednit při navrhování systému na ochranu proti potencionálně nebezpečným příkazům.

2.1 Analýza a návrh řešení webové části Virtlabu

Mezi nejviditelnější a nejčastěji napadané součástí každé internetové aplikace patří její webové rozhraní. Zvláště pak, jestliže je postaveno na dynamických stránkách, ať již ve formě JSP, ASP, či PHP. Je tedy logické, že se v prvé řadě budeme věnovat zabezpečení naší aplikace proti průniku skrz PHP[7][8]. Vzhledem k tomu, že mohutně používáme i MySQL, musíme dávat obzvlášť pozor i na zabezpečení databáze. Veškerý text se bude zabývat PHP verzí 4, konkrétně PHP 4.1.0 a vyšší.

2.1.1 Předávání parametrů

Základní bezpečnostní problém v PHP, spočívá v předávání parametrů ať již ve formě vyplněných formulářů od uživatelů nebo parametrů, které potřebujeme přenášet mezi jednotlivými stránkami, pro jejich další použití. Proměnné se do PHP skriptu mohou dostávat několika cestami:

- jsou definovány programátorem
- jsou předány pomocí URL
- jsou předány pomocí webového formuláře (metodou POST)
- jsou předány pomocí cookies

- jsou předány pomocí session

Problém nastává v případě, že dostaneme rozdílný obsah stejné proměnné z několika míst současně. Toto je v PHP řešeno pomocí tzv. globálních proměnných. Pokud chceme načíst obsah proměnné $jmeno, která přišla pomocí URL, dostaneme její obsah jako $_GET["jmeno"], pokud by přišla metodou POST, použijeme $_POST["jmeno"]. Analogicky pak $_SESSION["jmeno"] pro proměnnou ze session a $_COOKIE["jmeno"] pro hodnotu proměnné načítanou z cookies. Další globální proměnou je $_REQUEST, která obsahuje hodnoty proměnných z polí $_POST, $_GET a $_COOKIE. V případě, že je hodnota stejné proměnné obsažena ve více polích, upřednostňuje se hodnota v poli $_POST a poté $_GET. Tuto proměnnou nedoporučuji používat, protože její nesprávné použití může vést k přepsání hodnot proměnných útočníkem. Proto je všude v kódu používán očekávaný typ globální proměnné.

Základním prvkem pro bezpečnou práci s proměnnými v PHP je vypnutí *register_globals* (nastavení na hodnotu 0) v konfiguračním souboru PHP (php.ini). Od verze PHP 4.1 je toto nastavení defaultně vypnuto. Toto nastavení způsobuje, že veškeré globální proměnné jsou převedené do "normální" lokální proměnné a pro útočníka tak není problém měnit hodnotu proměnných v našem kódu podle svých představ (za předpokladu, že nebudeme všechny pečlivě inicializovat). Ukázku nalezneme na výpisu č. 1. V případě, že útočník přidá za URL hodnotu proměnné $valid_input nastavenou na jedničku (*URL?valid_input=1*), bude program vždy předpokládat, že vstup byl správný a provede funkci *show_financial_data()*. Právě tomuto se zabrání vypnutím výše zmíněné funkce *register_globals*.

```
if (input_is_okay()) {
    $valid_input = 1;
}
if ($valid_input) {
    show_financial_data();
}
else {
    show_order_form();
}
```

Výpis 1: Přepsání lokální proměnné

Dále je potřeba znemožnit případnému útočníkovi manipulovat s hodnotami proměnných, které budeme přijímat od uživatelů z formulářů, případně si je budeme předávat mezi jednotlivými PHP skripty. Máme v podstatě čtyři možnosti, jak tyto proměnné přijímat a předávat.

První způsob je v používaní cookies a předávání všech parametrů tímto způsobem. Tento způsob ale pro nás není žádoucí, protože v případě, že uživatelův prohlížeč nebude cookies podporovat, nebo bude mít tuto funkci vypnutou, stane se aplikace pro daného uživatele naprosto nepoužitelná.

Druhým způsobem je použití tzv. Sessions. Session je téměř ideální způsob, jak uchovávat informace patřící k přihlášenému uživateli. Velkou výhodou je, že všechny citlivé informace jsou ukládány na serveru, tedy bezpečným způsobem. Do prohlížeče se předává pouze identifikátor session. Webový server, přesněji řečeno PHP skript, si označí

každého uživatele jedinečným identifikátorem (v případě PHP 32 znaků dlouhým řetězcem). Identifikátor se pak předává společně s každým voláním skriptu uživatele. K tomu PHP využívá dvě metody. První je předávání pomocí cookies. Ačkoliv téměř každý prohlížeč podporuje cookies, není to vždy zaručeno (uživatel si může cookies v prohlížeči vypnout). S tím je potřeba počítat a v takovém případě je třeba použít druhou metodu. Tou je předávání identifikátoru metodou GET, jako parametr v URL (viz. dále). S přihlédnutím k původnímu návrhu systému, kdy se o použití sessions neuvažovalo a nutnosti v takovém případě celý systém od základu přepracovat (zvláště vzhledem k použití generovaného jednoznačného identifikátoru uživatele a jeho provázání s celým systémem včetně databáze, Cserveru i appletu), jsem se rozhodl tuhle možnost taktéž zavrhnout.

Třetí možností je používat metodu GET a předávat parametry přes *"adresu stránky"*. V tomto případě se parametry stanou součástí URL. Z toho plyne jasná nevýhoda a to, že každý uvidí obsah jednotlivých proměnných a může je jednoduše pozměnit (pokud je vhodným způsobem nezašifrujeme). Proto se tuto metodu doporučuje používat jen v nejnutnějších případech a pro parametry, které potřebujeme mít stále k dispozici a jiným způsobem je předávat nelze (v naší implementaci např. si - unikátní číslo uživatele).

Konečně čtvrtou možností je použití metody POST, kdy se parametry předávají v těle HTTP požadavku. Zde je případný útočník nemůže modifikovat přímo v URL, ale musí použít jiné, více sofistikovanější metody. Proto budeme tuto metodu používat nejčastěji a v kombinaci se šifrováním obsahu proměnných a jejich důsledným kontrolováním je naprosto bezpečná.

2.1.2 Ověřování uživatele

Typickým případem narušení bezpečnosti je snaha útočníka dostat se na stránku, ke které nemá oprávnění tzv. přímým přístupem. To lze jednoduše provést tak, že přímo zapíše její adresu do adresního řádku prohlížeče (útočník ji musí znát). Bránit se lze kontrolováním práva na přístup uživatele, ke každé stránce aplikace zvlášť. Jako vhodná implementace se jeví vkládání PHP souboru s kontrolou pro přístup do každé stránky. Toto lze provést příkazem *require "název_souboru"* a je toho využíváno i v našem projektu.

2.1.3 Podvržení Session id

Obecně při používání Sessions nastává problém v případě, že útočník získá session id některého jiného uživatele (např. z nějakého log souboru nebo jeho zjištěním z URL). Tento problém nelze kompletně ošetřit a proto je potřeba minimalizovat rizika s ním spojená. I když v naší aplikaci nepoužíváme Sessions, je využito generování unikátního čísla (si) při přihlášení uživatele a to simuluje použití session id u Sessions. Mezi základní pravidla patří generování nového id při každém přihlášení uživatele, jeho zneplatnění při korektním odhlášení a zajištění jeho expirace po určité době, pokud se uživatel korektně neodhlásí. Dále je potřeba u kritických akcí jako je např. změna hesla, požadovat i staré heslo a nespoléhat se jen na identifikaci uživatele pomocí id.

2.1.4 Kontrola vstupů od uživatele

Je zapotřebí důkladně kontrolovat všechny vstupy od uživatele, aby nám nemohl útočník podvrhnout škodlivý kód, který by mohl vést k nežádoucím akcím (zjištění hesel uživatelů, smazání části systému a podobně). Na první pohled nemusí být tohle bezpečnostní riziko zřetelné, proto si ukážeme příklad na výpisu č. 2. Uživatel zadává měsíc a rok pro přístup ke kalendáři, který je volán jako externí příkaz. Avšak útočník může zadat místo roku např: ";ls -la", čímž se bez dostatečné kontroly vstupů zobrazí výpis web adresáře daného www serveru, případně zadáním ";rm -rf *" dosáhne smazání celého webu !!!

```
$month = $_GET['month'];
$year = $_GET['year'];

exec("cal $month $year", $result);
print "<PRE>";
foreach ($result as $r) { print "$r<BR>"; }
print "</PRE>";
```

Výpis 2: Kontrola vstupů od uživatele

2.1.5 SQL Injection

SQL injection se nazývá typ techniky, kdy se útočník pokouší vložit do příkazu pro SQL část svého kódu a tím upravit, či úplně změnit původní význam SQL dotazu. Opět si to nejlépe ukážeme na příkladu, který je ve výpisu č. 3.

```
SELECT * FROM users WHERE name='$username' AND pass='$password';
```

Výpis 3: SQL Injection

Uživatele chceme autentizovat přes databázi pomocí SQL dotazu. Ovšem pokud útočník zapíše heslo ve tvaru *' OR '1'='1*, výsledný dotaz bude vypadat jako na výpisu č. 4. Jeho výsledkem je, že uživatel bude autentizován jako daný uživatel bez znalosti hesla a bude mu umožněn přístup k celé aplikaci.

```
SELECT * FROM users WHERE name='znamy_uzivatel' AND pass='' OR '1'='1';
```

Výpis 4: SQL Injection 2

Proti tomuto útoku je několik možností obrany. Například můžeme použít funkci *addslashes()*, která zajistí vložení zpětného lomítka před znaky jako apostrof, uvozovky a další. SQL je potom bude chápat jako součást řetězce a přestanou být nebezpečné. V závislosti na nastavení PHP může být aktivní vlastnost jazyka zvaná *magic quotes*, která nám zajišťuje používání funkce *addslashes()* na všechny proměnné, přicházející z POST, GET a cookies.

2.1.6 Chybová hlášení

Chybová hlášení by měla být v ostrém provozu z bezpečnostních důvodů vypnuta, protože v případě chyby např. v SQL dotazu se uživateli vypíše chybové hlášení včetně části

tohoto dotazu. Toto může útočník zneužít například pro útok typu SQL injection. Vypnutí zobrazování chybových hlášení se v PHP provádí změnou proměnné *display_errors* v souboru php.ini na hodnotu nula. Dále je vhodné nastavit logování chyb do souboru a tento soubor pravidelně prohlížet. Opět se nastavuje v souboru php.ini pomocí proměnné *error_log*. Další možností je vytvoření vlastního systému chybových hlášení a nakládání s nimi dle vlastních požadavků. Pro náš případ jsme zkombinovali obě možnosti a využíváme jak automatického logování chyb do souboru, tak i posílání závažných chyb emailem na určenou adresu (definována v souboru function.php v proměnné *alert_email*).

2.1.7 Práva webserveru

Důležitou součástí bezpečnosti je nastavení web serveru tak, aby nespouštěl PHP skripty jako superuživatel, ale jako uživatel s minimálními právy, nutnými pro jejich chod a pro přístup jen do klíčových adresářů (adresáře určené k ukládání dat pro danou web aplikaci, přístup k ostatním potřebným PHP souborům a pod.). V opačném případě by se mohlo stát (například využitím chyby při ověřování vstupních dat), že útočník si bude schopen vytvořit svůj účet na webovém serveru a poté s ním naložit dle libosti.

Dále by neměla být citlivá data ukládaná do souborů, které jsou přístupné pomocí dotazu na web server. Tzn. neměla by se nacházet ve veřejné části web serveru a už vůbec ne s koncovkou, kterou webserver vyhodnotí jako bezpečnou a pošle obsah tohoto souboru uživateli. V případě potřeby je můžeme načíst např. pomocí příkazu *include* z PHP.

2.2 Implementace řešení webové části Virtlabu

Po komplexní analýze problému bezpečnosti webového rozhraní a seznámení se se současným stavem aplikace, bylo nutno udělat několik kroků k zajištění bezpečnosti.

Kroky jako nastavení příslušných práv pro spouštění PHP skriptů, nastavení používání HTTPS protokolu, nastavení chybových hlášení a podobně je třeba vždy nechat na administrátorovi, který bude celý systém instalovat (veškeré potřebné kroky jsou popsány v dokumentaci).

Jedním z prvních požadavků pro zabezpečení, bylo ověřování přístupů uživatelů pomocí školního LDAP serveru. Tato možnost se ukázala jako velice vhodná, protože většina uživatelů Virtlabu bude zároveň i studenty školy a proto můžeme k jejich autentizaci použít školního LDAP serveru. Odpadá nám tím starost o bezpečnost hesel v databázi, kdy v položce pro heslo se namísto hesla nachází řetězec *–LDAP–*. V případě, že uživatel není zaveden ve školním LDAPu, autentizuje se klasickým způsobem, přičemž jeho heslo je v databázi zakódováno. Po úspěšném přihlášení se uživateli vygeneruje id, které je uloženo do tabulky Sessions a provází ho po celou dobu práce se systémem (až do odhlášení ze systému). Toto id je kontrolováno při přístupu na jakoukoliv stránku, aby se zamezil útočníkovi přímý přístup na nechráněnou stránku. V případě, že se uživatel neodhlásí ze systému, je jeho session id odstraněno ze systému automaticky, po nastavené době nečinnosti. Pro úspěšné ověřování uživatelů pomocí LDAP serveru je potřeba mít PHP nainstalováno s podporou pro LDAP.

S ohledem na bezpečnost jsem se rozhodl implementovat kryptování obsahu veškerých proměnných, které se předávají pomocí URL, případně se objevují ve výpisu zdrojového kódu stránky (např. při předávání proměnných mezi jednotlivými stránkami). Rozhodl jsem se použít dostatečně bezpečný způsob šifrování a proto je potřeba mít nainstalováno PHP s podporou balíku *mcrypt*. Tento balík umožňuje použití několika algoritmů pro šifrování jako DES, TripleDES, Blowfish, 3-WAY, SAFER-SK64 a podobně. Vybral jsem TripleDES, jehož délka klíče je dostatečně velká (192 bitů) a algoritmus je přiměřeně rychlý. Algoritmus potřebuje ke své práci klíč, pomocí kterého bude kryptovat požadované řetězce. Klíč se generuje ihned po úspěšném přihlášení uživatele, pomocí speciální funkce, která náhodně vybírá znaky ze znakové sady. Pro naší potřebu je vybráno 24 znaků ($24 * 8 = 192$ bitů) a ty jsou jako řetězec uloženy do tabulky Sessions a korespondují s unikátním id. Z toho vyplývá, že klíč zůstane stejný po dobu, dokud se uživatel neodhlásí ze systému. Administrátor systému má možnost zakázat kryptování (např. pro usnadnění práce při dalším vývoji) změnou proměnné $kryptuj v souboru *function.php* na hodnotu nula. Kryptovány jsou všechny hodnoty, u kterých to má smysl. Výjimku tvoří například unikátní id uživatele, které nelze z technických důvodů kryptovat. Neexistovala by potom možnost, jak zjistit, který uživatel v systému pracuje a tudíž, který klíč z databáze se má použít pro dekódování proměnných.

Díky kryptování proměnných se nemusíme obávat toho, že by útočník mohl pozměnit jejich hodnoty a tím získat přístup k nepovoleným akcím. Navíc jsou na všech místech, kde to bylo možné, proměnné předávány pomocí metody POST. Výjimku tvoří například část stránek, určená pro spuštění úlohy, kde jsou z technických důvodů předávány pomocí GET. Ale jsou opět kryptované, takže jejich pozměnění je téměř nemožné.

Na ostrém serveru s ohledem na bezpečnost doporučuji vypnout zobrazování chybových zpráv uživateli a zapnout jejich logování do zvoleného souboru. Zprávy ukládané do logu jsou upraveny tak, aby byly ve formátu *id_uzivatele:zprava*. Mimo to je při vážných chybách, jako je nemožnost připojit se k databázi, posílán email s chybovou hláškou na určenou emailovou adresu. Útoku typu SQL injection je zabráněno především zapnutím PHP vlastnosti *magic quotes*, která zabraňuje vložení nebezpečných znaků jako apostrof, uvozovky a pod. z proměnných do SQL příkazu tím, že před ně vloží zpětné lomítko.

2.3 Zabezpečení na straně Java appletu

Velkým bezpečnostním rizikem pro provoz celé virtuální laboratoře jsou samí uživatelé, kteří mohou úmyslně, či zcela neúmyslně zablokovat zařízení a tím znemožnit ostatním a případně i sobě práci na něm. Typickým příkladem jsou příkazy IOSu jako *enable password*, které umožňují nastavení hesel. Pokud uživatel nastaví na zařízení heslo, které je jiné než dohodnuté, znemožní uživatelům na tomto zařízení další práci bez jeho znalosti. V takovém případě nebude fungovat ani automatické vymazání zařízení před začátkem nové úlohy. Jedinou možností znovuzprovoznění zařízení tak zůstává manuální přemazání, k čemuž je ovšem potřeba fyzického přístupu k danému zařízení. Takové řešení je zcela nepoužitelné, pokud si představíme, že někdo zahesluje zařízení v pátek večer a až do pondělka nebude možno pracovat na žádné úloze, která toto zařízení využívá.

Toto vede k nutnosti implementovat do systému funkci, která nám umožní definovat pro každé zařízení zakázané příkazy, které nebude možno do zařízení zadat. Ideální se ukázalo implementovat tuto funkci v javaaapletu a testovat na zakázané příkazy každý řádek před odesláním na Cserver. V případě, že tento bude obsahovat zakázaný příkaz, bude uživateli zobrazeno varování a příkaz nebude na server odeslán. Vzhledem k tomu, že některé příkazy je potřeba zakázat úplně a některé je potřeba povolit jen s určitými kombinacemi slov (omezené příkazy), rozdělil jsem je na dvě skupiny. Každé zařízení má proto v tabulce Equipment dvě položky pro jejich uložení.

V první položce jsou ty, které jsou zakázané jako celek (např. *menu* u switche). Jsou implementovány s podporou regulárních výrazů. To znamená, že pokud bude v zakazovacím řetězci například *"menu.*"*, bude zakázán jakýkoliv řádek, který začíná slovem *menu*. Pozornost při psaní zakazovacího řetězce je potřeba věnovat znaku \, který je potřeba psát zdvojeně (vzhledem k implementaci práce s řetězci v javě). Jednotlivé zakázané příkazy od sebe oddělujeme zdvojeným znakem /. Typický řetězec zakázaných příkazů prvního typu pro switch může vypadat takto:

```
menu.*//login tacacs.*// enable secret 0  .*// enable secret 5  .*// enable secret level  .*// enable
    use−tacacs.*
```

Výpis 5: Zakázané příkazy typ 1

Ve druhé skupině jsou příkazy, které mohou pokračovat jen určitým způsobem, budeme je nazývat omezené příkazy. Sem patří například již výše zmiňované příkazy pro nastavení hesla. Následuje popis syntaxe zápisu celého řetězce zakázaných příkazů. Jako první napíšeme omezený příkaz (opět zde lze využit regulárních výrazů), oddělíme ho dvojicí znaků *&*. Následují řetězce, které jsou za daným příkazem povoleny. Tyto řetězce od sebe oddělujeme dvojicí znaků %. Další omezený příkaz se odděluje stejně jako u prvního typu pomocí zdvojeného znaku /. Ukážeme si typický příklad použití, opět pro switch:

```
*(en$|$ena$|$enab$|$enabl$|$enable)\{1\} +(s$|$se$|$sec$|$secr$|$secre$|$secret)\{1\} +\&\&
    cisco// *(en$|$ena$|$enab$|$enabl$|$enable)\{1\} +(p$|$pa$|$pas$|$pass$|$passw$|$
    passwo$|$passwor$|$password)\{1\} +(l$|$le$|$lev$|$leve$|$level)\{1\}
    +(1$|$2$|$3$|$4$|$5$|$6$|$7$|$8$|$9$|$10$|$11$|$12$|$13$|$14$|$15)\{1\} +\&\&cisco
```

Výpis 6: Zakázané příkazy typ 2

Jak si můžete všimnout, je při tvorbě zakázaných příkazů třeba klást důraz i na fakt, že příkazy lze do zařízení zadávat i v jejich zkrácené podobě. Je nutno tedy vystihnout všechny možné jejich kombinace, obdobně jako to demonstruje uvedený příklad. Toto je hlavní důvod, proč byl kladen důraz na implementaci s možností použití regulárních výrazů. Další problém nastává s rozlišením malých a velkých písmen, případně s jejich střídáním. Toto je implementováno přímo ve funkci, která se stará o zpracování zakázaných výrazů a tudíž na to nemusíme myslet při jejich vytváření.

V případě, že zařízení nemá definované žádné zakázané příkazy, nenechává se položka v databázi prázdná, ale zapisuje se do ní řetězec *"zadne prikazy"*. Toto je zavedeno s ohledem na bezpečnost, kdy při prázdném řetězci by nebyl pro útočníka problém spustit applet s prázdným parametrem pro zakázané příkazy a tím tuto ochranu vyřadit. Útočník

nemůže zakázané příkazy nijak pozměnit vzhledem k tomu, že jsou v HTML kódu i při předávání jako parametry appletu zakryptovány pomocí šifrovacího klíče. O použití a implementaci tohoto klíče se více dozvíte v kapitole 2.2.

3 Bezpečnost komunikace mezi komponenty Virtlabu

Bezpečnost komunikace mezi jednotlivými komponenty Virtlabu je velice důležitým prvkem Virtlabu. Zvláště pak, pokud si uvědomíme, že veškerá komunikace mezi uživatelem a Virtlabem (a jeho zařízeními) je přenášena zcela nechráněně přes internet. Komunikace mezi uživatelem a webovým rozhraním probíhá pomocí nezabezpečeného HTTP protokolu. A taktéž komunikace mezi uživatelem a Cserverem probíhá na úrovni nezabezpečeného TCP spojení. Tuto komunikaci je třeba zabezpečit, aby se zabránilo útočníkovi v odposlechnutí, případně změně přenášených informací.

3.1 HTTP protokol

V případě, že používáme klasický, nezabezpečený HTTP protokol, musíme si uvědomit, že veškerá textová data, která putují od nás k serveru a naopak se přenášejí v nijak nezabezpečeném, čitelném stavu. Pokud bude útočník schopen odchytávat tyto pakety, není pro něj žádný problém kontrolovat veškerou naší komunikaci se serverem a tím pádem i veškerá (i důvěrná) data, které serveru posíláme. Mezi tato důvěrná data patří obecně nejčastěji uživatelské jméno a heslo, případně číslo kreditní karty a pod.

Z tohoto důvodu se doporučuje používat HTTPS protokol, který veškerou komunikaci mezi námi a serverem kryptuje. Dále musíme vědět, že opravdu komunikujeme s požadovaným serverem a ne někým, kdo se za něj vydává. K tomuto účelu se používají certifikáty, které potvrzují, že server je opravdu tím, za koho se vydává. Certifikáty vydávají důvěryhodné certifikační autority (společnosti).

3.2 Komunikace mezi webovou aplikací a databází

Komunikace mezi PHP a případnou databází probíhá za běžných okolností také nekryptovaně a proto je vhodné zabezpečit i tuto oblast. Toto se provádí tím, že se k databázi nepřipojujeme běžným způsobem, ale využíváme kryptované spojení pomocí SSL, obdobně jako u HTTPS protokolu. V případě, že databáze i PHP běží na stejném stroji (jako v našem případě) není potřeba tuto komunikaci řešit, protože celá probíhá v rámci jednoho stroje.

3.3 Komunikace mezi webovou aplikací, Java appletem a Cserverem

Dalším problémem v bezpečnosti celé virtuální síťové laboratoře je otázka předávání parametrů mezi PHP a appletem a následná komunikace mezi appletem a C serverem.

3.3.1 Komunikace mezi webovými stránkami a Java appletem

Pro správné spuštění appletu je mu potřeba předat z webové stránky parametry, potřebné pro jeho inicializaci a běh. Útočník může tyto parametry jednoduše zjistit ve zdroji WWW stránky, vytvořit si vlastní stránku a tam je pozměnit. Tím může docílit pro něj prospěšného chování appletu jako např. pozměnění zakázaných příkazů pro jednotlivá zařízení.

Začlenění SSL protokolu do TCP/IP modelu	
TCP/IP vrstva	**Protokol**
Aplikační vrstva	HTTP, NTP, Telnet, a pod.
SSL vrstva	SSL
Transportní vrstva	TCP
Internetová vrstva	IP

Obrázek 2: TCP/IP model včetně SSL

Z tohoto důvodu jsou rovněž kryptovány veškeré důležité parametry appletu[6]. Útočníkovi je tak znemožněna jakákoliv manipulace s nimi, bez znalosti klíče. Applet samozřejmě musí tento klíč znát. Po spuštění appletu je posláno C serveru unikátní id uživatele. C server si ověří, zda takové id existuje a v případě kladného výsledku pošle appletu klíč, patřící k danému id. Applet si pomocí tohoto klíče dešifruje všechny parametry a může pokračovat v komunikaci se serverem. Hrozí zde nebezpečí odposlechnutí klíče při jeho posílání mezi Cserverem a appletem, proto je nutno tuto komunikaci patřičně zašifrovat. Podrobnosti uvádím v kapitole 3.3.2.

3.3.2 Komunikace mezi Java appletem a Cserverem

Komunikace mezi Cserverem a appletem probíhala nezabezpečeným způsobem pomocí TCP spojení. Obě komponenty si posílají komunikaci v textové podobě, tak jako to dělá například telnet. Jen několik speciálních řídících příkazů má svůj formát. Z toho vyplývá, že celá komunikace mohla být bez problému odposlouchávána a případně i pozměněna útočníkem. Jednou z možností, jak tomuto zabránit, je vytvoření speciálního protokolu. Tímto protokolem by se mohly posílané znaky různě kódovat, dělat s nimi například bitové posuny a podobně, čímž by se útočníkovi znemožnilo odposlouchávání, aniž by znal používaný protokol a kódování. Toto řešení má však jednu výraznou slabinu a to v podobě neměnnosti toho protokolu a kódování. Takže by bylo jen otázkou času, než by útočník použitý protokol rozluštil (zvláště, pokud je zdrojový kód volně k dispozici). Z tohoto důvodu jsem se rozhodl pro použití zabezpečení formou SSL protokolu[1][2]. SSL je protokol, který poskytuje zabezpečení na síti tím, že přidává další vrstvu do klasického TCP/IP modelu tak jak je zobrazeno na obrázku č. 2. Přesněji mezi transportní a aplikační vrstvu síťového modelu. TCP neposkytuje prostředky pro zabezpečené spojení a rovněž neumožňuje zajistit pravost uživatelů na koncích spojení. Naproti tomu SSL nám poskytuje jak ověření uživatelů, tak zabezpečené spojení mezi nimi. Dosahuje toho pomocí svých dvou podprotokolů. První je *SSL record protokol*, který definuje formát pro přenos dat a druhým je *SSL handshake protokol*, který definuje proces při úvodní výměně dat, včetně autentizace uživatelů na obou koncích spojení. Dále SSL používá hashovací funkci podobnou kontrolnímu součtu, kterou zahashuje zprávu a tento hash přilepí na konec zprávy před jejím zašifrováním. Tímto se brání proti pozměnění zprávy útočníkem. Java podporuje SSL od verze 1.4.2 nativně pomocí JSSE[9]. Pro použití v C++

bylo zapotřebí využít přídavný balík knihoven s názvem OpenSSL[1]. Tento balík umožní nastavit a vytvořit SSL spojení a poté s ním pracovat podobným způsobem jako s obyčejným socketem. Pro vytvoření úspěšného SSL spojení je nutno vybrat typ šifrování, podle kterého bude probíhat vzájemná komunikace, vygenerovat certifikát a odpovídající klíče k němu. Certifikát je digitálně podepsané sdělení, ručící za identitu a veřejný klíč určité entity (osoby, společnosti, počítače a podobně). Certifikáty mohou být v nejrůznějších formátech, z nichž nejrozšířenější je *X509*, který také budeme používat. Certifikáty jsou vydávané buď pověřenými certifikačními autoritami (za což se platí), nebo si je můžeme vydat sami a následně podepsat (tzv. *self-signed* certifikáty). Pro náš případ, kdy budeme certifikát potřebovat jen pro vytvoření spojení mezi klientem a serverem (applet a Cserver), nám plně postačí *self-signed certifikát*. Celý proces vytváření SSL spojení vypadá následovně:

- Cserver si načte ze souborů certifikát, privátní klíč k němu a čeká na požadavek o spojení
- Applet si načte ze svého úložiště symetrický klíč
- Applet požádá o připojení na server a nabídne možné algoritmy a šifry pro použití
- Cserver vybere nejvhodnější kombinaci a pošle appletu certifikát
- Applet zakóduje svůj symetrický klíč veřejným klíčem z certifikátu a pošle ho Cserveru
- Cserver pomocí svého privátního klíče dešifruje symetrický klíč appletu
- Dále už probíhá komunikace pomocí šifrování symetrickým klíčem s použitím hashe

[1]Bližší informace na adrese: http://www.openssl.org

4 Role tutora v systému

Práce studentů na úloze je koncipována jako samostatná činnost, ale ukázalo se, že by bylo vhodné, kdyby se mohl vyučující, či jiný uživatel s potřebnými zkušenostmi a právy (tutor) připojit ke konzoli zařízení, na kterém student pracuje a ukázat mu správný postup. Dalším případem využití je možnost záměrné změny konfigurace určitého zařízení tutorem (například k účelu simulování chyby na zařízení nebo nějakém ze spojů). Student poté řeší nastálou situaci a snaží se přijít na příčinu problému (tzv. troubleshooting).

4.1 Analýza

Po analýze celého problému se ukázalo jako nejvhodnější, vytvoření v systému nové role uživatele, tutora. Toto znázorňuje obrázek 3. Uživatel s právy tutora má možnost prohlédnout si jakoukoliv právě běžící úlohu, vybrat si zařízení (i obsazené) a připojit se k němu. To je mu umožněno tlačítkem z hlavní stránky, která se zobrazuje po přihlášení do systému. Vzhledem k tomu, že při ukázce správné konfigurace zařízení je zapotřebí, aby uživatel viděl práci tutora, ale při některých jiných zásazích do zařízení je toto nevhodné, rozhodl jsem se vytvořit dva způsoby, kterými se může tutor k zařízení připojit.

Prvním z nich je skrytý mód, kdy po připojení tutora k obsazenému zařízení je uživateli zobrazena informace o tom, že se na zařízení připojil tutor a že uživatel jeho práci neuvidí. Tutor může na zařízení pracovat, aniž by uživatel viděl jeho počínání. Po odpojení tutora od zařízení je uživatel informován o tom, že se tutor odpojil a může pokračovat v práci na zařízení.

Druhou možností je veřejný mód, kdy chce tutor například předvést uživateli správný postup práce na zařízení. V takovém případě se uživateli zobrazí upozornění, které mu říká, že se na jeho zařízení připojil tutor a uvidí jeho práci, nebude však do ní moci zasahovat. Tutor může zadat potřebné příkazy na zařízení a poté se odpojit. Jakmile se tutor odpojí, uživatel o tom bude informován a bude moci pokračovat v práci.

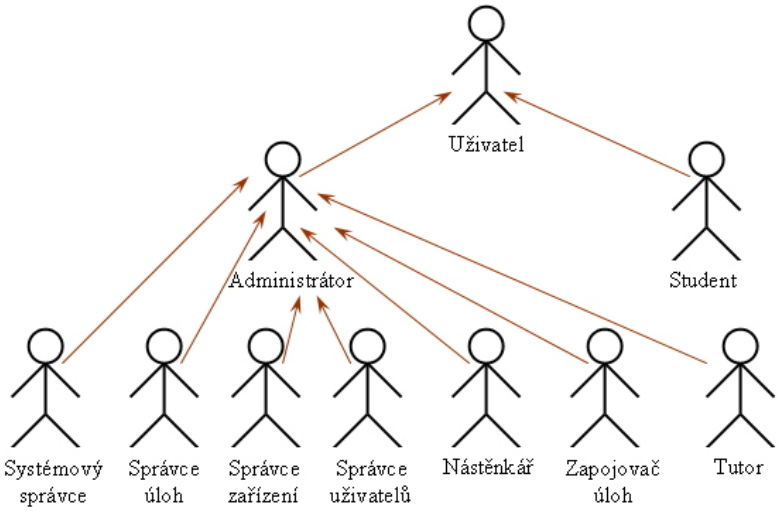
Počas práce tutora na zařízení je o tom uživatel informován změnou nápisu v okně appletu, jak je vidět na obrázku 4. Uživatel nemá v této době možnost na zařízení pracovat, protože applet nereaguje na jeho příkazy.

4.2 Implementace

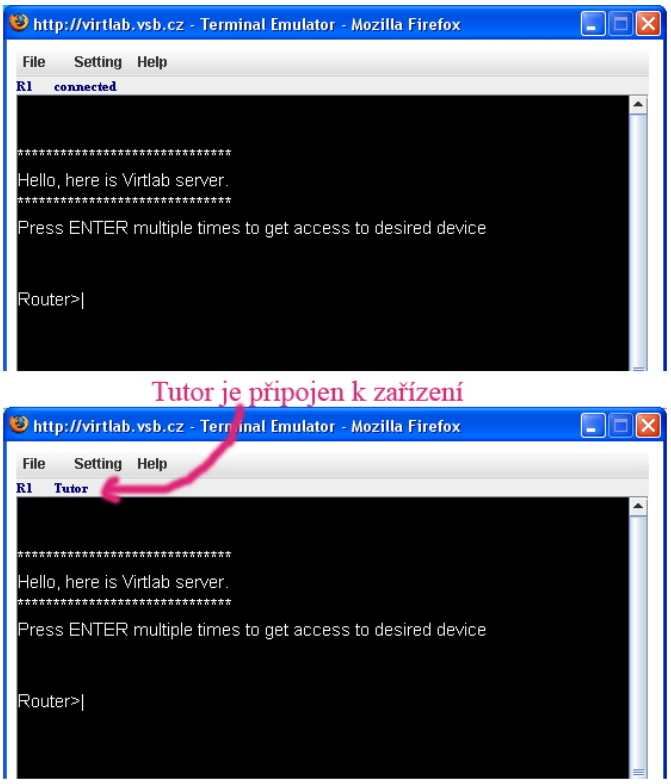
Vzhledem k tomu, že při implementaci tutora se vytvářela v systému zcela nová role uživatele (tutor), bylo nutno změnit kód nejen na straně appletu a Cserveru, ale i v PHP kódu webových stránek a dokonce změnit strukturu databáze. Protože si applet a Cserver musí nějak předat informace o tom, zda se chce připojit tutor, bylo rovněž zapotřebí pozměnit komunikační protokol (viz. kapitola 8.1) mezi těmito dvěma prvky. Všechny důležité změny si nyní popíšeme.

4.2.1 Úpravy v databázi a webových stránkách

S novou uživatelskou rolí tutora bylo zapotřebí změnit strukturu databáze, přesněji strukturu tabulky uživatelů (User). Přidal jsem do ní položku s názvem *admin_tutor*, která je



Obrázek 3: Role uživatelů v systému



Obrázek 4: Informování o práci tutora na zařízení

obdobně jako ostatní položky pro role uživatelů, výčtového typu a může nabývat hodnot 'N' nebo 'Y'. Zároveň se změnou v databázi bylo nutno upravit kód PHP stránek, zodpovědných za vytváření a editaci uživatelů. Bylo je potřeba rozšířit o možnost nastavovat i tuto roli jak při vytváření, tak při editaci uživatele. Dále jsem musel upravit hlavní stránku tak, aby umožňovala tutorovi dostat se do všech právě běžících úloh. Stránka pro spouštění úloh musela být také přepracována pro potřeby tutora (dvě řady tlačítek pro připojení k zařízením). Dále jsem přidal parametr předávaný appletu. Tento parametr appletu oznamuje, zda s ním bude pracovat tutor nebo obyčejný uživatel.

4.2.2 Úpravy v Java appletu

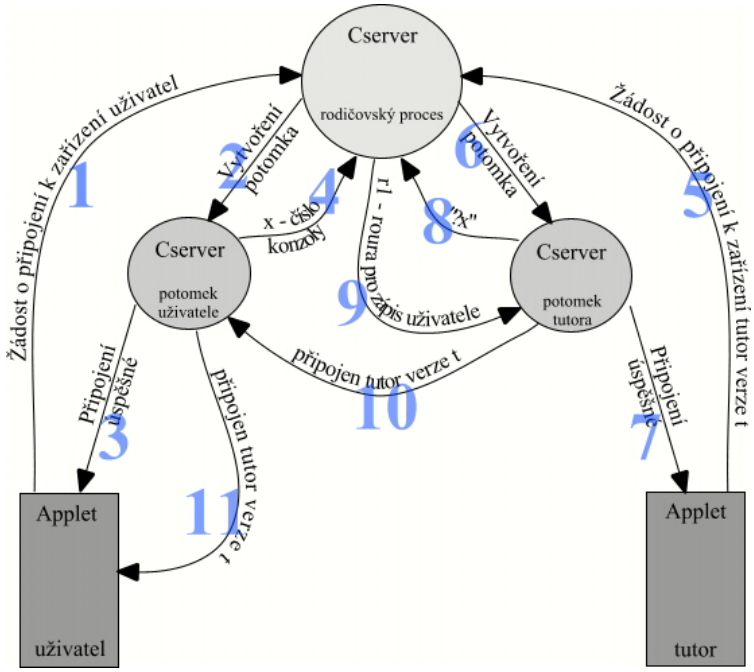
V appletu bylo zapotřebí provést hned několik úprav. Základní úpravou je připravení appletu pro příjem dalšího parametru z webové stránky a jeho případné dešifrování (pokud je šifrování v systému zapnuto). Po navázaní spojení s uživatelem, jeho ověření a získání klíče pro dešifrování dat je Cserveru poslána informace o tom, zda se daný uživatel hlásí jako tutor. Ten ještě pro jistotu ověří, zda uživatel s daným id opravdu má přiřazenu roli tutora a v kladném případě pokračuje v komunikačním protokolu s appletem. Applet je dále připraven na to, že může přijmout od Cserveru zprávu (definovanou v komunikačním protokolu) obsahující jednu ze tří aktivit tutora:

- Tutor se přihlásil k zařízení, ke kterému applet poskytuje přístup a bude pracovat ve skrytém módu.
- Tutor se přihlásil k zařízení a bude pracovat ve veřejném módu.
- Tutor se odhlásil od zařízení.

Podle typu zprávy applet zobrazí uživateli upozornění a provede změnu nápisu v okně appletu, který informuje o aktuálním stavu spojení se zařízením. V případě skrytého módu přestane uživateli zobrazovat znaky, přicházející ze zařízení. Dále applet zablokuje uživateli možnost vysílat požadavky na zařízení, pokud je k němu tutor připojen.

4.2.3 Úpravy v Cserveru

Začlenění tutora do systému si vyžádalo nejvíce úprav v Cserveru. Bylo zapotřebí vymyslet mechanismus, kterým se budou předávat zprávy o přihlášení tutora mezi jednotlivými procesy, které Cserver vytváří pro každý požadavek o připojení k zařízení. Nejvhodnější se k tomuto účelu ukázaly být roury[3][5] (*pipes*). Následující popis předávání zprávy o připojení tutora zachycuje obrázek 5 (čísla v obrázku označují pořadí událostí). Před vytvořením nového procesu se vytvoří dvě roury (jedna je později použita pro zápis a druhá pro čtení), které po jeho vytvoření fungují jako komunikační kanál mezi rodičem a potomkem. K jejich správné funkci je zapotřebí zavřít nepoužívané konce. Po vytvoření potomka čeká rodič na zprávu, která od něj přijde. Podle typu zprávy rozliší, zda je k zařízení připojen obyčejný uživatel nebo tutor. V případě obyčejného uživatele, si uloží do předem připraveného pole pid (unikátní identifikační číslo procesu) potomka a taky identifikátory roury pro čtení a zápis k tomuto potomkovi, které využívá dále. Tyto



Obrázek 5: Mechanismus předávání zprávy o připojení tutora

informace si ukládá na místo, shodné s číslem konzole, ke kterému je uživatel připojen (to získá z roury od potomka). Pokud se jedná o tutora, pošle mu číslo roury, pomocí které může předat informaci správnému procesu o tom, že se na stejné zařízení připojil tutor. Tutor tenhle identifikátor ihned použije a pošle určenému procesu zprávu, že je připojen. Tento proces zprávu příjme a podle obsahu zjistí, zda chce tutor pracovat skrytě nebo veřejně. Podle jeho volby zašle appletu typ zprávy o připojeném tutoru. Pokud se tutor odpojí, je opět odpovídajícímu procesu předána zpráva o jeho odpojení a ten ji přepošle dále appletu. Zde si je zapotřebí uvědomit, že vytvořením roury se procesy na sobě stávají závislými a tak po ukončení procesu nedojde k jeho odstranění, ale stane se z něj tzn. zoombie. Zůstane jí, dokud se neukončí rodičovský proces. Proto je zapotřebí v rodičovském procesu kontrolovat stav jeho potomků. Teprve převzetím potomkova návratového kódu se proces potomka zcela odstraní ze systému.

5 Komunikace systému s administrátory

Při používání Virtlabu se ukázalo, že v systému existují situace, o kterých je zapotřebí vhodně informovat vybranou skupinu uživatelů. Jako nejvhodnější prostředek pro informování jsem vybral komunikaci pomocí elektronické pošty. Jednak má každý uživatel v systému definovánu svou emailovou adresu a navíc je tato forma komunikace vcelku lehce implementovatelná. V průběhu používání a úprav systému se vyskytly dva hlavní případy, kdy je vhodné použít tohoto komunikačního prostředku.

Prvním z nich je případ, kdy se PHP skriptu nepovede připojit k databázi. Jedním z důvodů této chyby může být změněné heslo pro přístup k databázi (například útokem hackera), nebo jeho špatné nastavení. Dalším důvodem může být chyba na straně serveru, na kterém databáze běží (chyba souborového systému, neúmyslné smazání databáze a podobně). Tato chyba má za následek celkové vyřazení systému z provozu a proto je zapotřebí ji okamžitě vyřešit. PHP skript v tomto okamžiku nejenom uloží informaci o této chybě do logovacího souboru, ale také pošle na nastavenou emailovou adresu zprávu o nemožnosti přístupu k databázi s požadavkem o sjednání nápravy v nejbližší možné době. Emailová adresa je nastavená v souboru *function.php* v proměnné *alert_email*. Tuto adresu není možné načítat z databáze, protože varování na ní se bude posílat právě v okamžiku nepřístupné databáze.

Druhý případ, ve kterém je využita komunikace systému s uživateli nastává v případě, že je někým rezervována úloha, kterou je potřeba před jejím začátkem ručně zapojit. K tomuto účelu slouží pomocný program nazvaný *Check Timeslot*, do kterého byla doimplementována funkce kontroly ručního zapojování úlohy. Program zjišťuje dva časové úseky (timesloty) dopředu (toto lze nastavit parametrem v programu), zda v tomto čase nepoběží úloha, kterou je potřeba zajít ručně zapojit. Pokud je tomu tak, zjistí nahlédnutím do předcházejícího timeslotu, zda úloha jen nepokračuje. Pokud je zde opravdu začátek nové úlohy, vyhledá mezi uživateli všechny s rolí zapojovače úloh a všem na jejich emailové adresy pošle zprávu s žádostí o ruční zapojení konkrétní úlohy v konkrétním čase. Zde nastává otázka, zda je nezbytné posílat žádost o zapojení úlohy i v případě, že poslední spuštěná úloha měla totožné zapojení s následující úlohou. Bylo by vhodné toto zohlednit při implementaci a v případě, že není nutno měnit topologii zařízení, žádost neposílat. Tento postřeh byl získán příliš pozdě na to, aby mohl být korektně implementován. Pro je uveden v návrzích na další vylepšení Virtlabu, viz. kapitola 9.1.

V případě jazyka C se k poslání emailu využívá spuštění externího příkazu operačního systému, který je schopen pracovat s emaily. Jako parametr mu je předán soubor s obsahem a parametry emailu. V PHP je pro poslání emailu využíván speciální případ funkce pro logování chybových akcí. Varování se neukládá do souboru nastaveného pro logování, ale posílá se na uvedený email.

6 Úprava vzhledu a uživatelského rozhraní aplikace

Aplikace měla vzhledem k použití kaskádových stylů a jejich rozdílné implementaci v různých webových prohlížečích značné problémy se správným zobrazováním webových stránek pod jiným prohlížečem než je Internet Explorer. Tím se stala těžce použitelnou pro uživatele operačního systému jiného než Windows a také pro uživatele používající dnes stále rozšířenější webový prohlížeč Mozilla Firefox. Optimalizovat aplikaci pro všechny dostupné prohlížeče by bylo vzhledem k její rozsáhlosti velmi časově náročné, rozhodl jsem se ji optimalizovat pro dva hlavní prohlížeče, zastávající dnes převážnou většinu trhu. Jsou jimi Internet Explorer a Mozilla Firefox.

Vhodnou úpravou kaskádových stylů se podařilo odstranit největší problém a to zobrazování příliš malého fontu ve Firefoxu. Dále bylo zapotřebí posunout grafické prvky na některých místech webu (například procházení zařízení) tak, aby se nepřekrývaly s ostatními prvky na stránce.

Dalším potřebným krokem bylo vylepšení uživatelského rozhraní, jeho ergonomičnosti a případně doplnění chybějících údajů na klíčových částech webu. Hlavní změnou je zpříjemnění uživatelského rozhraní zminimalizováním počtu nutných kliknutí myší. Kdy při akcích jako rezervace úloh bylo zapotřebí po úspěšně provedené rezervaci odkliknout informaci o úspěšném provedení rezervace a teprve poté se uživatel dostal zpět na stránku s rezervacemi. Toto je nyní vyřešeno informací o úspěšné rezervaci přímo v horní části stránky, ve které se rezervace provádějí. Tato změna i když na první pohled nevýznamná značně zpříjemnila a zrychlila práci s webovým rozhraním Virtlabu. Takovou úpravu bylo zapotřebí provést na několika místech webového rozhraní.

Na hlavní stránku, která se zobrazí uživateli po úspěšném přihlášení přibyla informace o úlohách, které má uživatel rezervovány. Zobrazuje se čas začátku úlohy a její název. V případě, že úlohu je možno již spustit, zobrazuje se i tlačítko, které uživateli umožní tuto úlohu spustit. Dále je uživateli vypsána aktuální kvóta, kterou ještě může vyčerpat při rezervaci úloh pro aktuální týden. Pokud má uživatel roli tutora, zobrazují se mu na hlavní stránce také tlačítka pro spuštění všech právě běžících úloh.

Uživatelům, jejichž heslo je autentizováno pomocí školního LDAPu, byla zakázaná možnost nastavit si nové heslo pro přihlášení do Virtlabu a jsou nuceni z bezpečnostních důvodů používat jen ověřování pomocí LDAPu.

Pro administrátora, majícího na starosti vytváření a správu úloh bylo upraveno vytváření úloh tak, aby bylo možno vybírat zařízení do úlohy i v jiném prohlížeči, než Internet Exploreru. Původně měl celý proces vytváření úlohy spoustu nedostatků, projevujících se primárně v jiných prohlížečích než je Internet Explorer. Rovněž byly upraveny některé detaily při přidávání úloh pomocí XML, které nebylo uzpůsobeno na hierarchické řazení kategorií úloh. Původně se v XML zadávala kategorie jen s názvem kategorie, což bylo špatně. Nyní se zadává správně nejen pomocí názvu kategorie, ale včetně třídy kategorie, do které patří. Nový XML tag vypadá následovně:

```
<seznam_kategorii>
    <kategorie id="RIP v2" trida="Směrovací protokoly" />
    <kategorie id="SPS" trida="Předměty" />
</seznam_kategorii>
```

Výpis 7: XML tag pro přidání kategorie

U rezervací úloh se již nezobrazuje id uživatele, pokud ji má přihlášený uživatel rezervovánu, ale zobrazuje se text odstranit. Kliknutím na tento text může uživatel svou rezervaci zrušit. Dříve se rezervace rušila po kliknutí na login uživatele, což se ukázalo podle reakcí uživatelů jako matoucí. Také byla přidána možnost vrátit se při odstraňování rezervace o krok zpět kliknutím na odkaz, namísto doposud jediného možného řešení a to klinutím na tlačítko zpět v prohlížeči.

U spouštění úloh bylo znatelně vylepšeno zabraňování opětovného kliknutí na tlačítko pro přístup ke konzole zařízení a to jeho zašeděním. Toto bylo implementováno již dříve, ale často se stávalo, že tlačítko nebylo zašeděno a tím vznikl prostor pro vícenásobné přihlášení k jednomu zařízení. Bylo to možné také z důvodu, že v Cserveru nebyla implementována kontrola, zda již na zařízení někdo pracuje, či nikoliv. Případně po odpojení od zařízení, nebo zavření okna appletu nebylo tlačítko odšeděno a uživatel se nemohl k zařízení znovu připojit. Tato vlastnost se ukázala jako dosti nevhodná a proto bylo vyvinuto značné úsilí na její odstranění. Nyní je tato část vyřešená tím, že applet po svém úspěšném připojení k zařízení zavolá funkci, která provede obnovení okna prohlížeče, ve kterém je zobrazena stránka pro spuštění úlohy. Stránka se obnovuje i v případě odpojení se od zařízení. Takže po úspěšném připojení se v prohlížeči tlačítko zašedí a vedle něj se zobrazí id uživatele, který na něm právě pracuje. V případě odpojení je tlačítko odšeděno a id uživatele odstraněno. Applet bohužel ve většině případů nestihne zavolat funkci pro obnovení webové stránky v případě, že jeho okno je zavřeno bez předchozího odpojení uživatele. S tím bohužel na straně appletu nelze nic dělat, je to způsobeno provázáním appletu s webovým prohlížečem a operačním systémem. Naštěstí existuje možnost nechat obnovit webovou stránku se spuštěním úlohy pomocí okna prohlížeče, ve kterém se nachází applet. Zde je možno definovat funkci, která se má provést po zavření okna prohlížeče. Definujeme zde obnovení rodičovského okna prohlížeče, ale musíme s jeho obnovením určitý čas počkat (stačí jedna vteřina). Musíme počkat než Cserver zjistí, že se uživatel od zařízení odpojil a stihne v databázi uvolnit zařízení pro další použití. Veškeré obnovování stránky s formulářem pro spuštění úlohy je možné jen díky tomu, že jsou všechny potřebné parametry stránky předávány funkcí GET (pomocí URL adresy stránky) a ne pomocí metody POST. Právě z tohoto důvodu jsou na tomto jako na jednom z mála míst webového rozhraní Virtlabu předávány parametry metodou GET.

U rozhraní pro přidávání a editaci zařízení přibyla položka pro model zařízení. Model zařízení se dosud chybně zadával namísto jeho sériového čísla. Přidáním této položky se zpřehlednila práce se zařízeními, protože máme k dispozici nejenom jejich unikátní identifikátor ve Virtlabu, ale také jejich sériové číslo a model zařízení podle výrobce. Dále bylo nutno přidat položky pro definici zakázaných příkazů pro konkrétní zařízení. Jsou k tomu určeny dvě položky, označené jako *Kompletně zakázané příkazy* a *Částečně zakázané*

příkazy. Pro všechny nové položky bylo samozřejmě zapotřebí přidat patřičné záznamy do struktury databáze.

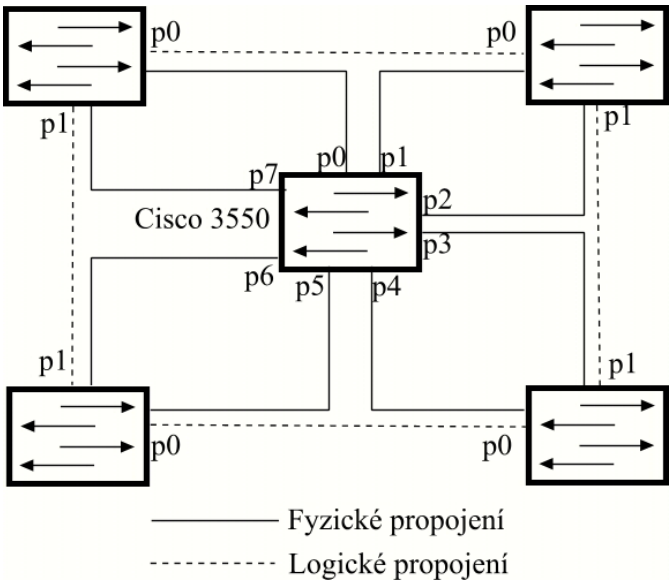
V části pro administraci uživatelů byla provedena řada změn, které usnadní jejich správu. Z těch, které přispívají k lepšímu uživatelskému komfortu jmenujme alespoň setřízení seznamu uživatelů podle abecedy, což značně usnadňuje jejich vyhledávání. Informace o dalších změnách v této části lze nalézt v kapitole 8.

7 Automatizované spojování trunk portů

Dosud používané automatizované spojovací pole umožňuje spojovat jak ethernetové, tak i sériové rozhraní zařízení. Počet portů je ovšem značně omezen a především s ohledem na množství ehternetových rozhraní u přepínačů je nedostatečný. Proto bylo navrženo řešení, využívat k propojování ethernetových rozhraní přepínače s podporou VLAN Tunnelingu. Jedním z takových přepínačů je například Cisco 3550.

Technologie VLAN Tunnelingu[10] neboli také tunelování protokolu na druhé vrstvě je primárně navržena pro použití u poskytovatelů připojení, jejichž páteřní síť je využívána mnoha zákazníky. Tito zákazníci mohou mít definovány své vlastní VLAN sítě a v případě, že se jejich VLAN sítě budou překrývat, docházelo by v síti poskytovatele ke kolizím. V případě VLAN Tunnelingu ale žádné kolize nehrozí, protože před vstupem rámce do sítě poskytovatele je mu přiřazeno další VLAN ID, které je platné pro daného zákazníka a celou síť poskytovatele. Vzhledem k tomuto jedinečnému VLAN ID nemůže nastat kolize mezi VLANy jednotlivých zákazníků, i když ve svých sítích používají totožné VLAN ID. Rámec je poté navigován přes sít poskytovatele připojení. Není zpracováván, ale pouze přeposílan a na hraničním zařízení je VLAN ID poskytovatele z rámce odstraněno. Zákazníkovi se tak jeví jeho síť jako přímo spojená jediným spojem mezi jeho dvěmi zařízeními, které komunikují se sítí poskytovatele. Zákazníkovi budou korektně fungovat všechny protokoly jako CDP či STP. Dokonce STP bude korektně fungovat pro každý VLAN zákazníkovy sítě. Rozhraní, ze kterého přicházejí data zákazníka musí být nastaveno jako 802.1Q trunk port a rozhraní na zařízení poskytovatele musí být nastaveno do módu 802.1Q Tunneling, neboli do tunelu. Nutnou podmínkou pro funkci celého systému je použití trunk portů mezi přepínači v síti poskytovatele.

Tohoto můžeme využít při našem řešení a pomocí přepínače podporujícího technologii VLAN Tunellingu propojovat jednotlivá ethernetová rozhraní zařízení mezi sebou. Do vhodného přepínače (případně do více těchto přepínačů) zapojíme ethernetové porty všech zařízení dostupných ve Virtlabu. Poznamenáme si, které rozhraní je připojeno ke kterému rozhraní použitého přepínače. Taktéž musíme mít poznačeno, které rozhraní kterých zařízení jsou zapojeny do automatizovaného spojovacího pole. Před úlohou speciálním programem načteme potřebné propojení úlohy, zjistíme které zařízení jsou zapojeny ke kterému přepínači, případně ke kterému automatizovanému spojovacímu poli a pošleme na tyto zařízení správné konfigurační příkazy. Tímto se nám zařízení propojí do požadované topologie, jak lze vidět na obrázku 6. Vytvoření tohoto speciálního programu bylo zadáno v rámci jiného projektu a proto nebylo řešeno v rámci této diplomové práce. Na mě bylo nachystat tomuto programu soubor s topologií sítě v úloze a tento program ve vhodnou dobu zavolat. Funkce tohoto systému byla experimentálně ověřena, ale ještě není plně implementována v ostré verzi Virtlabu.



Obrázek 6: Propojení zařízení pomocí technologie VLAN Tunnelingu

8 Významné úpravy a vylepšení Virtlabu

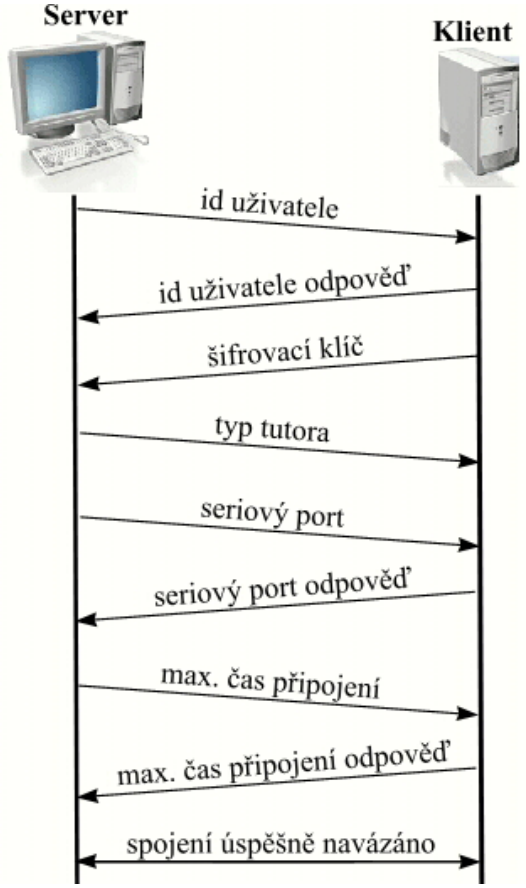
V této kapitole budou popsány významné úpravy a vylepšení, které byly provedeny na Virtlabu nad rámec původního zadání diplomové práce, případně byly nutné ke zdárnému dokončení zadání diplomové práce. Patří zde například opravy chyb, které se vyskytly v průběhu používání Virtlabu a nebyly odhaleny při jeho testování, které probíhalo v rámci diplomové práce Ing. Pavla Němce. Také se zde vyskytují úpravy, které byly provedeny v rámci vlastní iniciativy za účelem rozšíření Virtlabu o nové možnosti a zkvalitnění jeho funkce.

8.1 Komunikační protokol

Vzhledem k potřebě navázání spojení Cserveru s Java appletem a vzájemné výměně počátečních informací mezi nimi, byl k tomuto účelu vyvinut speciální protokol. Nyní, při rozšiřování Virtlabu ho bylo zapotřebí z několika důvodů změnit. Původní protokol byl zbytečně složitý, protože obsahoval hlavičku a poté až posílaná data. Tím neumožňoval snadné ladění pomocí čtení dat z odchytávaných paketů, špatně se zadávaly příkazy při ladění pomocí telnetu a podobně. Dalším důvodem jeho úpravy, zde spíše rozšířením, byla nutnost předávání více počátečních informací mezi Cserverem a appletem. Cserver musí posílat šifrovací klíč, jinak by applet nemohl dekódovat informace, které jsou mu předávány pomocí proměnných z webové stránky. Applet musí zase Cserveru poslat informaci o tom, jestli se k němu nesnaží připojit tutor.

Nyní si popíšeme celý proces navázání spojení mezi Cserverem a appletem. Jeho grafické znázornění lze nalézt na obrázku 7. V případě, že uživatel klikne na tlačítko pro připojení k zařízení, je otevřeno nové okno a do něj je nahrán Java applet. Tento applet (klient) se pokusí o připojení k Cserveru (server) a skrze něho k vybranému zařízení takto:

1. klient naváže TCP spojení se serverem,
2. klient posílá své unikátní identifikační číslo, které obdržel z webové stránky,
3. server ověří, zda toto číslo existuje v databázi a v případě kladného výsledku pošle kladnou odezvu,
4. server pošle klientu šifrovací klíč, patřící k obdrženému identifikačnímu číslu,
5. klient si tento klíč uloží a dešifruje veškeré potřebné informace,
6. klient pošle serveru informaci o tom, zde se snaží připojit tutor (případně jeho typ),
7. klient pošle serveru číslo sériového portu, ke kterému se hodlá připojit,
8. server ověří, zda má klient právo přístupu k obdrženému sériovému portu (tudíž k zařízení, které je k němu připojeno) a zda už k tomuto portu není nějaký uživatel připojen. V případě kladného výsledku, pošle klientovi kladnou odezvu a zablokuje zařízení pro klienta,
9. klient pošle serveru čas, dokdy chce se zařízením pracovat,



Obrázek 7: Komunikační protokol mezi Cserverem a Java appletem

10. server ověří, zda má klient právo tak dlouho na zařízení pracovat. V případě kladného výsledku, pošle klientovi kladnou odpověď a otevře znakové zařízení na určeném sériovém portu.

Při zasílání klíčových atributů se nejprve pošle jeden bajt (hlavička), který určuje jaký atribut za ním bude následovat. Toto se týká unikátního identifikačního čísla, sériového portu a času pro práci na zařízení. Tyto hlavičky byly vzhledem ke své jednoduchosti v komunikačním protokolu ponechány.

Dále se při komunikaci využívá speciálních kódů, pomocí nichž Cserver informuje applet o důležitých událostech. Tyto se dělí na dvě hlavní skupiny.

První je skupina událostí patřící k roli tutora. Je zde možnost informovat applet o připojení tutora typu jedna nebo dvě, případně o jeho odpojení ze zařízení.

Druhá skupina má na starosti tzv. časové události. Patří sem informování appletu o tom, že uživateli zbývá posledních deset, pět a jedna minuta práce na zařízení, případně o tom, že jeho čas pro práci na úloze již vypršel. A konečně poslední událost informuje applet o tom, že uživatel nejevil dlouhou dobu aktivitu a byl odpojen od Cserveru z tohoto

důvodu. Dobu neaktivity uživatele lze definovat v hlavičkovém souboru Cserveru a je defaultně nastavena na hodnotu patnácti minut.

Tyto speciální kódy jsou v podstatě jen jednobajtová slova. Jejich hodnota je volena s ohledem na typ přenášených dat takovým způsobem, aby se minimalizovala možnost jejich náhodného obsažení v běžně posílaných datech. Z tohoto důvodu je taktéž posílání speciálních události dvouúrovňové. Nejprve se pošle skupina události a ihned za ní typ události. V případě, že po skupině události je přijat neznámý typ události, je zpracováván jako obyčejný znak. Seznam všech speciálních kódů naleznete v příloze A.

8.2 Speciální úloha s uživatelem definovanou topologií sítě

Při používání Virtlabu se ukázalo, že zařízení jsou zbytečně nevytížena z důvodu, že uživatel si může rezervovat a poté pustit pouze úlohy, které jsou právě umístěny na nástěnce. Velice často nastává situace, že uživatel by si rád procvičil určité zapojení, ale v danou chvíli je na nástěnce pouze úloha, která toto neumožňuje. O tuto úlohu nemusí být zájem po celý den a tak jsou zařízení zbytečně nevyužita i když by mohla být využita k procvičení jiné úlohy. Z tohoto důvodu by bylo vhodné zejména pro zkušenější uživatele, vytvořit speciální typ úlohy ve které by si mohli definovat zařízení, které chtějí v úloze použít a také si zvolit propojení těchto zařízení. Vzhledem k použití automatického spojovacího pole není problém v propojení zařízení dle uživatelovy libosti. Na takto definované topologii si poté může uživatel provádět vlastní experimenty.

Při implementaci musela být vytvořena nová tabulka v databázi, obsahující pro každou rezervovanou speciální úlohu popis této úlohy. Je v ní uloženo jednoznačné id této úlohy, seznam zařízení, která budou v úloze používána, jejich propojení a podobně. Po ukončení úlohy již nejsou tyto data více zapotřebí a proto je vhodné jednou za čas z této tabulky vymazat již nepotřebné záznamy. V opačném případě, by se tabulka s postupem času zvětšovala a mohla značně zpomalovat chod systému. Uživatel si vybírá zařízení použitá v úloze a jejich vzájemné propojení ihned po rezervaci úlohy. Případně může tyto parametry měnit, nejpozději však pět minut před začátkem úlohy.

8.3 Mazání zařízení před začátkem úlohy

Před začátkem nové úlohy je potřeba vymazat všechna zařízení, které se v ní budou používat a nastavit je do výchozího stavu. Toto se provádí sérií tzv. mazacích příkazů, které má každé zařízení definováno v databázi. O mazání zařízení se stará pomocný program nazvaný *Check Timeslot*. Původně se zařízení mazala postupně a nebylo možno mezi jednotlivými příkazy posílanými do zařízení počkat určenou dobu. Mazání zařízení se provádí pět minut před začátkem úlohy a vzhledem k tomu, že restart jednoho zařízení může trvat až dvě minuty, nebylo možno zajistit smazání všech zařízení před začátkem úlohy. Proto bylo zapotřebí mazat zařízení paralelně.

Po zvážení všech možností jsem se rozhodl pro využití vláken, kdy pro každé zařízení patřící k úloze je v programu vytvořeno vlastní vlákno, které se postará o jeho smazání. Vlákna pracují paralelně a nezávisle na sobě.

Ještě stále ale zůstává problém při zadávání příkazů do zařízení, potřebných pro jeho vymazání. Některé příkazy zařízení vykonává určitou dobu a po tuto dobu nepřijímá žádné další příkazy. V původní verzi nebylo proto možno některá zařízení korektně smazat. Obdobně je to s kombinací kláves CTRL+Z, pomocí které se uživatel dostane do privilegovaného módu v IOSu zařízení a který je taktéž využit při mazání zařízení. Z tohoto důvodu byly pro mazání zařízení implementovány dva speciální příkazy. Prvním z jich je příkaz $\{\wedge Z\}$, který pošle na zařízení kombinaci kláves CTRL+Z. A umožní tak skok do privilegovaného módu v IOSu zařízení. Druhým příkazem je $\{WAIT\ xx\}$, který zastaví vykonávání dalších příkazů pro mazání zařízení po dobu 1-99 vteřin.

Po provedených úpravách a vytvoření vhodných mazacích příkazů již není problém s vymazáním dostupných zařízení a jejich mazání probíhá spolehlivě. Jediný problém může nastat, pokud se zařízení dostane do nestandardního stavu (například do některé položky v menu přepínače), pro který není možno vytvořit vhodný mazací příkaz. Jediné možné řešení je zakoupení nebo výroba speciálního HW zařízení, které by naše zařízení resetovalo hardwarově.

8.4 Ostatní úpravy a vylepšení

V této části popíši minoritní úpravy Virtlabu, o nichž by ovšem mělo být zmíněno v této diplomové práci.

Byla značně přepracována část webové aplikace, zodpovědná za správu uživatelů. Každého uživatele je nyní možno přiřadit do určité skupiny uživatelů a lze mu nastavit čas pro expiraci konta. Tyto informace jsou uloženy do databáze, ale nikde se s nimi dále nepracuje. Jsou zde pouze jako příprava pro budoucí rozšiřování Virtlabu, viz. kapitola 9.1. Značně byl vylepšen systém kvót uživatelů. Původně byla maximální kvóta pro všechny uživatelé stejná a v databázi byla pro každého uživatele uložena pouze jeho aktuální velikost kvóty. Nyní je v databázi jak aktuální kvóta daného uživatele, tak i jeho maximální výše kvóty. Ta se využívá při automatickém obnovování kvót. Takže každému uživateli je samostatně obnovena kvóta na pro něj maximální velikost. Administrátor uživatelů má nyní možnost nastavit uživateli heslo, případně nechat heslo uživatele autentizovat přes LDAP.

Uživatelé si stěžovali na pomalost Java appletu při opakovaném připojení k zařízení a proto byly některé jeho části přepsány, což vedlo k výraznému zvýšení rychlosti appletu. Applet měl taktéž problémy se zobrazováním znaků a stávalo se, že některé znaky nebyly zobrazeny korektně. Po dlouhém zkoumání příčiny tohoto problému jsem dospěl k závěru, že problém je na straně Javy, konkrétně v implementaci knihovny Swing pro práci s uživatelským rozhraním. V případě příliš rychlého posílání znaků pro zobrazování se začínají znaky zobrazovat nekorektně. Problém byl vyřešen přidáním malé pauzy po každém zobrazeném znaku. Další úpravou appletu je implementace schránky, která se chová jako schránka ve Windows a jiných operačních systémech a je s jejich schránkou svázána. Označený text z okna appletu je možno pomocí CTRL+C zkopírovat do schránky a poté kdekoliv vložit. Naopak, stiknutím CTRL+V se text ze schránky vloží do okna appletu a pošle na zařízení. Další možností jak si ukládat komunikaci se zařízením, je použití funkce capture. Tato funkce po jejím zapnutí z menu appletu zaznamenává

veškerou komunikaci se zařízením do souboru až do doby, kdy je uživatelem vypnuta, nebo se uživatel odpojí od zařízení.

Do Cserveru byla přidána možnost vypnutí logování aktivity uživatelů (provede se parametrem -n při spouštění). V takovém případě se loguje pouze jejich přihlášení a odhlášení ze serveru. Toto nastavení je vhodné zejména s ohledem na velikost logovacího souboru, který při větším počtu uživatelů velice rychle zvyšuje svou velikost. Při normálním nastavení se totiž loguje veškerá komunikace uživatelů se zařízeními.

9 Závěr

V práci jsem se zaměřil především na komplexní vypracování bezpečnosti virtuální síťové laboratoře s ohledem na její jednotlivé komponenty a komunikaci mezi nimi. Nejprve jsem teoreticky rozebral jednotlivé komponenty a definoval bezpečnostní rizika, které představují. Poté jsem uvedl typické příklady útoků a navrhl vhodnou ochranu, kterou jsem prakticky implementoval do stávajícího systému. Dalším neméně důležitým úkolem byla implementace nových funkcí do stávajícího systému, včetně analýzy, která implementaci předcházela a ukázala na nutnost změny některých částí stávajícího systému. Součástí práce bylo i zamyšlení nad možností automatizovaného spojování ethernetových rozhraní laboratorních zařízení pomocí přepínače, podporujícího technologii VLAN Tunneling (dot1QinQ), kterým je například přepínač Cisco 3550. Přepracoval jsem i uživatelské rozhraní u některých akcí, prováděných uživateli s ohledem na jejich ergonomičnost. Celkově je nyní celý systém virtuální síťové laboratoře mnohem lépe použitelný pro nasazení v opravdových podmínkách a může být bez vážnějších obav jak o bezpečnost, tak i o funkčnost používán k výuce studentů. Přesto se určitě najdou nějaké chybičky, či návrhy na vylepšení, což je u takto rozsáhlého projektu, jakým Virtlab bezpochyby je naprosto očekávané. Návrhy na další vylepšení a úpravy najdete k kapitole 9.1.

9.1 Vývoj Virtlabu v budoucnosti

Vzhledem k tomu, že Virtlab se neustále vyvíjí a rodí se nápady na jeho vylepšení, je vhodné se zmínit o možnostech jeho rozšiřování v budoucnu.

Vylepšení by se určitě daly provést v roli tutora, kdy sice může tutor převzít konzoli zařízení od uživatele a ukázat mu správný postup práce, ale nemají žádnou možnost vzájemné komunikace. Mohl by se implementovat systém, pomocí kterého by si mohli uživatelé v případě potřeby zavolat tutora na pomoc (alespoň v předem stanovených hodinách nebo úlohách). Taktéž by mohla být implementována vzájemná komunikace mezi uživateli přihlášenými v systému, případně spolupracujícími na úloze.

Nynější systém posílání emailů zapojovačům úloh, kdy se email zašle všech uživatelům s rolí zapojovače úloh by bylo vhodné doplnit o možnost přiřazení času ke každému zapojovači. Tento čas by určoval, ve kterých dnech a hodinách je k dispozici pro zapojování úloh a emaily s žádosti o zapojení úlohy by se posílaly pouze aktivním zapojovačům v době začátku úlohy (či těsně před ní). Dále by mohlo být kontrolováno, zda již není správná topologie zapojena. Toto se může zjistit například tím, že předcházející úloha využívala stejné topologie, nebo dokonce byla totožná.

S narůstajícím počtem uživatelů Virtlabu bude jistě zapotřebí zefektivnit jejich správu. První kroky jako přidání skupin uživatelů a expirace konta do databáze již byly provedeny, ale nejsou v systému patřičně implementovány. Toto je oblast, která si jistě zaslouží pozornost při dalším vylepšování.

Směr, kterým by se mohl Virtlab do budoucna ubírat, naznačuje speciální úloha (viz. kapitola 8.2), kdy si může uživatel sám zvolit topologii úlohy a vybrat si prvky, které bude v úloze využívat. V současném stavu je možno dát na nástěnku i více úloh ve stejném čase, ale zařízení použitá v úlohách se nesmějí překrývat. Takže uživatel si může vybrat

jen z omezeného počtu úloh v omezeném čase. Nic ale nebrání tomu, aby si uživatel zvolil úlohu na které chce pracovat sám a taktéž čas, ve kterém na ní chce pracovat. Samozřejmě již nesmí být ve zvoleném čase rezervovány zařízení potřebná k běhu úlohy. Pokročilejší studenti by měli možnost nejen vybírat z předem připravených úloh, ale rovnou si vybrat zařízení, která chtějí používat a zvolit si jejich vzájemná propojení.

Roman Kubín

10 Literatura

[1] Pravir Chandra, Matt Messier, John Viega, *Network Security with OpenSSL*, O'Reilly, June 2002.

[2] Matt Messier, John Viega, *Secure Programming Cookbook for C and C++*, O'Reilly, July 2003.

[3] Matthew N., Stones R., *Linux programujeme profesionálně*, Praha : Computer Press 2001.

[4] Hawlitzek F., *Java 2, příručka programátora*, Praha : Grada Publishing spol. s.r.o. 2002.

[5] Simson Garfinkel, Gene Spafford, *Practical UNIX & Internet Security*, O'Reilly, April 1996

[6] Sun Microsystems, *JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification*, [online].[cit.2006-05-08] dostupné na www: <http://java.sun.com/j2se/1.4.2/docs/api/>

[7] Dave Clark, *PHP Security Mistakes*, [online].[cit.2006-05-08] dostupné na www:<http://www.devshed.com/c/a/PHP/PHP-Security-Mistakes/>

[8] Dave Child, *Writing Secure PHP*, [online].[cit.2006-05-08] dostupné na www: <http://www.ilovejackdaniels.com/php/writing-secure-php/>

[9] Sun Microsystems, *JavaTM Secure Socket Extension (JSSE) Reference Guide*, [online].[cit.2006-05-08] dostupné na www: <http://java.sun.com/j2se/1.4.2/docs/guide/security/jsse/JSSERefGuide.html>

[10] Cisco Systems, *Configuring 802.1Q and Layer 2 Protocol Tunneling*, [online].[cit.2006-05-08] dostupné na www: <http://www.cisco.com/en/US/products/hw/switches/ps646/products_ configuration_guide_chapter09186a00801cdf50.html>

A Speciální kódy komunikačního protokolu

V tabulce 1 jsou uvedeny speciální kódy komunikačního protokolu, které používá Cserver s appletem pro vzájemné vyměňování důležitých zpráv.

kód	hodnota	popis
SI	′1′	unikátní id uživatele
OK_SI	′4′	potvrzení přijetí id uživatele
MAX_LEASE_TIME	′3′	čas ukončení
OK_MAX_LEASE_TIME	′6′	potvrzení přijetí času ukončení
COM_PORT	′2′	číslo konzoly
OK_COM_PORT	′5′	potvrzení přijetí čísla konzoly
DISCONNECT_EVENT	7	následuje událost typu disconnect
TUTOR_EVENT	8	následuje událost typu tutor
DISCONNECT_NOACTIVITY	17	odpojení - neaktivita
TIME_EXCEEDED	18	odpojení - vypršení času
DISCONNECT_AFTER_10_MIN	19	ukončení práce za 10 min
DISCONNECT_AFTER_5_MIN	20	ukončení práce za 5 min
DISCONNECT_AFTER_1_MIN	21	ukončení práce za 1 min
TUTOR_CONNECT	23	tutor se pripojil - typ 1
TUTOR_DISCONNECT	24	tutor se odpojil
TUTOR_CONNECT_VISIBLE	25	tutor se připojil - typ2

Tabulka 1: Kódy komunikačního protokolu