VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

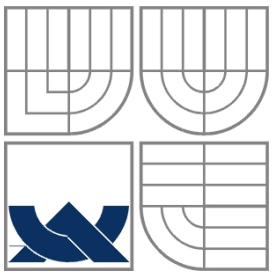
KOMUNIKACE S KAMEROU PROSTŘEDNICTVÍM HTTP KANÁLU

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

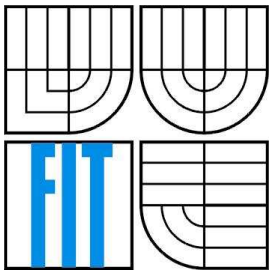
AUTOR PRÁCE
AUTHOR

MICHAL KABELKA

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KOMUNIKACE S KAMEROU PROSTŘEDNICTVÍM HTTP KANÁLU

HTTP TUNNELING OF CAMERA IMAGE DATA TRANSMISSION

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — MICHAL KABELKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ RYŠAVÝ, Ph.D.
SUPERVISOR

BRNO 2008

Abstrakt

Táto práca sa snaží popísať postup návrhu a implementácie softwaru pre komunikačné rozhranie kamery so serverom vyvíjaného pre spoločnosť wasatis. V prvej časti sa zaoberá popisom problematiky a snahou nájsť riešenia daných problémov. V ďalšej časti sa zaoberá komunikáciou prostredníctvom rozhrania nazvaného Call Back Connection. V ďalších častiach popisuje postup implementácie a princíp konfigurácie aplikácii. V záverečnej časti rieši bezpečnostné otázky komunikácie.

Kľúčové slová

Linux, Fedora, CGI, Wasatis, Apache, NAT, Firewall, SHA-1, kamera, zabezpečenie

Abstract

This thesis is trying to describe design and implementation of software for communication interface of camera with server developed for company Wasatis. The first part is describing difficulties and effort to finding solutions to given problems. The next part is about communication through interface called Call Back Connection. In next few parts it is describing process of implementation and principle of configuration of applications. In final part it solves security questions of communication.

Keywords

Linux, Fedora, CGI, Wasatis, Apache, NAT, Firewall, SHA-1, video camera, home security

Citácia

Michal Kabelka: Komunikace s kamerou prostřednictvím http kanálu, bakalárska práca, Brno, FIT VUT v Brne. 2008

Komunikace s kamerou prostřednictvím http kanálu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Ondřeje Ryšavého, PH.D.
Uviedol som všetky pramene a publikácie, z ktorých som čerpal.

……
Michal Kabelka
14. mája 2008

Obsah

1 Úvod

Zmyslom tejto bakalárskej práce je vytvorenie komunikačného rozhrania medzi kamerami a serverom pre spoločnosť Wasatis. Wasatis je spoločnosť založená skupinou ambicióznych ľudí v Paríži vo Francúzku, zaoberajúcich sa zabezpečovacími a kamerovými systémami. Navrhovaný systém vychádza z už existujúceho systému spoločnosti Axis. Použitie už vyvinutého softwaru Axis by bolo možné, avšak spoločnosť Axis vyžaduje využitie ich vlastných serverov, čo sa prieči predstavám spoločnosti Wasatis. Z tohto dôvodu sa rozhodli vyvinúť si vlastný software. Aplikácia vyvíjaná v rámci tejto bakalárskej práce je len malou častou celkového softwaru. Návrh rozhrania bol konzultovaný s predstaviteľmi spoločnosti Wasatis a riešený s dôrazom na potreby zabezpečovacích systémov.

V prvej časti sa venujem načrtnutiu celkovej problematiky a vyčleneniu konkrétnej časti určenej pre riešenie v mojej práci. Diskutovaná je voľba hardware a tiež správny výber softwarového vybavenia a jeho konfigurácia pre celkovú funkčnosť. Vo veľkej miere preberám problematiku a princípy firewallu a prekladu adries (NAT) a hlavne ich negatívneho pôsobenia na riešený problém. V celej časti sa tiež snažím načrtnúť teoretické aj praktické východiská pre riešenia problému spôsobeného práve spomínanými službami.

V druhej časti je preberaný komunikačný kanál medzi kamerou a serverom. Veľkú časť zaberá problematika rozhrania CGI, ktorá je využívaná ako v komunikačnom tak aj prenosovom kanáli. Samozrejme načrtnuté sú aj bezpečnostné problémy, ktoré sebou použitie rozhrania CGI prináša. Vo veľkej miere tam nájdete vysvetlenie Call Back Connection, čo je základný princíp komunikácie obchádzajúcej firewall a NAT, za použitia tunelovania http protokolu na porte 80 a spojenia inicializovaného kamerou a nie serverom.

Tretia časť vysvetľuje spôsoby konfigurácie serverovej aj klientskej aplikácie. Popisuje parametre konfiguračných súborov, funkcie pre ich spracovanie a v neposlednej rade štruktúry v ktorých sú načítané parametre uchovávané v bežiacich aplikáciách.

Posledná časť sa venuje dôležitej otázke bezpečnosti. Kamera a server navzájom autentizujú jeden druhého. Využívajú k tomu hashovacie algoritmy ako SHA1 za účasti náhodne generovaných hodnôt snonce a cnonce pre zníženie rizika napadnutia systému.

V závere sa snažím zhodnotiť svoj prínos pri realizácii celého projektu, zhodnotiť ďalší vývoj ako mnou realizovanej časti tak aj celého projektu a navrhnúť zlepšenia a zhrnúť, čo mi práca na bakalárskej práci dala.

2 Súčasný stav projektu, hardware a voľba software

2.1 Popis projektu Wasatis

2.1.1 Spoločnosť Wasatis

Wasatis je začínajúca spoločnosť založená skupinou ambicióznych ľudí vo Francúzku, zaoberajúca sa kamerovými systémami. Ich súčasný projekt vychádza z už existujúceho systému spoločnosti AXIS nazývaného STS (Subscriber Technology System). Je to systém dovoľujúci sledovanie a ukladanie videa z ktoréhokoľvek počítača pripojeného do siete internet, cez web rozhranie. Hlavnou myšlienkou je ponúknuť systém širokej a teda aj laickej verejnosti, z čoho vyplýva snaha spoločnosti vyvinúť systém schopný nulovej konfigurácie, teda konfigurácie nevyžadujúcej interakciu užívateľa.

2.1.2 Systém Wasatis STS

Systém Wasatis STS predstavuje komplexný projekt, zložený z mnohých častí. Kým časť projektu pojednávaná v tejto práci predstavuje komunikačnú infraštruktúru medzi serverom spoločnosti Wasatis a klientskými kamerami, iné, podstatne zložitejšie časti sú venované hlavne téme spracovania obrazu. Revolučná myšlienka totižto spočíva v spracovaní obrazu na strane serveru. To znamená, že kamera neustále posiela zachytené snímky na server, v intervale napríklad jednej snímky za sekundu. Plnohodnotné video nemôže byť neustále prijímané serverom, to by mohlo spôsobovať zahlcovanie prenosovej linky a následné znižovanie kvality poskytovaných služieb. Server potom prevádza analýzu prijatých snímok a v prípade zistenia neočakávanej udalosti, napríklad narušenia priestoru, požiada kameru o zvýšenie rýchlosti prenosu, prípadne upovedomí užívateľa o vzniknutej udalosti. Možností, aké udalosti sa dajú zisťovať pri spracovaní prijatého videa je nekonečne veľa, od narušenia priestoru až po sledovanie pohybu starých ľudí. Témou tejto práce však ostáva, len samotný komunikačný a prenosový kanál medzi serverom a kamerami.

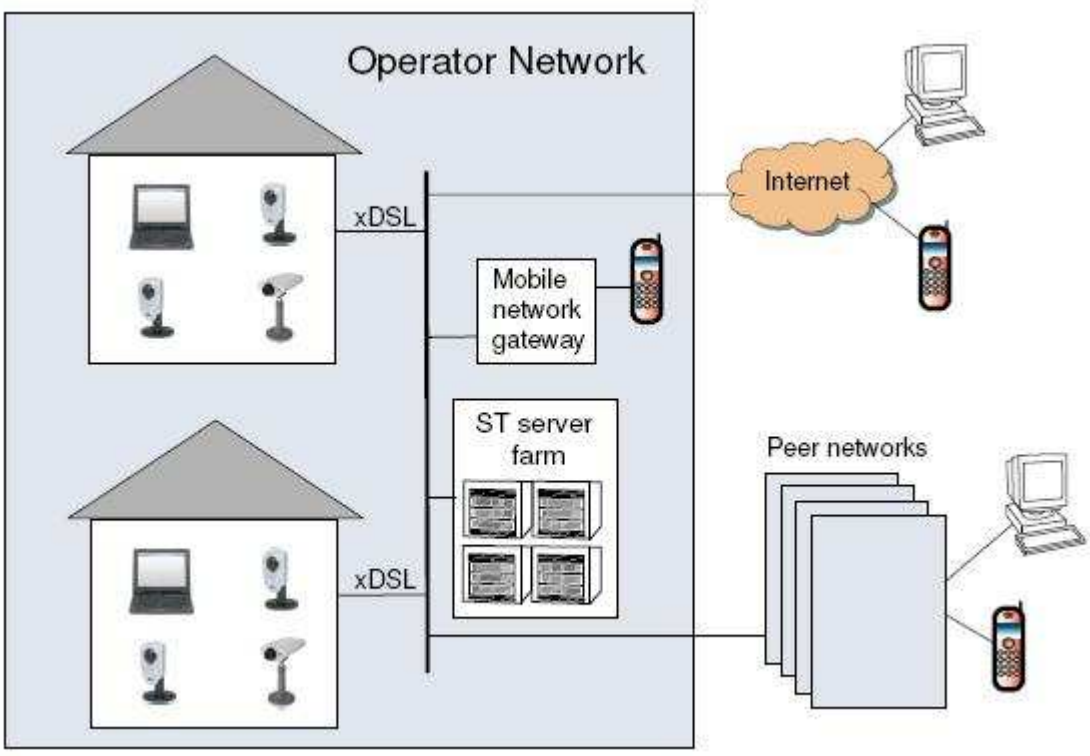
2.1.3 Neznalý užívateľ

Drvivá väčšina súčasných, existujúcich kamerových systémov vyžaduje aspoň minimálnu konfiguráciu pri pripojení do internetu. Nakoľko prakticky každá sieť sa nachádza za firewall-om, je

nutné práve do jeho konfigurácie pridať pravidlá povoľujúce komunikáciu na viacerých portoch, ktoré daný protokol vyžaduje. Podobný úkon môže pre neznalého človeka predstavovať neriešiteľný problém.

Ďalší problém sa naskytá v súvislosti s prekladom adries (NAT) a teda neverejnou klientskou IP adresou. Neverejná IP adresa znemožňuje naviazanie spojenia zo strany serveru. Iniciátorom spojenia sa teda musí stať klient (kamera).

Všetky hore uvedené problémy sú diskutované v nasledujúcich kapitolách, v ktorých sa snažím navrhnúť riešenia, prípadne načrtnúť ďalšie možné alternatívy. Zvolené riešenia sú neskôr implementované do výsledného programu, ktorý sa pričlení do celého komplexného projektu spoločnosti Wasatis.



Obrázok 2.1: Schéma systému Wasatis STS

2.2 Hardware

Wasatis používa kamery značky Axis rady 206 a vyššie. Kamery využívajú linuxové jadro 2.4, alebo 2.6 s uClibc a dash [3]. Z tohto dôvodu bolo možné celý systém prototypovať na počítači so štandardnou distribúciou linuxu.

Ako testovací server som používal vlastný stolný pc s procesorom AMD Athlon 64 X2 5200+ s dvomi jadrami (bežiaci na 3,03 Ghz), operačnou pamäťou 2024 MB DDR2 RAM a sieťové karty

firmy nvidia. Myslím, že daný počítač disponuje viac než dostatočným výkonom na počiatočné testovanie i prípadné neskoršie rozšírenia.

2.3 Firewall a NAT

Pri navrhovaní projektu bolo hlavnou myšlienkou, aby zákazník, ktorý si zakúpi kameru od spoločnosti Wasatis, nemusel nič nastavovať. Jediné, čo potrebuje urobiť bude zapojiť kameru do elektrickej siete a pripojiť ju k internetu pomocou LAN, WLAN, atd. Kamera sa sama pripojí k serveru, registruje sa a bude okamžite prístupná pre užívateľa cez web rozhranie.

2.3.1 Firewall

Veľká výzva spočívala v spôsobe ako prejsť cez firewall bez nutnosti otvárať porty. Pri prenose videa pomocou protokolu RTP, ktorý využíva premenné porty z rozsahu 16384-32767 [5], by vznikal ešte väčší problém. Bolo by zbytočné otvárať jeden, konkrétny port, keďže pri ďalšom spojení by bol pravdepodobne použitý iný.

Z tohto pohľadu sa núkala možnosť použiť pre všetku komunikáciu a prenos samotného videa medzi serverom a kamerami jeden port, najlepšie taký, ktorý je na všetkých firewalloch otvorený. A tým je port 80 (aspoň pre odchádzajúce spojenie), využívajúci protokol http. Preto sa ako najelegantnejší spôsob ako obísť firewall ukazuje možnosť tunelovať celú komunikáciu do tela http paketu.

Teoreticky by bolo možné používať port 80, bez nutnosti využívať protokol http a teda netunelovať, len priamo posielať v požadovanom protokole na port 80. Niektoré moderné firewally, však skúmajú aj obsah paketu. V prípade portu 80 a http protokolu skúmajú správnosť hlavičky paketu a v prípade, že odhalia, že požiadavka nie je požiadavkou na www server, je tento paket zahodený a teda firewallom neprejde. Existujú rôzne druhy firewallov a teda rôzne druhy filtrovania [6]

Paketové filtre

Jedná sa o najjednoduchšiu a najstaršiu formu firewallu, ktorá spočíva v tom, že pravidlá presne uvádzajú, z akej adresy a portu na akú adresu a port môže byť doručený prechádzajúci paket. Kontrola sa prevádza na tretej a štvrtej vrstve modelu ISO/OSI.

Aplikačné brány

Niekedy označované tiež ako proxy firewally, celkom oddeľujú siete medzi ktoré boli postavené. Všetka komunikácia cez aplikačnú bránu prebieha formou dvoch spojení – klient (iniciátor spojení) sa

pripojí na aplikačnú bránu (proxy), tá prichádzajúce spojenie spracuje a na základe požiadavku klienta otvorí nové spojenie k serveru, kde je klientom aplikačná brána. Dáta, ktoré aplikačná brána dostane od serveru potom zase v pôvodnom spojení predá klientovi. Kontrola sa prevádza na siedmej (aplikačnej) vrstve modelu ISO/OSI.

Stavové paketové filtre

Prevádzajú kontrolu rovnako ako jednoduché paketové filtre, naviac si však ukladajú informácie o povolených spojeniach, ktoré potom môžu využiť pri rozhodovaní, či prechádzajúce pakety patria do už povoleného spojenia a môžu byť prepustené, alebo musia prejsť znovu rozhodovacím procesom.

Stavové paketové filtre s kontrolou známych protokolov

Tieto druhy firewallov sú schopné kontrolovať prechádzajúce spojenie až na úroveň korektnosti prechádzajúcich dát známych protokolov a aplikácii. Môžu tak napríklad zakázať priechod http spojenia, v ktorom sa objavia indikátory, že sa nejedná o požiadavku na www server, ale tunelovanie iného protokolu.

Práve kvôli dokonalej kontrole prechádzajúcich paketov niektorými novými firewallmi a teda nutnosti vytvárať http hlavičku, ktorá by splňovala požiadavky RFC, vychádzala ako najlepšia a najschodnejšia varianta, použitie http serveru a rozhrania CGI (detailný popis rozhrania CGI nájdete neskôr v texte) na strane wasatis serveru. Klientska aplikácia na strane kamery si vytváranie hlavičky a jej spracovanie zabezpečuje sama na úrovní vyvíjaného softwaru.

2.3.2 NAT/PAT

NAT a PAT sú všeobecne zamieňané pojmy. PAT (port address translation) slúžiaci pre preklad portov, je oficiálne označovaný ako symetrický NAT (network address translation) a v Linuxovej komunite ako „maškaráda“ (MASQUERADE). Preklad je väčšinou prevádzaný technikou Full cone NAT – v texte ďalej prezívanej ako PAT – definovanej v RFC 3489 [8]. Tento dokument definuje štyri typy NAT a to: „Full cone NAT“, „restricted cone NAT“, „port restricted NAT“ a „symmetric NAT“.

Toto rozdelenie je ale diskutabilné. Bolo definované v dobe, keď existovalo niekoľko implementácií NAT/PAT, ktoré sa v rôznych ohľadoch líšili a táto štandardizácia nespôsobila, že by sa k sebe priblížili. Druhým dôvodom je, že toto vymedzenie vzniklo, až v pri príprave protokolu STUN, ktorý mal umožniť zistenie existencie a typu NATu a Firewallu medzi aplikáciou

a internetom. Dá sa predpokladať, že dodávatelia sieťovej infraštruktúry, ktorí nepodporujú STUN, ani neparticipujú na IETF toto rozdelení NATov nedodržujú.

PAT úzko súvisí s NAT – definuje statické, alebo dynamické prekladanie vnútorných adries v sieti na adresy vonkajšie, ktoré majú prístup ďalej do inej časti siete, alebo do internetu.

Pre príklad: na vnútornom rozhraní je sieť 192.168.1.0/24 a na vonkajšom 10.10.10.0/8. Pokiaľ paket smeruje z vnútornej siete von, zmení sa zdrojová adresa v hlavičke paketu z 192.168.1.X na 10.10.10.Y. Pokiaľ cieľové zariadenie zachytí tento paket, odpovie na adresu 10.10.10.Y, paket príde na smerovač, ktorý zabezpečil preklad, a ten podľa NAT tabuľky distribuuje paket ďalej do vnútornej siete. Nikto okrem smerovača nevie, kto je v skutočnosti odosielateľom dát – je to možné brať ako istú formu sťaženia práce útočníkovi, ale v žiadnom prípade nie ako bezpečnostná (security) záležitosť, či ochranu pred útokom [1].

NAT je možné požiť len v prípade, že máme dostatok verejných IP adries, na ktoré je treba tie neverejné prekladať.

PAT funguje rovnako ako NAT a naviac prekladá porty. Počítač z lokálnej siete odosiela paket, so svojou zdrojovou adresou a portom, ktorý príde na smerovač, ten priradí novo vzniknutému spojeniu číslo portu z množiny dostupných portov a vloží tento port do poľa zdrojová adresa hlavičky paketu. Smerovač potom odošle paket na výstupné rozhranie.

Smerovač si vytvorí záznam v prekladovej tabuľke, ktorý obsahuje

- vnútornú IP adresu,
- vnútorný zdrojový port a
- vonkajší port

Nasledujúce pakety v spojení sú už priradzované k tomuto záznamu.

Vzdialený počítač vloží svoju zdrojovú adresu a port do polí v hlavičke pre cieľovú adresu a cieľový port. Smerovač v lokálnej sieti už zaistí preklad hlavičky na údaje ďalej identifikujúce cestu ku klientovi.

PAT : nevýhody

1. So zariadením za prekladom portov nie je možné vytvoriť priame spojenie, ktoré je nutné napríklad pre FTP, SIP, H.323. Musí sa používať tzv. connection tracking moduly v jadre [9].
2. Protokol UDP je možné pretunelovať pomocou protokolu STUN (aplikácia chownat), ale to nie je možné u najčastejšie používaného symetrického PAT/NAT.
3. PAT porušuje pravidlo, na ktorom je postavený internet: Každé zariadenie má možnosť sa spojiť s akýmkoľvek iným zariadením.
4. PAT, a NAT obecne, nefunguje s IPSec, pretože NAT modifikuje hlavičku, čo nejde dohromady s kontrolami prevádzanými IPSecom. Iné tunelovacie protokoly sa stretávajú s podobným problémom.

5. PAT spomaľuje nástup IPv6, pretože z pohľadu užívateľa internetu (teda predovšetkým webu) ich pripojenie cez PAT neobmedzuje, a ne druhú stranu internet novej generácie neponúka nič, čo by v IPv6 internete nebolo.

PAT respektíve NAT (ďalej v texte označované spoločne ako NAT) má svoje nepochybné výhody, hlavne v dobe nedostatku voľných verejných IPv4 adries. Pri riešenom projekte, však NAT predstavuje problém, alebo aspoň sťaženie riešenia komunikácie medzi serverom a kamerami. Hlavný problém predstavuje nevýhoda spomínaná v bode tri a teda nemožnosť spojiť sa s akýmkoľvek zariadením, v našom prípade nemožnosť iniciovať spojenie so strany serveru pokiaľ je klient umiestnený za NATom. V prípade, že by sa klient pripojil na webové rozhranie serveru a požadoval by vidieť video so svojej kamery, server by nebol schopný spojiť sa s kamerou a predať jej príkazy k začatiu prenosu. Z tohto pohľadu sa ponúka jediná možnosť a teda, že spojenie začne kamera. Pri pripojení kamery do internetu sa táto pripojí na sever a vytvorí sa komunikačný kanál, ktorý je udržovaný po celú dobu pripojenia kamery. Týmto kanálom potom server posiela klientskej kamere príkazy. Takýto princíp komunikácie sa označuje ako Call Back Connection (CBC). Detaily o CBC a princípy spojenia kamery so serverom sú popísané v kapitole 3.2.

2.4 Voľba distribúcie operačného systému

Pretože súčasný server spoločnosti Wasatis používa Gentoo Linux, čo je meta distribúcia založená na kompilovaní aplikácií podľa daných predpisov [2], a sám som ho pred nedávnom používal, zvažoval som inštaláciu a použitie práve tohto systému. Keďže ho už ale nemám nainštalovaný a inštalácia Gentoo a jeho následné nakonfigurovanie nie je záležitosť na pár minút, dokonca ani pár hodín, rozhodol som sa siahnuť k inej obecnej distribúcii. Voľba padla na systém, ktorý už mám nainštalovaný na laptope a s ktorým už mám skúsenosti a teda Fedora. Mnohí by so mnou asi nesúhlasili pri výbere a použití Fedory ako operačného systému pre server, ale vzhľadom k tomu, že „môj server“ bude využívaný len ako vývojový a testovací, myslím, že použitie onoho systému nebude nijak na škodu.

Fedora je operačný systém pre všeobecné použitie, založený predovšetkým na slobodnom a otvorenom software. Je založená na balíčkovacom systéme RPM. Primárne je určený pre použitie na domácich počítačoch, avšak umožňuje využitie aj na serveroch. Je vyvíjaný komunitou vývojárov okolo Fedora Project, sponzorovaného spoločnosťou Red Hat. Fedora sa vyskytuje v niekoľkých verziách, v dobe písania tejto práce sú to tieto [4]

- Fedora 8 – súčasná stabilná verzia – udržovaná
- Fedora 7 – predchádzajúca stabilná verzia – udržovaná
- Fedora Core 6 – predchádzajúca stabilná verzia – posledná udržovaná

- Fedora 9 – testovacia/budúca stabilná verzia

Všetky staršie verzie počínajúc Fedora Core 1 a končiac Fedora Core 5 už nie sú udržované a teda ich distribúcia je ukončená. Zatiaľ posledná stabilná verzia Fedora 8 vyšla 8. novembra 2007. Práve túto verziu som si zvolil ako operačný systém pre testovací server.

2.4.1 Inštalácia Fedora 8

Inštalácia Fedory nie je rozhodne nič zložité a dá sa zvládnuť v priebehu asi hodiny. Pre inštaláciu som si stiahol inštalačný súbor Fedora-8-x86_64-DVD.iso zo stránky download.fedoraproject.org. Jedná sa o verziu pre 64bitové procesory. Ako súborový systém som použil základný ext3fs a ako balíčkovú výbavu zvolil webový server.

Po dokončení inštalácie prišlo na rad skontrolovanie nastavenia siete pomocou

```
/sbin/ifconfig eth0
```

a spustenie dhcp klienta pre prijatie nastavení od dhcp serveru

```
/sbin/dhclient
```

Ďalším krokom bolo nainštalovanie balíčka livna-release-8.rpm pre povolenie prístupu do repozitára livna cez balíčkový manažér yum. Ako posledný krok inštalácie prišiel na rad update systému a nainštalovaných balíčkov pomocou príkazu

```
yum update
```

Teraz je už systém nainštalovaný a plne aktualizovaný, nasleduje nainštalovanie ďalšieho potrebného softwaru, hlavne web serveru.

2.4.2 Výber web serveru

Do úvahy prichádzalo viacero možností komunikácie medzi kamerami a serverom. Ako najlepší spôsob pre dané použitie sa ukázalo použitie web serveru bežiaceho na firemnom servery a použitie rozhrania CGI. Rozhranie CGI je podrobne popísané v kapitole 3.1.

Existuje viacero druhov web serverov, medzi najznámejšie a najpoužívanejšie patria [11]

Lighttpd

Jedná sa o web server optimalizovaný pre prostredia, kde je dôležitá rýchlosť, napriek tomu je bezpečný, rýchly a splňuje všetky štandardy. Jeho základným znakom je nízka náročnosť na pamäť, malé zaťaženie procesora a jeho zameranie na rýchlosť ho robí ako stvoreným pre servery trpiace problémami pri veľkom zaťažení, alebo pre oddelené obsluhovanie statických a dynamických dát. Lighttpd je slobodný a otvorený software, šírený pod licenciou BSD. Podporuje FastCGI, SCGI

a CGI rozhranie pre externé programy napísané v ľubovoľnom programovacom jazyku. Zvláštnu pozornosť venovali vývojári podpore PHP [10].

Internet Information Services

IIS, formálne tiež prezívaný ako Internet Information Server, je súbor internetových služieb pre servery používajúce Microsoft Windows. Je to druhý najpopulárnejší webový server, podľa počtu navštívených stránok. Server v súčasnosti zahŕňa FTP, SMTP, NNTP, a HTTP/HTTPS [12].

Apache

Apache je web server, ktorý zohral kľúčovú rolu pri počiatočnom raste internetu. Bola to prvá konkurencie schopná alternatíva k web serveru spoločnosti Netscape Communications Corporation (dnes známeho ako Sun Java System Web Server). Od tých čias sa vyvíjal a stal sa schopným konkurovať ostatným Unix-based web serverom v otázkach funkcionality a výkonu. Hovorí sa, že meno Apache tvorcovia zvolili z dvoch dôvodov. Z úcty k domorodému indiánskemu kmeňu Apačov, dobre známych pre ich vytrvalosť a umenie v boji a kvôli základom projektu ako súbore patchov (záplat) na základný kód NCSA HTTPd 1.3, teda „A patchy server“ (záplatový server) [13].

Vývoj Apache sa začal v roku 1993 v NCSA (National Center for Supercomputing Aplications) na Illinoiské univerzite. Pôvodný názov projektu bol NCSA HTTPd. V ďalšom roku však vývojársky tím opustil hlavný programátor, tým došlo k spomaleniu vývoja, až v roku 1998 k úplnému zastaveniu. NCSA HTTPd už medzi tým používali správcovia webových serverov a dodávali k nemu vlastné patche. Hlavnú úlohu v ďalšom vývoji zohrali dvaja mladíci, ktorí založili e-mailovú konferenciu a začali zber úprav (patchov) a ich distribúciu koordinovať. Prvá verejná verzia s označením 0.6.2 bola vydaná v apríli 1995. Nasledovalo prepísanie kódu (Apache2 už neobsahuje nič z pôvodného NCSA HTTPd) a založenie Apache Group.

Apache je vyvíjaný a udržovaný otvorenou komunitou vývojárov pod záštitou Apache Software Foundation. Aplikácia je dostupná pre širokú škálu operačných systémov od Unix až po Microsoft Windows. Je charakterizovaný ako slobodný a otvorený software, vydaný pod licenciou Apache.

Apache podporuje širokú škálu funkcii, mnohé z nich implementované ako kompilované moduly, ktoré rozširujú základnú funkcionalitu od programovacích jazykov určených pre programovanie na strane serveru až po podporu autentizačných schém.

O IIS od Microsoftu som ani neuvažoval, uvádzam ho tu len pre kompletnosť dnes najpoužívanejších webových serverov na internete. Do úvahy prichádzali Lighttpd a Apache. Pre prevádzku reálneho serveru by bolo pravdepodobne výhodnejšie použitie Lighttps, hlavne v prípade, keby sa počet užívateľov začal počítať v stovkách, či tisíckach, keďže Lighttpd je server prevažne určený na prevádzku v miestach veľkým vyťažením.

Napriek tomu som nakoniec zvolil Apache. Predsa len je to najpoužívanejší serverový software súčasného internetu a testovanie na mojom počítači rozhodne nepresiahne počet niekoľkých klientov. Koniec koncov Apache je priamo obsiahnutý v základnej balíčkovej výbave Fedory.

2.4.3 Inštalácia web serveru

Inštalácia najnovšej verzie Apache serveru pomocou balíčkovacieho softwaru yum

```
yum install httpd httpd-devel httpd-manual
```

Nasledovala konfigurácia automatického spúšťania služby httpd, aby nebolo nutné server pri každom reštarte pc spúšťať

```
/sbin/chkconfig httpd on
```

A nakoniec spustenie serveru pomocou

```
/sbin/service httpd start
```

3 Rozhranie CGI, komunikačný a video kanál

Riešený projekt, alebo tiež vyvíjaná aplikácia sa dá rozdeliť do dvoch častí. Jedna časť pojednáva o komunikačnom kanáli, tá druhá o kanáli prenosovom. Prvá časť, by sa dala považovať za dôležitejšiu, aj keď prenos multimediálnych dát je nemenej dôležitý. Bez neho by samotná aplikácia nemala zmysel, avšak komunikačné rozhranie je svojou implementáciou podstatne zložitejšie a rozsiahlejšie. Obe rozhrania pracujú na princípe CGI, rozoberaného práve v nasledujúcej časti.

Ako už bolo spomínané v časti 2.3.2, kvôli nutnosti prechádzať cez NAT a Firewall, je komunikácia riešená pomocou tzv. Call Back Connection. Detailný popis tohto spôsobu komunikácie ako aj princípy jeho implementácie v riešenom projekte nájdete spracované kapitole 3.2.

3.1 CGI: Common Gateway Interface

Common Gateway Interface (CGI) je štandard v komunikácii medzi externými aplikáciami a informačným serverom, akým je HTTP, alebo Web server. Prostý html dokument prijatý Web démonom je statický, čo znamená, že sa nemení: textový súbor bude stále rovnaký. Na druhú stranu, CGI program, je vykonávaný v reálnom čase, to znamená, že jeho výstup môže obsahovať dynamické informácie [14].

Povedzme napríklad, že by ste chceli spojiť vašu Unixovú databázu z WWW, aby ste umožnili dotazovať sa na ňu ľuďom z celého sveta. V podstate potrebujete vytvoriť CGI program, ktorý bude schopný spustiť web démon, aby preniesol informácie do databáze a prijaté výsledky mohol spätne zobraziť klientovi. CGI program môže byť napísaný v akomkoľvek jazyku spustiteľnom na požadovanom systéme, ako napríklad [14]

C/C++

Fortran

PERL

Unix shell a mnoho ďalších.

Nakoľko CGI je spustiteľný program, je to v podstate to isté ako nechať celý svet spúšťať program na vašom systéme, čo nie je práve najbezpečnejšie. Existuje však niekoľko bezpečnostných pravidiel, ktoré zmierňujú určité bezpečnostné riziká. Jedným z nich je napríklad, že CGI programy, môžu byť spúšťané len zo špeciálneho adresára. V prípade Apache a teda aj v našom je to **`/var/www/cgi-bin/ .`**

3.1.1 Konfigurácia Apache pre povolenie CGI

Aby CGI programy fungovali korektne, alebo aby vôbec fungovali, je potrebné pre ich podporu nakonfigurovať Apache. Direktíva **`ScriptAlias`** povie Apache, že príslušný adresár je určený pre CGI programy. Apache bude teda predpokladať, že každý súbor v tomto adresári je CGI program a pokúsi sa o jeho vykonanie, v prípade, že daný súbor je požadovaný klientom [15].

Pre nastavenie adresára **`/var/www/cgi-bin/`** ako adresára obsahujúceho CGI programy, bolo potrebné pridať nasledujúci riadok do konfiguračného súboru Apache serveru:

```
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
```

Nachádzajúceho sa v **`/etc/httpd/conf/httpd.conf`** .

Pri prvom testovaní CGI skriptu, keď server neustále vracal hlášku „Internal server error“, som zistil, že server beží ako neprivilegovaný užívateľ, zvyčajne „nobody“, alebo „www“ a všetky súbory v adresári /var/www/ sú pod vlastníctvom roota.

Najjednoduchším riešením bola prostá zmena prístupových práv adresára v ktorom sídlia CGI programy a teda pridaním práva na spúšťanie pre všetkých príkazom **`chmod –R a+x /var/www/cgi-bin/`** .

Druhým spôsobom je využite funkcie Apache **suExec**, popísanej nižšie.

suEXEC

Funkcia suEXEC, predstavená v Apache 1.2., poskytuje užívateľom Apache možnosť spúšťať CGI programy pod užívateľským ID odlišným od užívateľského ID, pod ktorým je web server štandardne beží. Obvykle, keď je CGI program spúšťaný, je spúšťaný ako rovnaký užívateľ pod ktorým beží web server.

Pre použite funkcie suExec je nutné ju najskôr nakonfigurovať. Jednou z možností je konfigurácia pomocou konfiguračného skriptu “src/configure” a úpravy suExec hlavičkového súboru. Druhou možnosťou je konfigurácia pomocou autokonfiguračného nástoja APACI.

Príklad výstupu APACI:

```
 suEXEC setup:
             suexec binary: /usr/local/apache/sbin/suexec
             document root: /usr/local/apache/share/htdocs
            userdir suffix: public_html
                   logfile: /usr/local/apache/var/log/suexec_log
                 safe path: /usr/local/bin:/usr/bin:/bin
                 caller ID: www
           minimum user ID: 100
          minimum group ID: 100
```

Po nakonfigurovaní je nutné nastroj suEXEC prekompilovať príkazom **`make`** a nakoniec nainštalovať pomocou **`make install.`** Binárny obraz „suexec“ je umiestnený do adresára definovaného pri konfigurácii. Implicitné umiestnenie je **`"/usr/local/apache/sbin/suexec"`** [16].

3.1.2 CGI knižnice

Pre prácu s rozhraním CGI existuje veľká rada knižníc, ktoré uľahčujú prácu programátorom. Dlho som zvažoval použitie niektorého z dostupných nástrojov. Počas implementácie som však zistil, že pre môj projekt je použitie knižnice zbytočné. CGI totiž to funguje pomerne jednoducho. V prípade požiadavku typu GET sú parametre od web serveru predávané priamo pomocou premenných prostredia (enviroment variable), konkrétne premennej **`“QUERY_STRING”`**. Tá sa dá v programe jednoducho získať volaním funkcie **`getenv("QUERY_STRING")`**, potom už stačí prijatý reťazec len spracovať a máme k dispozícii, kompletnú informáciu od klienta. V prípade požiadavku typu POST je to obdobné, akurát že dáta sú od serveru predávané na štandardný vstup (stdin). Počet znakov predávaných na stdin je definovaný v premennej prostredia (enviroment variable) **`“CONTENT_LENGTH”`**. Je dôležité načítať len počet znakov daných touto premenou, pretože pri pokuse o načítanie väčšieho počtu znakov, nie je definované chovanie programu a môžu nastať neočakávané situácie [19].

Odpoveď späť ku klientovi je ešte jednoduchšia. Je realizovaná zápisom na štandardný výstup (stdout), pomocou klasickej funkcie printf(). Dáta zapísané na stdout si web server (Apache) odchytí a pošle ako odpoveď klientovi. Jediné, čo je potrebné nastaviť je parameter Content-Type podľa typu dát, ktoré posielame, aby server vedel, čo posiela. Nastavenie premennej sa robí opäť zápisom na štandardný výstup. Príklad pre názornosť:

```
printf("%s%c%c\n","Content-Type:text/html;charset=iso-8859-1",13,10)
```

Knižnice obsahujú v sebe podporu pre prácu s cookies, funkcie pre spracúvanie reťazcov a mnoho ďalších, ktoré by v mojom projekte pravdepodobne nenašli uplatnenie.

qDecoder

qDecoder je vývojová sada pre programovanie sieťových aplikácii v jazykoch C/C++ zahŕňajúca podporu pre rozhranie CGI. Algoritmus Query Fetch qDecoderu založený na zreťazených záznamoch poskytuje nezávislosť nad nižšími vrstvami jednoduchým rozhraním knižnice, bez ohľadu na COOKIE/GET/POST(zahŕňajúc Upload súborov). Dovoľuje, aby web-ový software mohol byť navrhovaný a implementovaný viac intuitívne. qDecoder je vyvíjaný podľa modelu slobodného softwaru a je verejne šírený (vydaný pod licenciu LGPL). Posledná a súčasná stabilná verzia je 8.0 [18].

CGIC

Ďalšou z knižníc pre podporu CGI je knižnica CGIC. Je to knižnica vytvorená pre jazyk ANSI C, pre vytváranie www aplikácii založených na CGI rozhraní. CGIC podporuje úlohy pre spracúvanie dát z formulárov, príma dáta pomocou GET s POST, poskytuje funkcie pre nastavenie a obnovovanie „cookies“, načítava CGI premenné prostredia (enviroment variables) do C-éčkových reťazcov a v neposlednej rade poskytuje podporu pri ladení CGI programov. CGIC je kompatibilné z väčšinou súčasne používaných serverov, vrátane Apache využitého v tomto projekte [17].

Neskôr pri ladení programu, som si uvedomil, že ladenie programu založeného na rozhraní CGI nie je vôbec jednoduché. Veď ako ladiť program, ktorý vlastne spúšťa iná aplikácia (web server) a nie je možné ho v spustiť v nejakom nástroji pre ladenie, akým je napríklad GDB. V tom okamihu by som podporu niektorej z knižníc prijal s nadšením. Avšak program už bol napísaný a prípadná dodatočná integrácia niektorého z nástrojov by prestavovala mnohonásobne viac práce než pracné ladenie. Pri ladení som si teda musel vystačiť s logmi, ktoré sa v prípade Apache nachádzajú v adresári /etc/httpd/logs/. V danom adresári sa nachádzajú dva typy logov: Access_log a Error_log.

3.2 Call Back Connection

Call back connection ako už doslovný preklad napovedá, je forma komunikácie založená na posielaní informácii, v našom prípade príkazov pre kameru, nie ako požiadavku (request), ale ako odpoveď na požiadavku (response). Kvôli dôvodom spomínaným vyššie, spôsobených kamerou umiestnenou za NATom, prípadne Firewallom nemôže spojenie naväzovať server. Iniciátorom spojenia sa teda stáva kamera. Tá v súlade s predstavou o nulovej konfigurácii, okamžite po zapnutí a pripojení do internetu načíta adresu a port serveru z konfiguračného súboru, umiestneného v **`/home/camera/apps/camera.cfg`** a pokúsi sa k nemu pripojiť.

3.2.1 Wasatis Server

Serverová aplikácia sa dá chápať ako proxy medzi všetkými kamerami a užívateľmi vo Wasatis systéme. Dovoľuje užívateľom prístup k ich kamerám s jedinou podmienkou, aby kamery mali prístup do internetu ako http klienti. To znamená, že kamera je prístupná z internetu, aj v prípade, že je umiestnená za zariadeniami akými sú NAT, alebo Firewall.

Server čaká na spojenia od kamier a vytvorí komunikačný kanál pre každú kameru, ktorá sa pripojí. Kamera udržuje komunikačný kanál po celú dobu spojenia a v momente keď užívateľ požaduje prístup ku kamere, je vytvorené okamžite spojenie cez server. Princíp pripojenia kamery je

popísaný v kapitole 3.2.4 . Príkazy posielané cez tento kanál dovoľujú prístup k ovládaniu kamery. Serverová aplikácia je nazvaná ako **`cbc.cgi`** .

3.2.2 Klient – Kamera

Na všetkých kamerách systému Wasatis bude bežať klientska aplikácia vyvíjaná práve v rámci tohto projektu. Klient sa pripojí k wasatis serveru definovaného v konfiguračnom súbore a príjme všetky príkazy a požiadavky prostredníctvom http spojenia. Komunikačný kanál medzi kamerou a serverom je šifrovaný a na oboch stranách prebieha autentizácia.

Pri zapnutí kamery sa klientska aplikácia pripojí na wasatis server použitím HTTP a tak môže prejsť cez väčšinu routerov a firewallov. Konfiguračný súbor v kamere špecifikuje, ku ktorému serveru sa má kamera pripojiť. V súčasnosti disponuje wasatis len jedným serverom. V budúcnosti, v prípade, že by vlastnili viacero serverov kvôli rozloženiu záťaže a prípadným poruchám, by bolo možné aby konfiguračný súbor obsahoval adresy viacerých serverov a v prípade neúspešného pripojenia k jednému z nich by sa kamera pokúsila pripojiť k ďalšiemu zo zoznamu.

Wasatis server posiela príkazy pre kameru ako HTTP odpoveď na prvotnú HTTP požiadavku. Spojenie je teda od úvodnej požiadavky stále udržované a je prezívané Call Back Connection, CBC. V prípade prerušenia spojenia, klient okamžite otvorí nové.

4 Konfigurácia

4.1 Konfigurácia serveru

Pre konfiguráciu serveru je potrebný len jediný údaj a tým je názov a umiestnenie adresára s údajmi o kamerách. Kvôli jedinému parametru mi pripadalo zbytočné vytváranie konfiguračného súboru a tak tento údaj je natvrdo nastavený priamo v zdrojovom kóde serverovej aplikácie. Jeho prípadná zmena by bola možná, zmenením obsahu premennej **`path`** v súbore **`cbc.cgi.c`**, pred skompilovaním zdrojových kódov, avšak zmena umiestnenia adresára by bola zbytočná, nakoľko CGI aplikácie nemôžu z dôvodu bezpečnosti pristupovať mimo adresár pre nich vyhradený. Prednastavená hodnota premennej **`path`** je **`"./cams".`** Teda adresár je umiestnený v aktuálnom adresári v ktorom sídli aplikácia **`cbc.cgi`** a jeho názov je **`cams`**.

V adresári **`cams`** sú umiestnené adresáre všetkých kamier ktoré majú povolenie sa pripojiť k serveru. Každý adresár nesie názov podľa SN (10 miestneho sériového čísla kamery) kamery a obsahuje tri súbory:

- **`cam.cfg`** – konfiguračný súbor posielaný kamere, obsahuje dva parametre:

send_video	Podľa tohto parametru kamera vie, či server požaduje posielanie videa, alebo nie. hodnoty: yes/no
fps	Určuje počet záberov za sekundu hodnoty: 1-30

Tabuľka 3.1: Parametre konfiguračného súboru

- **`reg.bin`** – registračný súbor, do ktorého si server ukladá informácie o autentizácii a údaje potrebné pre autentizáciu kamery, obsahuje tri parametre

status	Stav autentizácie kamery, či je alebo nie je autentizovaná hodnoty: auth/unauth
snonce	Hodnota snonce vygenerovaná serverom a použitá pri autentizácii posledného spojenia s kamerou hodnota: 20 hexa znakov šifry SHA1
cnonce	Hodnota cnonce vygenerovaná klientom a použitá pri autentizácii posledného spojenia so serverom hodnota: 20 hexa znakov šifry SHA1

Tabuľka 3.2: Parametre registračného súboru

- **key.bin** – súbor obsahuje kľúč, používaný k autentizácii klienta.

key	20 miestny hexa kľúč kamery používaný k autentizácii klienta. Je to výsledok hash funkcie sériového čísla a špecifického kódu operátora. key = SHA1(SN, brandspec)

Tabuľka 3.3: Parametre súboru s autentizačnými kľúčmi

Pre príklad. Kamera so sériovým číslom 1234567890 má registračný súbor uložený v súbore: **/var/www/cgi-bin/cams/1234567890/reg.bin**

a samotný súbor má tvar:

unauth 127ca0725ba08ec74a0b1d9cc988538a152caa19 0

„Unauth“ teda znamená, že kamera doposiaľ nebola úspešne autorizovaná. Zatiaľ bola vygenerovaná len hodnota snonce ako výzva na autorizáciu pre kameru. Po prijatí odpovede a úspešnej autorizácii kamery sa do súboru uloží aj hodnota cnonce, zatiaľ prezentovaná ako **“0“**.

Vo vnútri aplikácie sú súbory **cam.cfg** aj **reg.bin** reprezentované vlastnými štruktúrami, v ktorých každá položka reprezentuje jeden parameter z daného súboru.

Štruktúra nesúca údaje z registračného súboru reg.bin:

```
struct Register {
      char status[10];
      char snonce[41];
      char cnonce[41];
};
```

Štruktúra nesúca údaje z konfiguračného súboru cam.cfg:

```
struct Camcfg {
      char send_video[4];
      char fps[3];
};
```

Význam jednotlivých členov štruktúr je zhodný s popisom parametrov v súboroch, z ktorých sú načítané.

Súbor **reg.bin** je určený nie len na čítanie, ale aj pre zápis. Pre zápisu do súboru je určená funkcia **regFileWrite**, majúca tri parametre. Prvým je cesta k súboru, druhým sériové číslo obsluhovanej kamery a tretím štruktúra, v ktorej sú uložené dáta určené pre zápis do súboru.

int regFileWrite(char *path, char *serial, struct Register *regInfo)

V prípade úspešného zápisu funkcia vracia hodnotu 0, v prípade neúspechu 1.

K načítaniu údajov z registračného súboru bola vytvorená funkcia s obdobným názvom **regFileRead** a zhodnými parametrami. Jediným rozdielom je, že tretí parameter slúži ako štruktúra pre načítané parametre. Návratové hodnoty sú rovnaké ako v predchádzajúcej funkcii.

```
int regFileRead (char *path, char *serial, struct Register *regInfo)
```

Pre naplnenie štruktúry **Camcfg** slúži funkcia **camcfgFileRead**. Syntax má zhodnú s už spomínanými funkciami pre správu registračného súboru.

```
int camcfgFileRead (char *path, char *serial, struct Camcfg *camcfg)
```

4.2 Konfigurácia kamery

Klientska aplikácia je konfigurovaná pomocou konfiguračného súboru **camera.cfg** umiestneného v **/home/camera/app/camera.cfg** na file systéme kamery. Súbor definuje radu parametrov používaných klientskou aplikáciou.

Konfiguračný súbor začína menom v hranatých zátvorkách. Ďalej je tvorený parametrami popísanými v tabuľke 3.4. Každý parameter stojí na samostatnom riadku a hodnota mu je pridelená pomocou znamienka rovná sa (“=“).

Syntax:

<parameter>	Popis
enabled	Parameter používaný pre rozhodovanie, či konfiguračný súbor ma, alebo nemá byť použitý. Hodnota: yes/no
serverurl	Adresa Wasatis serveru. http://<addr>:<port> <addr> DNS meno, alebo IP serveru <port> port, na ktorom server čaká na spojenie
video	Cesta k video rozhraniu
cbc	Cesta k rozhraniu pre komunikačný kanál
vbuffer	Cesta k video bufferu kamery
timeout	Definuje, koľko sekúnd má kamera čakať kým sa skúsi znovu pripojiť k serveru. Definuje po akej dobe je kamera považovaná za odpojenú.
brandspec	Špecifické číslo operátora je definované desiatimi číslicami int. Toto číslo je súčasťou kľúča, používaného k autentizácii a kryptografii.
serialnum	Sériové číslo kamery

Tabuľka 3.4: Konfiguračné parametre kamery

Parametre sa do konfiguračného súboru zapisujú v tvare <meno_parametru> = <hodnota>. Pre názornosť môže konfiguračný súbor vyzerať nasledovne:

```
[wasatis_camera_1]
enabled = yes
serverurl = b04-1008b.kn.vutbr.cz:80
video = /cgi-bin/video.cgi
cbc = /cgi-bin/cbc.cgi
vbuffer = ./image.jpg
timeout = 120
brandspec = 2233445566
serialnum = 1234567890
```

Načítanie konfiguračného súboru má na starosti funkcia **`configFileRead`**, volaná s dvomi parametrami. Prvým je názov a cesta ku konfiguračnému súboru, druhým štruktúra, v ktorej funkcia vracia konfiguračné parametre načítané so súboru.

```
int configFileRead (char *configFile, struct Config *cfgFile),
```

Funkcia je volaná vždy pri štarte klientskej aplikácie v kamere. Preto v prípade zmeny konfiguračného súboru je pre načítanie nových nastavení nutný reštart kamery a tým aj reštart klientskej aplikácie. Nasleduje štruktúra **`struct Config`**, nesúca parametre načítané z konfiguračného súboru

```
struct Config {
        char name[25];
        bool enabled;
        char serv_ip[25];
        char serv_port[6];
        char video[50];
        char cbc[50];
        char vbuffer[50];
        int timeout;
        char brandspec[11];
        char serial[11];
};
```

5 Rozhrania

Pre komunikáciu s wasatis serverom sú používané dva typy komunikačných rozhraní:

- rozhranie pre pripojenia kamery a posielanie príkazov – CBC
- a rozhranie pre prenos multimediálnych dát

Pre dve spomínané rozhrania sú používané dva rôzne typy http požiadaviek (http request). Vo všeobecnosti je HTTP požiadavka konštruovaná adresou a cestou k rozhraniu podľa konfiguračného súboru, nasledovanou radou parametrov.

wasatis server interface: **`http://<addr>:<port>/<path>`**

Cesta **`(path)`** závisí od typu rozhrania a je konštruovaná na základe parametrov obsiahnutých v konfiguračnom súbore kamery. Parameter „cbc“ je určený pre komunikačný kanál, „video“ zas pre kanál prenosový.

5.1 Komunikačný kanál – CBC

Pre uskutočnenie spojenia medzi kamerou a serverom, kamera vytvorí HTTP GET požiadavku a pošle ju serveru. Požiadavka pozostáva z URL, podľa parametru **`serverurl`** z konfiguračného

súboru, nasledovaná cestou ku konkrétnemu rozhraniu (parameter video, alebo cbc z konfiguračného súboru kamery) a potrebnými parametrami.

Formát požiadavku na server pre vytvorenie komunikačného kanálu je nasledovná:

http://<serv_ip>:<serv_port>/<cbc_path>?<serial_num>=<hodnota>

Keď sa kamera prvý krát pripája k wasatis serveru, server odpovie s autentizačnou výzvou pre kameru a spojenie sa uzavrie. Kamera sa pripojí znovu a ak uspeje v autentizačnom procese so serverom, bezpečné spojenie môže byť uskutočnené. Spojenie sa potom stáva komunikačným kanálom medzi kamerou a serverom a je udržované po celú dobu. Všetka komunikácia na tomto kanáli podlieha autentizácii.

Príklad ako môže vyzerať prvá požiadavka na spojenie od kamery so sériovým číslom 1234567890, na server bežiaci na IP 147.229.192.86 a porte samozrejme 80:

http://147.229.192.86:80/cgi-bin/cbc.cgi?serial=1234567890

V prípade takejto požiadavky aplikácia v kamere naviaže spojenie so serverom na adrese 147.229.192.86 a porte 80 a pošle mu požiadavku v tvare:

```
GET /cgi-bin/cbc.cgi?serial=1234567890 HTTP/1.1
Host: 147.229.192.86
Connection: close
```

O naviazanie spojenia sa stará funkcia **cbcConnect**, volaná vždy pri štarte kamery, alebo vždy keď je potreba opätovného naviazania spojenia, pri prípadnom výpadku.

```
int cbcConnect( int sockfd , struct sockaddr_in serv_addr ,
 struct Register *regData , struct Config *cfgFile )
```

Správu samotného komunikačného kanálu a teda aj prijímanie príkazov od serveru má na starosti funkcia **cbc**, volaná periodicky každých desať sekúnd.

```
int cbc( int sockfd , struct sockaddr_in serv_addr ,
 struct Register *regData , struct Config *cfgFile )
```

CBC posiela v rámci požiadavku na server a v rámci udržania spojenia pakety obsahujúce len základné údaje potrebné pre autentizáciu:

- **serial** – sériové číslo kamery, slúži k jednoznačnej identifikácii kamery
- **cnonce** – 20miestne hexa číslo, slúži ako autentizačná výzva pre server
- **signature** – 20miestne hexa číslo ako výsledok hash funkcie SHA1, pre podpis paketu

Jedinou zmenou oproti cbcConnect je položka Connection, ktorá je zmenená z **close** na **keep-alive**. To z dôvodu, aby nebolo nutné pri každom spojení ho nanovo vytvárať. Spojenie má tým pádom nižšiu réžiu a nezaťažuje komunikačnú linku zbytočnými paketmi nutnými na vytvorenie

spojenia. Na servery je hodnota **keep-alive** nastavená na hodnotu pätnásť sekúnd. To znamená, že ak server neobdrží od kamery nejaké dáta aspoň raz za pätnásť sekúnd, bude spojenie zrušené a kamera bude musieť naväzovať spojenie na novo.

Keďže server je vytvorený na princípe rozhrania CGI, web server pri každej žiadosti o spojenie so serverovou aplikáciou spustí nový proces tejto aplikácie. Z tohto dôvodu nebolo nutné riešiť vybavovanie jednotlivých požiadaviek pomocou nekonečného cyklu, alebo vytvárania synovských procesov pomocou **fork**. Každé volanie programu obslúži totižto len jednu požiadavku od klienta a v zápätí sa ukončí.

Pri každom spojení generuje server novú výzvu snonce pre autentizáciu. V prípade úspešnej autentizácie pridá server do odpovede údaje z konfiguračného súboru. Volaním funkcie **camcfgFileRead** načíta parametre do štruktúry typu **Camcfg** a predá ich kamere.

Odpoveď kamere po úspešnej autorizácii teda obsahuje štyri parametre:

- **snonce**
- **signature**
- **send_video**
- **fps**

Význam jednotlivých položiek je zhodný z popisom uvedeným v časti 3.2.3.

5.2 Video kanál

Video kanál slúži k prenosu multimediálnych dát, teda videa, medzi kamerou a wasatis serverom. Za normálnych okolností, keď server nepožaduje obrazové dáta od kamery, nie je spojenie pomocou video kanálu vôbec naviazané. V prípade, keď by niektorá z aplikácii bežiacich na servery, napríklad software pre detekciu narušenia, alebo streaming server, požadovala video od kamery, musí v konfiguračnom súbore konkrétnej kamery podľa sériového čísla **/var/www/cgi-bin/<serial_number>/cam.cfg** zmeniť parameter **send_video** na **„yes“** a nastaviť počet snímok za sekundu, ktoré požaduje. Server potom tieto údaje načíta a pošle kamere v rámci odpovede na pravidelné dotazy, ktoré táto na server posiela.

V momente, keď kamera po spracovaní prijatých dát od serveru zistí, že parameter **send_video** obsahuje hodnotu **„yes“**, vytvorí spojenie so serverom pomocou video kanálu.

Pre vytvorenie video kanálu medzi kamerou a serverom, kamera vytvorí HTTP POST požiadavku a pošle ju serveru. Požiadavka pozostáva z URL, podľa parametru **serverurl** z konfiguračného súboru, nasledovaná cestou ku konkrétnemu rozhraniu.

http://<serv_ip>:<serv_port>/<cbc_path>

Parametre za url sa v prípade požiadavku typu POST nedávajú. Všetky dáta sú posielané v tele, v dátovej časti http paketu. Pre názornosť uvádzam príklad http hlavičky pri posielaní obrazových dát na server:

```
POST /cgi-bin/video.cgi HTTP/1.1
Host: 147.229.192.86
Connection: close
Content-type: application/x-www-form-urlencoded
Content-length: 1000
<data>
```

6 Bezpečnosť

Každá kamera disponuje vlastným unikátnym sériovým číslom. Hashovaním tohto sériového čísla pomocou hashovacej SHA-1 funkcie s desať miestnym špecifickým číslom operátora (definovaného v súbore), vznikne špecifický kľuč (KEY). Tento kľúč je používaný pre autentizáciu a šifrovanie spojenia medzi kamerou a serverom. Wasatis server uchováva databázu všetkých kľúčov a teda všetkých kamier dostupných v systéme. V prípade, že by daný software používalo viacero spoločností, umožnil by tento princíp overovania kamier odfiltrovanie tých ktoré patria k inému poskytovateľovi služieb. Požiadavka na pripojenie od kamery, ktorej číslo nie je serveru známe by bola odmietnutá, aj napriek tomu, že používa rovnaký software a komunikuje na rovnakom protokole. Táto funkcia zrejme nebude využitá, keďže software bude používať pravdepodobne len spoločnosť Wasatis.

Pre autentizáciu som sa rozhodol použiť jednosmernú, alebo tiež hash funkciu. Princíp autentizácie spočíva vo vytváraní podpisu (signature) a jeho pridávaní do každého komunikačného paketu.

SHA-1

SHA: je to päť kryptografických hash funkcii navrhnutých NSA (National Security Agency), čo je národná bezpečnostná agentúra v Spojených štátoch, a publikované NIST (National Institute of Standards and Technology). SHA je skratka od Secure Hash Algorithm. Hash algoritmy vytvárajú digitálnu reprezentáciu fixnej dĺžky (známu ako message digest) sekvencie vstupných dát ľubovoľnej dĺžky. Hash algoritmy sú označované ako „bezpečné“ keď:

1. Je výpočetne nemožné nájsť správu z ktorej bola vytvorená daná „message digest“
2. Je výpočetne nemožné nájsť dve rôzne správy, z ktorých by vznikla rovnaká „message digest“
3. Každá zmena v správe (aj zmena jediného bitu) bude s veľmi veľkou pravdepodobnosťou viesť k úplne rozdielnej „message digest“

Päť spomínaných algoritmov je označovaných ako SHA-1, SHA-224, SHA-256, SHA-384 a SHA-512. Neskoršie štyri varianty sú niekedy kolektívne označované ako SHA-2. SHA-1 vytvára „message digest“ o dĺžke 160 bitov. Číslo v názve pri ostatných štyroch algoritmoch znamená bitovú dĺžku správy ktorú vytvárajú.

SHA-1 je používaný v niekoľkých všeobecne rozšírených bezpečnostných aplikáciách a protokoloch, vrátane TLS a SSL, PGP, SSH, S/MIME, and IPsec. Je považovaný za nástupcu za MD5, staršej, široko rozšírenej hash funkcie [20].

Zo začiatku som váhal medzi použitím MD5 a SHA1. Nakoľko MD5 je starší algoritmus a hlavne bol už prelomený, rozhodol som sa použiť SHA1. Existuje niekoľko voľne šíriteľných knižníc pre jazyk C obsahujúcich implementácie kryptografických funkcii. Vari najznámejšia z nich je OpenSSL, priamo obsiahnutá vo väčšine štandardných linuxových distribúcii, vrátane Fedory. Rozhodol som sa teda pre OpenSSL. Pri kompilácii sa však vyskytol problém s importovaním hlavičkových súborov do mojich zdrojových textov, ktorý sa mi nepodarilo vyriešiť a tak som hľadal náhradu. Za náhradné riešenie som zvolil XySSL, voľne šíriteľnú implementáciu podobnú OpenSSL s otvorenými zdrojovými kódmi, dostupnú na stránkach xyssl.org. XySSL obsahuje implementovanú radu šifrovacích a hashovacích algoritmov ako AES, MD5, SHA-1 a ďalšie.

Pre potreby aplikácie bola vytvorená funkcia **`sha1Value`**, ktorá vo svojom tele zabezpečuje nastavenia potrebných premenných a vygenerovanie samotnej SHA1 správy. Keďže výsledkom generovania správy je dvadsať miestne číslo v šestnástkovej sústave, je toto na konci funkcie skonvertované na reťazec o dĺžke štyridsať znakov.

```
int sha1Value(char *input, char *output)
```

Podpis vznikne hashovaním troch základných údajov. Jedným je kľúč (KEY) špecifický pre každú kameru, druhým je hodnota snonce, generovaná serverom ako autentizačná výzva pre kameru. Tretím je hodnota cnonce, generovaná naopak kamerou ako autentizačná výzva pre server. Dôležitá je autentizácia na oboch stranách. V prípade slabej kontroly na strane serveru tento mohol povoliť pripojenie zariadeniu (kamere), ktorá nemá povolený prístup do systému Wasatis. Naopak v prípade slabej, alebo žiadnej kontroly na strane kamery, by mohlo dôjsť k podvrhnutiu paketu a teda aj príkazov pre kameru.

Príklad vytvárania podpisu paketu (signature):

```
signature=SHA1(cnonce, snonce, KEY).
```

6.1 Princíp pripojenia a autentizácie

1. Klient – kamera sa pripojí na Wasatis server
2. Kamera pošle na server HTTP požiadavku aby sa identifikovala

   Http požiadavka vyzerá nasledovne:

GET /cgi-bin/cbc.cgi?serial=<serial number> HTTP/1.1

Host: <server_ip>

Connection: close

3. Wasatis server odpovie s výzvou (snonce) vygenerovanou pomocou random()
4. Wasatis server následne ukončí spojenie
5. Kamera sa pripojí k serveru znova

   Toto spojenie je prezívané CBC – Call Back Connection
6. Kamera pošle na server HTTP požiadavku s hodnotou cnonce

   Http požiadavka vyzerá nasledovne:

   GET /cgi-bin/cbc.cgi?serial=<serial number>

   &cnonce=<cnonce value>&signature=SHA1(

   snonce, cnonce, KEY) HTTP/1.1

   Host: <server_ip>

   Connection: keep-alive

   V prípade, že podpis (signature) je nesprávny, alebo úplne chýba, server odpovie s

   incorrect_signature

   a ukončí spojenie. Kamera začne autentizačný proces odznova, od kroku 1.

   Ak je podpis (signature) správny, kamera je autorizovaná a autentizovaná. Server odpovie:

   HTTP/1.1 200 OK

   Bezpečný komunikačný kanál bol práve uskutočnený a príkazy sú posielané z wasatis serveru ku kamere.

7 Záver

Prácou na bakalárskej práci som si mal možnosť zistiť aké požiadavky majú spoločnosti na kamerové systémy, používané v oblasti zabezpečenia. Získal som náhľad do štruktúry siete a komunikačných protokolov používaných pre prenos multimediálnych dát a dovolila mi spojiť môj záujem o programovací jazyk C so znalosťami z oboru sietí získanými z povinných, ale predovšetkým voliteľných predmetov na Fakulte informačných technológii. Za zvlášť zaujímavé považujem, to že som si mal možnosť vyskúšať vývoj softwaru, ktorý je súčasťou väčšieho projektu a bude využitý v praxi a práca na ňom odvedená nebude samoúčelná.

Dôležitá bola pre mňa spätná väzba, ktorú som mal po dobu tvorby softwaru s vývojovým tímom spoločnosti Wasatis.

Z obecného hľadiska by bolo možné za najprínosnejšie považovať nájdenie riešenia ako prejsť cez firewall a NAT, bez nutnosti zmeny nastavení, teda bez nutnosti akéhokoľvek nastavovania, čo bola hlavná a prvotná myšlienka predstaviteľov spoločnosti Wasatis. Výsledný software by mal byť schopný po integrácii do kamery sám vytvoriť spojenie a komunikovať so serverovou aplikáciou, bez nutnosti zásahu užívateľa. Za prínosné sa dá určite považovať aj návrh a implementácia autentizačnej schémy, ktorá rozhodne prispeje k bezpečnosti celého systému.

V priebehu práce som sa musel vysporiadať s niekoľkými problémami, predovšetkým pri nastavovaní Apache a povoľovaní komunikácie pomocou rozhrania CGI. Hlavný problém ale predstavovala nemožnosť ladiť program klasickými metódami a pomocou klasických nástrojov, čo predstavovalo značné sťaženie pri hľadaní chýb a tým aj spomalenie celého vývoja aplikácie.

Nakoľko v dobe písania tejto práce neboli ešte dokončené ostatné časti vyvíjaného systému, nebolo možné mnou implementované aplikácie nasadiť do testovacej prevádzky a otestovať funkčnosť v reálnej prevádzke. Rozhodným prínosom by bola možnosť software testovať priamo v kamere a overiť tak jeho funkčnosť, avšak tú som zatiaľ nemal k dispozícii.

Nakoľko problematika riešeného projektu je dosť zložitá a komplexná, rozhodne v ňom existuje množstvo vecí, ktoré sa dajú vylepšiť. Určite by som do budúcnosti uvažoval implementácii protokolu RTP/RTCP do video rozhrania a riešil celú réžiu riadenia prenosu na báze týchto protokolov. Ďalším vylepšením z hľadiska bezpečnosti by mohlo byť okrem autentizácie pridanie aj šifrovania na komunikačnom kanáli, napríklad formou algoritmu rc4, pre ktorého použitie je dokonca vytvorená funkcia priamo v aplikácii, avšak v konečnom riešení nie je aplikovaná.

Verím, že s prípadnými vylepšeniami sa podarí môjmu softwaru uspieť v praxi a splní predstavy spoločnosti Wasatis. Úspech celého projektu ešte závisí na dokončení ostatných častí, venujúcich sa predovšetkým problematike spracovania obrazu a v neposlednej rade tvorte webového rozhrania pre prístup užívateľov k službám spoločnosti Wasatis.

Literatúra

[1] WWW stránky. Cisco: How nat works.
http://www.cisco.com/warp/public/556/nat-cisco.shtml.

[2] WWW stránky. Gentoo linux: About.
http://www.gentoo.org/main/en/about.xml.

[3] WWW stránky. Axis Communications- Network Camera Developer Pages
http://www.axis.com/techsup/cam_servers/dev/product_interface_guide.htm

[4] WWW stránky. Wikipedia: Fedora
http://cs.wikipedia.org/wiki/Fedora

[5] WWW stránky. Wikipedia: Real-Time Transport Protokol
http://en.wikipedia.org/wiki/Real-time_Transport_Protocol

[6] WWW stránky. Wikipedia: Firewall
http://cs.wikipedia.org/wiki/Firewall

[7] WWW stránky. Instaling apache on Fedora
http://www.flmnh.ufl.edu/linux/install_apache.htm

[8] WWW stránky. Rfc 3489: Stun - simple traversal of user datagram protocol (udp)
through network address translators (nats).
http://www.ietf.org/rfc/rfc3489.txt.

[9] WWW stránky. Wikipedia: Nat#drawbacks.
http://en.wikipedia.org/wiki/Network_address_translation#Drawbacks.

[10] WWW stránky. Wikipedia: Lighttpd
http://en.wikipedia.org/wiki/Lighttpd

[11] WWW stránky. Wikipedia: Web server
http://en.wikipedia.org/wiki/Web_server

[12] WWW stránky. Wikipedia: Internet information services
http://en.wikipedia.org/wiki/Internet_Information_Services

[13] WWW stránky. Wikipedia: Apache http server
http://en.wikipedia.org/wiki/Apache_HTTP_Server

[14] WWW stránky. CGI: Common Gateway Interface
http://hoohoo.ncsa.uiuc.edu/cgi/intro.html

[15] WWW stránky. Apache Tutorial: Dynamic content with CGI
http://httpd.apache.org/docs/1.3/howto/cgi.html

[16] WWW stránky. Apache suExex support
http://httpd.apache.org/docs/1.3/suexec.html

[17] WWW stránky. Cgic: an ANSI C library for CGI programing
http://www.boutell.com/cgic/

[18] WWW stránky. The qDecoder project: About qDecoder
http://www.qdecoder.org/about.qsp

[19] WWW stránky. Getting started with CGI programming in C
http://www.cs.tut.fi/~jkorpela/forms/cgic.html

[20] WWW stránky. SHA hash functions
http://en.wikipedia.org/wiki/SHA_hash_functions

Zoznam príloh

Makefile

```
# Makefile for wasatis project
CBC             = cbc.cgi
VIDEO      = video.cgi
CLIENT          = wasatis_client
CBC_SOURCES     = cbc.cgi.c
VIDEO_SOURCES   = video.cgi.c
CLIENT_SOURCES  = wasatis_client.c

CFLAGS          = -std=gnu99 -Wall -pedantic -W

CC              = gcc
CBC_OBJECTS     = $(CBC_SOURCES:.c=.o)
VIDEO_OBJECTS   = $(VIDEO_SOURCES:.c=.o)
CLIENT_OBJECTS  = $(CLIENT_SOURCES:.c=.o)

################################################

.SUFFIXES: .c .o

.c.o:
          $(CC) $(CFLAGS) -c $<

################################################

all:       $(CBC) $(CLIENT) $(VIDEO)

rebuild:   clean all

$(CBC):          $(CBC_OBJECTS)
          $(CC) $(CBC_OBJECTS) $(LDFLAGS) -o $@
          rm $(CBC).o
          cp cbc.cgi /var/www/cgi-bin/

$(CLIENT): $(CLIENT_OBJECTS)
```

```
	$(CC) $(CLIENT_OBJECTS) $(LDFLAGS) -o $@
	rm $(CLIENT).o
#	cp wasatis_client /home/camera/apps/


$(VIDEO):  $(VIDEO_OBJECTS)
	$(CC) $(VIDEO_OBJECTS) $(LDFLAGS) -o $@
	rm $(VIDEO).o
	cp video.cgi /var/www/cgi-bin/

	cp -R cams /var/www/cgi-bin/
	chmod -R 777 /var/www/cgi-bin/cams/
###############################################
```