B E S I P

Bright Embedded Solution for IP Telephony

verze 1.0

CESNET, z.s.p.o.

2011

Document No: BESIP-V-1.0
Version / date: December 10, 2011
Original language : CZ
Original title: “Bright Embedded Solution for IP Telephony”
Contact: besip.developers@gmail.com

Stránka projektu: https://sip.cesnet.cz/cs/implementace/besip
Vývoj: https://homeproj.cesnet.cz/projects/besip/
Kontakt: mailto:besip.developers@gmail.com
Mirror distribuce: http://liptel.vsb.cz/mirror/

Editor :

Miroslav Vozňák

Autoři přispívající do jednotlivých kapitol:

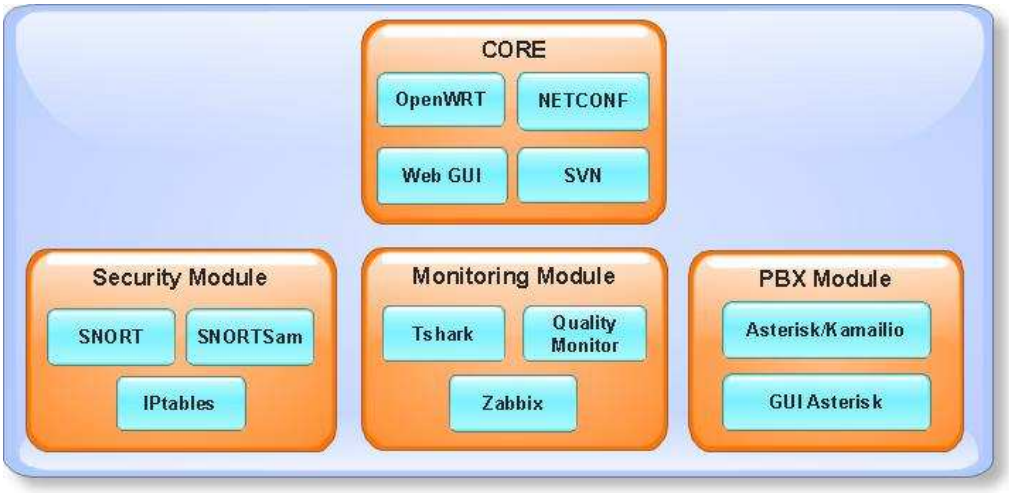
Lukáš Kapičák, kap. 3
Václav Svatoň, kap. 2
Jakub Šafařík, kap. 4
Jiří Šlachta, kap. 2
Karel Tomala, kap. 2,
Martin Tomeš, kap. 5
Jan Vogl, kap. 5
Lukáš Vojáček, kap. 2
Jiří Vychodil, kap. 3

verze 1.0

Obsah

1. Úvod

Cílem projektu s pracovním názvem Bright Embedded Solution for IP Telephony (BESIP) je vývoj a snadná implementace jednoduchého a efektivního embedded řešení SIP komunikačního serveru se snadnou integrací nejenom do sítě CESNET založeného na open-source řešení. Toto zařízení slouží pro zajištění IP telefonie na pracovištích menší velikosti (SOHO – small office/home office). Architektura systému BESIP je znázorněna na obr. 1.



Obr. 1: Architektura BESIP

Práce na projektu byla rozdělena do následujících balíčků (WP), na kterých pracovali studenti Fakulty informatiky a elektrotechniky VŠB – Technické univerzity v Ostravě:

- **WP1:** balíček vývoje a implementace nástrojů a algoritmů,
- **WP2:** balíček s cílem ověřit a verifikovat výsledky ve WP1,
- **WP3:** celková finalizace, implementace uživatelského rozhraní a tvorba dokumentace.

Tab. 1: Rozklad projektu na úlohy.

Work package	Popis
WP1.1	Analýza požadavků konečného embedded zařízení v závislosti na výkonu, ceně, spotřebě a nabízených službách hardwarové platformy.
WP1.2	Evaluace hardwarových platforem na základě předem definovaných požadavků.
WP1.3	Implementace nástrojů, uživatelského a grafického rozhraní do vyvíjeného komunikačního systému.
WP2.1	Testování funkčnosti, stability a kompatibility jednotlivých komponent systému.
WP2.2	Zátěžové testy s cílem definovat maximální zatížení vyvíjeného produktu.
WP2.3	Zdokonalování řešení a optimalizace běhu systému při nasazení v reálném provozu.
WP3.1	Dokončení a odevzdání finální verze produktu pro sdružení CESNET, včetně archivace operačního systému pro jednoduchou aplikaci na další zařízení.
WP3.2	Tvorba uživatelské a administrátorské dokumentace.

2. Jádro systému BESIP

Jádro systému BESIP se skládá z linuxové distribuce OpenWRT, jeho webového rozhraní pro základní síťová nastavení, SVN systému pro správu zdrojových kódů projektu a v neposlední řadě je připraven pro budoucí použití protokol NETCONF, který je určen pro konfiguraci síťového zařízení.

2.1. OpenWRT

OpenWRT je operačním systémem postaveným na jádře GNU/Linux. Předpokladem OpenWRT jsou jeho nízké nároky na různé architektury a konfigurace, a to především pro "embedded" zařízení. Nároky na hardware jsou v případě OpenWRT výrazně nižší, nežli u běžných distribucí postavených na GNU/Linux. Společně s různými optimalizacemi v OpenWRT jej lze pak nasazovat na téměř jakýkoliv hardware, a to jak z pohledu architektury, tak z pohledu hardwarových prostředků. Optimalizace provedené v OpenWRT ústí ve vyšší konfigurovatelnost, tak též ve vyšší výkonnost vzhledem k nižším hardwarovým požadavkům, než mají běžné distribuce.

Tvorba OpenWRT obrazů pro BESIP

V rámci vývoje pro OpenWRT bylo naším cílem naportovat určité aplikace, které nebyly k dispozici v repozitářích OpenWRT, nebo jejich verze nebyly dostatečné pro jejich provoz v rámci projektu BESIP. Aplikace, které jsme naportovali, zatím realizujeme pomocí repozitářů pro OpenWRT buildroot, jejichž funkčnost je podobná, jako v různých distribucích, což spočívá v přidání řádků do specifického souboru obsahující dané repozitáře. Narozdíl od distribučních repozitářů běžných distribucí obsahují repozitáře OpenWRT a naše repozitáře pouze soubory Makefile, které nám popisují průběh stažení, konfigurace, kompilace a instalace jednotlivých balíků, které pak budou pro OpenWRT zkompilovány v procesu kompilace.

Vývojáři OpenWRT pravidelně kompilují poslední verzi OpenWRT ve specifických intervalech (obvykle denně) balíčky a image OpenWRT, a to pro všechny architektury, pro něž jsou "optimalizovány". Tyto balíčky a image se kompilují v plné konfiguraci, kde konfigurace image je podmnožinou této konfigurace a nadbytečné balíčky jsou zkompilovány jako moduly, což je výhodné pro ty, kteří potřebují mít minimalistický image a po určitém čase si potřebují dodatečně nějaký balíček doinstalovat. Balíčky, které jsou zvoleny jako moduly se při procesu kompilace zkompilují a zabalí do ipk balíčku, který pak lze dodatečně do běžícího OpenWRT systému doinstalovat.

Následující postup bude obsahovat lehce modifikovaný postup od uvedeného postupu na wiki OpenWRT, jehož změny spočívají především v přidání naší konfigurace a našich balíčků do buildrootu OpenWRT.

- OpenWRT buildroot je množina souborů Makefile a patchů, které umožňují vygenerovat toolchain pro kompilaci na různé architektury a vygenerování image pro daný hardware. Tento toolchain využívá malé knihovny jazyka C, která se nazývá uClibc.

- Toolchain je množina nástrojů určená pro kompilaci aplikací ze zdrojových kódů.

Vytváření vlastního obrazu OpenWRT

Předtím, nežli začneme připravovat OpenWRT buildroot je zapotřebí, aby operační systém, na němž kompilujeme OpenWRT měl všechny dostupné aplikace nutné pro jeho kompilaci. Na Debianu Squeeze, na němž se obrazy sestavovaly, jde o následující balíky:

- Pro x86 verzi Debianu Squeeze:

```
# apt-get install build-essential asciidoc binutils bzip2 gawk gettext \
git libncurses5-dev libz-dev patch unzip zlib1g-dev flex quilt
```

- Pro 64b verzi Debianu Squeeze:

```
# apt-get install build-essential asciidoc binutils bzip2 gawk gettext \
git libncurses5-dev libz-dev patch unzip zlib1g-dev ia32-libs \
lib32gcc1 libc6-dev-i386 flex quilt
```

Po instalaci těchto aplikací bude nutné stáhnout pracovní kopii SVN repozitáře BESIPu. Pracovní kopie obsahuje naportované aplikace, sadu patchů a skriptů, které lehce modifikují OpenWRT. Pracovní kopii SVN repozitáře získáme pomocí příkazu:

```
# svn co https://liptel.vsb.cz/svn/besip
```

Po stažení pracovní kopie OpenWRT stáhneme pracovní kopii SVN repozitáře OpenWRT [https://dev.openwrt.org/wiki/GetSource] , a to konkrétně větev Backfire 10.03, pro níž byl vývoj přizpůsoben, což realizujeme pomocí příkazu:

```
# svn co svn://svn.openwrt.org/openwrt/branches/backfire
```

Konfigurace OpenWRT buildrootu

Vývoj jsme přizpůsobovali hardwaru, který byl pro tento projekt zakoupen, což spočívalo v konfiguraci jádra a image OpenWRT. Vzhledem ke složitějšímu ručnímu zařazování patchů a balíčků byl sestaven skript, který provede patche nad buildrootem OpenWRT a přidá naše repozitáře do feeds.conf v kořenovém adresáři OpenWRT buildrootu. Tento skript se nazývá apply.sh a je umístěn ve větvi OpenWRT repozitáře projektu BESIP (momentálně nyní jen trunk). Aplikaci tohoto skriptu provedeme v kořenovém adresáři OpenWRT buildrootu:

```
:/cesta/k/openwrtbuildroot/sh cesta/ke/skriptu/apply.sh
```

Skript apply.sh provádí následující:

- kontroluje, zda-li je spuštěn z OpenWRT buildrootu,
- aktualizuje a nainstaluje feedy (dodatečné repozitáře balíčků pro OpenWRT),
- přidá náš repozitář do feeds.conf ,
- rekurzivně vrátí do původního stavu pracovní kopii SVN repozitáře OpenWRT,
- provede patche nad OpenWRT buildrootem a jeho balíčky,
- aktualizuje a nainstaluje feedy (potřebné pro zavedení naší práce).

Shrneme-li proceduru, tak celá spočívá v:

- získání pracovní kopie OpenWRT,
- získání pracovní kopie BESIPu,
- v OpenWRT buildrootu spustíme bash skript apply.sh, který je umístěn v pracovní kopii BESIPu (besip/Trunk/OpenWrt/apply.sh),

- nakonfigurujeme OpenWRT,
- zkompilujeme OpenWRT.

V repozitáři BESIPu je také připraveny konfigurace jádra a image, které jsou umístěny v repozitáři besipu – besip/Trunk/OpenWrt/apply.sh

Konfigurace image OpenWRT

Konfigurace obrazu spočívá v kontrole výběru a výběru volitelných balíčků oproti výchozí konfiguraci. Prvním krokem tvorby obrazu OpenWRT by měla být kontrola závislostí a prerekvizit build systému, která se provádí pomocí:

```
# make defconfig
```

Defconfig hlásí chyby týkající se chybějících závislostí. Obvykle stačí doinstalovat chybějící balík a znovu spustit make s parametrem defconfig. Druhým krokem je vytvoření konfigurace obrazu OpenWRT. Menuconfig je nástroj, který slouží k výběru cílové platformy, specifických parametrů obrazu, parametrů jádra, volitelných balíčků a také jaderných modulů. Menuconfig spouštíme v kořenu složky stažené větve OpenWRT následujícím příkazem:

```
# make menuconfig
```

Kde po spuštění menuconfigu můžeme nahrát výchozí konfiguraci, kterou rovněž vkládáme do kořenové složky. Obecná konvence pro pojmenování souborů je .config. V našem případě uložíme konfiguraci jádra z pracovní kopie BESIPu, z adresáře besip/Trunk/OpenWrt/cfg/besiphw-image.config do příslušného adresáře. Vzhledem k různým způsobům nasazení OpenWRT jsme občas nuceni experimentovat s konfigurací, kde si nevystačíme pouze s konfigurací obrazu, ale také musíme zasáhnout i do konfigurace jádra. Vytvořený obraz jádra je pak zkomprimován do souboru vmlinuz pomocí algoritmu zlib, LZMA či BZIP2. Konfiguraci jádra obrazu OpenWRT spouštíme pomocí parametru kernel_menuconfig:

```
# make kernel_menuconfig
```

Po spuštění výše uvedeného příkazu se stáhne jádro (v mém případě linux-2.6.32.27), patche a další nástroje do adresáře build\underline{~}dir v kořenové složce větve OpenWRT (backfire), a to konkrétně do:

```
backfire/build_dir/linux-x86_generic/linux-2.6.32.27
```

Toto je následováno podobným konfiguračním nástrojem jako v případě menuconfigu. Rovněž si pro jádro můžeme nahrát vlastní konfiguraci, kterou je třeba uložit do výše uvedeného adresáře. V našem případě uložíme konfiguraci jádra z pracovní kopie BESIPu, z adresáře besip/Trunk/OpenWrt/cfg/besiphw-kernel.config do příslušného adresáře.

Vytvoření obrazu OpenWRT

Jestliže disponujeme vícejádrovým procesorem, můžeme spustit více build procesů, a to pomocí parametru –j.

```
# make –j <počet souběžně běžících build procesů>+1
```

Jestliže dojde během kompilace k chybám, je vhodné spustit make s maximální verbozitou, aby bylo možné lokalizovat chybu ve výpisech. Spouštíme make s parametrem V=99.

```
# make V=99
```

Kompilace OpenWRT je relativně časově náročný proces, obzvláště tehdy, když vybíráme větší množství balíků z feedů OpenWRT. Výsledkem kompilace je kolekce souborů umístěná do adresáře bin v kořenovém adresáři větve OpenWRT. V tomto adresáři pak nalezneme adresář pojmenovaný architekturou, pro níž kompilujeme obrazy OpenWRT a v něm jsou vygenerované virtuální disky, veškeré balíčky a bitové kopie. V menuconfigu si můžeme zvolit jako vygenerovaný obraz virtuální disk pro VMware a VirtualBox, které jsou okamžitě použitelné pro virtuální stroje.

Při kompilaci také může dojít ke komplikacím, které jsou nejčastěji způsobeny volbou některého balíčku z feedů. Feedy nezaručují 100\% funkčnost balíčku, některé mohou být neudržované, některé mohou mít chyby v Makefile (špatné odkazy na archivy zdrojových kódů). Pak také může nastat situace, kdy balíčky na sobě při kompilaci závisí, aniž by to bylo zmíněno v jejich Makefile, některé balíčky zase mají závislosti jen při okamžiku kompilace. Tyto závislosti mohou způsobit nezkompilování image OpenWRT, jelikož dojde často k tomu, že některé balíčky, ačkoliv jsou vybrány pro image OpenWRT, se kompilují později, než samotný balíček, jež má tyto závislosti, což způsobí pád při kompilaci. Pro tyto situace je vhodné zkompilovat image s ignorováním chyb (viz příkaz níže) a poté jej zkompilovat klasickým make (třeba i s verbozitou V=99).

```
# make -i
```

V případě kompilace image se nám osvědčilo zkompilovat image nejdříve s ignorováním chyb a teprve poté jej zkompilovat klasickým způsobem. Ve výjimečných případech kompilace neprocházela ani s touto kombinací, nicméně přidání maximální verbozity (V=99) se zdálo býti postačujícím v procesu kompilace s ignorováním chyb a následně klasické kompilace.

Vygenerované bitové kopie jsou pak použitelné okamžitě pro nasazení na fyzickém stroji. Tyto soubory aplikujeme na fyzický stroj pomocí programu dd a to následujícím způsobem:

```
# dd if=openwrt-x86-generic-rootfs-ext2.img of=/dev/diskFyzickéhoStroje
```

Vytvoření virtuálního stroje VMware

Vytvoření virtuálního stroje pro VMware se skládá ze dvou kroků - vytvoření definice virtuálního stroje (vmx file) a vytvoření virtuálního disku.

Wiki OpenWRT http://wiki.openwrt.org/oldwiki/RunningKamikazeOnVMwareHowTo uvádí funkční konfiguraci pro výchozí konfiguraci OpenWRT, u níž se počítá s absencí ovladačů SATA, USB řadičů a dalších. Pro vygenerované obrazy na liptel.vsb.cz/mirror se s těmito řadiči počítá, tudíž stačí použít vmx soubor na výše zmíněném odkazu na Wiki OpenWRT, nebo vytvořit virtuální stroj ve VMware pro existující disky a vmx soubor poté doplnit o následující řádky:

```
+serial0.fileType = "pipe"
-serial0.fileName = "COM1"
+serial0.fileName = "\\.\pipe\com_1"
+serial0.pipe.endPoint = "server"
+floppy0.present = "FALSE"
```

Virtuální disk pro virtuální stroj VMware můžeme vytvořit třemi způsoby:

1. První způsob je přímé využití virtuálního disku vygenerovaného z kompilace

OpenWRT pomocí make. Tento způsob pak vyžaduje korekci velikosti virtuálního disku pro danou platformu pomocí funkce Expand u Workstation verze Vmware.

2. Druhý způsob je konverze vygenerované bitové kopie pro danou platformu, a to pomocí příkazu:

```
qemu-img convert -f raw disk.img -O vmdk disk.vmdk
```

3. Třetí způsob je ve vytvoření bitové kopie pevného disku na fyzickém stroji pro získání přesné velikosti pevného disku pomocí:

```
dd if=/dev/fyzickýDisk of=bitováKopie.img
```

Poté je třeba převést bitovou kopii pomocí příkazu uvedeného v druhém způsobu. Tento virtuální disk je poté vhodné přepsat nulami k získání minimální velikosti virtuálního disku. Tento virtuální disk připojíme do virtuálního stroje, nabootujeme libovolnou linuxovou distribuci a pomocí následujícího příkazu vynulujeme disk:

```
dd if=/dev/zero of=/dev/virtuálníDisk
```

Následně překlopíme vygenerovaný obraz na vmdk disk pomocí:

```
dd if=openwrt-x86-generic-combined-ext2.img of=/dev/virtuálníDisk
```

Práce s vygenerovanými obrazy OpenWRT

Popis níže popisuje realizaci tvorby virtuálního stroje, na liptel.vsb.cz/mirror budou ale ke stažení i image předvytvořených virtuálních strojů. S vygenerovanými obrazy, které budou na liptel.vsb.cz/mirror lze zacházet různými způsoby. Zajímat nás budou pouze soubory:

1. openwrt-x86-generic-combined-squashfs.img
2. openwrt-x86-generic-combined-ext2.vmdk
3. openwrt-x86-generic-combined-ext2.img

Soubor obsahující squashfs v názvu je obrazem, který je rozdělen na 2 partice - na boot partici v ext2 (read/write) a na root partici, která je pouze read only. Obrazy s ext2 v názvu mají narozdíl od squashfs i root partici read/write - jsou totiž na souborovém systému ext2. Jedná se o "bitovou kopii", tudíž stačí rovnou spustit tento image pomocí virtualizéru KVM, nebo jej převést na VMDK (pro Vmware či VirtualBox) pomocí příkazu:

```
qemu-img convert -f raw openwrt-x86-generic-combined-squashfs.img -O vmdk
disk.vmdk
```

Spuštění obrazu pomocí virtualizéru KVM je velmi jednoduché.

```
# qemu-system-x86 openwrt-x86-generic-combined-ext2.vmdk
```

Nejjednodušší postup spuštění pomocí VMware je ve stažení vmx souboru, který je definicí virtuálního stroje - který je přiložen zde k wikině jako openwrt.vmx. Je to textový soubor, v němž je definovaný virtuální disk na řádku začínajícím na *ide0:0.fileName* . Hodnotou tohoto parametru by měl být název souboru vmdk image. Pak již stačí jen spustit vmx soubor, který si načte image a spustí virtuální stroj. Složitější postup je popsán na konci kapitoly, který ukazuje, jak vytvořit stroj o vlastní velikosti disku, než je definovaná při kompilaci OpenWRT.

Spuštění pomocí VirtualBoxu, klasickou procedurou vytvoříme virtuální stroj,

použijeme existující pevný disk - zvolíme vmdk soubor a proklikáme se na konec procedury - zatím všechny parametry kromě disku jsou volitelné. Klikneme na název virtuálního stroje, zvolíme "Sériové porty" a u jakéhokoliv z portů (alespoň jeden musí být), zaklikneme "Povolit sériový port". Pak již můžeme spustit virtuální stroj.

Skripty sestavené pro tento projekt

Apply.sh

Skript apply.sh byl vytvořen pro jednodušší zařazení našich patchů do OpenWRT. Tento skript je nutné spouštět z OpenWRT buildroot formou:

```
# sh cesta/ke/skriptu/apply.sh
```

Skript apply.sh provádí následující:

- kontroluje, zda-li je spuštěn z OpenWRT buildrootu,
- aktualizuje a nainstaluje feedy (dodatečné repozitáře balíčků pro OpenWRT),
- přidá náš repozitář do feeds.conf,
- rekurzivně vrátí do původního stavu pracovní kopii SVN repozitáře OpenWRT,
- provede patche nad OpenWRT buildrootem a jeho balíčky,
- aktualizuje a nainstaluje feedy (potřebné pro zavedení naší práce).

Autobuild.sh

Skript autobuild.sh byl vytvořen pro automatizovanou kompilaci OpenWRT. Jeho činnost spočívá v automatizovaném procesu kompilace, při němž provádí všechny vyžadované kroky, které by normálně prováděl uživatel. Nutnou podmínkou je nastavit při jeho spuštění adresář, v němž bude umístěn buildroot OpenWRT a pracovní kopie SVN repozitáře BESIPu.

[https://homeproj.cesnet.cz/projects/besip/repository/revisions/169/entry/Trunk/OpenWrt/autobuild.sh]

keyDownloader.py

Skript keyDownloader.py byl vytvořen pro jednodušší zařazování veřejných SSH klíčů do běžícího image OpenWRT. Skript ze specifické adresy stáhne na HTTP protokolu stáhne veřejné klíče do vymezeného adresáře, zvaliduje je pomocí ssh-keygen aplikace a následně je zařadí do specifického souboru, který je určen pro veřejné SSH klíče. Skript kontroluje také duplicity klíčů, tudíž se do daného souboru zařazují jen klíče, které jsou z pohledu skriptu "nové". Skript je umístěn na http://liptel.vsb.cz/mirror/besip/scripts/keyDownloader.py. Tento skript je lehce modifikovatelný. Nemá sice parametry pro příkazovou řádku, nicméně na počátku skriptu stačí změnit hodnoty, z nichž stahuje klíče a adresáře, do nichž klíče ukládá.

Portované aplikace pro OpenWRT

Aplikace, nebo jejich dílčí části, které jsme naportovali pro OpenWRT:

- Asterisk 1.8.4.4-1 (moduly pbx-lua a další..)
- lua-crypt

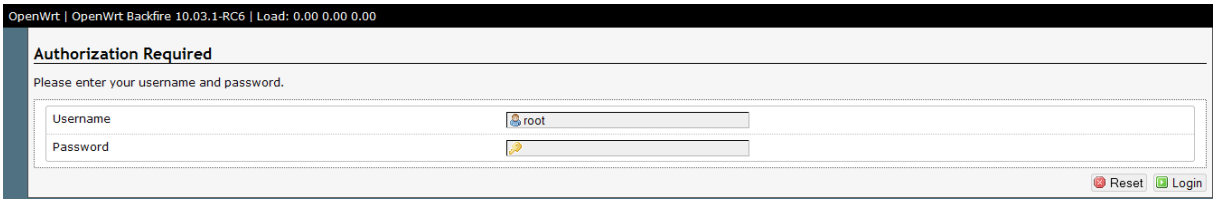
- ipqbdb
- Kamailio 3.1.4-1
- Opravy v Sqlite3 3070701-1
- Opravy v Sqlite2 2.8.17-2
- yang-utils 1.15-4-1
- netconfd 1.15-4-1
- xorp 1.8.3-1
- snortsam 2.70

2.2. OpenWRT - LuCI

Pro správu a administraci základních nastavení systému a ústředny Asterisk slouží webová rozhraní pro která byl vytvořen rozcestník, který je zobrazen na obrázku 2.

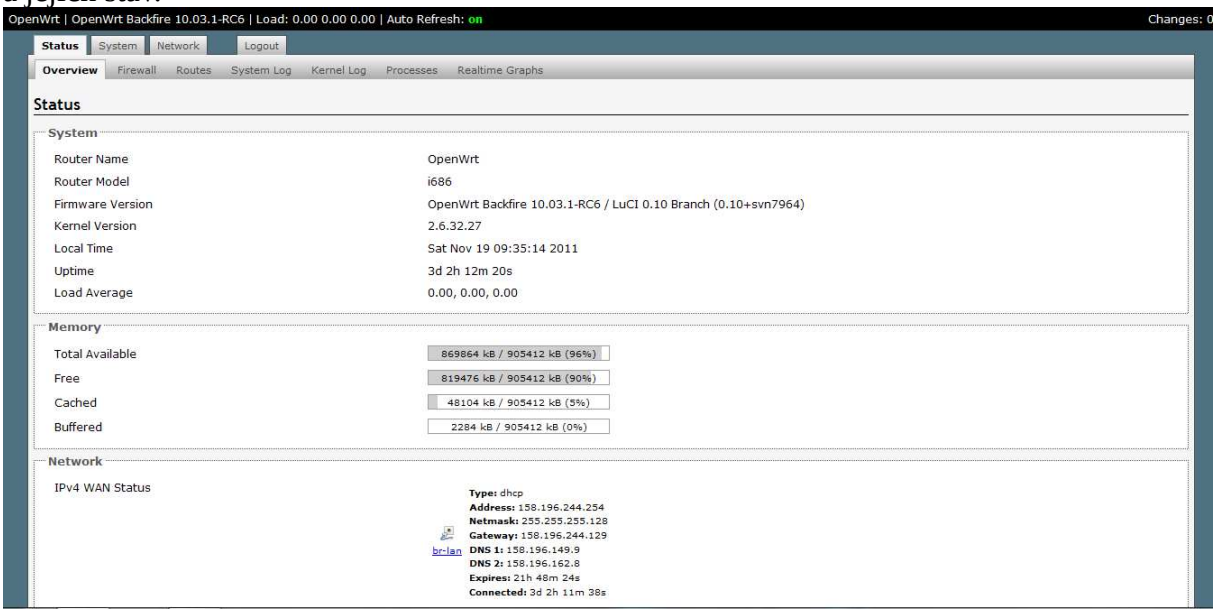
Obr. 2: Rozcestník BESIP

Pro připojení k základnímu nastavení systému BESIP slouží webové rozhraní OpenWRT – LuCI. Po spuštění se objeví přihlašovací obrazovka, kde je pole *Username a Password.*



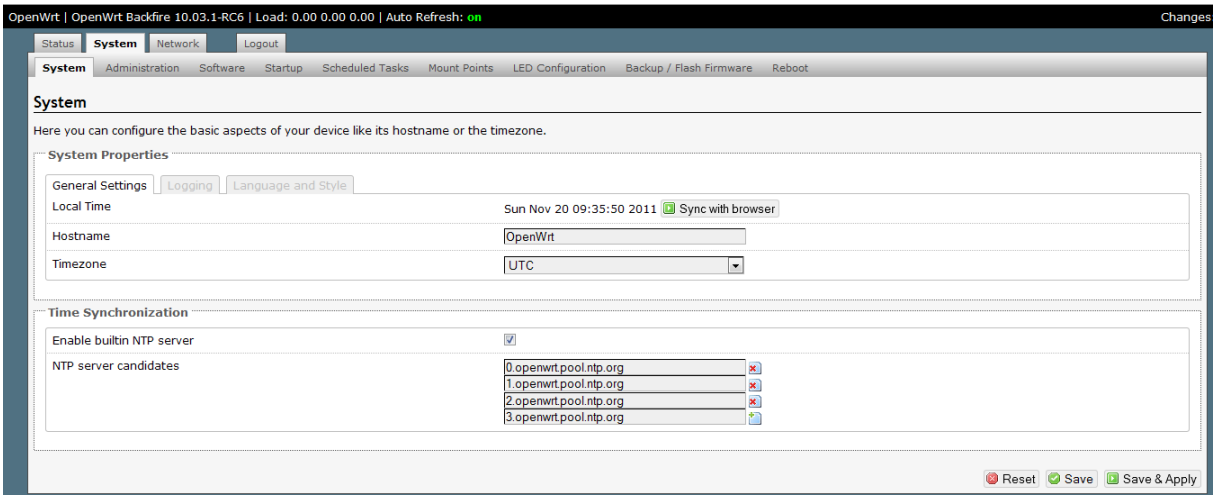
Obr. 3: Přihlašovací obrazovka pro OpenWRT – LuCI

Obrázek 4 ukazuje záložku **Status,** kde je možné mít přehled o systému, informace o firewallu, aktuálně nastavená aktivní pravidla pro směrování, přehled o záznamech činnosti systému a jádra operačního systému a také seznam aktuálně spuštěných systémových procesů a jejich stav.



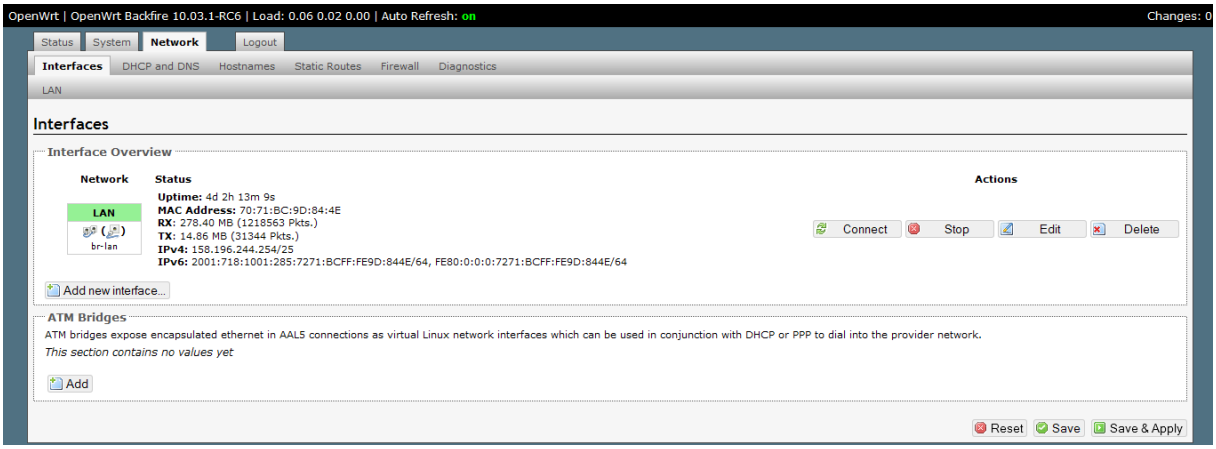
Obr. 4: Stav systému

Obrázek 5 ukazuje záložku **System,** kde jsou pole pro základní nastavení.



Obr. 5: Nastavení systému

Obrázek 6 ukazuje záložku **Network,** kde jsou pole pro síťová nastavení.



Obr. 6: Nastavení sítě

2.3. Hardware pro testování

Pro testování byl zvolen následující hardware (obr. 7), na kterém byly prováděny implementační testy a také testy spojené s dimenzováním výkonu. Hardwarové parametry jsou:

- Základní deska Intel Packton D410PT:
  - CPU: x86 Intel Atom D410 – 1,66 GHz
  - Chipset: Intel NM10 Express
  - Síťová karta: Realtek 10/100 Mbps
  - USB 2.0: 8
  - Max. Velikost RAM: 4 GB
- Paměť RAM: Kingston 1GB 667 MHz CL 5
- Pevný disk: Kingston 16GB SSD Now S100
  - Rozhraní: SATA 3 Gb/s
  - Velikost: 2,5“
  - Rychlost čtení: 230 Mb/s
  - Rychlost zápisu: 75 Mb/s
  - Trim: Ano

Obr. 7. Hardwarové zařízení

2.4. SVN, webové rozhraní, databáze a NETCONF

Cílem bylo vybrat vhodný verzovací nástroj pro systém, vytvořit tabulky v SQL databázi pro uchovávání záznamů z Asterisku a do budoucna připravit pro konfiguraci síťových zařízení NETCONF v kombinaci s YANG.

SVN

SVN je SCM systém (Source Code Management) pro správu třídy programů a postupů, které mají usnadnit spolupráci více lidí na společných projektech. Na určitém serveru je umístěn repozitář. Jednotlivý uživatelé si vytvoří lokální kopie repozitáře (checkout). Mohou si pak stahovat aktualizace repozitáře (update), nebo naopak posílat své změny do repozitáře (commit). Kazdý commit zvýší číslo revize o 1. U nasazeného SVN je stahování dat možné i bez autentizace uživatele. Modifikace je možná pouze přes SSL spojení, kde autentizace se provádí pomocí LDAP. Každý uživatel má oprávnění jen na jemu přidělené adresáře. Cílem použití SVN bylo udržovat vývoj pod kontrolou, to znamená jaké změny a kdy byli provedeny. Dále kdo tyto změny provedl a případně možnost vrátit se ke stabilní verzi.

Webové rozhraní

Klíčovým kritériem při výběru technologie pro tvorbu webového rozhraní byl zvolený programovací jazyk a funkcionalita, jež byla od rozhraní požadována. Vzhledem ke zvolené platformě OpenWRT, na níž je zařízení postaveno, bylo rozhodnuto o použití skriptovacího jazyka PHP. Tento nástroj zaručuje rychlý návrh dynamických webových stránek s veškerou požadovanou funkcionalitou, mezi kterou patřilo například volání CLI příkazů a spouštění scriptů. Jako webový server byl zvolen Lighttpd. Cílem zde bylo, rozjet na cílovém zařízení nenáročný webový server a vytvořit dynamické webové stránky bežící na tomto serveru, jež jsou určeny pro konfiguraci zařízení prostřednictvím PHP či CGI scriptů.
Lighttpd:

- nízké nároky na paměť, vhodné pro embedded zařízení
- rychlost, flexibilita, bezpečnost
- opensource pod BSD
- podpora FastCGI, PHP
- použité např. u wikipedia, youtube, meebo, …

Databáze

Pro účely uchovávání nastavení systému Asterisk bylo nutno vybrat nenáročnou embedded databázi. Z toho hlediska se jako nejlepší volbou jevilo použití databázového systému SQLite. Jedná se o relační embedded databázi s malou paměťovou náročností postavenou na standardu SQL. Byl vytvořen skript, který vytvoří tři SQL databáze potřebné pro Asterisk. Klíčovými prvky bylo získat informace ohledně použité SQL platformy a pak hodnoty, které se mají ukládat, na jejich základě se vytvořili jednotlivé tabulky. První krok byl potřebná ke zjištění syntaxe v dané databázi.

YANG a NETCONF

YANG je datový modelovací jazyk, používaný pro popis konfigurací a stavových informací, přičemž také umožňuje manipulaci s těmito daty prostřednictvím protokolu

NETCONF. Komunikace probíhá prostřednictvím zpráv ve formátu XML. YANG je nástroj pro popis dat, jež jsou následně fyzicky uložena na NETCONF serveru. YANG se také používá pro popis samotné komunikace probíhající mezi klientem a NETCONF serverem. Pomocí předem definovaných YANG modulů je tedy zaručen bezpečný a jednotný přístup ke konfiguračním parametrům, jež jsou uloženy v NETCONF serveru. Po nastudování dané problematiky bylo rozhodnuto o použití programového balíku s názvem YUMA. Tento balík obsahuje vše potřebné pro samotnou praktickou implementaci této technologie, včetně připravené klientské (yangcli - NETCONF over SSH client) i serverové (netconfd - NETCONF over SSH server) strany.

Standardy obsažené v balíku YUMA:

- Network Configuration Protocol (RFC 4741)
- NETCONF over SSH (RFC 4742)
- NETCONF Notifications (RFC 5277)
- Partial Lock RPC for NETCONF (RFC 5717)
- YANG Data Modeling Language (RFC 6020)
- Common YANG Data Types (RFC 6021)
- NETCONF Monitoring Schema (RFC 6022)
- With-defaults capability for NETCONF (RFC TBD)
- SSH2 (RFC 4252 – 4254)
- XML 1.0
- XPath 1.0
- YANG Data modeling language (RFC 6020)

Pro tvorbu konfiguračního webového rozhraní byl zvolen skriptovací jazyk PHP, takto vytvořené dynamické webové stránky budou funkčí na webovém serveru lighttpd. Server je aktuálně nasazen a webové stránky jsou aktuálně ve vývoji. Jako databázový systém byl zvolen SQLite jež byl prakticky nasazen. Byly vytvořeny tabulky dle specifikace a databázový systém je připraven pro ostrý provoz.

V rámci práce s technologiemi YANG A NETCONF, byl nasazen balík do OpenWRT s názvem YUMA, obsahující klienta s názvem *yangcli* a server *netconfd*. Protože je pro praktické použití nutno vytvořit YANG moduly ke každé konfigurovatelné části systému, byly vytvořeny na webu https://homeproj.cesnet.cz/projects/besip/wiki/NETCONF návody, tutoriály a šablony, obsahující ukázkové příklady, jež mají pomoc ostatním členům týmu v tvorbě jejich dílčích modulů.

Použití technologií YANG a NETCONF má výhodu v bezpečné, strukturované a přesně definované komunikaci při správě managementu webových prvků. Konfigurační parametry jsou bezpečně uchovávány na NETCONF serveru a veškeré požadavky a odpovědi musí dodržovat pevně definovanou strukturu, specifikovanou použitými YANG moduly. Po vytvoření veškerých požadovaných YANG modulů je cílem použít NETCONF server jako globální databázi veškerých konfigurovatelných parametrů systému. Parametry zadány uživatelem do webového rozhraní budou uloženy na NETCONF serveru a z tohoto místa také načítány pro samotné konfigurace. K takto uloženým datům lze pak přistupovat prostřednictvím jednoduchých dotazů, či provádět rychlé a snadné zálohy a obnovy konfigurací jednotlivých částí systému.

3. PBX

Jako primární, modulární a vysoce konfigurovatelná open-source ústředna pro BESIP byla zvolena ústředna Asterisk. Ve vytvořeném obrazu systému BESIP se počítá i s budoucím nasazením a použitím další open-source ústředny Kamailio, kterou již BESIP ve svém jádře obsahuje.

3.1. Asterisk

Oficiálně je Asterisk open source hybrid TDM a packet voice PBX, jedná se o IVR (Interactive Voice Response) platformu s funkčností Automatic Call Distribution (ACD). Neoficiálně jde možná o jedno z „nejsilnějších“, flexibilních a rozšířitelných řešení v oblasti integrovaného telekomunikačního softwaru. Jde tedy o kompletní open source softwarovou PBX, běžící na platformách Linux a Unix, poskytující veškeré vlastnosti, které byste očekávali od PBX. Jedná se o obecnou distribuci pod podmínkami GNU (General Public Licence). Systém je navržen tak, aby vytvořil rozhraní telefonnímu hardwaru, softwaru a libovolné telefonní aplikaci. Do systému BESIP byla naportována verze Asterisku 1.8.4.4-1.

3.2. GUI Asterisk

Asterisk GUI přináší podporu pro webovou správu softwarové telefonní ústředny Asterisk. Je vyvíjen jako Open-source projekt firmou Digium. Současná verze Asterisk GUI je 2.0 a je primárně určen pro Asterisk verze 1.6, ale měl by být kompatibilní s vyšší verzi 1.8 softwarové telefonní ústředny Asterisk. Při testování funkčnosti jsme však narazili na nekompatibilitu s OpenWRT a Asterisk verze 1.8. Z tohoto důvodu muselo být použito GUI ve verzi 2.1.0 rc1. Image je dostupný zde:

http://downloads.asterisk.org/pub/telephony/asterisk-gui/asterisk-gui-2.1.0-rc1.tar.gz

Cílem samotné implementace grafického rozhraní pro softwarovou telefonní ústřednu Asterisk bylo přinést zjednodušení ovládání a umožnit tak jednoduchou konfiguraci i pro méně technicky zdatné jedince.

Poslední verze systému BESIP přináší plně funkční ovládání softwarové ústředny Asterisk pomoci internetového prohlížeče. Byla implementována poslední verze GUI rozhraní – konkrétně verze 2.1.0 rc1. Tato verze je plně kompatibilní s poslední verzi Asterisku a taktéž s OpenWRT. Prozatím je implementováno jen základní rozhraní Asterisk GUI. Budou probíhat úpravy, které budou vycházet ze specifických požadavků tohoto projektu.

Pro instalaci je potřeba provést nad staženou složkou příkaz configure, make a make install. Poté dojde k instalaci samotného grafického rozhraní pro telefonní ústřednu Asterisk. Pro instalaci na distribuci OpenWRT je potřeba nejprve zkompilovat Asterisk GUI a poté nainstalovat pomoci příkazu opkg.

Dále je nutné nastavit konfigurační soubory v adresáři `/etc/asterisk/`. Jedná se konkrétně o konfigurační soubory `http.conf a manager.conf`. Konfigurace `http.conf` je následující:

```
[general]
```

```
enabled=yes
bindaddr=0.0.0.0
bindport=8088
prefix=gui
enablestatic=yes
```

Do konfiguračního souboru `manager.conf` je nutné přidat následující nastavení:

```
[general]
enabled=yes
webenabled=yes
port=5038
bindaddr=0.0.0.0
[bkruse]
secret=secret
read=system,call,log,verbose,command,agent,user,config,originate,read,write
write=system,call,log,verbose,command,agent,user,config,originate,read,writ
e
```

V těchto dvou konfiguračních souborech je nezbytně nutné nastavit správné uživatelské jméno a heslo a dostatečně zabezpečit přístup k těmto souborům, jelikož při nesprávném nastavení hrozí zneužití telefonní ústředny Asterisk. Asterisk GUI je plně upravitelné grafické rozhraní pro ovládání telefonní ústředny Asterisk. Konfigurace probíhá úpravou souborů v adresáři:

```
/var/lib/asterisk/static-http/
```

Tyto soubory jsou psány v jazyce html. V tomto adresáři můžete také nalézt obrázky, které doplňují samotný vzhled grafického rozhraní. Přihlašování probíhá pomoci uživatelského jména a hesla, které bylo zadáno do souboru `manager.conf` v poli `secret`. Uživatelské jméno je v tomto případě `general`.

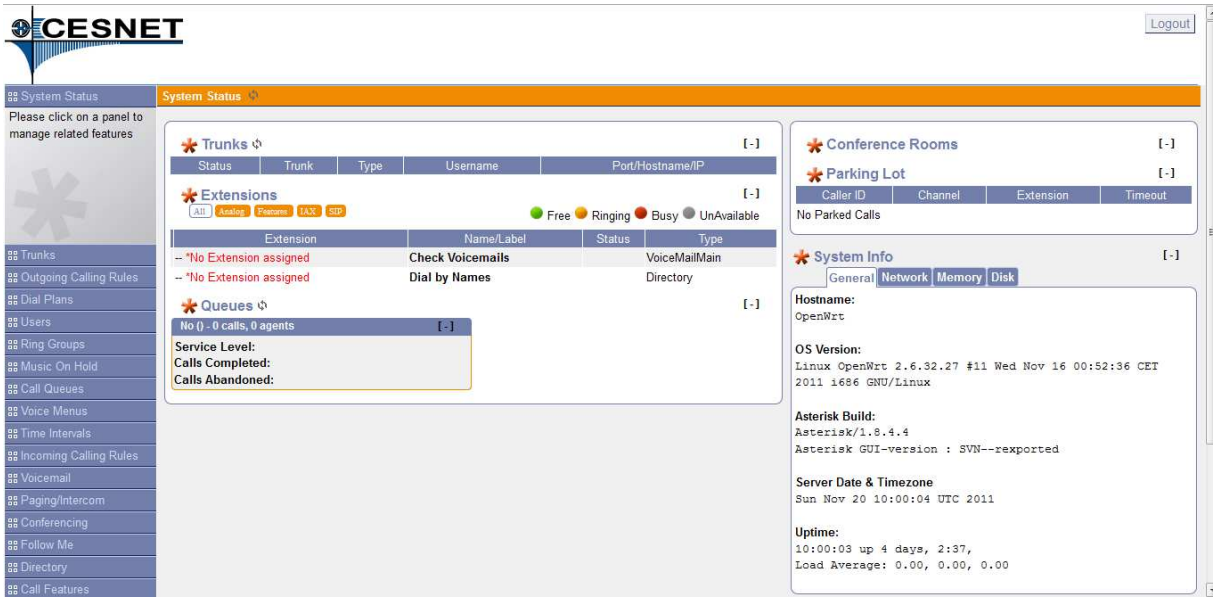
Připojení do grafického rozhraní probíhá přes základní přihlašovací obrazovku na adrese:

```
http://localhost:8088/gui/static/config/index.html
```

Pro připojení z jiného počítače je nutné zadat patřičnou IP adresu počítače, na kterém právě funguje BeSIP systém. Uživatelské jméno a heslo vychází z konfigurace, která byla nastavena v konfiguračním souboru `manager.conf`.

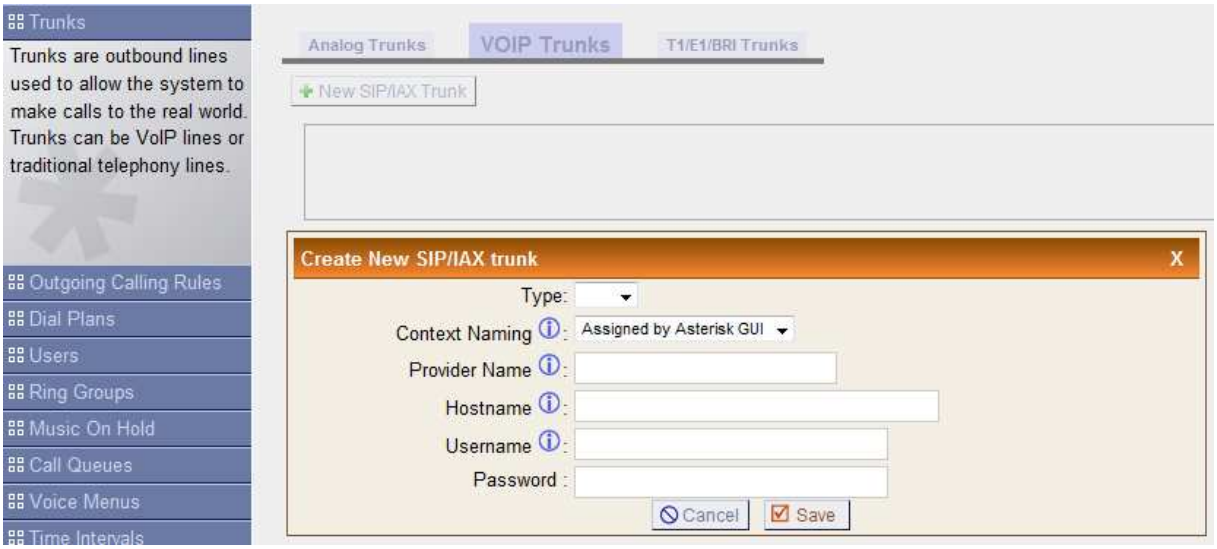
Obr. 8. Přihlašovací obrazovka pro Asterisk GUI

Možnosti samotné konfigurace softwarové telefonní ústředny jsou velmi rozsáhlé. Na následujícím obrázku jsou zobrazeny všechny volby, které je možno nastavit pomoci grafického rozhraní Asterisk GUI.



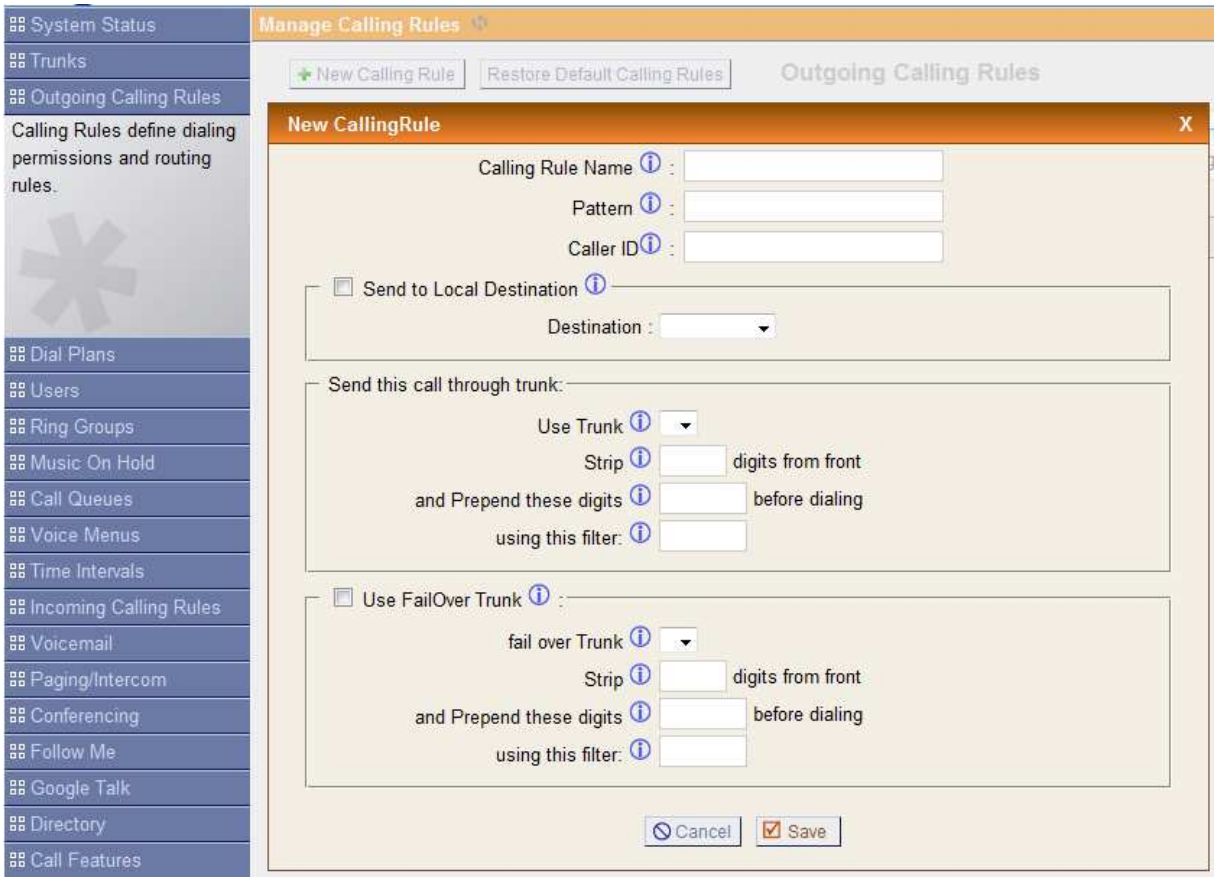
Obr. 9. Základní informace, jenž poskytuje Asterisk GUI

Na obrázku č. 9 je patrné, které informace Asterisk GUI může předat koncovému uživateli. Mezi tyto informace patří: Informace o Truncích, Záznamy v `extensions.conf`, informace o aktivních hovorech a informace o systému – Informace o síti, paměti a disku. Zobrazeny jsou však jen informace, které byly zadány do `extensions.conf` pomoci Asterisk GUI.



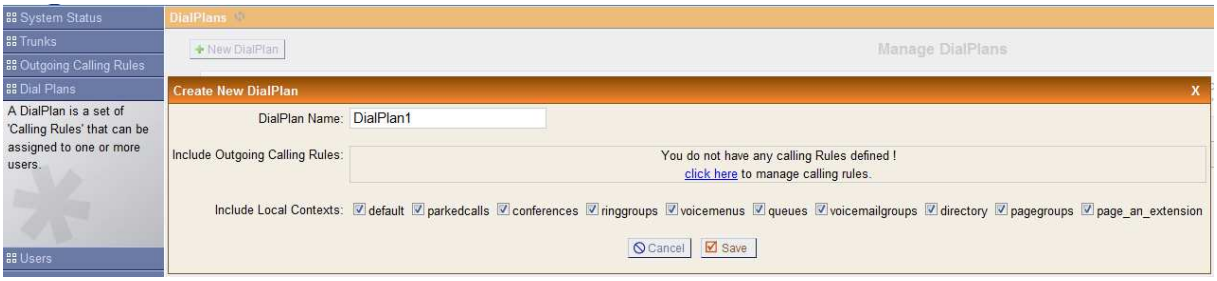
Obr. 10. Nastavení Trunků

Na obrázku č. 10 se nachází ukázka tvorby Trunků přes GUI rozhraní. Je zde možnost nastavit Trunky – Analogové, VOIP a T1/E1/BRI.



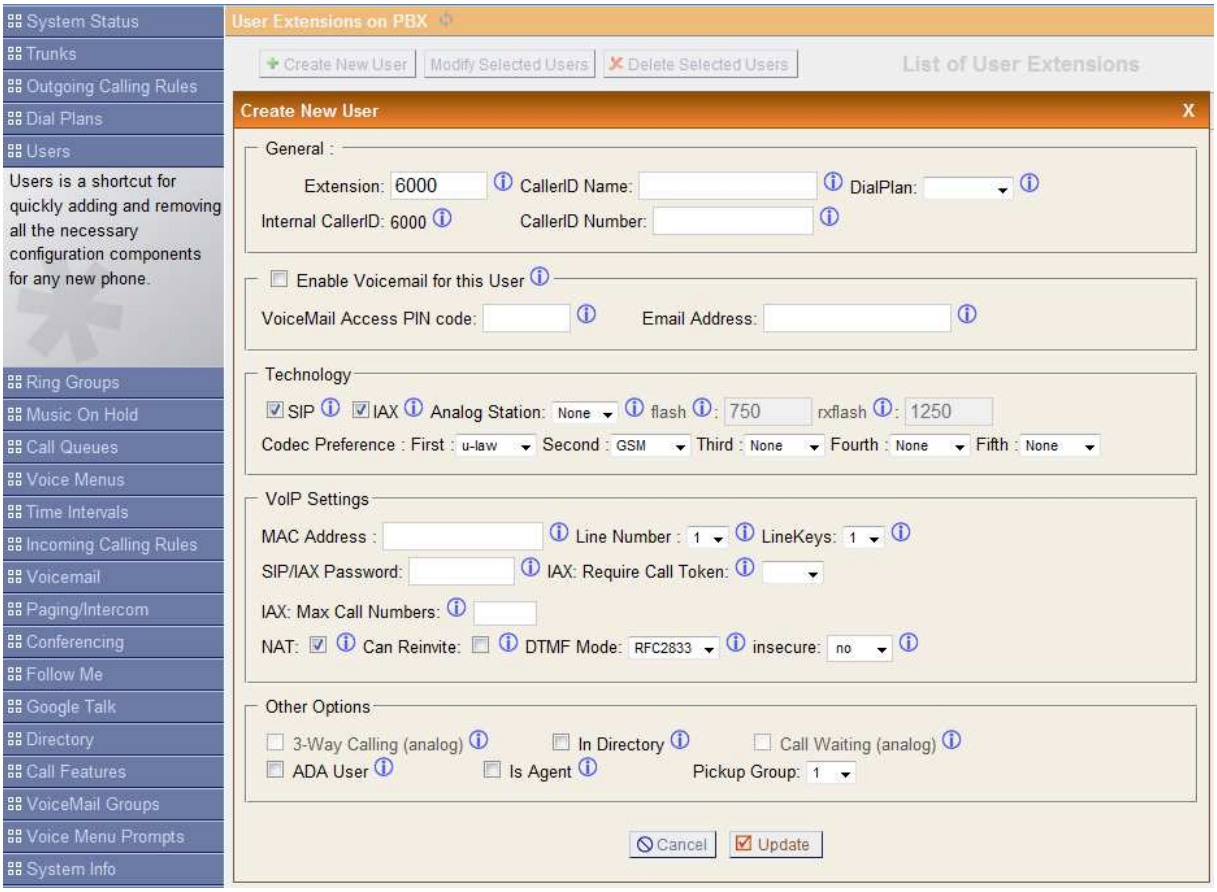
Obr. 11. Pravidla pro odchozí hovory

Pravidla pro odchozí hovory je také možno nastavovat pomoci Asterisk GUI.



Obr. 12. Nastavení Dial plánů

Dial plány je možno nastavovat v Dial plan menu. Nastavený Dial plán je také možné nalézt v `extensions.conf`.



Obr. 13. Vytvoření uživatelů

Pomocí volby Users je možno vytvořit uživatele. Při vytváření uživatele je možno nastavit specifické vlastnosti – lze nalézt na obrázku č. 13.

Obr. 14. Všechny možnosti, jenž podporuje Asterisk GUI

Na obrázku č. 14 je možno vidět všechny možnosti, které podporuje Asterisk GUI. Možnosti budou v budoucnu redukovány podle potřeb tohoto projektu.

Konfigurační adresář se všemi HTML stránkami je možno nalézt:

`/usr/lib/asterisk/static-http`

V tomto adresáři se nachází `index.html` soubor a také adresář `config`, který obsahuje výše zmíněné možnosti úpravy Asterisk GUI. Grafické prvky – obrázky apod. se nacházejí ve složce:

`/usr/lib/asterisk/static-http/config/images`

Všechny možnosti grafických úprav vychází z tohoto adresáře a z objektů, které se v tomto adresáři nacházejí.

V současné době probíhá definice vlastností, které má Asterisk GUI poskytnout koncovému uživateli systému BESIP. Jsou vybírány ty části, které by koncový uživatel mohl potřebovat a jsou také vybírány ty části, které koncový uživatel systému BESIP neupotřebí a tyto části jsou následně odebrány. Probíhá také úprava grafické stránky Asterisk UI tak, aby samotný vzhled podtrhoval samotnou podstatu celého projektu a aby co nejvíce usnadňoval orientaci při ovládání softwarové telefonní ústředny Asterisk. Konečná podoba vzejde během testování v testovacím provozu projektu BESIP. Dojde k odladění samotného grafického rozhraní přesně podle potřeb koncových uživatelů.

4. Bezpečnost

Vzhledem k velkému rozvoji v oblasti VoIP vzniklo v posledních letech velké množství implementací VoIP serverů i různých protokolů. Blíží se doba, kdy budou PSTN sítě kompletně nahrazeny. Ačkoliv nebylo zabezpečení vždy hlavním cílem, s rostoucí popularitou roste i potřeba po bezpečnosti. Projekt BESIP bude používat SIP protokol, který má některé znaky podobné s HTTP a SMTP protokoly, potenciální útočník tedy může využít již existující slabiny těchto protokolů proti SIP. Mezi nejpoužívanější útoky v poslední době patří i DoS, zejména díky své velké účinnosti a relativně jednoduché proveditelnosti.

Základním prvkem bezpečnosti bude firewall založený na open-source aplikaci iptables. Takovýto SIP server by byl ale i nadále ještě velmi zranitelný, bylo tedy nezbytné nalézt další metody, jak ještě zvýšit odolnost daného zařízení, zvláště pak proti útokům typu DoS. Dalším důležitým bodem je kromě zabezpečení vlastní služby i zabezpečení konfiguračního rozhraní. Zvláště pak tedy možnosti připojení k zařízení pomocí SSH a jeho správě pomocí CLI. Důležité je i informovat administrátora o blokovaných útocích, nejlépe pak pomocí zprávy na administrátorův e-mail.

Cílem je dosáhnout stavu, kdy bude zařízení schopné odolat nejběžnějším útokům bez narušení jeho funkcionality a to i v takovém případě, kdy v produkční síti nebudou použity žádné další bezpečnostní mechanismy a zařízení bude plně exponováno proti jakémukoliv útoku z vnější i vnitřní části sítě. BESIP server by tyto útoky měl zaznamenat, zablokovat a podat informaci o útoku bezpečnostnímu administrátoru, který může podniknout další kroky pro efektivnější blokování útoků v rámci celé produkční sítě. Protože bude bezpečnostní mechanismus implementován pouze přímo na daném zařízení, nelze se bránit proti všem typům útoků, jelikož existují hrozby, proti nimž lze vytvořit efektivní obranu pouze pomocí úpravy síťové topologie a hierarchickému systému kontroly uvnitř stávající topologie.

4.1. Zvažovaná řešení obrany před útoky

Jak bylo zmíněno výše, útoky zamezující přístup ke službě patří do skupiny nejobvyklejších útoků vůbec. Proto patří obrana proti nim k základní obraně zařízení. Otestovány a zvažovány byly následující aplikace.

Fail2ban

Nástroj umožňující blokování IP adres ve firewallu na základě skenování logovacích souborů. Dokáže kontrolovat celou řadu aplikací, v závislosti na nastavených konfiguračních souborech. V konfiguračních souborech je také obsažen regulární výraz. Pomocí tohoto regulárního výrazu následně detekuje fail2ban podezřelou aktivitu. Vlastní zablokování se provádí prostřednictvím vlastních pravidel pro iptables firewall. Blokování určité IP adresy je omezeno nastavenou dobou. Pro každou hlídanou službu lze také nastavit počet výskytů shod s regulárním výrazem. Tyto shody jsou sledovány v čase, aby bylo možné nastavit i intenzitu výskytu nežádoucích událostí. Pokud se v určitém časovém intervalu nevyskytne dostatečný počet nálezů, dojde k vynulování počítadla výskytů. Výsledkem je tedy poměrně efektivní zablokování zdroje útoku, přičemž po uplynutí nastavené doby lze opět komunikovat i s tímto zdrojem. V případě falešně detekovaného útoku může dojít k opětovnému aktivování blokované služby bez lidského zásahu.

Nevýhodou řešení fail2ban je stav, kdy není během útoku do logu zapsána žádná informace či jsou dostupné pouze strohé informace o chybě (bez vazby na IP adresu). V takovém případě nemůže fail2ban nijak zasáhnout a útok blokovat. Reakce na útok může být ovlivněna i dobou, po kterou je analyzován logovací soubor. K dalšímu prodlení dochází při použití bufferu u zápisu do logovacího souboru.

IPQ BDB

Slouží podobně jako fail2ban pro blokování uživatelů. Aplikace sleduje netfilter queue (iptables) pomocí ibd-judge daemona. Zachycené pakety následně porovnává se záznamy ve vlastní databázi – Berkeley DB. Při nalezení shody dojde k posouzení hodnoty pro blokování a aplikace rozhodne o blokaci či povolení provozu. Záznamy do databáze jsou přidávány, podobně jako i v případě fail2ban, na základě výskytu hledaného výrazu v logovacím souboru. Záznam do databáze lze přidat i prostřednictvím CLI rozhraní. Algoritmus rozhodující o blokování funguje následovně. Pro každé hledané pravidlo je nutné nastavit počet výskytů, po nichž dojde k jistému zablokování provozu. Podle tohoto se následně vypočítá procentuální šance na zablokování, která se zdvojnásobuje při každé detekci nastaveného pravidla. Nežádoucí provoz je tedy velmi rychle zablokován, přičemž současné řešení poskytuje i prostor pro částečně nežádoucí chování – tzv. šedá zóna, kdy dojde jen částečnému blokování (ovšem s postupujícími nálezy se pravděpodobnost blokování značně zvyšují).

V průběhu času se hodnoty pravděpodobnosti zablokování samy snižují, systém se tedy opět navrátí k původnímu neblokujícímu stavu, pokud z dané IP adresy již nepřichází žádný útok. Za nastavený časový úsek se pravděpodobnost blokace sníží o polovinu. I přes vhodněji navržený přístup pro blokování trpí i toto řešení podobným problémem jako fail2ban. Pokud není v logovacím souboru informace o útoku nebo je pouze strohá, nelze daný útok nijak blokovat.

SSI (Snort, SnortSam, Iptables)

Zde se nejedná o jedinou aplikaci, ale o vzájemnou kombinaci několika řešení. Výsledkem je pak IPS systém na bázi IDS systému Snort. Zdrojové kódy Snortu se musí opatchovat, aby bylo možné komunikovat s aplikací SnortSam, která umožňuje zasahovat do pravidel iptables, ale i konfigurací routerů, ACL listů cisco routerů, PIX a Juniper firewallů, …

Vlastní žádosti o blokování obsluhuje snortsam server, samozřejmostí je i šifrovaný přenos těchto žádostí. Snortsam podporuje také nastavení whitelistu, tedy seznamu nikdy neblokovaných IP adres. Vyhodnocení blokování probíhá jinak než v předchozích případech. Pro Snort je třeba upravit pravidla, která detekují daný nežádoucí provoz tak, aby SnortSam získal informace o zdroji a čase blokování. SnortSam z těchto informací vytvoří pravidlo pro iptables a sám sleduje další výskyty a čas trvání události. Pokud za nastavený interval nepřijde žádná další žádost o blokaci, dojde k povolení tohoto provozu, obdobně jako i u předchozích řešení.

Mezi výhody tohoto řešení patří především široké možnosti nastavení detekce nežádoucího provozu, tedy důkladnější kontrola a blokování většího množství útoků. Díky těmto vlastnostem bylo rozhodnuto o použití právě tohoto posledního řešení pro zabezpečení BESIP.

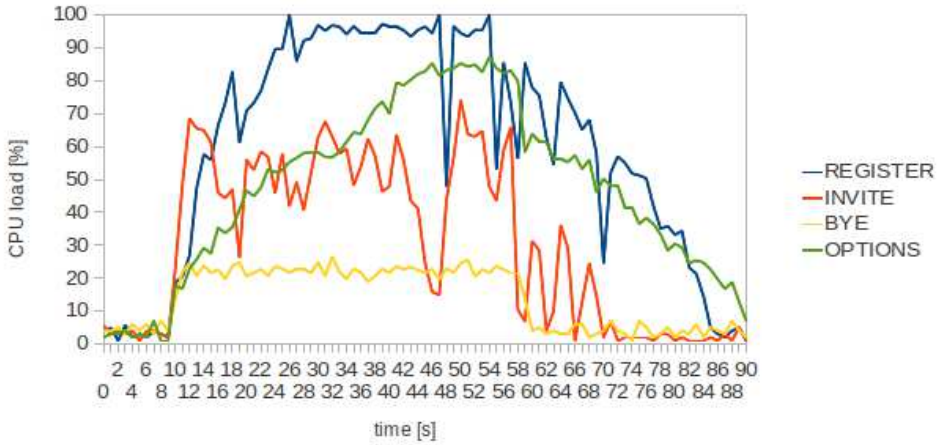
4.2. Metodika řešení

Obrana BESIP serveru (dále jen server) spoléhá na open-source firewall iptables. Tento firewall kontroluje veškerý provoz mezi serverem a ostatní sítí. Povolena je pouze komunikace na vyhrazených portech – tedy komunikace s webovým rozhraním serveru, ssh a porty potřebné pro provoz SIP ústředny Asterisk. Komunikace serveru s okolím není nijak omezena, může tedy komunikovat s jakýmkoliv zařízením. Standardně je také zapnuta ochrana proti SYN flood útoku.

V produkčním nasazení se ale mohou seznamy aktuálně blokovaných provozů často měnit, zejména pak díky aplikaci snortsam, která může do iptables dynamicky přidávat další pravidla. Konfigurace firewallu může být provedena rozdílnými způsoby, avšak pouze prostřednictvím CLI, typicky pomocí ssh připojení k serveru. Z webového rozhraní lze pouze deaktivovat IPS systém, firewall toto nastavení neovlivní. Nejjednodušší možností nastavení firewallu je využití UCI firewall rozhraní, které zjednodušuje nastavení iptables. Konfigurační soubor UCI firewallu je uložen v **`/etc/config/firewall`** a obsahuje sadu pravidel, které nastavují iptables. Především se jedná o povolení provozu pro potřeby BESIP serveru.

4.3. SSI

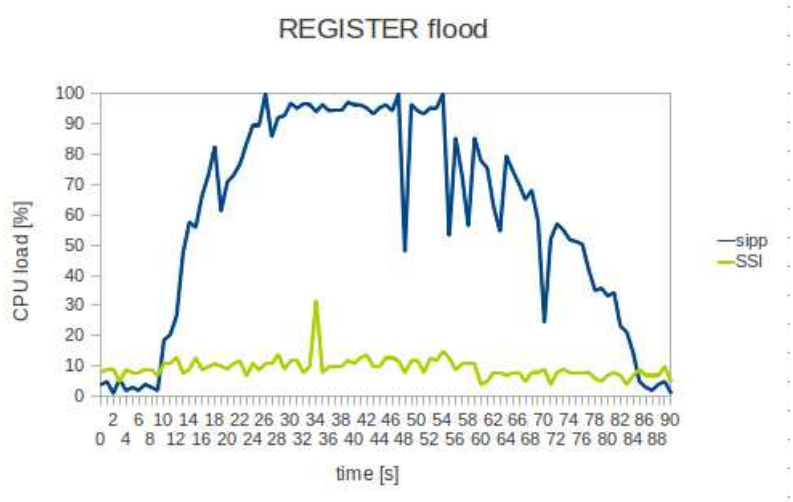
Řešením problematiky DoS útoků je nasazení IPS systému na bázi IDS Snort popsané výše. Pro korektní práci tohoto systému však musela být provedena řada testů vůči danému serveru. Testy se zaměřovaly především na identifikování nejúčinnějších metod způsobení DoS. K těmto testům byly použity různé nástroje, tak aby získaná data co nejlépe reflektovala současný stav. Testování se zaměřilo na několik kategorií, přičemž první z nich spočívala k dosažení maximálního zatížení na testovacím serveru. Testovány byly jednotlivé SIP zprávy zasílané vždy ve stejné míře zpráv za sekundu (viz. obr. 15). Útok začíná v čase 10s, končí v době 60s. Zbývající část grafu znázorňuje schopnost serveru vyrovnat se s útokem.



Obr. 15: Efektivita útoku pomocí různých zpráv

Z provedených testů byly identifikovány kritické zprávy, zvláště pak REGISTER a OPTIONS, které jsou pro DoS cílený na vyčerpání CPU nejvíce vhodné. Výsledky byly

použity pro vytvoření pravidel v aplikaci Snort. Díky těmto pravidlům je případný útok detekován a jsou zaslány nezbytné informace aplikaci SnortSam, která zprostředkuje zablokování pomocí pravidel iptables. Základní doba, po kterou bude útočník blokován je nastaven na 10 minut. Po uplynutí této doby dojde k opětovnému povolení provozu (pokud tedy útok z této adresy nepokračuje, pak zůstává daná adresa nadále blokována). Účinnost bezpečnostního mechanismu SSI byla následně opět testována, přičemž výsledky potlačení útoku jsou velice dobré. Protože všechny aplikace běží na daném serveru, nedochází téměř k žádnému zpoždění a útok je téměř okamžitě zablokován. Zatížení procesoru při útoku bez obranného mechanismu a s obranným mechanismem shrnuje obrázek 16, znázorňující průběh útoku zprávami REGISTER. Modrá křivka znázorňuje průběh útoku bez SSI, zelená křivka pak zatížení při aktivní ochraně kombinací SSI.



Obr. 16: Účinnost obrany při REGISTER flood útoku

Vyšší zatížení během klidové fáze je způsobeno právě běžícími aplikacemi Snort a SnortSam. Uživatelské rozhraní neumožňuje žádný zásah do nastavení detekčního mechanismu. Všechny potřebná pravidla jsou již uložena a připravena pro použití. Spuštění bezpečnostního mechanismu probíhá při bootování OpenWRT, zvlášť vytvořeným skriptem, jenž nakonfiguruje a spustí aplikace Snort i SnortSam.

```
#!/bin/bash
# nastavit promenne dle aktualniho stavu
INTERFACE="eth0"
SSCONF="/etc/snortsam.conf"
SSLOG="/var/log/security/snortsam.log"
SNORTCFG="/etc/snort/snort.conf"
SNORTTMP="/etc/snort/snortconftemp"
SECLOG="/var/log/security"
SNORTLOG="/var/log/snort"

WANIP=$(ifconfig $INTERFACE | grep 'inet addr:' | cut -d: -f2 | awk '{print $1}')

echo $WANIP
#1 pripravit snortsam config - smazat stary, vytvorit novy dle wanip
if [ -f $SSCONF ]
then
rm -f $SSCONF
echo "old $SSCONF deleted, creating new conf file"
else
echo "creating ssa configuration file"
fi
echo "defaultkey besipsecurity" >> $SSCONF
echo "accept $WANIP, besipsamagent" >> $SSCONF
echo "logfile $SSLOG" >> $SSCONF
echo "loglevel 3" >> $SSCONF
echo "daemon" >> $SSCONF
echo "iptables $INTERFACE syslog.info" >> $SSCONF

#1.5 kontrola existence log slozek
if [ ! -d $SECLOG ]
then
mkdir $SECLOG
fi
if [ ! -d $SNORTLOG ]
then
mkdir $SNORTLOG
fi
```

```
#2 spustit snortsam
echo "Starting SSA!"
snortsam
#3 spustit snort - tj. vygenerovat snort.conf
echo "Starting IDS Snort"
if [ -f $SNORTTMP ]
then
echo "snort temp config found"
rm -f $SNORTTMP
fi
echo "recreating snort.conf"
cp $SNORTCFG $SNORTTMP
rm -f $SNORTCFG
echo "var SIP_PROXY $WANIP" >> $SNORTCFG
echo "#portvar SIP_PORT 5060?" >> $SNORTCFG
tail +3 $SNORTTMP >> $SNORTCFG
echo "starting snort deamon"
#snort -D
snort -i $INTERFACE -c /etc/snort/snort.conf -D
```

Obr. 17: Startup skript

Pokročilá konfigurace Snort

Pro zkušené uživatele zůstává možnost úpravy stávajících pravidel obdobně jako u aplikace Snort. Soubory s pravidly pro blokování obsahuje soubor **`/etc/snort/rules/local.rules`.** Pro aplikování změn je třeba restartovat Snort. Pro upravení konfiguračního nastavení může být editován i konfigurační soubor **`/etc/snort/snort.conf`**, dodržujte však zásady uvedené v tomto souboru. Při spuštění totiž dochází znovuvytvoření tohoto souboru a případné změny by se nemusely korektně zachovat i po restartu celého systému.

SSH

K serveru bude možné přistupovat i pomocí SSH, zabezpečení tohoto přístupu by tedy nemělo být opomenuto. Pro přihlášení k zařízení je nutné využít certifikátu, nelze se přihlašovat pouze kombinací uživatelského jména a hesla. Pokud tedy potřebujete k zařízení přistupovat prostřednictvím SSH, je nutné se nejdříve přihlásit přímo na fyzickém zařízení a přidat svůj certifikát do souboru **`/root/ssh/authorized_keys`**. Klíče lze vygenerovat například pomocí příkazu:

```
ssh-keygen -t rsa
```

Případně lze i specifikovat délku klíče pomocí parametru b. Nyní se lze přihlásit k serveru pomocí soukromého klíče.

```
ssh -l root -i privateKey <Besip IP>
```

Firewall

Operační systém OpenWRT obsahuje už připravený firewall postavený na bázi iptables. Řešení SSI bude využívat právě tento firewall k blokování útoků, využijeme tedy jeho přítomnosti v systému a pouze upravíme již existující pravidla.

Konfiguraci firewallu lze provádět pomocí vlastních příkazů iptables, openwrt však obohatilo systém o další možnost konfigurace firewallu, a to pomocí UCI rozhranní. Konfiguraci firewallu tedy nalezneme v souboru **`/etc/config/firewall`**. Tento soubor obsahuje pouze popis základních pravidel, které jsou při startu operačního systému přeloženy na pravidla pro iptables. Protože je OpenWRT zamýšlen jako OS pro nasazení uvnitř routerů, odpovídá této situaci i výchozí nastavení firewallu, které bylo nutné přizpůsobit realitě, kdy bude zařízení sloužit jako SIP server.

Pro potřeby besipu byl tedy vytvořen nový konfigurační soubor, stále využívající UCI rozhranní, tak aby byla zachována možnost další konfigurace firewallu (pomocí luci nebo prostřednictvím editace konfiguračního souboru). Z původních zón zůstává zachována pouze jedna, jsou přidány pravidla povolující provoz jen na nezbytně nutných portech. K těmto pravidlům jsou následně dynamicky přidávány další pravidla pomocí snortsamu.

Besip server připravený pro produkční nasazení obsahuje již nakonfigurovaný firewall, poskytující základní stupeň ochrany. Toto řešení je dále doplněno o IPS systém dynamicky reagující na stávající situaci v síti, umožňující blokování nežádoucího provozu. Řešení obrany proti DoS útokům je důležité zejména díky jejich velké oblíbenosti a jednoduchosti provedení. Pro zařízení typu malého SIP serveru lze také očekávat tento typ útoku jako nejpravděpodobnější. Vzhledem k omezenému výpočetnímu výkonu zařízení je také útok tohoto typu pro server mnohem nebezpečnější, než proti serveru s větším výpočetním výkonem.

O případných útocích může být informován uživatel pomocí e-mailu a při přihlášení do systému. Přístup k danému zařízení je možný prostřednictvím SSH a celý server tak lze konfigurovat i přes CLI rozhraní. Protože se některé z konfiguračních souborů vytváří dynamicky, je nutné vždy sledovat případné instrukce uvnitř těchto souborů. Přihlášení prostřednictvím SSH vyžaduje použití certifikátů, je tedy nutné před prvním vzdáleným připojením tyto certifikáty na zařízení přidat. Použitím přihlašování za využití certifikátů dále zvyšuje bezpečnostní úroveň serveru.

Vlastní zařízení je schopné odolávat zmíněným útokům, u nichž je to z povahy útoku možné. Při produkčním nasazení by měly být doplněny další bezpečnostní mechanismy a sledovány informace o útocích na server. Doporučujeme používat vždy poslední stable verzi řešení BESIP obsahující nejaktuálnější bezpečnostní opatření.

5. Monitoring

Cílem monitoringu v BESIP projektu je sledování (lépe řečeno odhadování) kvality řeči úspěšně sestavených hovorů. Monitoring kvality řeči je důležitou, avšak často velmi přehlíženou součástí nejen IP telefonie. Nasazení prostředku pro sledování kvality řeči přímo do komunikačního prostředku, kterým je VoIP ústředna může sloužit ke spoustě věcí, jako je statistický přehled kvality sestavených hovorů, indikace administrátora při rapidním dlouhodobějším poklesu kvality, podávání reportů koncovým uživatelům (zákazníkům) a jiné.

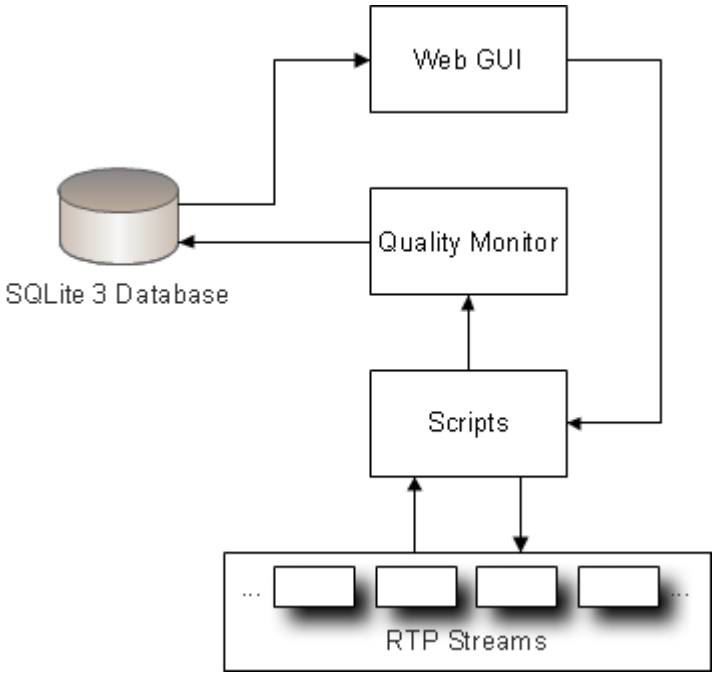
5.1. Kvalita řeči

K realizaci VQM (Voice Quality Measurement) modulu v BESIP projektu byla nejprve provedena analýza stávajících řešení, avšak nebyl nalezen žádný kompletní prostředek, který by splňoval všechny zadání projektu, tedy, aby šlo o otevřené open-source řešení a dále aby bylo možné monitorovat všechny hovory v reálném čase. Zvláště požadavek ke sledování kvality řeči v reálném čase sebou přináší nejvíce restrikcí, co se týče výběru vhodného algoritmu či modelu pro stanovení kvality řeči. Touto možností automaticky odpadá použití všech subjektivních metod, které jsou založeny na přidání lidské perspektivy do komunikačního řetězce, tedy by bylo potřeba, aby uživatel připojený k naší ústředně po každém hovoru subjektivně ohodnotil, jak byl s výsledkem spokojen. Je jasné, že takováto možnost není realizovatelná z několika hledisek, jako je například ochota uživatelů toto hodnocení provést a dále samotná subjektivní složka samotného hodnocení a další. K použití nám tedy zůstali pouze objektivní metody, avšak k samotné realizaci nejsou všechny dostupné objektivní hodnoty vhodné. Samotné objektivní hodnocení kvality řeči se dělí na dvě kategorie na intrusivní a neintrusivní podle dostupnosti k „původnímu“ hlasovému vzorku. V případě použití intrusivních metod je nutné tento původní vzorek před přenesením přes komunikační řetězec vlastnit a dále také samozřejmě stejný obsah vlastnit po přenesení přes samotný komunikační systém. Je tedy jasné, že ani tato možnost nemůže být využita hlavně z důvodu nemožnosti přístupu k vzorkům na konci a začátku spojení. Na výběr nám tedy zůstalo použít některou z objektivních neintrusivních metod.

V našem řešení jsme přistoupili k použití výpočetního matematického modelu zvaného E-model, který je popsán v doporučení G.107 vydaném studijní skupinou SG12 patřící pod ITU-T. Jak již bylo zmíněno, jde o výpočetní model, který ke svému stanovení kvality nepotřebuje pracovat se vzorky řeči, nýbrž v sobě skrývá parametry samotného komunikačního řetězce, jako je zpoždění, ztrátovost, vlivy kodeku a jiné. Některé tyto parametry jsme schopni z komunikačního systému za použití vhodných prostředků získat, zatímco některé ostatní není možno získat přímo, ale je nutné je odhadnout, tedy je nutno použití jakéhosi zjednodušeného E-modelu a výsledkem není hodnota *konverzační* kvality, nýbrž *poslechové.*

Celkové řešení monitorovacího systému v projektu BESIP je sestaveno z několika

různých open-source komponent a také z části, která byla přímo vyvinuta za tímto účelem, aby splňovala dané požadavky. Struktura systému je patrná z obrázku č. 1. Samotný systém se skládá ze tří logických komponent, kterými jsou – webové rozhraní sloužící k obsluze (Web GUI), skriptová část (Scripts), která řídí samotnou obsluhu získávání informací nutných k výpočtu za pomocí E-modelu a v neposlední řadě je to samotná část Quality Monitor, která v sobě obsahuje jak samotný výpočet, tak také zpracování dat získaných za pomocí skriptů. V přehledu je také patrná SQLite3 databáze, která slouží k ukládání získaných výsledků, což nám usnadňuje pozdější práci s daty, jako je zobrazení ve webovém prostředí či vymazání některých dat.



Obr. 18: Přehled logické struktury monitoringu kvality řeči.

Webové rozhraní je hlavní částí uživatelské interakce s monitorovacím nástrojem projektu BESIP. V defaultní konfiguraci je monitorovací nástroj vypnut a pomocí intuitivního rozhraní jej může obsluha ústředny zapnout. Tato část monitorovacího nástroje také slouží jako prostředek k zobrazení naměřených a spočtených výsledků. Struktura prezentovaných dat je následující:

```
Zdrojová IP – Cílová IP – Kvalita MOS – Kvalita R-faktor – Použité kodeky
```

Pro případnou prezentaci můžeme uvést příklad výstupu pro několik hovorů:

```
192.168.21.3    192.168.21.6    61.69     3.01    G.711/G.711
192.168.21.3    192.168.21.1    93.36     4,41      G.711
```

Samotné webové rozhraní je napsáno zcela za pomoci skriptovacího jazyka PHP neboť ke spuštění nebo zastavení monitorovacího systému je zapotřebí spouštět skripty pro shell operačního systému OpenWRT. Zároveň také, jak již bylo v úvodu zmíněno, jsou data uložena v SQLite3 databázi, tak nám použití jazyka PHP usnadní manipulaci s těmito daty.

Spuštění skriptů je inicializováno pomocí webového rozhraní monitorovacího nástroje samotným zapnutím monitorování. Prakticky to znamená spuštění zachytávání síťového provozu pomocí nástroje *tshark*, kde je aktivován RTP filtr. Použití RTP filtru nám velmi usnadní práci s RTP streamy neboť již dokáže z těchto streamů dostat některá důležitá statistická data (ztrátovost paketů, jitter) a data jiná (zdrojová/cílová IP, kodek) potřebná k výpočtu E-modelu. Jeden z příkazů, k zapnutí samotného tsharku má následující strukturu:

```
tshark -a duration:900 -qp -z rtp,streams > output.log
```

Nasbíraná data jsou bohužel uložena v textové podobě až po ukončení samotného tshark systémového procesu, je tedy nutno z tohoto důvodu a také důvodu šetření místa na disku (sbíraná data před vyhodnocením RTP filtrem mohou mít i několik GB v závislosti na objemu síťového provozu – sestavených spojení) celý proces opakovat s určitou periodou. Za vhodnou periodu bylo zvoleno 15 minut (duration:900). Jakmile jsou data získaná za pomoci tsharku a RTP filtru k dispozici může skript spustit výpočet kvality, který je hlouběji popsán v další části. Po zpracování získaných dat jsou z disku vymazány z důvodu šetření místa.

Část Quality Monitor získala svůj název podle jména třídy v programovacím jazyce Java, ve kterém byl napsán a zpracován samotný algoritmus výpočtu E-modelu. Nejprve je však nutné zpracovat data získaná pomocí tsharku a vybrat z nich jen pro nás podstatné věci. Po rozparsování dat je pro všechny hovory jednotlivě spočtena kvalita pomocí zjednodušeného E-modelu. Struktura matematické relace E-modelu vypadá dle doporučení následovně:

$$R = Ro - Is - Id - Ie - eff + A$$

Ro představuje základní odstup signálu od šumu, ve kterém jsou zahrnuty všechny možné druhy šumu včetně šumů způsobenými elektrickými obvody zařízení a šumy způsobených na vedení. *Is* zahrnuje všechny možné kombinace zhoršení, které se objevují více či méně souběžně s užitečným hlasovým signálem. Faktor *Id* představuje všechna zhoršení, která jsou způsobena různými kombinacemi zpoždění. *Ie-eff* zahrnuje zhoršení způsobené užitím určitého hlasového kodeku, projevením možné ztrátovosti paketů, a jeho odolnosti vůči ztrátovosti. V poslední řade nám parametr *A* celkovou výslednou kvalitu mírně zlepšuje, neboť např. v případě satelitního telefonu je uživatel na hovor více soustředěný než při běžném použití pevného terminálu v domácím prostředí.

Pro naši aplikaci však nepotřebujeme stanovovat všechny tyto parametry a vystačíme si pouze s jakousi zjednodušenou formou E-modelu. Zjednodušení E-modelu spočívá v nahrazení některých jeho proměnných statickými-defaultními hodnotami. Výsledná relace poté vypadá následovně:

$$R = 94{,}7688 - 1{,}4136 - 0 - Ie - eff + 0$$

Parametr Ie-eff v sobě, jak již bylo zmíněno, obsahuje vliv kodeku a ztrátovosti paketů a spočte se následovně:

$$Ie-eff = Ie + (95 - Ie) \cdot \frac{Ppl}{\frac{Ppl}{BurstR} + Bpl}$$

Ie je tabulková hodnota dle použitého kodeku, Ppl značí ztrátovost paketů, BurstR charakter ztrátovosti (zhlukovost), kde je tato hodnota pevně nastavena na 1 a parametr Bpl značí odolnost hlasového kodeku vůči ztrátovosti (hodnota opět nastavena fixně na 1).
Výstupem E-modelu je hodnota zvaná R faktor. Abychom byli schopni získat hodnotu MOS, která je jakousi unifikovanou jednotkou všech metod hodnocení kvality řeči, je potřeba pomocí matematických relací hodnotu R faktoru přepočítat:

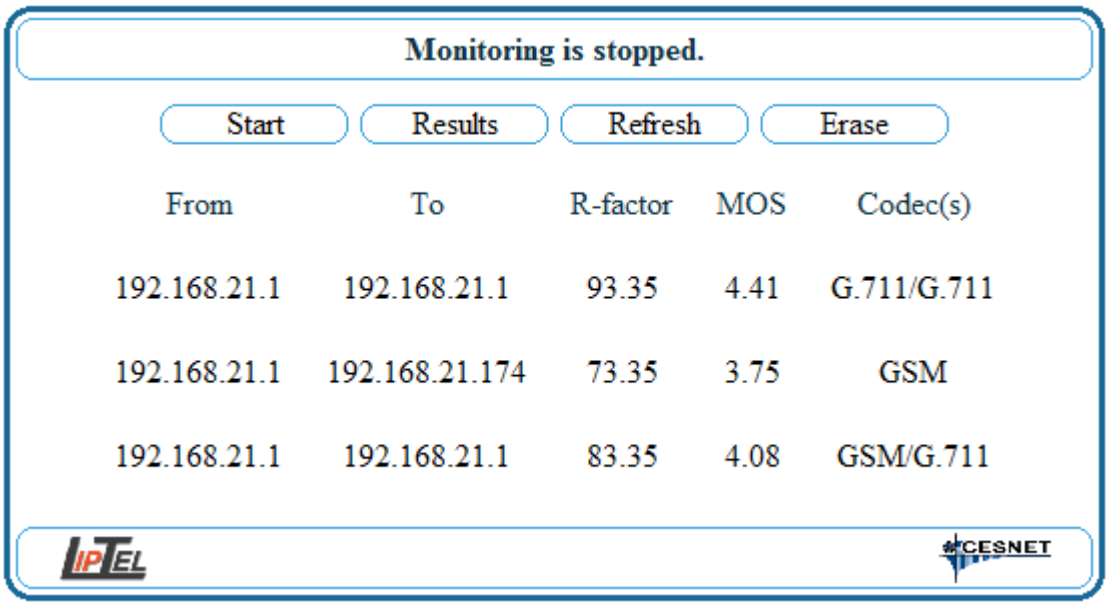
$$MOS = \begin{cases} 1, R < 6{,}5 \\ 1 + 0{,}035 \cdot R + R \cdot (R - 60) \cdot (100 - R) \cdot 7 \cdot 10^{-6}, 6{,}5 \leqslant R \leqslant 100 \\ 4{,}5, R > 100 \end{cases}$$

Všechny získaná data, jak byly prezentována v kapitole 2.1. jsou poté ukládána do SQLite3 databáze. Pro komunikaci s databází slouží open-source Java API *SQLiteJDBC*, která je do aplikace importována ve formě jar balíčku.

Jak již bylo výše zmíněno, ovládání monitoringu kvality řeči je řešeno pomocí webového GUI, které je přístupné z hlavního rozcestníku BESIP managementového rozhraní. Ukázka uživatelského rozhraní je uvedena na obrázku č. 19.

V horní části grafického rozhraní se nachází indikátor stavu, zda je samotný monitoring na ústředně aktivní (Monitoring is running…), či je aktuálně vypnutý (Monitoring is stopped.). Samotné ovládání modulu je realizováno pomocí čtyř tlačítek nacházející se pod indikátorem stavu aplikace.

Start/Stop – V závislosti na aktuálním stavu (zapnuto/vypnuto) je první tlačítko buď Start, nebo Stop. Jak již samotný název napovídá, slouží k samotné aktivaci monitoringu, popřípadě k jeho zastavení.
Programátorské vysvětlení – Klinutím na tlačítko Start se pomocí Java skriptu a Ajaxu na straně serveru spustí php skript *_start.php*, který spustí na pozadí (pomocí nohup) shell skript *_start.sh*, který aktivuje samotný proces tsharku, který se vždy po 600 sekundách (10 minut) ukončí a získaná data z RTP filtru se zapíšou do souboru *streams.log*. Při tomto stavu nedochází vždy po každém ukončení aplikace k vypočítání výsledků a uložení do databáze, ale vždy proběhnou 3 takovéto rotace, než dojde k automatickému výpočtu a stávající soubor *streams.log* se vyčistí. Kliknutím na tlačítko Stop se pomocí Java skriptu a Ajaxu na straně serveru spustí php skript *_stop.php*, který spustí shell skript *_stop.sh*, který ukončí procesy tsharku a sh _start.sh pomocí příkazu kill.



Obr. 19: Ukázka webového GUI modulu monitoringu kvality řeči.

Results – tlačítko sloužící k manuální inicializaci výpočtu E-modelu. Samotná aplikace vypočítá výsledky automaticky po 3 proběhnutích cyklu tsharku, tedy po 30 minutách, tímto tlačítkem však můžeme výsledky nechat spočítat a zobrazit okamžitě. Tlačítko aktualizuje stránku, avšak především proběhlo-li větší množství hovorů po posledním výpočtu, bude potřeba stránku aktualizovat několikrát, aby se všechny výsledky zobrazily korektně.

Programátorské vysvětlení - Kliknutím na tlačítko Results se pomocí Java skriptu a Ajaxu na straně serveru spustí php skript *_perform.php*, který spouští shell skript *_perform.sh*. Pokud je na pozadí monitoring aktuálně běžící, php skript nejprve spustí *_stop.sh*, poté provede výpočet (*_perform.sh*) a poté monitoring znovu obnoví pomocí _start.sh. Je-li monitoring vypnutý, provede se pouze část s *_perform.sh*. Shell skript *_perform.sh* v sobě obsahuje spuštění Java aplikace QualityParser, která byla za tímto účelem napsaná a posléze vymaže obsah souboru, kde jsou uloženy informace z tsharku (*streams.log*), neboť tyto data již byla zpracována.

Refresh – Tlačítko Refresh pouze obnoví stránku. Má naprosto stejnou funkci jako klávesa F5 nebo tlačítko obnovení v prohlížeči. Programátorské vysvětlení – Provede refresh stránky pomocí Java skriptu.

Erase – Provede vymazání všech záznamů v databázi. Stisknutím toho tlačítka nedojde k automatickému obnovení stránky a stránku je nutno obnovit ručně (např. pomocí tlačítka Refresh). Programátorské vysvětlení - Kliknutím na tlačítko Erase se pomocí Java skriptu a Ajaxu na straně serveru spustí php skript *_erase.php*, který pomocí SQL příkazu vymaže všechny záznamy z databáze. V hlavní (spodní) části grafického prostředí se zobrazují získané výsledky ve formátu, jak bylo avizováno.

Nástroj pro monitorování kvality řeči, který je v podobě pro projekt BESIP stačí k základnímu přehlednému monitorování kvality řeči, která může sloužit pro přehled kvality sestavovaných hovorů, ale také jako pomůcka k řešení případných problémů netýkajících se pouze samotné VoIP ústředny. Do budoucna je možno tento nástroj rozšířit v oblasti podpory

více kodeků (v současném stavu dokáže pracovat pouze s kodeky G.711, GSM a G.729) a v neposlední řadě také zahrnout více faktorů, které mají na výslednou kvalitu hovoru vliv, než jsou pouze ztrátovost paketů a vlastnosti hlasových kodeků, a provést optimalizaci celého řešení. Měření kvalitu hovoru má v prosazování QoS (Quality of Service) svůj obrovský potenciál, i když je tato oblast často opomíjena a rozhodně se vyplatí tato řešení nasazovat i v ústřednách neboť máme přístup k výsledkům v podstatě v reálném čase bez závislosti na testovacích hovorech nebo drahých testovacích řešeních či systémech.

6. Dostupnost systému BESIP

K projektu BESIP byly vytvořeny webové stránky, které slouži k vývoji a dále slouží jako uživatelská dokumentace pro koncové uživatele, jak si zprovoznit systém BESIP. Níže jsou uvedeny veškeré webové odkazy, které slouží pro stažení BESIPu a jeho práci s ním:

Webová stránka pro práci s BESIPem:

https://homeproj.cesnet.cz/projects/besip/wiki

https://sip.cesnet.cz/cs/implementace/besip

Link na stažení image BESIP:

http://liptel.vsb.cz/mirror/besip/nightly/

Link na SVN (zdrojové kódy):

https://liptel.vsb.cz/svn/besip/

Veškeré návody jak pracovat se staženými soubory jsou popsány v kapitole 2.1 nebo na výše uvedené webové stránce.

7. Závěr

Cílem projektu bylo vytvořit jednoduché řešení SIP komunikačního serveru pro embedded zařízení. Původně bylo zamýšleno využití pro menší tzv. SOHO pracoviště avšak využití námi vytvořeného systému je mnohem komplexnější. Kompletní řešení systému je postaveno na open - source nástrojích, což umožňuje téměř nulové náklady. Pro funkci a využití systému je pouze nutné vynaložit náklady na pořízení hardwarového zařízení.

Jádro systému bylo postaveno na operačním systému linuxové distribuce OpenWRT. Díky konfigurovatelnosti OpenWRT byl vytvořen a přizpůsoben image pomocí balíčků dle požadavků na systém BESIP. Pro softwarovou ústřednu byl použit Asterisk jako flexibilní a rozšiřitelné řešení. Bylo vytvořeno úvodní uživatelské rozhraní pro základní konfiguraci a nastavení OpenWRT, Asterisku a v neposlední řadě rozhraní pro přístup k aplikaci pro určení kvality hovoru.

Pro bezpečnost BESPIPu byla použita aplikace SSI. Nejedná se o jedinou aplikaci, ale o vzájemnou kombinaci několika řešení. Výsledkem je pak IPS systém na bázi IDS systému Snort. Obrana BESIP serveru (dále jen server) spoléhá na open-source firewall iptables. Tento firewall kontroluje veškerý provoz mezi serverem a ostatní sítí. Komunikace serveru s okolím není nijak omezena, může tedy komunikovat s jakýmkoliv zařízením. Standardně je také zapnuta ochrana proti SYN flood útoku.

Veškeré informace o projektu, image BESIP nebo zdrojových kódech lze nalézt na odkazech, které jsou uvedeny v kapitole 6.

Rejstřík