Modular SIP Server on Embedded Platform

MIROSLAV VOZNAK, LUKAS MACURA, JIRI SLACHTA
Department of Multimedia
CESNET
Zikova 4, 160 00 Prague
CZECH REPUBLIC
voznak@ieee.org, lukas.macura@cesnet.cz and jiri.slachta@cesnet.cz
http://www.ces.net

Abstract: - The paper deals with implementation of multiplatform embedded SIP communication server with unified configuration interface which has been developing within the framework of a BESIP project (Bright Embedded Solution for IP Telephony) since May 2011. The BESIP consists of several components which are distributed under GPL as an open-source solution and the whole project is supported by CESNET (Czech Education and Scientific NETwork association). The paper explains and describes the whole concept and individual modules, acquaints with the current state and with the future intents. SIP server is portet into OpenWRT project core and Asterisk with Kamailio inside are used as SIP engines. Kamailio was selected for security and reliability, Asterisk for extensive PBX functions. Next to this, there is implemented Web frontend for user-friendly management. We put emphasis on security, which is ensured by the Security module with IDS/IPS (Intrusion Detection and Protection Systems) and on the Spech quality monitoring with own developed tool. Last, but very important component, is NETCONF server which is part of image and enables an advanced unified management.

Key-Words: - OpenWRT, SIP server, BESIP, Speech quality, VoIP security.

1 Introduction

The BESIP aims to become VoIP PBX system available for anybody, the users need not know complex software features and hidden internals of VoIP software. BESIP offers the prepared solution with integrated key components, the entire system is distributed as an image or individual packages can be installed from SVN and users do not care about dependencies, they just configure VoIP system which works. Every software solution includes own configuration and management. BESIP aims to be scalable solution with security and unified configuration in mind [1]. After research and project discussion we made that decisions:

OpenWRT for good scalability and simple embedding:

- Kamailio for reliability and high availability
- Asterisk and Kamailio as B2BUA (Back-to-Back User Agent) and SIP Proxy
- YUMA as NETCONF server
- OpenWRT UCI as configuration backend

Several open-source applications were adopted and implemented into developed modules, however within the implementation many modifications were required, especially in the core module (OpenWRT) due to complicated porting of applications into OpenWRT buildroot [2]. Our patches were verified and accepted by OpenWRT community. The speech quality monitoring tool was developed from scratch and implemented in Java. BESIP can run on low-end devices with 32MB RAM at least and supports OpenWRT MIPS architecture.

2 Selection of suitable Platform

The most important step which had to be done, was choosing right software distribution/platform. There was an idea to modify Debian distribution, this is probably the easiest way for developers. Debian includes many ports and packages which are available for many software services but Debian is not suitable for embedding. A modification of Debian, in order to be easily embedded into small device with read-only flash, is really a difficult task and the expected results of such work can not lead to a source distribution.

Next solution was adopting some low-level distribution for embedding. There are several possibilities like FreeWrt, OpenWrt, DebWrt etc. After discussion and projects observations, OpenWrt was selected as primary platform. There are many packages included and packages which are

not included and can be added into applications tree. Even if it is not easy procedure for some kind of packages (especially for packages without configure script), we decided for this way. OpenWrt is well-know for great support, ticket system, relatively well documentation and cooperation with community of developers.

3 Architecture and Technology

The BESIP architecture is depicted in Fig. 1, it is created entirely from opensource parts. This was main presumption for project management and developing. There are four basic modules: Core, Security, Monitoring and PBX. Core is divided into following parts:

- OpenWRT as build platform;
- NETCONF For administration of entire system, YUMA implementation was adopted;
- Web GUI for user-friendly configuration;
- SUBVERSION as revision control system providing a support and better orientation for developers, it is not a part of the released BESIP image.

The security module is based on SNORT, SNORTSam and iptables [7]. In addition to this, the Kamailio ratelimit and pike module is used for defending attacks. The monitoring module exploits a tshark package and our java code which interprets its results and gives information about particular speech quality. The Zabbix agent is used to report basic states of entire system and finally the PBX module is made from kamailio in conjunction with Asterisk.

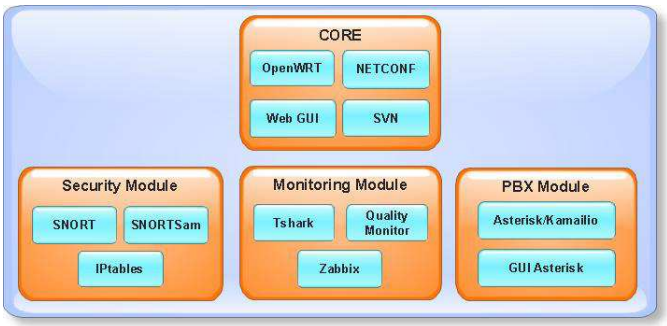

Fig. 1. BESIP architecture.

3.1 Configuration Module

The NETCONF protocol exploits a specified mechanism for exchanging the configuration data among an administrator and network devices. This protocol allows the device to send and receive configuration data through XML documents using the RPC paradigm [5]. These XML documents are handed over the RPC calls, the RPC request is initiated by a client that requests the configuration data or a command to be performed on the server. While these requests are being performed, the client is blocked until he receives the RPC reply from NETCONF server [4]. That replies consists of a configuration that is complete or a partial. Another reply is a message informing us if a command was successfully performed on the server or not. This communication is transferred over a transport protocol which has to be secured and to allow an authentification and authorization. Most probable and secure way, how to communicate with the NETCONF server, is to use SSH2 protocol (RPC calls over SSH subsystem).

The structure of configuration data on NETCONF server in YUMA package (netconfd) is specified by the YANG module which defines the semantics and syntax of a management feature. It provides complex data structures which allow design any data structures that will meet the requirements of developers.

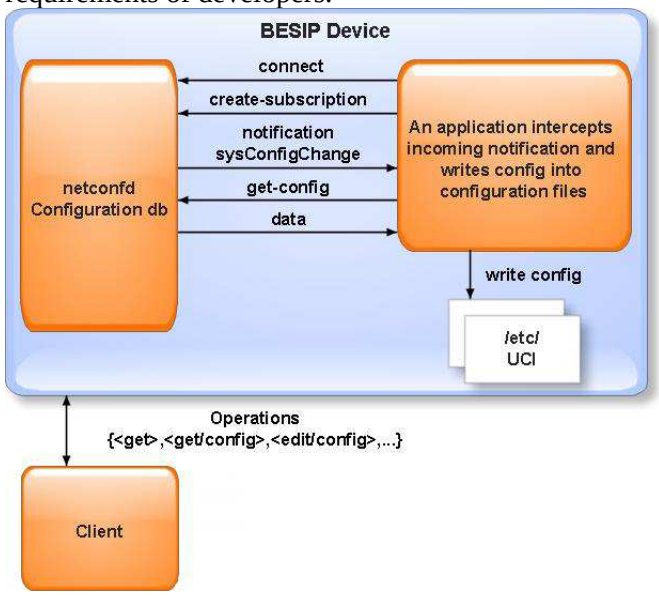

Fig. 2. NETCONF usage.

The configuration data are stored securely on the NETCONF server and all requests and responses must comply with firmly defined structure, specified by YANG modules. Next, global database of all the configurable parameters is required and it is ensured by NETCONF server. The configuration parameters are inserted by the user and stored in the NETCONF server. Consequently, the stored configuration data are available through simple queries. It makes the device quickly configurable, therefore a backup or a restore of configuration can be simply and quickly performed [6].

Yuma is a package which provides tools for the network management. It consists of a NETCONF

client yangcli, NETCONF server netconfd, validation tools yangdiff and yangdump and netconf-subsystem, which allows us to communicate with NETCONF server through a SSH2 subsystem. We ported this package into OpenWrt in both latest versions (version 1.15-6 and 2.2-1). Because there are some issues we have to solve, we have not delivered these packages to OpenWrt community yet although we use them. Nevertheless we continue in development of the mentioned packages.

OpenWRT uses UCI as configuration backend, it is a group of configuration files which can be read or modified by common UCI API. We decided to provide a glue between NETCONF and UCI.

The long term goal of this project is to make the BESIP configuration independent on clients. Today, many systems are configurable using web, ssh or telnet and each of them offers its own semantics and configuration file. BESIP project aims to change this situation, using NETCONF as defined communication and management protocol, configuration independent syntax will be available on all modules. At first stage of project, applications and libraries had been ported, afterwards we focused on implementation of NETCONF, UCI, PBX, Security and monitoring modules. See Fig. 2 to understand configuration data flow which has been defined in BESIP.

3.2 PBX Module

The PBX module is key part of the BESIP project. It operates as SIP proxy or SIP B2BUA, depending on configuration, and ensures a call routing. Asterisk is used for call manipulation and for the PBX functions. Kamailio is used for the proxying SIP requests, the traffic normalization and for the security [11]. There are always two factors when developing VoIP solution, the first one is high availability and reliability, the second one is an issue of advanced functions. Many developers try to find a compromise, we have implemented both and our BESIP is able to adapt to the users requirements. More complex system can handle many PBX functions such as a call recording or an interactive voice response but due to the bigger complexity, it is more susceptible to fault. On the opposite side, pure SIP proxy is easier software which can perform call routing, more fault tolerant but it is more difficult to use the advanced PBX functions [8], [13]. The BESIP offers users an option to choose how system will work. From this reason, the BESIP includes both Kamailio and Asterisk. Today, only one of these engines can be configured but in future, both engines will work together and will be configured by common NETCONF server. Kamailio will route requests even if Asterisk will be out of order, only advanced PBX functions will be unavailable in such situation [13], [15].

3.2.1 Asterisk GUI

Asterisk-GUI is very flexible web solution of Asterisk management. Even if it is not NETCONF based Asterisk-GUI was added to the first BESIP release. The reason was that at this time, there was not completed an interoperability between NETCONF and Asterisk. It is available in the next release and the implementation involved very complex task. The users can decide to use easy Asterisk-GUI for PBX setup at initial version of BESIP. Nevertheless in future version we would like to remove the Asterisk-GUI package from BESIP image and the configuration will be accessible only through new developed NETCONF based management. During implementation, we solved several technical issues concerning Asterisk-GUI in OpenWRT environment and finally we made a decision on disuse Asterisk-GUI in BESIP roadmap. The last release still links this GUI on the BESIP main page [9].

3.2.2 Accounting

In many systems, an accounting is divided into two separate parts. The individual calls are processed and a call detail record (CDR) is generated to every performed call, these CDRs are stored in text file or a repository. The next part of the accounting is an application which enables to perform statistics over stored data, it means to search and display in accordance with requested criteria. This is a conventional scenario, classical approach of many accounting applications and highly reliable because the PBX function is not affected by accounting and even if there is problem with accounting software, PBX still operates properly. However there is one big disadvantage, during a call setup, the PBX knows nothing about call price and cannot provide an authorization which is well-known from pre-paid services offered by mobile operators. Having this information, we are able to perform more checks and operations at the call setup level. For example, we can look up into user credit and do not permit a call if the credit is depleted or low. Similar to this, we can authorize every call against a threshold, such as maximal price per minute/trunk/global. These thresholds can be pre-set or dynamically changed according to the actual user credit. Having this information, the PBX will be safer and resistant against attacks aimed at an exploitation of the PBX [10].

3.3 Security Module

Security module is very important part of BESIP and all the time, it was considered to make the developed system as secure as possible. Next to this, entire system has to be fault-tolerant, monitored and protected from attacks. It means that if the device is under attack, only attacker has to be blocked, not entire system or other users. If there is some security incident, BESIP immediately solves the situation and notifies this event in detailed report to the administrator. The attack are recognized and processed by SNORT rules, the source IP address is automatically sent into firewall by SNORTSam and the intruder's IP is blocked. This is very flexible, reliable and effective implementation. Dropping attack based on IP directly in the Linux kernel is much more efficient than to check messages on the application level. Only first messages are going to SNORT filter. When SNORT identifies a suspicious traffic, next messages from the same IP are blocked. In next BESIP releases, we are going to to implement ipqbdb mechanism which will be even more self-defending. It is based on IP denoting.

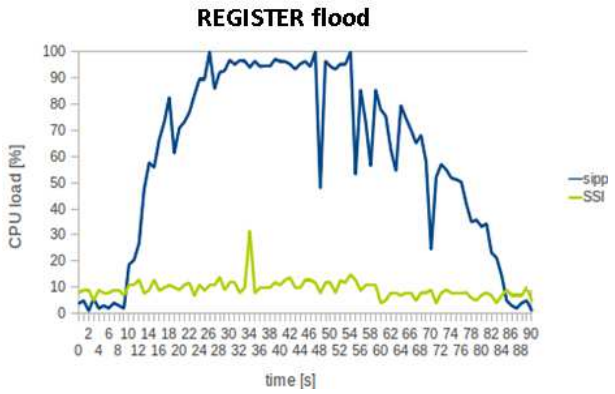

Fig. 3. Attack effectivity based on REGISTER flood.

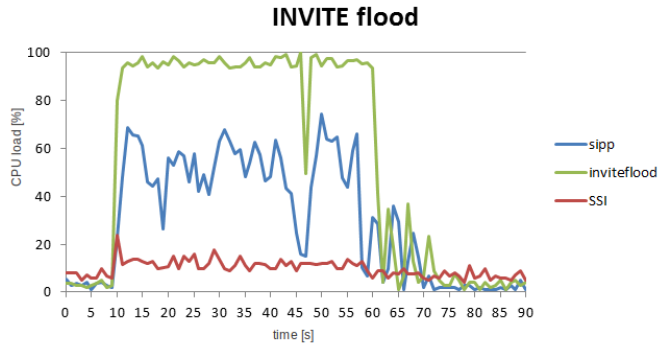

Fig. 4. Attack effectivity based on INVITE flood.

If more soft faults apper from some IP, it is blocked at the IPTABLES level, this approach can effectively block incorrectly configured clients and servers. For example, if client sends REGISTER with proper credentials, it is not obviously security attack but the client attempt to register again and again, with every registration requires computing sources at SIP REGISTRAR server. Such attempts can be denoted and blocked for a time interval. Administrators can use Zabbix or NAGIOS agent inside BESIP to gather all information directly into their monitoring system. The monitoring is very important part of the security module and BESIP team was already focused on the issue in early design [14].

Partially, BESIP is resistant to some kind of DoS attacks. It depends on hardware used. If hardware is strong enough to detect some security incidents on application level, the source IP is immediately dropped. But for weak hardware it can be serious problem. In such case, it is better to stop DoS attacks before it reaches BESIP. For example, SNORT on a dedicated machine will be much more flexible than if is an integral part of VoIP system. Therefore, we recommend to use an external IPS system to make VoIP service robust and secure. Nevertheless BESIP includes own IPS/IDS system [11].

The features of our security module were verified in test-bed and results are depicted in Fig. 3 and 4. The CPU load was monitored during trivial SIP attacks. The line SSI (Snort, SnortSam, IPtables) represents the response in case of active security module in BESIP whereas next dependencies were measured without SSI. There were emulated only two types of DoS attacks, namely REGISTER flood and INVITE flood. In order to generate these attacks, we used sipp generator and in case of INVITE also inviteflood tool. The dependencies in both figures clearly prove the ability of security module to mitigate the performed attacks.

3.4 Monitoring Module

The overall solution of the monitoring system consists of several different open source components and also of the part that was directly developed for this purpose to meet the defined requirements. System structure is depicted in Fig. 5. The system itself consists of three logical components, which are – web interface that serves the administrators (Web GUI), part of the script (Scripts) that controls the obtaining the information necessary to compute the speech quality in the simplified E-model [13]. Last component is part of the Quality Monitor, which contains the logic for calculation itself and performs processing of data obtained by scripts. In the overview SQLite3 database, which is used to store the results.

ISBN: 978-1-61804-109-8

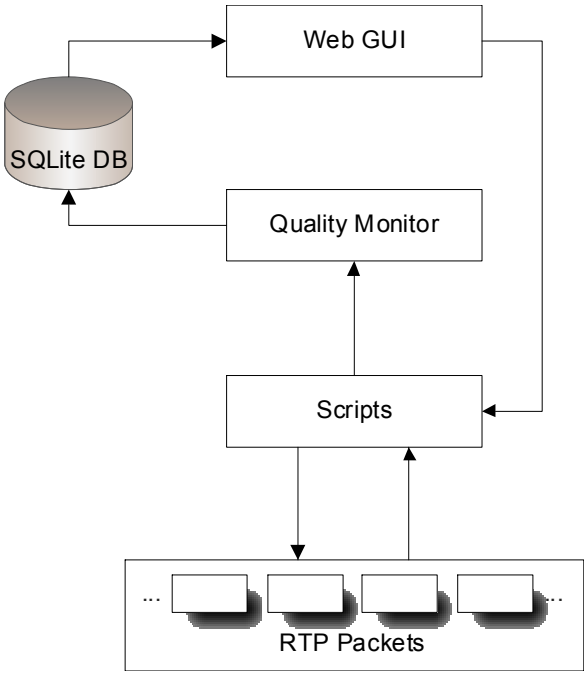

Fig. 5. Overview of the logical structure of VQM.

The developed application offers the comfort of management in a web application, the developed interface aggregates required functions. Web interface is the main part of user interaction with a monitoring tool. Monitoring tool is turned off in the default configuration and can be enabled using the intuitive main interface of BESIP any time . This part of the monitoring tools is also used as a mean to display the measured and computed results. Structure of the presented data is as follows: Time, Source IP, Destination IP, MOS and used Codec. An example of user interface is shown in Fig. 6.

Monitoring is running...

Stop	Results	Refresh	Erase	
Date	From	To	MOS	Codec
23.04.2012 05:46	192.168.21.50	192.168.21.55	2.79	G.711
23.04.2012 05:55	192.168.21.50	192.168.21.55	3.38	G.711
23.04.2012 05:59	192.168.21.50	192.168.21.55	3.01	G.711

Fig. 6. Sample of web GUI of monitoring speech quality.

4 Conclusion

The contribution of our work is entire BESIP concept and its implementation. As we have mentioned, BESIP consists of several components which are distributed under GPL as an open-source solution. A few of them have been fully adopted such as the components in Security and PBX modules, some of them modified, concerning the CORE module and finally we have developed own tool for Speech quality assessment. The contribution of our work is not only few hundreds of hours spent on the development, on the coding BESIP system, we bring a new idea of the unified configuration management, with unified CLI syntax which enables to configure different systems, Asterisk and Kamailio in our case. We perceive that we need to solve a lot of issues, Individual packages are working and after several pre-releases, the version 1.0 was released in November 2011. BESIP is distributed as a functional image for x86 platform but is possible to run it on Vmware or KVM. Configuration is available through web-browser or SSH client. Today, there is a trunk version in SVN which is actively developed and individual improvements are included in next subversions. After testing, version 2.0 will be released, new release 2.0 will be based completely on NETCONF with one API to configure entire system. Next to this, CLI syntax is developed and will be connected to NETCONF. CLI will be independent of internal software so if some internal software is modified, there will be no change in configuration. Even more, CLI and NETCONF configuration will be independent on hardware and version. To export configuration from one box and to import it to the next one will be simple task. Users will modify only one configuration file to manage entire box. After this step, all internals of configuration will be hidden as was mentioned in introduction. Entire BESIP management and development is available at [13] and Binary images from nightly autobild can be downloaded from [14].

Acknowledgement
This work has been supported by the Ministry of Education of the Czech Republic within the project LM2010005.

References:

[1] F. Rezac, M. Voznak, J. Ruzicka, Security Risks in IP Telephony, *CESNET Conference 2008-Security*, pp.31-38, 2008.
[2] L. Spitzner, *Honeypots: Tracking Hackers*, Addison-Wesley Proffesional, 2002.
[3] D. Sisalem, J. Kuthan, T.S. Elhert, F. Fraunhofer, Denial of Service Attacks Targeting SIP VoIP Infrastructure: Attack Scenarios and Prevention Mechanisms. *IEEE Network*, 2006.
[4] M. Voznak, F. Rezac, Threats to voice over IP communications systems, *WSEAS Transactions*

on Computers, Volume 9, Issue 11, November 2010, Pages 1348-1358

[5] M. Voznak, F. Rezac, SIP Threats Detection System, *9th WSEAS International Conference on Data Networks, Communications, Computers*, Faro, 2010, pp. 125-130

[6] D. Endler, M. Collier, *Hacking Exposed VoIP*, McGraw-Hill Osborne Media, 2009.

[7] N. Provos, T. Holz, *Virtual honeypots*, Addison-Wesley Professional, 2007.

[8] R.C. Joshi, A. Sardana Honeypots: *A New Paradigm to Information Security*, Science Publishers,2011.

[9] M. Voznak, F. Rezac and K. Tomala, SIP Penetration Test System, *33rd International Conference on Telecommunications and Signal Processing (TSP 2010)*, pp. 504-508, 2010.

[10] M. Voznak, F. Rezac, Web-based IP telephony penetration system evaluating level of protection from attacks and threats, *WSEAS Transactions on Communications*, Volume 10, Issue 2, February 2011, Pages 66-76

[11] M. Voznak, J. Safarik, DoS attacks targeting SIP server and improvements of robustness, *International Journal of Mathematics and Computers in Simulation*, Volume 6, Issue 1, 2012, Pages 177-184.

[12] M. Voznak, M. Tomes, Z. Vaclavikova, M. Halas, SIP Threats Detection System, *9th WSEAS International Conference on Data Networks, Communications, Computers*, Faro, 2010, pp. 119-124

[13] Management of BESIP Project, LipTel Team, 2011. URL https://homeproj.cesnet.cz/projects/besip/wiki

[14] Source code of BESIP Project, LipTel Team, 2011.URL http://liptel.vsb.cz/mirror/besip/nightly