Kamailio Syntax Generator and Configuration File Parser

MIROSLAV VOZNAK [1], LUKAS MACURA [2]

[1] VSB-Technical University of Ostrava
Department of Telecommunications
17. listopadu 15, 708 33 Ostrava
CZECH REPUBLIC
miroslav.voznak@vsb.cz

[2] Silesian University of Opava
Faculty Of Bussines Administration
Univerzitni nam. 1934/3, 733 40 Karvina
CZECH REPUBLIC
macura@opf.slu.cz

Abstract: - This papers deals with simplification of Kamailo and OpenSIPS configuration. We developed a tool that is able to generate complex Kamailio syntax. Kamailio is very powerful and flexible SIP server but its config is difficult to understand and our tool brings option to simplify significantly the config file creation process. Its important feature is the fact that the tool is independent on used modules because blocks in global template are selected only if a particular given option is used. Our tool generates a Kamailio config file that is customized for a particular scenario from a generic main routing script. The tool was verified in practice and we expect our result to be tested by the community using and developing Kamailio. Theoretically, with small modifications, this tool could generate configs for Asterisk and other SIP servers.

Key-Words: - SIP server, Kamailio, OpenSIPS, Syntax, Script generator.

1 Introduction

Security is of paramount importance for today’s Internet. We try to find a solution enabling to secure our applications and services. Security is also the hot topic for IP telephony. These days, many hackers and tools are sniffing over the Internet and looking for potential loopholes. This is the reason why we need to implement SIP proxies to hide internal VoIP infrastructure and servers. The SIP Express Router project is a well-known example of SIP proxy implementation and has already been forked into several independent projects, such as openSER, openSIPS and Kamailio. The last mentioned project is the youngest; we decided to enhance its settings because we expect it to expand widely in the academic environment. This article describes a tool enabling to configure Kamailio. We developed a generic tool that can be adopted as a suitable solution for Kamailio SIP proxy implementation. Kamailio is very powerful and flexible but its config is difficult to understand. Even more, one small change in the behavior can evoke many changes in the configuration. We tried to make the process easier for other users by providing them our script and we consider this tool to be very useful.

To enhance security, we need to hide our VoIP infrastructure behind a SIP proxy. It means we have to implement a SIP proxy on the top of our SIP network and split internal and external traffic [1], [2]. External communication with our network is routed through the SIP proxy. There are more solutions how to built a SIP proxy but if we intend to implement an open-source solution, we need to use a high-performance SIP proxy based on the SIP Express Router project (SER) [3], [4]. SER was developed by the Fraunhofer Institute for Open Communication Systems in Berlin in 2002 and later it was acquired by US-based Tekelec. The original SER had been forked into several projects, the most recent of which is Kamailio. This project was established in 2008 when the former openSER project was renamed Kamailio. Kamailio and SIP Express Router teamed up to build the SIP Router project and recently, Kamailio was awarded the Best

Open Source Networking Software 2009. The latest major version v3 was released in 2010.

We consider Kamailio the best open-source solution for SIP proxy implementation and we expect it to expand in the academic environment [5]. Our script generator is able to prepare a Kamailio config template in accordance with user requests. This customized configuration is produced by a PHP script with Smarty engine.

2 Kamailio

Kamailio (former OpenSER) is an Open Source SIP Server released under GPL, able to handle thousands of call setups per second [6]. Kamailio supports the following transport protocols: TCP, UDP and secure TLS [7].

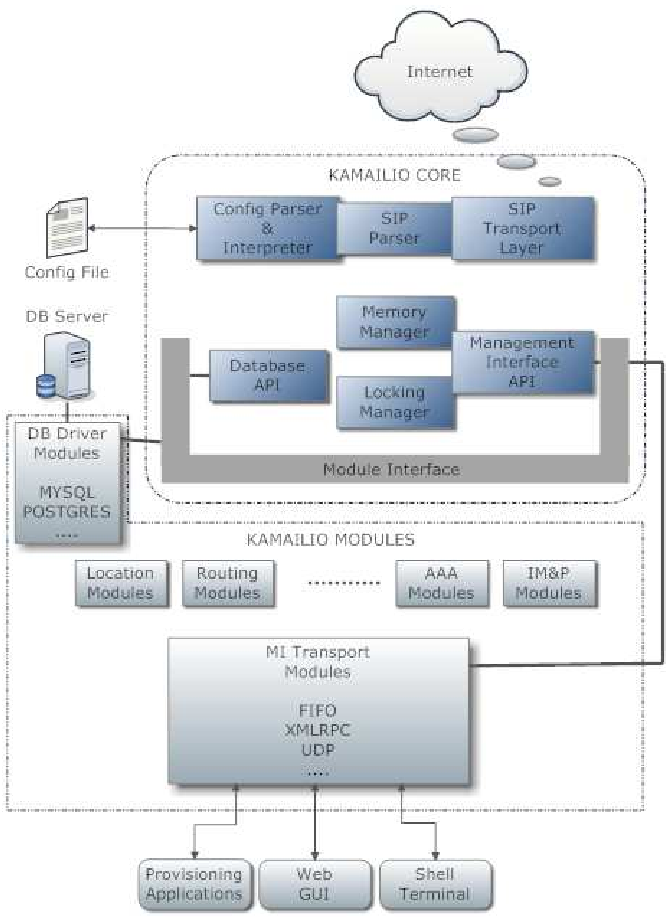

Fig. 1. Kamailio architecture.

This project offers many features such as SIMPLE instant messaging and presence, ENUM, least cost routing, load balancing, routing fail-over, accounting, authentication and authorization against MySQL, Postgres, Oracle, Radius, LDAP, XMLRPC control interface and SNMP monitoring. It can be used to build large VoIP platforms or to scale up SIP-to-PSTN gateways, PBX systems or media servers such FreeSWITCH, SIP Express Media Server or Asterisk.

Kamailio can play a role of proxy, registrar or redirect server, or any combination thereof [8], [9]. Kamailio has a modular architecture, depicted on figure 1. There are two main components: the core providing the low-level functionalities, and the modules ensuring additional functionalities. The core includes a memory manager, config parser and intepreter, SIP parser, transport layer memory and locking manager, database API and management interface API. The module interface provides access to Kamilio modules.

The config file is in fact a script and we need to finetune many parameters to achieve a proper interpretation and interaction with other required technologies such as TLS, MySQL, LDAP or ENUM [10]. The config file consists of many code lines and many SIP dialogues have to be written directly in it. It is really difficult to create some “generic" config as the code needs to be modifiable by everyone to allow enabling or disabling individual features. The figure below describes the simple part of a config example, a simple modification of one module or feature can cause many changes in the code. For example, in our code, there is a function “is_uri_host_local()” which is a part of one module. This function can be used many times. But we wanted to make our config modules and technology independent.

```
route[RESTDIALOG] {
 if (loose_route()) {
  if (is_method("BYE")) {
   setflag(1); setflag(3);
  }
  if ($tU =~ +420) {
  } else if ($tU =~ +421 ) {
 t_relay(sip:gw1:5060);
  } else if ($tU =~ +1 ) {
 t_relay(sip:gw2:5060);
  } else if ($tU =~ +47 ) {
 t_relay(sip:gw3:5060);
  }
  route(RELAY);
 exit;
}
```

Fig. 2. Part of original configuration.

3 Used Technology

A Kamailio config file can become a nightmare for many potential users. This is why we focused our work on creating a tool which generates the Kamailio config file based on input parameters. The generated config should be editable so that certain parameters could be modified after its creation. Many users have been waiting for a similar tool and many institutions do not use Kamailio because of the complexity of its configuration file.

3.1 Features

When we designed our tool we found the following features as important:

- Scalable. From smallest SIP proxy up to big one with external databases.
- Secure. As much as possible, including sanitize, ratelimit, header checks etc.
- Modular. Modules should have basic dependencies, e.g. LDAP authentication depends on general LDAP support.
- Unlimited. All needed options should not be limited by a number of records.

The config file should support at least the following modules: LDAP, ENUM, Authentication (directly from kam3cfg or external databases), Location, NAT and RTP proxy.

3.2 Programming Language

We had to choose the right technology (programming language and libraries) for our tool. We adopted PHP and Smarty because of their support and portability. Even though there are other powerful languages we wanted to use some templating system and Smarty seemed to be ideal. Many administrators know PHP, it is a well-known language and in combination with Smarty we get a really powerful tool. Smarty is a template engine for PHP. More specifically, it facilitates a manageable way to separate application logic and content from its presentation. Smarty provides built-in and custom functions to be used in your templates. These functions are like the API of Smarty templates. Even though smarty looks like a templating system for HTML pages, it is powerful for other config and text files too. Smarty supports variables, loops, user functions and much more. Figure 3 represents the same part of config as figure 2 but prepared using Smarty and PHP. Variables are filled in PHP and the rest of work is performed by Smarty. The code is clear and flexible.

```
route[RESTDIALOG] {
 if (loose_route()) {
  if (is_method("BYE")) {
   setflag(1); setflag(3);
  }
<foreach from=$local_prefixes item=prefix>
  if ($tU =~ <$prefix.prefix>) {
  t_relay(<$prefix.gw>);
  } else {
<foreachelse>
  if (0) {
</foreach>
```

Fig. 3. Part of configuration with Smarty.

4 Results

We developed a tool generating Kamailio config templates based on kam3cfg script and supporting SIP proxy functions and local call routing. This tool is able to receive arguments also from files and to create config in accordance with requests. Its important feature is the fact that the tool is independent on used modules because blocks in global template are selected only if a particular given option is used. Our tool generates a Kamailio config file that is customized for a particular scenario from a generic main routing script. For debug purposes, we have created macro xlog which will log all messages with the same prefix and suffix. Anybody can change suffix and prefix to fit his needs. In common situation, this would be very complex because there can be many xlog lines in a script. Even some macros are modified automatically based on modules used. The tool has many parameters and explaining all of them is beyond the scope of this document.

```
./kam3cfg.php \
  --local-ips 192.168.1.0/24^192.168.3.0/24 \
  --local-domains local.edu^sip.local.edu \
  --local-prefixes '123/556/sip:gw:5060' \
  --force-rtp \
  --listen \
   udp:192.168.1.1:5060^tls:192.168.1.1:5061
```

Fig. 4. Simple SBC forcing as RTP proxy.

Only a small number of cases can be seen in figures 4, 5 and 6. Some parameters can have multiple values. Unfortunately, Console_Getopt module cannot read the same multiple parameters, so we use delimiter ", *" to split it into multiple values. Moreover, all multiple values can be loaded from a file provided it starts with "@". If you want to see all parameters, run "./kam3cfg.php –help".

```
./kam3cfg.php \
  --local-domains @domains.txt \
  --local-prefixes @prefixes.txt \
  --with-nat \
  --listen udp:192.168.1.1:5060
```

Fig. 5. Simple SBC with NAT traversal.

```
./kam3cfg.php \
  --local-domains local.edu^sip.local.edu \
  --with-enum \
  --enum-suffixes e164.localnet.edu^e164.arpa. \
  --listen udp:192.168.1.1:5060
```

Fig. 6. Simple SBC with ENUM routing.

Xlog suffix and prefix are configurable. Almost any virtual variable of Kamailio can be used. All prefixes will be taken from prefixes.txt and all domains will be read from domains.txt. The file format is one value per line. This is very useful if we have many prefixes and domains. It is important to set local domains, mostly there is only one domain but a multidomain config can be created too. Local prefix, in figure 4, can be served directly by Kamailio or served by an external gateway. We optimize some functions in case a request comes from local addresses therefore we set local IPs. This is used for „force-rtp" option too. This means that when we set „force-rtp" to 1, all requests from „local-ips" are proxied by the RTP proxy. Our Kamailio, in figure 4, is listening on specified UDP port 5060 and TLS port 5061. Where ENUM support is enabled, all non-local requests are checked against ENUM database. Even more, local routing policy can be forced to local ENUM prefix. We limit requests by source IP and requests by methods because there are too many scenarios which we cannot predict; we use three predefined modes for ratelimit: small, medium and large site. Our script can use an external LDAP database for authentication, AVP load and local aliases. During each call, a LDAP server is asked to retrieve data.

The big advantage of our script is that it can automatically insert debug messages into the generated script. There are four levels of debug (0-4). We standardized the log format in all messages so that it is easy to find all lines belonging to a single call. A log line can be customized by „xlog-suffix" and „xlog-prefix". In a standard kamailio config, this would be more complicated because the log line needs to be changed completely in many places in the script file.

More complex scenarios use external databases, such a situation is depicted on figure 7 where local domains „local.edu" and „sip.local.edu" are used. NAT support and LDAP server are enabled. The destination URI is checked against LDAP, therefore an extension should exist in LDAP (achieved by ldapaliases-uri filter).. This option is useful in a multiPBX environment where the central config is at a LDAP server. Even more, we can map LDAP attributes to avps (ldap-attrmap).

```
./kam3cfg.php \
  --local-domains local.edu^sip.local.edu \
  --with-nat \
  --with-ldap \
  --with-ldapaliases \
  --ldapauth-uri
\'ldap://ldap/o=su?cn,pwd?sub?(|(cn=$au)(numb
er=$fU))' \
  --ldapaliases-uri
\'ldap://ldap/o=su?cn,number?sub?(number=$f
U)' \
  --ldap-attrmap
\'cn=s:username^pwd=s:password^name=s:disp
layname' \
  --enum-suffixes
\e164.localnet.edu^e164.arpa^nrenum.net^e164
.org \
  --listen
udp:192.168.1.1:5060^tls:192.168.1.1:5061 \
  --with-tls \
  --tls-key '/etc/kamailio/key.pem' \
  --tls-certificate '/etc/kamailio/cert.pem'
```

Fig. 7. More complex example.

5 Conclusion

We achieved to implement most of features required. Nevertheless, there is still potential to improve our work, especially in a global template file. Our tool was tested in practice at the Silesian university, and we expect other members of the CESNET association to use it. We discovered that there is a limit in Kamailio script parsing. If a script is longer and more complex, Kamailio fails and returns "out of memory" message. This situation happens with scenarios where we had in use 10.000

gateways loaded from a file. We suppose that such a value is not achieved in a real setup and there is a possibility to solve it using routing from an external database. We will be enhancing this feature in future versions. Our tool is available on Internet [11] and the best way to download it is to follow SVN instructions on the page. We expect our tool to be tested by the community using and developing Kamailio and we would appreciate any feedback to further improve our work and enable releasing next version. We intend to create a web-based frontend for the tool and enable everyone using the config generator online. We will be working on it but we need to receive feedback on the current version. Theoretically, with small modifications, this tool could generate configs for Asterisk.

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

References:

[1] M. Voznak and F. Rezac, Threats to voice over IP communications systems, *WSEAS Transactions on Computers*, Volume 9, Issue 11, November 2010, Pages 1348-1358.

[2] D. Sisalem, J. Floroiu, J. Kuthan, U. Abend and H. Schulzrinne, *SIP Security*, Wiley, 350p., 2009.

[3] A. Pelinescu, J. Janak and J. Kuthan, SIP Express Router (SER) , In *IEEE NETWORK*, vol. 17, issue 4, p.9, 2003.

[4] F. Goncalves, *Building Telephony Systems with OpenSIPS 1.6*, Packt Publishing, 274p., 2010.

[5] R. Chochelinski and I. Baronak, Private Telecommunication Network Based on NGN, In *32nd International Conference on Telecommunications and Signal Processing*, 2009, Dunakiliti, HUNGARY, pp. 162-167

[6] M. Voznak and J. Rozhon, SIP Infrastructure Performance Testing, In *9th International Conference on Telecommunications an Informatics*, Catania, Italy, 2010, pp. 153-158.

[7] M. Voznak,Speech bandwith requirements in IPsec and TLS environment, In *13th WSEAS International Conference on Computers*, 2009 Rhodes, GREECE, 2009, pp. 217-220.

[8] J. Bates, C. Gallon, M. Bocci, S. Walker and T. Taylor, *Converged Multimedia Networks*, Wiley, 364 p., 2006.

[9] I. Pravda and J. Vodrazka, Voice quality planning for NGN including mobile networks, *12th International Conference on Personal Wireless Communications (PWC 2007)*, 2007, Prague.

[10] P. Falstrom and M. Mealling, The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM), *IETF*, Request for Comments: 2915, April 2004

[11] L. Macura, Kam3cfg, 2010, *Available online* URL http://open.phonyx.eu/wiki/kam3cfg.

[12] D. Komosny and I. Herman, Voice/data integration in municipal transport management, *WSEAS Transactions and Communications,* Volume 4, Issue 1, January 2005, Pages 20-23.