SIP Server Security with TLS: Relative Performance Evaluation

Merima Kulin, Tarik Kazaz and Sasa Mrdovic
Faculty of Electrical Engineering
University of Sarajevo
Sarajevo, Bosnia and Herzegovina
{ mk15172, tarik.kazaz, sasa.mrdovic }@etf.unsa.ba

***Abstract*—VoIP (Voice over Internet) provides delivery of voice information over unsecured IP-based networks like the Internet. VoIP data, signaling and voice, needs to be secured in such an environment. Security mechanisms take their toll on VoIP system performance. SIP is dominant signaling protocol for VoIP. This paper measures relative decrease in VoIP performance of system with secured SIP signaling over one without it. It compares SIP with authentication enabled over three transport protocols: UDP, TCP and TLS. Peak throughput of concurrent calls, registration request delay, session request delay, SIP server CPU and RAM usage are measured. Testbed environment consists of Asterisk IP private branch exchange (PBX) as a part of Elastix server, several SIP user agents and SIPp traffic generator. Test results show that performance of SIP over TLS based signaling is four times lower than the SIP signaling over UDP in most metrics.**

***Index Terms*—VoIP, SIPS, TLS, IP PBX, Elastix, Asterisk, SIPp, Performance testing**

I. Introduction

VoIP systems are becoming more widespread. In order to ensure their full adoption it is necessary to provide acceptable level of security. In conventional telephone system (PSTN) security is provided by physically restricted access to telephone lines and private branch exchange (PBX). With VoIP systems, voice and signaling data are packetized and sent through IP network. Packets, on their way from caller to called party, might pass through different systems that are not under the control of either of the parties, their organization or ISP. Such packets, if not protected, can be seen, changed, dropped or new packets might be inserted.

VoIP system protocols separate signaling and media on different channels. In most VoIP systems, Session Initiation Protocol (SIP) protocol is used for signaling and the Real-Time Transport Protocol (RTP) for media transport. Consequently, appropriate security mechanisms must be provided for securing them. Secure media transport on VoIP communications is realized using either IPSec or Secure RTP (SRTP). SRTP is more efficient in terms of bandwidth [1]. SIP RFC3261 [2] specifies several security mechanisms: Transport Layer Security (TLS) at transport level, IPSec at network level, SIPS URI Scheme for secure access to resources, HTTP Authentication for authentication and S/MIME for SIP messages body end-to-end confidentiality and integrity. Usually, only one or two of these mechanisms are enough and are used in VoIP systems. However, security comes at a cost. That cost is in additional processing in all nodes. Additional processing lowers number of calls that can be established and increases call establishment delays. This is in part caused by increased CPU load and memory usage of SIP servers.

This paper presents experimental performance study of costs related to using SIP security. It compares three SIP usage cases: over UDP with authentication, over TCP with authentication and TLS. For each case the following metrics are compared: peak throughput of concurrent calls, registration request delay, SIP server processor load and consumed RAM memory. We believe that these are the most representative usage scenarios. We base our metrics on IETF RFC6076 [3] recommendations. This study is conducted using Elastix Linux distribution as widely deployed Asterisk implementation that includes SIP server. During testing we had to tweak some of the components. All changes are described in the paper. Our results show relative cost of using TLS and are helpful for design of future SIP networks.

The rest of the paper is organized as follows. Related work is mentioned in section 2. Section 3 explains the practical implementation of SIPS. Section 4 explains the used testbed for performance testing. In section 5 the impact of TLS on SIP server performance is compared to SIP over UDP and TCP. Conclusion and discussion on future work are in section 6.

II. Related Work

Ever since SIP was proposed as standard signaling protocol for VoIP there was an interest in its performance. In 2002. SIPstone project proposed simple set of metrics for evaluating and benchmarking the performance of SIP servers [4]. Four different implementations of SIP server functions were compared in [5]. Transport protocol they used was UDP. Gurbani et al. analyzed different SIP performance metrics including end-to-end mean response times, availability and probability of loss on a model they created [6]. They found limits on number of call attempts that still provide acceptable mean response time. Nahum et al. used OpenSER SIP server implementation and SIPp call generator to evaluate SIP proxy server performance [7]. Their conclusion was that performance varies by order of magnitude depending on transport protocol used, TCP or UDP, and if authentication was enabled or not. They report

throughput of hundreds to thousands operations per second. Interesting findings were reported by [8]. They showed that principal reason for OpenSER inferior performance using TCP instead of UDP is server design. They argue that performance using TCP could be made competitive to performance using UDP with improvements on OpenSER architecture. In order to provide a standard set of common metrics that will allow interoperable performance measurements IETF recently issued RFC6076 devoted to SIP performance metrics [3]. Voznak at al. focused their research on creating a methodology that would allow administrators to precisely measure the SIP Server performance and compare it to other software and hardware platform [9]. Their work resulted in best practice document for SIP performance evaluation [10]. An interesting branch of research of SIP performance is focused on overload conditions. In order for SIP to be used effectively in production environment it has to be able to handle overload condition gracefully. Papers by Shen et al. [11], Hilt and Widjaja [12], Noel and Johnson [13], and Abdelal and Matragi [14] led to IETF RFC6357 on SIP overload control [15].

SSL/TLS adds a layer between application and transport layer. This layer performs cryptographic operations and it increases processing time. Cost of TLS processing was subject of [16]. Their conclusion was that tested Web servers were couple of orders of magnitude slower when serving pages over TLS. Another paper [17] analyzed architectural impact of SSL. They found that SSL increases computational cost of the transactions by a factor of 5-7. Zhao et al. provided detailed analysis of various cryptographic operations in SSL [18]. They showed that major overhead incurred during SSL processing lies in the session negotiation phase and about 70% of the total processing time of an HTTPS transaction is spent in SSL processing. Measurements made by Coarfa et al. show that RSA computations are the most expensive of all TLS operations and count for 20 to 58% time of web server [19].

All above papers analyzed influence of SSL/TLS on web traffic and web servers. Influence of TLS on SIP server performance was thoroughly tested in [20]. They compared SIP over TLS with SIP over TCP and UDP. Their results showed that SIP authorization has the biggest impact on total cost while impact of SSL was rather small. It is different from latter results by other researchers and might have something to do with used hardware and software and rather small volume of calls, 21 calls per second. Results of simulation that measured SIP call setup delay of different security protocols (TLS, DTLS and IPSec) using different transport protocols (UDP, TCP and SCTP) showed that this delay doubles for TLS compared to no SIP security scenario [21]. A possible issue with results might be that they are product of simulation and not a test of a real system. Test on real system was performed by Sureshkumar and Dutta [22]. They measured Call setup time, Mean number of calls, Memory utilization, CPU utilization and queue size for four different scenarios: UDP and TCP with no security, TLS-authentication and TLS-encryption. They report increase of all measured parameters for TLS vs. UDP to be less than 100% and even smaller increase for TLS vs. TCP. The most relevant is work of Shen et al. [23]. They did a very thorough testing. First, they had four different deployment models: proxy chain, outbound proxy, inbound proxy and local proxy. In addition they tested various combinations of transport protocols (UDP, TCP, TLS with 3DES and TLS with AES) with and without authentication, for a total of eight. Further, they tested different TLS configurations (e.g. with or without mutual authentication or session resumption). Their testbed consisted of OpenSIP, OpenSSL on Linux with Intel-based server hardware. That paper provides detail analysis of costs on CPU and its causes. They state that using TLS can reduce performance by up to a factor of 17 compared to the typical case of SIP-over-UDP. It must be pointed out that this factor is for the case of TLS with mutual authentication compared to UDP without authentication. We consider fewer cases that we believe are more relevant but we test for more SIP to performance metrics as per [3]. In fact, we measured the performance of SIP server by monitoring the processor load and consumed RAM memory. In comparison to the other mentioned papers, we used Elastix as possibly the most powerful packaging of a very powerful and popular telephony development toolkit, the Asterisk server. As opposed to [23] we measured simultaneously the performance of SIP protocol. This was evaluated using RRD and SRD metrics as defined in [3].

III. Implementing Secure VoIP Communication

The key aspect of secure VoIP communication is the security of the signaling path, which is provided by SIP protocol. The main components for securing SIP communication are: confidentiality and integrity of signaling messages and authentication of parties. Several SIP security mechanisms are specified in [4] as mentioned before. TLS is a widespread and well-known transport protocol for secure communication. It provides confidentiality and integrity of the transmission channel for data exchanged between applications at higher level. TLS makes use of X.509 certificates to associate a public key with the certificate subject. This relationship is confirmed by the digital signature of the certificate authority (CA), which involves public key cryptography. TLS allows both entities in a communication link to authenticate each other. However, TLS with mutual authentication is impractical and a challenging implementation due to problems with key distribution. In [23] it is shown that using TLS with mutual authentication can reduce performance by up to a factor of 17 compared to SIP-over-UDP. Further, TLS support is not yet fully implemented in all currently available SIP UAC softphone solutions. Some of them with TLS support are: Blink, Bria, Linphone, MicroSIP and Yate. Not all support mutual authentication using client certificates. For this reasons, in our proposed VoIP security approach we used HTTP digest authentication scheme for verifying the identity of users and performing message authentication. On the other side, with TLS clients authenticate the server using the server's public key associated with the server's certificate. This assured SIP mutual authentication. SIP message privacy protection is en-

abled using encryption with keys that are exchanged during TLS handshake procedure. TLS message integrity is ensured by sending additionally a keyed digest of the original message using a secret key shared between the sender and receiver.

This paper is focused on relative performance evaluation of SIP server in local proxy operation mode when using secured SIP message transport (TLS) over non-secured transport protocols (TCP/UDP).

IV. Experiment Setup

In order to perform performance testing of SIP server and measure some basic SIP performance metrics in different cases, when using SIP/UDP, SIP/TCP and SIP/TLS, we used the testbed shown in Fig. 1. Our testing platform consisted of three main elements: a SIP server, a traffic generator and a monitoring system. In addition, we used a DNS server based on `bind9` software that translated the domain name of the Elastix server (`centrala1.ntpmng.com`) into its IP address. All software we used is freely-available and Open Source. We installed the particular software on virtual machines using VirtualBox virtualization tool. Below we describe in detail the hardware and software used in our experiments.

A. SIP server software

We used the latest stable release of Elastix server, 64-bit version of Elastix 2.3.0, as SIP server. Elastix project begun as a call report interface for Asterisk but today it includes multiple features and functionalities (VoIP, Mail server, IM server, Fax server etc.) all realized through Open Source tools compiled together. However, we used the IP PBX functionality. The core IP PBX represented Asterisk. The currently integrated version of Asterisk in Elastix 2.3.0 is Asterisk 1.8.11.0-0, which is a Long Term Support (LTS) release. Asterisk is built on loadable components called `modules`. All of them are loaded based on the `/etc/asterisk/modules.conf` file. For example, Asterisk uses `res_srtp` module for SRTP support. Asterisk supports TLS for SIP signaling encryption and SRTP for media streams encryption. For this purpose the packages OpenSSL (`openssl-devel` on CentOS) and LibSRTP (`libsrtp-devel` on CentOS) must be installed. Elastix 2.3.0 includes both of them and the RTSP module `res_srtp` is also compiled and installed. The main configuration files are `extensions.conf` and `sip.conf`, which include variety of other files, all located in the `/etc/asterisk/` directory. The file `sip_additional.conf` includes information about user extensions. We created 2000 user SIP peers, 1000 per side. This was effectively done by importing a `.csv` file through Elastix Web User Interface.

The default transport protocol used in Elastix server is UDP. We used target options in the `sip.conf` file to enable SIP over TCP, and SIP over TLS. For TCP support the options `tcpenable` and `tcpbindaddr` were set. Analogous, the global options `tlsenable` and `tlsbindaddr` were set for TLS support. Additionally, in the `sip_additional.conf` file for every user extension the transport option was set to `tcp` or `tls`, depending of the testing scenario. The next step was to generate a self-signed certificate. This was done using the OpenSSL tool. A

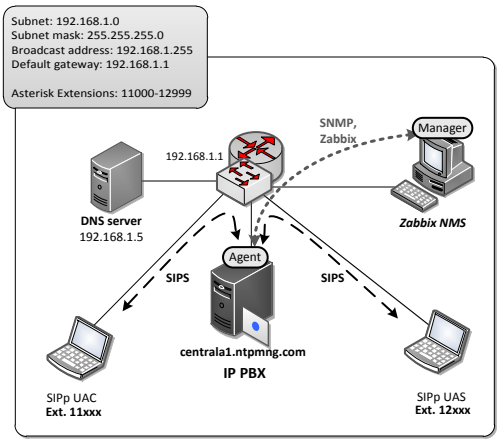

Figure 1. Testbed diagram

CA certificate (`ca.crt`) and a server certificate (`server.pem`) were generated. The `ca.crt` connects the identity of the CA with its public key that the client uses to validate the server's certificate. Finally, we needed to point Elastix to the CA and server certificate location using `tlscertfile` and `tlscafile` options in the `sip.conf` file, respectively, after we had moved both certificates to the appropriate locations in Elastix server. The default path of the files is `/var/lib/asterisk/keys/`. Lastly, the certificate authority (`ca.crt`) was attached to the client machines, so they could validate the server certificate. We installed Elastix server on a Guest OS CentOS 5.7 (File Descriptor Limit=1024, Stack Size Limit=10240) with:

- 4x Virtual x64 Processor Core @ 2.4 GHz,
- 3430 MB Virtual RAM.

B. Traffic generator software

For the purpose of SIP server performance testing we needed to generate hundreds of calls. It is difficult to achieve quite a few established calls using softphones on both sides on single physical machines. Consequently, we used a SIP load generator, called SIPp (version 2.3 built with TLS support), to simulate client sessions [24]. SIPp enabled the generation of high SIP load to run performance tests and also the measurement of delay parameters of individual SIP calls. SIP call flows are defined in XML scenario files which are loaded when running SIPp. SIPp supports TCP and UDP over multiple sockets and advanced features like TLS, SIP authentication, UDP retransmissions and SIP header field injection from external CSV file to emulate live users. SIPp also allows generation of RTP traffic, but does not support SRTP. On the SIP UAC side we used multi socket mode in all cases UDP/TCP/TLS to emulate user agents calling a SIP server. For each new call a new socket was opened. UDP retransmission followed the mechanism described in [2].

Since we wanted to test the performance of a relatively busy SIP server, we needed to generate 1000 client sessions on the calling party side. However, during testing we noticed that SIPp is capable of creating about 250 simultaneous calls.

Therefore, due to the software limitations of SIPp we needed to install 4 virtual machines each running a SIPp UAC to achieve 1000 calls per second. Another issue with SIPp was that when started it could act only as UAC or UAS, but not both at the same time on the same machine. The problem became clear after taking a look at the SIPp call flow diagram of our desired testing scenario on the SIP UAS side shown in Fig. 2. The calling client party acted as classical as SIP UAC during register procedure and call setup procedures. It sent SIP requests and waited for the corresponding answers. However, it is required that the called party is registered with Elastix server in order to be able to accept calls. Considering Fig. 2 it is obvious that the called party must simultaneously act as SIP UAC (when sending SIP REGISTER) and SIP UAS (when receiving SIP requests from UAC). Running a separate UAC-mode script for registering the SIP called parties is useful only in UDP testing mode. In TCP and TLS mode after the registration script is executed a TCP message with set FIN flag is sent for each simulated client session. This caused Elastix server to discard the registrations. For this purpose, some VoIP platforms were useful (Yate, Linphone or PJSUA) but were limited in terms of ineffective user accounts creation or speed of processing incoming calls. Fortunately, modifying the SIPp source code allowed us to run SIPp in a mixture scenario of UAC and UAS mode concurrently. At [25] there is a patch that was created by Matthew Briggs for this purpose. It has to be mentioned that the first, UAC mode, script had run long enough to enable finishing the execution of the second, UAS, script. This was achieved by setting appropriate `<pause>` message command at the end of the first script.

Below are the characteristics of the used SIPp UAC and SIPp UAC/UAS virtual machines both with 20 GB HDD, File Descriptor Limit 1024 and Stack Size Limit 8192.

SIPp UAC VM:

- Guest OS Ubuntu 12.04 x64 Server Edition,
- 2x Virtual x64 Processor Core @ 3.4 GHz,
- 1024 MB Virtual RAM.

SIPp UAC/UAS VM:

- Guest OS Ubuntu 12.04 x64 Server Edition,
- 4x Virtual x64 Processor Core @ 2.27 GHz,
- 2452 MB Virtual RAM.

C. Monitoring system software

In Fig. 1 the three principal components of network management architecture can be identified: a managing entity (the manager), the managed devices called agent (Elastix server) and a network management protocol. The function of the managing entity was realized by Zabbix [26] network monitoring application. Zabbix controlled the collection, processing, analysis and display of network management information in real-time. The values about the processor load and consumed RAM were retrieved using the protocol specific for Zabbix over TCP, whereas the number of concurrent calls using the Simple Network Management Protocol (SNMP). Accordingly, we installed and configured Zabbix agent and SNMP agent on

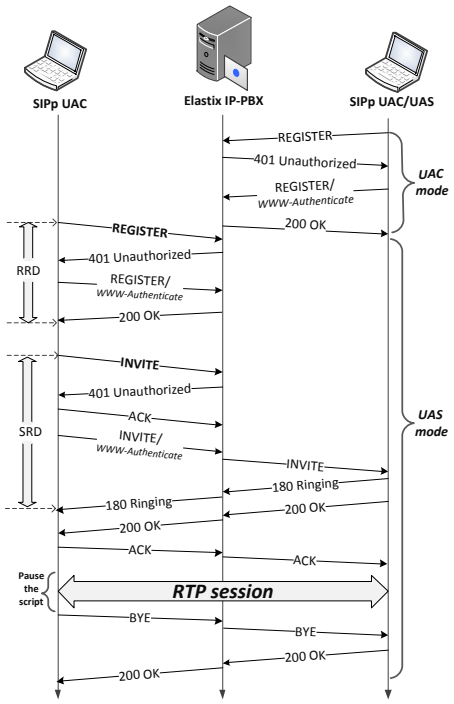

Figure 2. MSC diagram for testing XML scenario

Elastix server and added Asterisk and Digium MIB files into `/usr/share/snmp/mibs` directory on Elastix server.

D. Hardware and connectivity

The Elastix server hardware had Intel Core i5 2.4GHz processor with 6GB RAM and 500GB hard drives. All 4 SIPp UAC virtual machines were on the same host machine that had Intel Core i7 3.4GHz processor with 8GB RAM and 720GB hard drives. SIPp UAC/UAS virtual machine was on host machine that has Intel Core i3 2.27GHz processor with 4GB RAM and 320GB hard drives. The devices were connected through 1 Gbit Ethernet cable with UBIQUITI AirRouter.

V. Performance Evaluation

The goal of our experiment was to analyze the ability of VoIP PBX Elastix server to handle multiple simultaneous registrations and call setups. We performed performance evaluation through testing hardware utilization of Elastix server during the SIP scenario defined in Fig. 2 as well as some standard metrics for measuring and reporting SIP performance. It is important to emphasize that our test scenario did not involve the exchange of media (RTP) traffic, because our goal was to test the impact of SIP signaling protocol over different transport protocols on the performance of VoIP PBX server and VoIP service. Because of this after a call was established in the SIPp scenario script we set a pause so that the call remained active for some time. Processor load and consumed RAM measuring was achieved using Zabbix network monitoring system as described in the previous section. SIP performance was evaluated using the proposed methodology

for testing and benchmarking SIP infrastructure in RFC6076 [3]. The metrics we chose for testing are: RRD (Registration Request Delay) and SRD (Session Request Delay) [10]. RRD is a measurement of delay in responding to a UA REGISTER request. SRD is the time interval from when the first bit of the initial INVITE message is sent by the originating user agent to the intended destination agent, until the last bit of the first provisional response is received (180 Ringing). Both, RRD and SRD are illustrated on Fig. 2. We measured RRD and SRD as specified in [3] at the originating SIPp UA just for successful session setup. This was achieved using the ability of SIPp to dynamically display statistics about running tests, including response times. For this purpose we specified the start (with `start_rtd` attribute) and the endpoint for two counters (with `rtd` attribute), each one for RRD and SRD computation.

To dump the response times in an external `.csv` file the `-trace_rtt` option was additionally used at the SIPp command line. This allowed us to effectively analyze and process the obtained results. Results for every single parallel SIP session that are collected from all machines from which we generated traffic were arithmetically averaged and as such are presented in the following section.

A. Results

Depending on the transport protocols over which SIP signaling is established, each figure has three corresponding characteristic curves. For each of the protocols we used the same scenario, but with different load level, or more precisely with different rates for generating concurrent calls. Each tested configuration regardless of the transport protocol has SIP authentication enabled. During testing we used TLS with TLS-AES chipper suite. The call generation rates were increased until we noticed that SIPp enters in saturation. Saturation occurs when SIPp starts to generate calls with rate less than rate that we specified when starting the generator. This problem was easily avoided by distributing the generation of calls on multiple machines. Due to the limitations of available equipment, the maximum achieved number of concurrent calls was 1300, after the distribution of SIPp generator on 4 machines. For each protocol we measured peek throughput of concurrent calls. Also for each scenario we measured RRD, SRD, processor load, and consumed RAM memory.

Fig. 3 shows the peek throughput of concurrent calls depending on transport protocol. As we expected SIP over UDP gives the best SIP server performance, followed by SIP over TCP and SIP over TLS respectively.

Fig. 4 and Fig. 5 show RRD and SRD respectively, for different transport protocol configurations and call rates. Note that TCP and TLS from the standpoint of RRD and SRD have worse performance than UDP because of the time needed for establishing connections. These results should be viewed relatively, because SIPp needs extra time to create call statistics. A better way to collect statistics would be to analyze packets captured by a network protocol analyzer.

Fig. 6 and Fig. 7 show peek processor load and consumed RAM memory, for different transport protocol configurations and call rates. Higher call rates cause a greater processor load. Again UDP has best performance. The graphics of consumed RAM memory give interesting results. It should be noted that the consumption of RAM memory for each protocol individually is relatively constant regardless of the call rate. Similar results are shown in paper [22]. However, TLS requires more RAM memory in regard to TCP and UDP, which have approximately the same demands on.

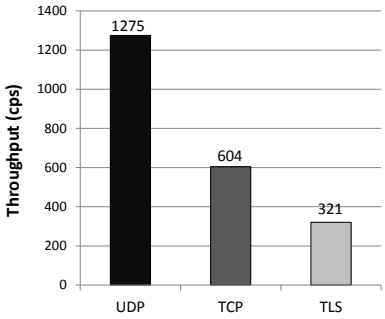

Figure 3. Peak Throughput of concurrent calls

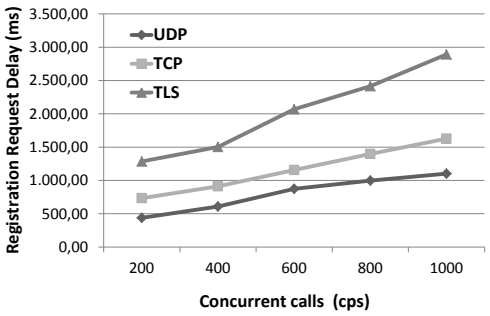

Figure 4. Average Registration Request Delay vs. concurrent calls

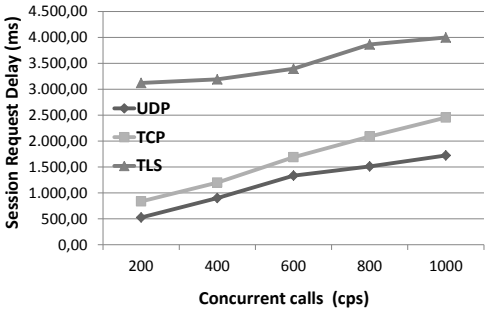

Figure 5. Average Session Request Delay vs. concurrent calls

VI. Conclusion and Future Work

Securing the SIP signalization is one of the primary goals when implementing secure VoIP networks. By using SIP over TLS based signaling, SIP signalization is secured. However the

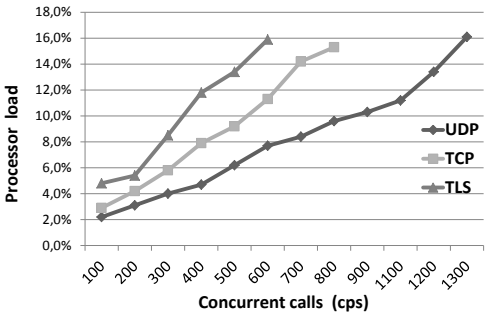

Figure 6. Peek processor load vs. Concurrent calls

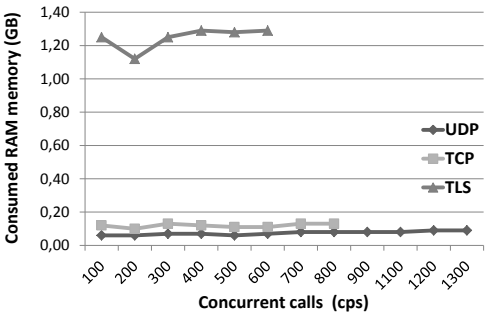

Figure 7. Consumed RAM memory vs. concurrent calls

use of TLS brings additional overhead that affects performance of VoIP service. In this paper we provided experimental results, obtained by testing performance in our testbed environment. The results show noticeable decreases in performance of VoIP service and higher demands on hardware when using SIP over TLS based signaling. In most metrics the performance of SIP over TLS based signaling are four times lower than the SIP signaling over UDP. When we analyzed the consumption of RAM memory, SIP over TLS based signaling has approximately 1 GB greater need than SIP over UDP and SIP over TCP signaling regardless of number of concurrent calls. Our testbed environment was based on virtual machines. This is justified by the fact that we observed relative performance.

In the future we plan to form a testbed environment based on multiple hardware machines with higher performance. In this way we expect to get the opportunity for generating higher load. Also, we plan to extend the capabilities of SIPp tool in order to provide support for generating SRTP protocol traffic. After that we will be able to test, measure and compare performances of VoIP services and server hardware for RTP and SRTP protocol.

REFERENCES

[1] J. Bilien, E. Eliasson, J. Orrblad, and J. O. Vatn, "Secure voip: call establishment and media protection," *2nd Workshop on Securing Voice over IP*, Jun. 2005.

[2] J. Rosenberg, H. Schulzrinne *et al.*, "Sip: Session initiation protocol," *IETF RFC 3261*, Jun. 2002.

[3] D. Malas and A. Morton, "Basic telephony sip end-to-end performance metrics," *IETF RFC 6076*, Jan. 2011.

[4] H. Schulzrinne, S. Narayanan, J. Lennox, and M. Doyle, "Sipstone - benchmarking sip server performance," 2002. [Online]. Available: http://hdl.handle.net/10022/AC:P:29277

[5] M. Cortes, J. R. Ensor, and J. O. Esteban, "On sip performance," *Bell Labs Technical Journal*, vol. 9, no. 3, pp. 155–172, 2004. [Online]. Available: http://dx.doi.org/10.1002/bltj.20048

[6] V. Gurbani, L. Jagadeesan, and V. Mendiratta, "Characterizing session initiation protocol (sip) network performance and reliability," *Service Availability*, vol. 3694, pp. 196–211, 2005.

[7] E. M. Nahum, J. Tracey, and C. P. Wright, "Evaluating sip server performance," *SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 1, pp. 349–350, Jun. 2007. [Online]. Available: http://doi.acm.org/10.1145/1269899.1254924

[8] K. K. Ram, I. C. Fedeli, A. L. Cox, and S. Rixner, "Explaining the impact of network transport protocols on sip proxy performance," *IEEE International Symposium on Performance Analysis of Systems and software (ISPASS)*, pp. 75 –84, Apr. 2008.

[9] M. Voznak and J. Rozhon, "Approach to stress tests in sip environment based on marginal analysis," *Telecommunication Systems*, pp. 1–11, 2011, 10.1007/s11235-011-9525-1. [Online]. Available: http://dx.doi.org/10.1007/s11235-011-9525-1

[10] M. Voznak, "Evaluating the performance of sip infrastructure," *Geant - Terena*, 2011. [Online]. Available: http://www.terena.org/activities/campus-bp/pdf/gn3-na3-t4-cbpd163.pdf

[11] C. Shen, H. Schulzrinne, and E. Nahum, "Session initiation protocol (sip) server overload control: Design and evaluation," *Principles, Systems and Applications of IP Telecommunications. Services and Security for Next Generation Networks*, vol. 5310, pp. 149–173, 2008.

[12] V. Hilt and I. Widjaja, "Controlling overload in networks of sip servers," *IEEE International Conference on Network Protocols (ICNP)*, pp. 83 – 93, Oct. 2008.

[13] E. Noel and C. R. Johnson, "Novel overload controls for sip networks," *21st Internationa Teletraffic Congress (ITC)*, pp. 1 –8, Sep. 2009.

[14] A. Abdelal and W. Matragi, "Signal-based overload control for sip servers," *7th IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 1 –7, Jan. 2010.

[15] V. Hilt, E. Noel, C. Shen, and A. Abdelal, "Design considerations forsession initiation protocol (sip) overload control," *IETF RFC 6357*, Aug. 2011.

[16] G. Apostolopoulos, V. Peris, and D. Saha, "Transport layer security: how much does it really cost?" *18'th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 2, pp. 717 –725, Mar. 1999.

[17] K. Kant, R. Iyer, and P. Mohapatra, "Architectural impact of secure socket layer on internet servers," *International Conference on Computer Design (ICCD)*, pp. 7 –14, 2000.

[18] L. Zhao, R. Iyer, S. Makineni, and L. Bhuyan, "Anatomy and performance of ssl processing," *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 197 –206, Mar. 2005.

[19] C. Coarfa, P. Druschel, and D. S. Wallach, "Performance analysis of tls web servers," *ACM Trans. Comput. Syst.*, vol. 24, no. 1, pp. 39–69, Feb. 2006. [Online]. Available: http://doi.acm.org/10.1145/1124153.1124155

[20] S. Salsano, L. Veltri, and D. Papalilo, "Sip security issues: the sip authentication procedure and its processing load," *Network, IEEE*, vol. 16, no. 6, pp. 38 – 44, Nov./Dec. 2002.

[21] E. C. Cha, H. K. Choi, and S. J. Cho, "Evaluation of security protocols for the session initiation protocol," *16th International Conference on Computer Communications and Networks (ICCCN)*, pp. 611 –616, Aug. 2007.

[22] S. V. Subramanian and R. Dutta, "Comparative study of secure vs non-secure transport protocols on the sip proxy server performance: An experimental approach," *International Conference on Advances in Recent Technologies in Communication and Computing*, pp. 301–305, Oct. 2010.

[23] C. Shen, E. Nahum, H. Schulzrinne, and C. P. Wright, "The impact of tls on sip server performance: Measurement and modeling," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 4, pp. 1217 – 1230, Aug. 2012.

[24] R. Gayraud and O. Jacques, "Sipp." [Online]. Available: http://sipp.sourceforge.net

[25] "Mail-archive on sipp." [Online]. Available: http://www.mail-archive.com/sipp-users@lists.sourceforge.net/msg05579.html

[26] R. Olups, "Zabbix 1.8 network monitoring," *PACKT Publishing Ltd.*, 2010.