A POLICY-BASED DESIGN METHODOLOGY AND PERFORMANCE EVALUATION FRAMEWORK FOR A SECURE VOIP INFRASTRUCTURE

Valentina Casola, Raffaele Chianese, Nicola Mazzocca, Massimiliano Rak
Seconda Universita' di Napoli - Dipartimento di Ingegneria dell'Informazione
Aversa (CE) - Italy

Antonino Mazzeo
Universita' degli Studi di Napoli, Federico II - Dipartimento di Informatica e Sistemistica
Naples, Italy

Keywords: Security Mechanisms, Design Methodology, SIP Performance.

Abstract: The increasing interest in telecommunication systems and the wide spreading of computer networks within commercial and scientific field, are going in the direction of a deep integration of phone and data systems into a single network infrastructure. We are particular interested in security issues that arise in such context, thinking, for example, about authentication and billing problems. The available security mechanisms can offer different guarantees but their introduction greatly affect the whole system performances. To guarantee system usability and an efficient resource usage, we propose a design methodology and a framework to evaluate how each security choice affects the whole system performances and help system designers in evaluating the performance-security trade-off.

1 INTRODUCTION

Traditional telephone systems are based on circuit switching mechanisms; this necessarily implies that all needed resources need to be allocated along the entire communication, even during break times; now we are assisting the first evolution in telecommunication systems, thanks to digital transmission: in fact, the real innovation has happened when vocal signals have begun to be transmitted on Internet. Voice over IP (VoIP) concept is based on this innovation. Actually there are two protocols which are considered standard in the VoIP context: H.323 and SIP. Each standard defines communication rules between the two terminals, and also provides the definition of complex architectures for managing calls establishment, and holding on. But while the first one provides for architectural dedicated solutions, the second one is a textual protocol, which places itself at the Application level of the TCP/IP architecture, and it is more interesting for us. For this reason in Section 2 a very brief overview on SIP architecture (components and protocols) is laid; it will be very brief because the target of this paper is quite different; we are particularly interested in security issues that arise in such context, thinking, for example, of the authentication and billing problems. The available security mechanisms can offer different guarantees but their introduction greatly affects the whole system performances. In order to guarantee system usability and an efficient resource usage, we propose a security policy (representative of different security mechanisms) and a methodology which could help a VoIP designer in evaluating how each security choice affects the whole system performances.

The paper is structured as follows: in the next Section we will illustrate a brief overview of a SIP-based VoIP architecture; in Sections 3 and 4 we will describe the architecture design parameters (both architectural and security) and all the steps of the design methodology needed to support system designers in evaluating the performance-security trade-off and, so, to help them in choosing system configuration parameters to meet performance and security requirements. In Section 5 we will propose a performance evaluation development framework and we will illustrate the definition of interesting tests, very suitable to take in count chosen security mechanisms. Finally, in Section 6 a full Case Study will be illustrated and some conclusion and future works will be discussed.

2 SIP AND SECURITY

In this Section we report an overview on SIP architecture and some security mechanisms (RFC3329, 2003; Cisco1, 2002 ; Cisco2, 2002) which could be found in literature, this section is absolutely not exhaustive but a lot of useful references will be provided.

2.1 SIP Architecture Overview

SIP (Session Initiation Protocol) is a protocol able to set up a session among one or several users; the definition of session, which we refer to, is given in (RFC3261, 2002), that is: "A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session".

For brevity sake, here we just report a number of entities which take part in SIP communication, the most important components of the SIP architecture are:

- **SIP User Agent** which represents an extremity of the connection, that is the application used by the final user in order to take part in the conversation. It can act as client (User Agent Client - UAC), if it makes requests, and it can act as server (User Agent Server - UAS), if it must grant a request.
- **SIP Network Server**, which manages the signaling to set up the connection. It can be of different types: *a Proxy Server*: it is an intermediate application which routes received SIP messages; its aim is putting in touch the calling part with the called part. *A Redirect Server*: it has only the task to accept the SIP requests, making addresses translation to set up the call. *A Registrar* is a server which accepts particular requests of registration by new SIP users, which give the necessary information to allow their location; such information are appropriately inserted in a database called "location service".

According to the SIP protocol, the described entities communicate through SIP suitable messages. The protocol is of peer-to-peer type, even if the User Agent interaction is of client-server type. The SIP element, who takes the initiative in the communication for settling a session, assumes the role of client, sending on a request of INVITE type towards a SIP user, who is invited to the dialogue (SIP request). A proxy has both client and server functionalities, while a redirect server simply transmits responses to client request, by mapping a specified address into actual addresses.

2.2 Security in SIP

Currently, SIP requirements identify either HTTP Digest or S-MIME as security mechanisms for providing selective security at SIP level. Both mechanisms do not guarantee a high level of security, furthermore, only the first one provides an authentication mechanism. (radsip, 2001) examines the proposal for a SIP extension to allow authentication of an user-agent through a proxy with a RADIUS (RFC2865, 2000) server; basic RADIUS protocol is not too much secure, especially because all messages are exchanged in plain text.

Some proposals which face such problems are based on TLS session between the user-agent and the proxy (TLS, 1999); with TLS, all SIP messages are exchanged in a secure environment. TLS seems to be a good security solution but it requires overhead and complexity, and the client needs to be changed *ad hoc* to support the protocol. In (kerbpki, 2003) is described how to use Kerberos-PKI mechanism (RFC1510, 1993) for providing end-to-end security between SIP clients. Once the client is authenticated, and Kerberos key is generated, SIP client can use the given key for encrypting and authenticating each others. Details about these and other security mechanisms and protocols are omitted for brevity sake; however, we are carrying on our analysis on different mechanisms (diameter, 2003; RFC1994, 1996), trying to let the SIP-architecture support different security-levels, because, as explained later, these could strongly affect the definitive performance of the architecture.

3 ARCHITECTURE DESIGN PARAMETERS

The design of systems, which are critical for time and security, is a very complex task that needs to be faced in a systematic way to take in count all critical parameters. Almost all choices, needed from the early stages of the design process, could heavy influence the global system security/performance; this suggests us to consider all the playing parameters and analyze their possible impact on those critical aspects, as soon as possible during the design phases. At this aim, in this paper we propose a "policy-based and performance-oriented framework" to be able to design a SIP architecture with pre-assigned performance and security requirements. The framework has been designed on standard SIP components and specifications and on a set of security rules about the way these components must be interconnected to guarantee a pre-assigned set of functionality and a pre-defined security level which characterizes the specific security mechanism. As said, the way to interconnect the basic elements is one of the most critical aspect, it has a great impact on system security, on availability and on performances, too.

Table 1: Logical Components

Component Name	Description
Proxy	The SIP proxy
Radius	The Radius Server
AS	Kerberos Authentication Server
TGS	Kerberos Ticket Granting Server

In particular, the defined framework takes in consideration the following elements:

- the logical base components of the SIP architecture,
- a set of security rules (the policy),
- the physical components of the real SIP architecture to implement.

Logical Components

The Logical components are the base components of a SIP architecture as seen in Section 2, they are reported in Table 1.

An instance of the Logical architecture will be represented in this paper by means of a vector V of 4 components whose numerical values represent the number of logical components really present in the architecture; for example V=(2,0,0,0) indicates that there are two Proxies and no one security server.

Security Policy

The security policy rules are reported by means of tables; as shown in Table 2 (a) the parameters that we consider critical for security are just four: the authentication system, the communication type between UA and Proxy and the communication type among Proxies and finally the number of domains. This policy is very simple because we have just considered security problems related to authentication mechanisms at application and network levels, indeed, the methodology we are proposing remains still valid if we add other security parameters (further authentication server types, authorization mechanisms, and so on). The security policy adopted, the description of its parameters and the values they could actually assume are reported in Table 2 (b,c,d)

Physical Components

The physical components we have located are just two: the *physical nodes* on which the logical components must be mapped and the *network components* (their numbers and performance/features usually represent an important parameter that designers need to consider to build the effective infrastructure).

Table 2: Security Policy and its Parameters

Parameter	Description
Authentication System	....
UA-Proxy Communication	....
Proxy-Proxy Communication	....
Domains Number	....

Authentication System Values	Description
No Mechanism	....
Radius	....
Kerberos	....

UA-Proxy and Proxy-Proxy Communication Values	Description
Plain text	....
Encrypted text	....
SSL Protocol	....
Kerberos Protocol	....

Domains Number Values	Description
Integer Value	....

4 DESIGN METHODOLOGY

In this Section, we will describe the design methodology which could help any system builder while taking in count critical aspects in all design phases. The methodology we propose here is made of 6 steps which are described here in details:

1. Choosing Phase;
2. Logical Architecture definition;
3. Physical Architecture definition;
4. Architecture Evaluation;
5. Repeat from step 4;
6. Choosing the optimal architecture.

Steps 1-2 The first step of our methodology consists of choosing the security policy; this step in fact strongly affects the logical architecture as it implies the presence of a number of minimal specific logical components and a number of specific functionalities (for example if we decide to implement SSL protocol among Proxies, specific mechanisms must be implemented). According to the defined policy, if we represent each policy instance by a vector P of four components (the number of components is representative of the number of the parameters of the policy), and if we consider the basic *logical elements* of the

Table 3: Policy and Logical Components relation

Policy	Proxy	Radius	AS,TGS
(0,x,x,x)	X	-	-
(1,x,x,x)	X	X	-
(2,3,0,x)	X	-	X
(2,0,3,x)	X	-	X
(2,3,3,x)	X	-	X

architecture, all the possible combinations are illustrated in Table 3. To each component of the vector is associated a numerical value which ordinally correspond to the real semantic value of that parameter (the row index of the corresponding table, starting by row zero) as it appears in Tables 2 (b,c,d); for example the instance (1,2,0,x) indicates that the 1-st component (Authentication System="1") assumes the value of the 2-nd row: "Radius"; the 2-nd component (UA-Proxy Communication="2") assumes the value "SSL Protocol"; the 3-rd component (Proxy-Proxy Communication="0") assumes the value "Plain Text"; the 4-rd component (Domains Number="x") could assume any value ("x" lower case); an X (upper case) in the other columns indicates the presence of the corresponding logical components in the final logical architecture.

With one policy instance, it is possible to combine logical components in different ways but the *kind* and the *minimal* number is defined by choosing the policy (see for example Table 4 of the case study); this implies that the functionalities of the global system, the number of user-domains, user's authentication mechanisms and the *role* that each logical components will assume in the architecture, are defined in this phase by properly choosing the policy.

Step 3 This phase includes the choice of the real available components on which the logical components will be mapped. Physical components do not affect system functionalities (if they permit the allocation of all minimal logical components) but global performances and security are certainly depended on this parameter (for example if the Radius Server is running on the same physical node of the Proxy to authenticate, the security involved is certainly less critical than a distributed approach).

Steps 4-5 These two steps consist in evaluating the performances of all possible final architectures. There are in fact more than one way to combine and map the logical components on the physical ones (see for example Table 5 of the case study). Once we have addressed all possible combination, we have to repeat the evaluation process for all corresponding solutions. The evaluation is made against performance measures for all solutions and the optimal one is implemented.

Step 6 The optimal architecture derived by the first steps is implemented.

5 PERFORMANCE EVALUATION

The security mechanisms introduced in the proposed architecture have a side effect on the overall performances of the system. In order to compare the different proposed solutions a set of performance indexes are needed. At the state of the art no standard benchmark exists for SIP-based architecture, even if the SIPStone (sipstone, 2002) benchmark is probably the most appropriate solution; it is very useful to choose the architecture to deploy in a real environment, but its metrics are of a too *high level* to really understand the performance effects of low level design choice. These considerations lead us to develop an evaluation framework that helps in developing dedicated tests.

In the following a brief overview of our performance evaluation development framework will be carried on. We will describe the way in which the workload will be described, how we organize and develop new tests and the performance indexes that we can evaluate. A real system evaluation will be given in the next section (Case Study).

5.1 Workload Modelling

While a common benchmark aims to study the system performances under real workloads, that usually reproduce common server usages, our tests aim at pointing out performance bottlenecks due to design choice. This means that instead of pointing out the system behavior under common user population, we aims to evaluate many different *system working conditions*[1]. This information can be adopted in future to predict the server behavior under many different workload condition (see future works). Note that many SIP Server implementations (almost all) create a new thread each time a new request arrives (this is true if a thread pool policy is adopted) so the number of the contemporary requests heavily affects the system performances (as will be shown in the final results). This consideration leads us to generate the tests as *closed loops*, in a closed-loop test, each new request is only issued after the completion of the previous one. In

[1]A working condition will represent a system state under which the architecture will show the same performance behavior to the same requests

this way, the number of waiting requests on the server is (almost) constant if we assume that the time spent in communication is less than the time spent by the Proxy to complete the requests[2]. Another important consideration is that in our tests the client behavior has to affect as few as possible the tests, so the testing UAC should be as simple as possible, closed-loops does not need to maintain and manage information about the sent requests. Thanks to this approach the number of parallel clients (N parameters) corresponds to the waiting requests on the server, and so the server working condition.

5.2 Tests Description

Our tests aim at pointing out the performance of the system design proposed for each functionality implemented. Tests can be organized in a hierarchic way as:

- **Transaction Level**: These tests aim at measuring the time spent in a SIP transaction. Different tests are needed for different transactions.
- **Session Level**: These tests aim at evaluating the performances of a complete SIP Session.
- **Workload Level**: This is the level at which SIPStone operates: it describes the System performances under known workload condition.

It is out of the scope of this paper to point out a more detailed classification of the tests and point out the state of our SIP evaluation framework. In this paper we will focus on a transaction level test for the REGISTER SIP operation and a simple Session Level test, because this tests will clearly exploit the effects of the security mechanism on the system performance, as we will point out in the following sections. Each test measures its own index, which is usually defined as the response time to the given sequence of SIP services on the client side. It is important to point out that all the performance index can be gathered from the client side, so that it is possible to perform an evaluation of an existing architecture without changing it.

REGISTER Transaction Level Test This test simply measures a REGISTER transaction on the target system, and its response. The result of this test will be the response time on the client, we will call it RRT (Register Response Time). Of course, security mechanisms have heavy impact on performances of this tests, in fact the response time will be different for any different authentication process.

[2]The testbed should assure this condition through the choice of the network

INVITE-BYE Session Level Test The Session is simply described as the Sequence INVITE-BYE from one UAC to another one. This simple test aims at pointing out the system behavior when managing many complete sessions. The result index will be named SRT (Session Response Time). In the test evaluation we will consider the response time to the RINGING signal as zero.

Final Considerations The Measurement methodology carried on is similar to the one proposed by SipStone: request rates (and so the number of parallel clients) are increased until the number of the transaction failure reachs an high value (15%). A graph for each test, reporting its index and the different proposed architectures, will be proposed. The proposed evaluation approach is oriented to exploit the effects of the design choice done on the target system, so it is of great help in building a high performance system; however, it is not a substitute of a standard benchmark, like SIPSTone, but it is proposed to work in conjunction with it to give a clear idea of the final result of the proposed architecture.

6 CASE STUDY

The laboratory dedicated to the SIP security architecture is composed by two node (Pentium II, 400Mhz, 512 MB and Peniutm III, 733 Mhz, 256 MB) and a fast ethernet switch. We adopted a RedHat Linux 7.3 as operating system, SUN Java 1.4.2 and NIST-SIP (nistsip, 2002; jainsip, 2003)as SIP stack implementation. A four node, biprocessor cluster is available as workload generator, this helps us to grant that client side do not affect heavily the performance indexes. In the following we will show step by step the analysis carried on during the system design.

6.1 Choose the Security Policy and the logical Architecture

Due to space limitations we will evaluate only few of the available security policies. Note that each policy can be mapped onto many logical architectures. Table 4 briefly describes the analyzed policies and the logical architectures that satisfy the policy requirements.

6.2 Choose the Physical Architecture

Each logical architecture can be carried out in many diverse physical configuration, tables 5 and 6 describe the analyzed configurations.
The first column of table 5 contains the name of the

Table 4: Security Policies and Logical Architecture

Policy	Logical Architectures
(0,0,0,0)	(1,0,0,0)
(0,0,0,1)	(2,0,0,0)
(1,0,0,0)	(1,1,0,0) (2,1,0,0)
(1,0,0,1)	(2,2,0,0) (1,2,0,0)

configuration and the second column the security policy and the logical chosen architecture, while the second and third columns of table 6 contain the logical components on the available nodes, where P identifies a Proxy Server Component, and R a Radius Server on the nodes SipSec1 and SipSec2.

Table 5: Configuration Name and Logical Architecture

Configuration	Logical Architecture
C1	[(0,0,0,0) ; (1,0,0,0)]
C2	[(0,0,0,0) ; (1,0,0,0)]
C3	[(1,0,0,0) ; (1,1,0,0)]
C4	[(1,0,0,0) ; (1,1,0,0)]
C5	[(1,0,0,0) ; (1,1,0,0)]
C6	[(1,0,0,0) ; (1,1,0,0)]
C7	[(1,0,0,0) ; (2,1,0,0)]
C8	[(1,0,0,0) ; (2,1,0,0)]

Table 6: Logical Architecture Mapped on the physical configuration

Configuration	SipSec1	SipSec2
C1	P	-
C2	-	P
C3	P	R
C4	R	P
C5	P+R	-
C6	-	P+R
C7	P+R	P
C8	P	P+R

Last two configurations correspond to a unique user domain (stored in the Radius server) whose users adopts two different access points (Proxies). For sake of simplicity we equally subdivide the load between the proxies.

6.3 Architectures Evaluation

In this last section we report some considerations to show the correctness of the approach. An interesting result is presented in Figure 1 which shows RRT values against the number of parallel clients for each configuration. In particular, the diagrams C1-reg.dat and C2-reg.dat respectively refer to REGISTER in a Register Server on the SipSec1 machine and on the SipSec2 machine, without any authentication mechanism. The diagrams C3-reg.dat and C4-reg.dat respectively refer to REGISTER in a Register Server on the SipSec1 machine and on the SipSec2 machine, with a local Radius Server as authentication mechanism. Finally, the diagrams C5-reg.dat and C6-reg.dat are different from the previous ones because the Radius Server is invoked remotely. We explicitly note that configurations C3 and C5, such as C4 and C6, have similar behavior; this means that the use of two nodes for the Radius server and the proxy has no real positive performance impact whilst it is a more expensive solution.

7 CONCLUSIONS AND FUTURE WORKS

Security issues and their effects on the system performances in VoIP architectures is an open research issue, on which only few work was done. In this paper we have proposed a design methodology to help in carrying out real SIP-based architectures integrated with security mechanisms and a framework for performance evaluation of the built systems. We have applied the proposed approach to a real case study, showing the effects of different design choices (both in terms of security mechanisms and of their implementation) on the target system performances. We have shown how to apply the methodology on a real case, building a real system in many different configurations and comparing them.

Future evolution of the work will cover both methodology and framework extensions. The methodology should be integrated with performance prediction tools that will help to predict design choice performance effects before real implementation on the system. Framework can be used both for testing existing systems to tune the predicting models and to validate final results in late development stages. Framework should be extended with new tests, pointing out relationship between tests and design choices. SIPstone benchmark results could be predicted from framework measurements and predictive models.

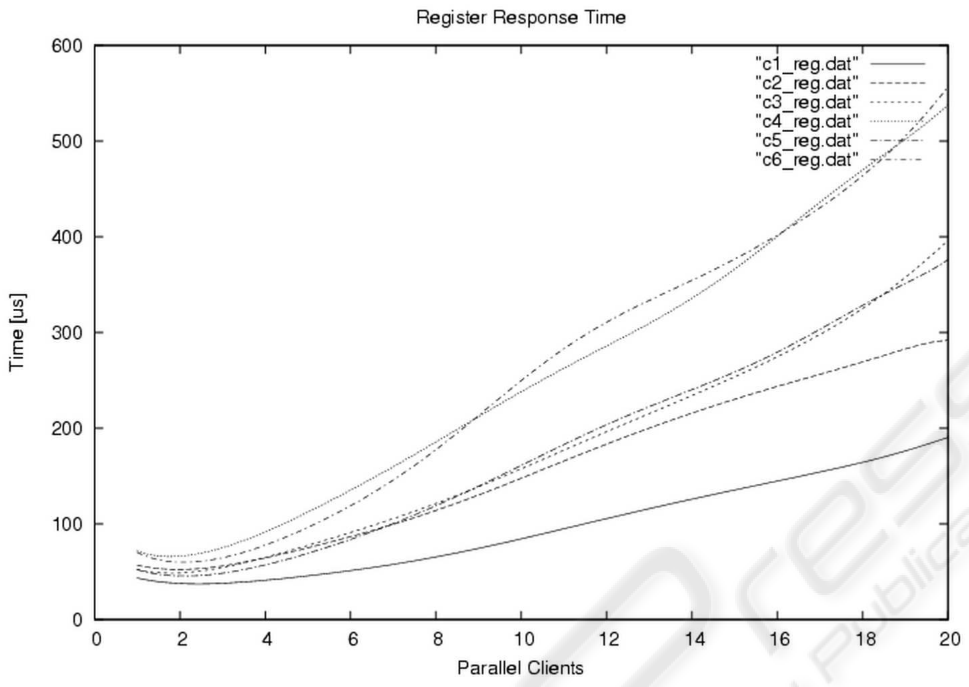

Figure 1: REGISTER Response Time Results

ACKNOWLEDGEMENT

This work was partially supported by ”Centro di Competenze” Regione Campania. We want to thank Roberto Torella, Pietro Ferrara and Fabio Sibiano for their technical support.

REFERENCES

J. Kohl and C. Neuman (1993). RFC1510 - The Kerberos Network Authentication Service V5. http://www.ietf.org, 1993.

M. Wagle and R. Aradhaya (2003). An Out-of-Band Authentication Procedure for SIP. http://www.ietf.org, 2003.

C. Rigney, S. Willens, A. Rubens, and W. Simpson (2000). RFC2865 – Remote Authentication Dial in User Service RADIUS. http://www.ietf.org, 2000.

C. Allen and T. Dierks (1999). RFC2246 – The TLS Protocol version 1.0. http://www.ietf.org, 1999.

J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler (2000). RFC3261 - Session Initiation Protocol SIP. http://www.ietf.org, 2002.

J. Arkko, V. Torvinen, G. Camarillo, A. Niemi, and T. Haukka (2003). RFC3329 - Security Mechanism Agreement for the Session Initiation Protocol (SIP). http://www.ietf.org, 2003.

W. Simpson (1994). RFC1994 - PPP Challenge Handshake Authentication Protocol. http://www.ietf.org, 1996.

Cisco White Paper. High-Availability Solutions for SIP Enabled Voice-over-IP Networks, 2002.

Cisco White Paper. Security in SIP-Based Networks, 2002.

M. Ranganathan (2002). JAIN-SIP: Architecture, Implementation, Testing. NIST. http://www-x.antd.nist.gov/proj/iptel/index.html, 2002.

H. Schulzrinne, S. Narayanan, J. Lennox, and M. Doyle (2002). SIPstone – Benchmarking SIP Server Performance. Columbia University, Ubiquity. http://www.sipstone.org, 2002.

JAIN SIP Relase 1.2 Specification Standard. Java Interface to the Session Initiation Protocol (SIP). Sun Microsystems, 2003.

B. Sterman (2001) Digest Authentication in SIP using RADIUS. SIP WorkGroup – INTERNET DRAFT, 2001.

P. Calhoun, J. Loughney, E. Guttman, G. Zorn and J. Arkko (2003). RFC3588 – Diameter Base Protocol. http://www.ietf.org, 2003.