Security Validation through Fault Injector Plug-in

Renato Galantini
State University of Campinas, UNICAMP, Brazil
Limeira, Brazil
renato.galantini@gmail.com

***Abstract*—Secure Socket Layer (SSL) and Transport Layer Security (TLS) are the protocol above TCP most widely deployed for guarantee privacy and authenticity of information exchanged between a Web Server and a Web client. SSL/TLS Handshake Protocol employs cryptosystem to ensure confidentiality, integrity and source authentication of sensitive data. This is an important protocol, but, due to the expansion of Internet, in fact, there is a long way to make the SSL/TLS protocol perfectly secure. In addition, there are still shortcomings and problems during the development of SSL/TLS and correctly configuration in Web Server. Successive attack is fatal for both the user and the company who uses these protocols to establish a safe channel to transfer information and can compromise the reliability of the whole software system. This work presents a plug-in developed to perform the security validation that is provided by SSL/TLS Handshake. The plug-in introduces typical attacks, selected from reported attacks, using the fault injection technique and was developed according to the architecture of current version of csXception tool, where the plug-in was integrated, allowing to perform the validation of security issues, preparing the tool to adequate itself to security aspects of evolving software system.**

Keywords: fault injection; software faults; software test; security tes

I. INTRODUCTION

Internet and Web Services have become popular, increasing the importance of considering the system security. The SSL/TLS can set up a valid secure channel between server and client, which can encode the plaintext, so that the third part, who intercept the message, can not disclose the original message without decode it.

SSL/TLS consists of two phases: handshake and data transfer. During the handshake process, the client and server use cipher suites to determine the parameters of security used during data transfer. There are potential dangers both during handshake and data transfer state, and, although the latest TLS version has fixed several secure hole of the old version, the successive attack, in practice, is not only a terrible event for the user who trusts the SSL/TLS, but also a challenge for software security area.

The work of Lee *et al*. [1] showed that in a study of more than 19.000 Web Servers, 98.36% of the servers have support for TLS, 97.92% of these servers have support for SSL 3.0, and still 85.37% have support for SSL 2.0. This statistical result, even related to 2006, shows the importance of SSL/TLS. This knowledge causes us to direct our efforts for the development of security tool and the validation using software fault injection techniques seems to be appropriated.

Fault injection techniques are widely used to assess the level of reliability and availability of a system and validate their mechanisms for handling errors, allowing solutions that are designed to handle exceptional situations to assist in discovering and implementing fault location and project with low interference in the system under test [2]. The plug-in presented in this paper aims to validate the systems security, which can improve higher levels of reliability in these systems. It is a prototype to test the handshake phase of SSL/TLS, developed in Java and tested using the architecture of current version of csXception tool [3].

The structure of the paper is as follows: Section II presents the related work, which established the basis of this work; Section III describes the approach for developing the plug-in. Section IV presents a case study using the plug-in and validating its operation. Section V presents the discussion about the results of the case study and Section VI presents conclusions and future works.

II. RELATED WORK

To develop this work was necessary to study different fields: fault injections, security protocols, mechanisms for security analysis, and attacks on protocols. We are focusing on guns of failure that can emulate communication failures and applications, the specification of SSL/TLS, and models and analysis of attacks reported in various sources.

A. Fault Injections

Fault injectors offer all features needed to test a system under faults. On the other hand, the known fault injectors limit the test experiment scope, because they are also responsible to offer the mechanisms to define the faults that can be injected.

The main drawback concerning the tools for fault injection is related to their unavailability for practical use. Both source code and binary files are, most of time, unavailable. The next paragraphs describe some tools and csXception, the tool that the proposed plug-in was integrated in and for this reason it was used in this work.

MENDOSUS implements fault injection into emulated networks. The faults that can be injected are related to crash and delay, applied to network components. To describe faultloads, this tool uses a script language.

FIRMAMENT is a fault injector that works at operating system level. The faults that can be injected are drop and delay of packets, besides arbitrary modification and duplication of packets.

JACA is based on computational reflection to inject interface faults in Java applications and is adequate to robustness tests. It does not need the source code of the application to be tested and has a graphical interface to make easy the specification of the characteristics of the system to be tested and the faults to be injected. Although Jaca tool injects faults in Java applications, its use is limited because it can inject only interface faults, which causes different impact in the system when compared with software faults, according to empirical evidences.

The csXception is a Software Implemented Fault Injection (SWIFI) and Monitoring tool that takes advantages on Processor Function Units. The csXception is the tool used in this work, boasts an automatic fault injection allows the verification and validation of a flexible manner. It was designed to accommodate a variety of techniques for fault injection and emulate software faults in embedded devices (hardware) and applications (software). The graphical user interface (GUI) provides a means for defining the faults to be loaded (faultload), running the experiment and analyzing the results, task that is provided by Easy Fault Definition (DPD) and csXception Analysis Tool (Xtract). It uses a standard SQL database to manage and to automate the experiments. Being used in several fields, it can deal with a set of injection techniques such as Boundary Scan Based Fault Injection (BSFI), Pin-Level Fault Injection (PLFI) beyond the traditional Software Implemented Fault Injection (SWIFI), allowing the use of a specific or multiple scenarios in accordance with the conditions (source code, others) and purposes.

Its architecture has characteristics of client/server model. It has a front-end application - Experiment Manager Environment (EME), which runs on a host computer is responsible for the control, management experience, collection, and statistical analysis. A core of the injection system to be evaluated (Target System) is responsible for the insertion of faults, in connection using TCP/IP. Fig. 1 illustrates this architecture.

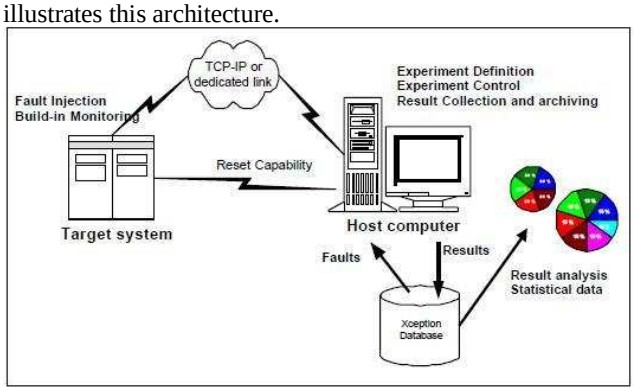

Figure 1. Xception Arquitecture [3]

Originally developed in C language [3], currently uses Java version, thus allowing the portability needed to run on multiple platforms or operating systems. In his structure coupling of new plug-ins, demanding that it fits the requirements of EME, the injection logic-based campaigns, and their way of communicating with the target (InfoBus) for TCP/IP or dedicated link. Keeping to these features our plug-in was done without any interference in the tool.

B. SSL/TLS

SSL/TLS implement security mechanisms based on cryptographic techniques, in order to meet security needs in the exchange of information. of the parties (client/server) through negotiation, mutual authentication, encrypted communication and integrity protection.

TLS is the standardization of SSL by IETF (Internet Engineering Task Force), which created a working group that established the versions RFC2246 [4], RFC4346 [4], RFC5246 [4] and RFC5746 [4]. The basic structure was maintained and there are occasional differences between TLS and SSL, for example, different cryptographic algorithms and new mail alerts.

The SSL/TLS is essentially composed of two main phases. The authentication phase handshake involving the parties to the exchange of keys and one with configuration information. Once the handshake is completed begins the process of information exchange encrypted.

The phase handshake is a multiphase process operating in general there are four main operational steps that are initiated by one party (client/server) in creating and closing a connection SSL/TLS. The phases are illustrated in Fig. 2 and a brief description of each phase [5]

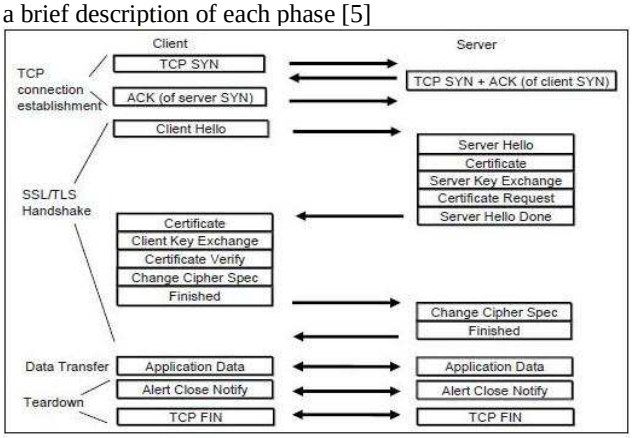

Figure 2. Handshake Protocol – SSL/TLS [5].

Phase 1: Setting up a TCP connection. At this stage, basically runs a three-way handshake between client and server in order to establish a TCP connection. The connection is peer-to-peer network between two nodes. The session is the association between a client and a server, to define a set of security parameters, such as the cryptographic ciphers supported.

Phase 2: Handshake Protocol. Responsible for establishing the connection SSL/TLS. At this stage, basically, two processes occur. The first is the key exchange, which means that the client and server must use asymmetric cryptographic techniques, and this process called pre-master secret. The pre-master secret is used independently to calculate a shared key or symmetric, called the master secret, which will be used later to derive the keys needed to ensure the integrity and confidentiality of communication. The

second process is the authentication of the parties normally done through digital certificates, and optional customer certification.

Phase 3: Data transfer. At this stage, the application data is exchanged between client and server. The SSL/TLS protocol encrypts all application layer data with a symmetric encryption algorithm and session key negotiated by the handshake protocol. The symmetric algorithm can be used to stream ciphers or block ciphers, depending on the ciphersuite negotiated during the handshake.

Phase 4: Closure. The main role of this phase is to close the connection SSL/TLS so that the parties client/server is notified that all application data has been transferred and the connection will be terminated.

C. Security analysis.

In our work we analyze the set of common properties of protocols (handshaking, message formats) and the correct usage of cryptographic ciphers (or cipher suites).

The tests are organized into three (3) procedures for evaluating the handshake phase of SSL/TLS, these can be run individually or in combinations:

- Test Charge Message: Test applied for the analysis of distributed systems, to change the contents of messages between the client and the server after establishing the initial parameters of connection. Allows the analysis of the behavior of the exceptions in the content of their broadcasts.
- Test Cipher Suites: Test to analyze the correct use of cryptographic ciphers. In the establishment phase connection (handshake) the set of cryptographic ciphers supported by both parties (client/server) are determined.
- Truncation Test: The Truncation Test does not allow you to receive the alert message (close_notify), closing the connection between the client and the server. Several types of security compromise such as embodiment, data tampering, are allowed.

The tests covering respectively the phases: transmission between the parties (client/server), connection establishment with all appropriate cipher suites, and the communication termination. The definition of the type of tests to implemented were based on a set of reports of public knowledge from several sources (NIST - National Vulnerability Database (NVD), SANS Institute, CERT Coordination Center) and adapted to the peculiarities of the tool. The tool was developed in Java language and is flexible to allow the inclusion of other attacks.

III. PLUG-IN

The plug-in is developed in Java using the package javax.net, the JESSIE (Java Secure Sockets Extension) and GNU Crypto that provides the facilities necessary to use the cryptographic ciphers. These packages do not have restrictions on use and changes, allowing their suitability to interact with the EME graphical user interface (GUI) of csXception.

csXception is an environment and platform-based development tool that is flexible enough to couple a plug-in. The work methodology used in this work conforms to the environmental circumstances of csXception and is schematic represented by the flow in Fig. 3.

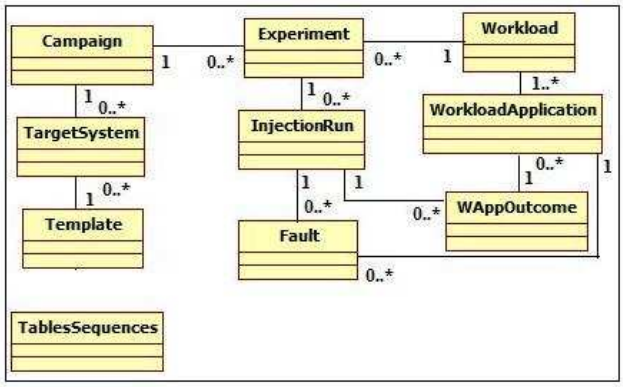

Figure 3. Flow Diagram of the project (partial and simplified).

The diagram of Fig. 3 shows that the tool follows a stream, which begins with the determination of the test campaign. Subsequently the experiment is defined. Given the workload (Workload), the system feature (Target System) defines the representative scenarios and, apply the injection of faults at runtime of these scenarios (Injection Run), taking into account the Faultload (defined in the class "Fault"). Finally, the analysis of experiments is performed (WAppOutcome). After these definitions, the tool presents the main interface of the plug-in, shown in Fig. 4.

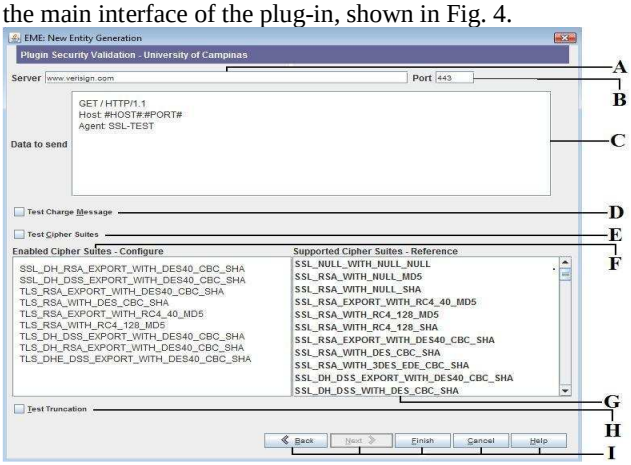

Figure 4. Interface plug-in for Validation of Computer Security.

The plug-in interface was developed aiming to be as simple and intuitive as possible. The final result is an interface that needs little configuration and that demands almost no experience of the user to use it. The interface areas that are emphasized in Fig. 4 correspond to:

Server (A): This field is intended to identify the address of the application (server) to be tested.

Port (B): Allows the user to entry the port for the application (server) that provides the services of SSL/TLS.

Being the port (443) standard, the field is filled by the same set and may be amended when necessary.

Data do send (C): Allows insertion of command to establish connection. Fig. 4 presents in this field the appropriated command for HTTPS (Hyper Text Transfer Protocol Secure) requesting and can be changed to other protocols that also use the concepts of SSL/TLS.

Test Charge Message (D): Must be selected to run the first type of test of the plug-in, which realizes the change of messages between the client and the server to verify/validate their behavior.

Test Cipher Suites (E): Must be selected to run the second type of test of the plug-in that attempts to connect to weak cryptographic ciphers that do not provide adequate security to existing software products.

Enable Cipher Suites – Configure (F): Allows the user to specify the cipher suites to be used in the second type of test of the plug-in. This selection requires the knowledge of cryptographic ciphers, given the completion of weak cryptographic ciphers and may be amended when necessary, using the items described in Supported Cipher Suites – Reference (G).

Supported Cipher Suites – Reference (G): Allows the selection of the set of cryptographic ciphers to be tested. The list was obtained through the use of GNU Crypto providers and Bouncy Castle that have no government restrictions and allow changes (add, delete) to adapt to your set of cryptographic ciphers.

Test Truncation (H): Must be selected to run the third type of test, which runs the truncation of the alert message to close the connection.

Basic Control Tool (I).

IV. Case Study

The tests were run on a laptop with an Intel Core 2 Duo P8400 2.26GHz, 4GB RAM, 250GB hard drive in Windows 7 Enterprise Edition. The Database Management System (DBMS) used was the PostgreSQL 9.0.2 for Windows 64bit and was installed on the same machine where the tests were performed.

The Test Cases specifications are defined as follows:

TABLE I. Test Charge Message

URL	Define the server of the application
Port	443 (or specify another)
Box to Check	Test Change Message

TABLE II. Test Cipher Suites

URL	Define the server of the application
Port	443 (or specify another)
Box to Check	Test Cipher Suites
Cipher Suite to Transfer	Define cipher suite in box Reference (G) and transfer to box Configure (F)

TABLE III. Test 3: Truncation

URL	Define the server of the application
Port	443 (or specify another)
Box to Check	Test Truncation

TABLE IV. Test Charge Message and Cipher Suites

URL	Define the server of the application
Port	443 (or specify another)
Box to Check	Test Change Message and Test Cipher Suites

TABLE V. Test Cipher Suites and Truncation

URL	Define the server of the application
Port	443 (or specify another)
Box to Check	Test Cipher Suites and Test Truncation
Cipher Suite to Transfer	Define cipher suite in box Reference (G) and transfer to box Configure (F)

V. Results and discussions

The plug-in integration and the execution of test campaign were performed by Critical Software, organization that is the owner of csXception tool.

This campaign was the first performed with the plug-in after it was integrated in the csXception. All testing equipment used and applications (server) were tested in controlled environments. At this moment new tests campaigns are scheduled to be performed in a commercial environment.

VI. Conclusions

The results show that it is possible to easily integrate a new plug-in to csXception tool. They show also that a fault injection techniques for software validation aspects of computer security can complement existent SWIFI tools, in our case the handshake phase of the SSL/TLS protocol. The plug-in is still a prototype that must undergo improvements in many aspects before being released as a commercial product.

Acknowledgment

Project being developed in collaboration with Critical Software SA.

References

[1] Lee H.K.; Malkin, T.; and Nahum, E. Cryptographic strength of SSL/TLS servers: current and recent practices. In IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, pages 83-92, New York, NY, USA, 2007. ACM.

[2] Arlat, J., Aguera, M., Amat, L., Crouzet, Y., Fabre, J., Laprie, J., Martins, E. and Powell, D. "Fault Injection for Dependability Validation–A Methodology and some Applications", IEEE Transactions on Software Engineering, 16 (2), Feb/1990, pp. 166-182.

[3] Xception. Available in http://www.csxception.com, December/2010.

[4] RFC(s). Availabe in http://www.ietf.org/rfc.html, December/2010.

[5] Berbecaru, D.; Lioy, A.; "On the Robustness of Applications Based on the SSL and TLS Security Protocols", LNCS 2007, pp.248-264.