

Robustness testing of SIP implementations

Christian Wieser, Ari Takanen
Codenomicon Ltd.
Kaitoväylä 1
FIN-90570 Oulu, Finland
Email: info@codenomicon.com

November 18, 2003

Abstract

Software vulnerabilities are found on a daily basis. In our research, robustness tests proved to be an efficient way to proactively eliminate the most trivial and most exploited vulnerabilities. We introduce our testing methodology using SIP implementations as a case study. We summarize the lessons learned and suggest possible improvements. We conclude by considering the integration and use of either free robustness testing material or commercial test tools in the development of Voice over IP software.

1 Introduction

SIP[4] will be a part of future IP telephony infrastructure, raising new security concerns and issues. As past experience has shown, a majority of these problems are due to software vulnerabilities. One class of software vulnerabilities is introduced during implementation, in the coding process, due to non-robust handling of input data. An example listing of vulnerability types and failures is in Table 1.

One way to counter these vulnerabilities is to do white box testing and to perform code audits. Unfortunately, often the source code is not available or is too complex for effective assessment of robustness and security. Black box testing methods are often needed.

In PROTOS[2] research, we applied a combination of syntax testing, fault injection and penetration testing to simulate attacks against the software. In this black box approach, varying inputs that are known to reveal robustness errors are fed to the implementation and the results are analyzed.

2 Available robustness tests

SIP interoperability testing events[1] have been held since April of 1999. During the events problematic messages were noted and released as an IETF-draft[3].

It defines 42 valid and invalid messages, describes them and gives directions as to how the SIP application should react.

The publicly available PROTOS robustness tests contain 4527 test cases in 54 test groups. Varying SIP header fields in INVITE messages are tested. The PROTOS c07-sip test suite uncovered numerous security problems in a sample of nine implementations that were tested, with only one passing all tests without crashing[5]. Later, numerous other products were tested and vendors announced their own test results[6]. This test suite is freely usable, as it is released under the GNU public license.

The Codenomicon SIP Test Tool is the next generation of the above work, generated using the Codenomicon test generator. Currently it contains over 20000 test cases in over 2000 test groups and tests all 44 SIP header fields and all 15 SDP fields. It contains test a documentation system and a GUI to assist in the test execution and fault elimination.

3 Adaptation in SW development

The adaptation of robustness testing tools in software development varies. Some developers run such tools only once. Others integrate them into their software development process. As robustness tests are designed on the basis of the protocol definition, they rarely expire and they can be considered as extended regression testing. Not only do they seek out problems present earlier in a single piece of software, but they look for the most probable errors in any implementations of the software interface they were designed for.

For the acceptance of a new category of tests in software development, changes are required in processes and standards. Some developers call this the *hardening phase* in development. Others call these tests *torture* or *robustness tests*, or categorize them under regression testing. Clearly, some standard name as well as a well-defined place in the software development process is required in order to promote this adaptation.

Table 1: Implementation vulnerabilities

Type	Failure mode
Buffer overflow	Memory corruption / Arbitrary code execution
Memory allocation	Wait forever / out of memory
Recursive parser	Memory corruption
Integer boundary value	Read from illegal memory space
Signed index or length	Memory corruption / Arbitrary code execution
Format string	Memory corruption / Arbitrary code execution
Missing data elements	Denial of service
Too small data types	Denial of service

4 Conclusions and future work

As with any testing method, robustness testing can not claim that software that passes all tests is free of bugs: it can only show the absence of certain software vulnerabilities. Free tests are one step in promoting a baseline for software security. Hopefully this will alert the software vendors and their customers, and promote proactive means to prevent software vulnerabilities by increasing software quality and test coverage.

References

- [1] SiPit - SIP interoperability test event. <http://www.sipit.net>.
- [2] PROTOS - Security Testing of Protocol Implementations. <http://www.ee.oulu.fi/research/ouspg/protos/>.
- [3] Alan Johnston and Jonathan Rosenberg and Henning Schulzrinne. 2002. Session Initiation Protocol Torture Test Messages. <http://www.ietf.org/internet-drafts/draft-ietf-sipping-torture-tests-00.txt>.
- [4] J. Rosenberg and H. Schulzrinne and G. Camarillo and A. Johnston and J. Peterson and R. Sparks and M. Handley and E. Schooler. 2002. RFC3261 - SIP: Session Initiation Protocol. <http://www.ietf.org/rfc/rfc3261.txt>.
- [5] PROTOS Test-Suite: c07-sip <http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip/>.
- [6] CERT Advisory CA-2003-06 Multiple vulnerabilities in implementations of the Session Initiation Protocol (SIP). <http://www.cert.org/advisories/CA-2003-06.html>.