J Comput Virol (2010) 6:57–64
DOI 10.1007/s11416-009-0123-7

ORIGINAL PAPER

Advanced fuzzing in the VoIP space

Humberto Abdelnur · Radu State · Olivier Festor

Received: 20 January 2008 / Accepted: 17 July 2008 / Published online: 21 June 2009

Abstract Voice over IP is emerging as the key technology in the current and future Internet. This paper shares some essential practical experience gathered over a two years period in searching for vulnerabilities in the VoIP space. We will show a terrifying landscape of the most dangerous vulnerabilities capable to lead to a complete compromise of an internal network. All of the described vulnerabilities have been disclosed responsibly by our group and were discovered using our in-house developed fuzzing software KIF. The paper provides also mitigation techniques for all described vulnerabilities.

1 Introduction

Over the past few years, protocol fuzzing emerged as a key approach for discovering vulnerabilities in software implementations. The conceptual idea behind fuzzing is very simple: generate random and malicious input data and inject it in an application. This approach is different from the well established discipline of software testing where functional verification is checked. In fuzzing, this functional testing is marginal; much more relevant is the goal to rapidly find potential vulnerabilities. Protocol fuzzing is important for two main reasons. Firstly, having an automated approach eases the overall analysis process. Such an process is usually tedious and time consuming, requiring advanced knowledge in software debugging and reverse engineering. Second, there are many cases where no access to the source code/binaries is possible, and where a "black box" type of testing is the only viable solution. Protocol fuzzing can be applied to a broad scope of applications, ranging from device level implementations [9] and up to presentation layer [15].

We describe in this paper the practical experience gained over a two years period with fuzzing in the VoIP space. We performed fuzzing of different devices and SIP stacks in order to validate our research activity on automated and smart fuzzing. All of the described tests, were performed with our developed tool, described in [8]. Our fuzzing approach is based on stateful protocol fuzzing for complex protocols (like for instance SIP). To the best of our knowledge, this is the first SIP fuzzer capable to go beyond the simple generation of random input data. Our method is based on a learning algorithm where real network traces are used to learn and train an attack automata. This automata is evolving during the fuzzing process. Our work in this area is motivated by two major factors: firstly we validate practically the formal research contributions in the area of fuzzing. Secondly, we discover vulnerabilities and follow an responsible disclosure policy by helping vendors to fix them and notifying the affected parties via large distribution mailing lists, web sites and podcast.

We will cover these issues in depth in our paper, which is structured as follows:

Section 2 starts with a short overview on the VoIP infrastructure that has been used for the study described in this paper. The next remaining sections detail the broad scope of the types of vulnerabilities, ranging from simple input validation vulnerabilities and up to cross-layer and multi-technology comprising examples. The final section in this paper concludes the paper and point out future relevant evolutions.

H. Abdelnur (✉) · O. Festor
INRIA Nancy – Grand Est, Vandoevre-Les-Nancy, France
e-mail: abdelnur@loria.fr

O. Festor
e-mail: festor@loria.fr

R. State
University of Luxembourg, Luxembourg, Luxembourg
e-mail: state@uni.lu

2 Fuzzing voice over IP devices

Voice over IP infrastructures are application level specific devices using internet technology as underlying transport layer. End users operate simple end devices (phones) by leveraging different types of servers in order to manage the mobility, localization and user to user call establishment. This call establishment is performed by a signaling protocol, where SIP [14] is becoming the de-facto standard body endorsed protocol. Therefore all VoIP devices do embed SIP stacks which are responsible to process SIP messages and to implement a rather complex state machine. In most cases, the access to the source code of the SIP stacks is impossible and for most VoIP hardphones, running in dedicated equipment, no debugging possibility exists for an independent security researcher. The only resort to perform a security assessment is in this case to perform black-box security testing. We have performed our security and fuzzing experiments over a broad scoped and heterogeneous testbed which is summarized in Table 1

All the experiments were performed with our tool, KiF [8]. KiF consists in two autonomous components, the Syntax Fuzzer and the State Protocol Fuzzer, which jointly provide a stateful data validation entity. The tests may be similar to the normal behavior or can flood the device with malicious input data. Such malicious data can be syntactically non compliant (with respect to the protocol data units), or contain semantic and content wide attack payload (buffer overflows, integer overflows, formatted strings, or heap overflows).

The Syntax Fuzzer takes a fuzzer scenario and the provided ABNF [10] syntax grammar to generate new and crafted messages. The fuzzer scenario drives the generation of the rules in the syntax grammar and may also depend on the State Protocol Fuzzer in order to generate the final message (appropriated or not) to be sent to the target entity.

The State Protocol Fuzzer does passive and active testing. Therefore, two state machines are required: (1) one specifying the SIP state machine and (2) one specifying the testing state machine. The first state machine is used for the passive testing and it controls if there is any abnormal behavior coming from the target entity during the execution of the tests. This state machine may be inferred from the SIP traces of the target entity. The second state machine is used for the active testing and it's driving the profile of the security test. This state machine is defined by the user and can evolve over time. Figure 1 shows the overall framework of KiF.

Table 1 Tested equipment

Device	Firmware
Asterisk	v1.2.16, v1.4.1
	asterisk-addons-v1.2.8
	asterisk-addons-v1.4.4
Cisco 7940/7960	vP0S3-07-4-00
	vP0S3-08-6-00
	vP0S3-08-7-00
Cisco CallManager	v5.1.1
FreePBX	v2.3.00
Grandstream Budge Tone-200	v1.1.1.14
Grandstream GXV-3000	v1.0.1.7
Linksys SPA941	v5.1.5
	v5.1.8
Nokia N95	v12.0.013
OpenSer	v1.2.2
Thomson ST2030	v1.52.1
Trixbox	v2.3.1

3 Weak input validation

The most frequent vulnerability that we encountered is related to weak filtering of input data. This filtering does not properly deal with metacharacters, special characters, over lengthy input data and special formatting characters. Most of these vulnerabilities are due to buffer/heap overflows, or format string vulnerabilities. The most probably cause is that developers assumed a threat model in which VoIP signaling data would be generated only by legitimate SIP stacks. The real danger of this vulnerability comes from the fact that in most cases, one or very few packets can completely take down a VoIP network. This is even more dangerous when realizing that in these cases the SIP traffic is carried over UDP, such that highly effective attacks can be performed stealthy via simple IP spoofing techniques. Table 2 shows some of our published vulnerabilities, where we have decided to highlight two extreme cases: The first vulnerability (disclosed in *CVE-2007-4753*) reveals that even the simplest check for the existence of the input is not performed and that even simple attacks can lead to effective attacks. The second case, (*CVE-2007-1561*) is situated at the opposite site, since a VoIP server is concerned by an attack with a rather complex input structure. The danger in this case is that one single packet will take down the core VoIP server and thus lead to a complete take down of the whole VoIP network.

Preventing these types of attacks at a network defense level is possible with deep packet inspection techniques and proper domain and application specific packet filtering devices.

3.1 Attacks against the internal network

Most VoIP devices have embedded web servers that are typically used to configure them, or to allow the user to see the missed calls, and all the call log history. The important issue is that the user will check the missed calls and other device related information from her machine, which is usually on the

Fig. 1 KiF framework

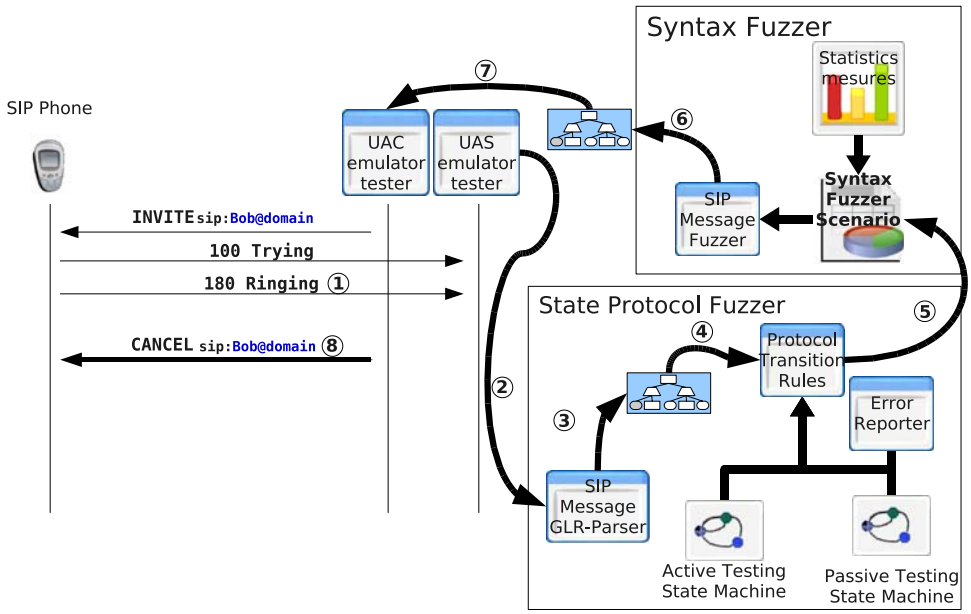


Table 2 Input validation vulnerabilities

Device	Synopsis	CVE-Identifier	Impact
Asterisk v1.4.1	Invalid IP address in the SDP body	CVE-2007-1561	DoS
Cisco 7940/7960	Invalid Remote-Party-ID header	CVE-2007-1542	DoS
vP0S3-07-4-00			
Grandstream Budge	Invalid WWW-Authenticate header	CVE-2007-1590	DoS
Tone-200 v1.1.1.14			
Linksys SPA941 v5.1.5	Invalid handling of the \377 character	CVE-2007-2270	DoS/String overflow
Thomson ST2030 v1.52.1	Invalid SIP version in the Via header	CVE-2007-4553	DoS
	Invalid URI in the To header	CVE-2007-4753	
	Empty packet		
Linksys SPA-941 v5.1.8	Unescaped user info	CVE-2007-5411	XSS attacks
Asterisk v1.4.3	Unescaped URI in the To header	CVE-2007-54881	SQL injection and Toll-fraud
FreePBX v2.3.00	Unescaped URI in the To header	[7]	XSS attacks
Trixbox v2.3.1			

internal network. If the information presented is not properly filtered, this same user will expose her machine (located on the internal network) to malicious and highly effective malware. We will illustrate the following example discovered during a fuzzing process (see *CVE-2007-5411*). The VoIP Phone Linksys SPA-941 (Version 5.1.8) has an integrated web server that allows for configuration and call history checking. A Cross Site Scripting vulnerability (XSS) [11] allows a malicious entity to perform XSS injection because the ”FROM” field coming from the SIP message is not properly filtered. By sending a crafted SIP packet with the FROM field set to :

```
 "<script x='"
<sip:'src='beef.js'>@domain>;tag=1"
```

the browser is redirected to include a javascript file (y.js) from an external machine (baloo) as shown in Fig. 2. This external machine is under the control of an attacker and the injected javascript [11] allows a remote attacker to use the victim's machine in order to scan the internal network, perform XSRF (Cross Site Request Forgery) attacks, as well as obtain highly sensitive information (call record history, configuration of the internal network), deactivate firewalls or even redirect the browser towards malware infested web pages (like for instance MPACK [2]) to compromise the victim's machine. The major and structural vulnerability comes in this case, by the venture of two technologies (SIP and WEB) without addressing the security of the cross-technological information flow.

Fig. 2 Linksys SPA-941 XSS attack

The impact of this vulnerability is very high : most firewalls/IPS will not protect the internal network against XSS attacks delivered over SIP. Additionally, users will connect to these devices directly from the internal network and therefore the internal network can be compromised. Jeremiah Grossmann [11] showed how firewalls can be deactivated with XSS attacks and many other malicious usages do exist. Unfortunately, most VoIP devices have weak embedded WEB applications, such that other vulnerable systems exist and are probably exploited in the wild.

4 Protocol tracking vulnerabilities

Protocol tracking vulnerabilities go beyond simple input filtering of single messages. In this type of vulnerability, several messages will lead a targeted device in an inconsistent state, albeit each message on its own does not violate the SIP RFC [14]. These vulnerabilities are caused by weak implementations of protocol state engines. Exploiting this vulnerability can be done in three main ways:

1. the device might receive inputs that are not expected in its current protocol state: for instance, when waiting for a BYE method, an INVITE is received
2. the input might consist in simultaneous messages addressed to different protocol states
3. slight variations in SIP dialog/transaction tracking fields leading a device towards an inconsistent state

The discovery of such vulnerabilities is truly difficult. The fuzzing process should be able to identify where a targeted device is not properly tracking the signaling messages and which fields can be fuzzed in order to detect it. The search space in this case is huge being spread over many messages and numerous protocol fields, requiring thus advanced and machine learning driven fuzzing approaches. Table 3 shows such disclosed vulnerabilities having different complexity grades.

A simple case (*CVE-2007-6371*), where a CANCEL message arrives earlier than expected, can turn the device into an inconsistent state which will end up in a Denial of Service state, as showed in Fig. 3.

The major danger with this type of attacks is that no application level firewall can completely track so many flows in real time and that even in the case of known signatures, polymorphic versions of known attacks can be obtained easily and these will remain under the security radar. As of today, unfortunately no effective solution to prevent this type of attacks exists.

4.1 Toll fraud vulnerabilities

Toll frauds occur when the true source of a call is not charged. This can happen by the usage of a compromised VoIP infrastructure or by manipulating the signaling traffic. It is rather amazing to see that although technology evolved, the basic conceptual trick of the 70's, where phreakers reproduced the 2600 Hz signal used by the carriers is still working. Thirty years after, the signaling plane can be still tampered with and manipulated by a malicious user. What did change however, is the needed technology. Nowadays, we can inject SQL commands (Chapter VI [13]) in the signaling plane, and the toll fraud is possible. In the following, we will describe in detail one vulnerability found during a fuzzing process [7]. Some SIP proxies store information gathered from SIP headers into databases. This is necessary for billing and accounting purposes. If this information is not properly filtered, once it will be displayed to the administrator it can perform a second order SQL injection, that is during the display, the data is interpreted as SQL code by the application. In this case, two consequences can result: First, the database can be changed—for instance the call length can be changed to a small value—and thus the caller can do toll fraud. If we consider Asterisk [3], the popular and largely deployed open source VoIP PBX, Call Detail Records (CDR) are stored in a MySQL database.

FreePBX [1] and Trixbox [5] use the information stored in such database in order to manage, compute generate billing reports or display the load of the PBX.

Some functions do not properly escape all the input characters from fields in the signaling packets.

A first flavor of this specific attack can be performed by an subscribed user of the domain and the attack consists of injecting negative numbers in the CDR table in order to change the recorded length/other parameters of a given call. The direct consequence is that no accurate accounting is performed and the charging process is completely controlled by an attacker.

Table 3 Stateful vulnerabilities

Device	Synopsis	CVE-Identifier	Impact
Cisco 7940/7960 vP0S3-08-6-00	Does not handle unexpected messages (e.g. OPTIONS) inside an existing INVITE transaction	CVE-2007-4459	DoS
Grandstream GXV-3000 v1.0.1.7	Unexpected message inside an INVITE transaction allows to remotely accept the call	CVE-2007-4498	Remote Eavesdropping
CallManager v5.1.1	Authentication uses not one-time nonces	CVE-2007-5468	Replay Attacks
OpenSer v1.2.2		CVE-2007-5469	Â
SIP Protocol Relay Attack	Attacker can trigger the target entity to authenticate to him	[6]	Toll-Fraud
Cisco 7940/7960 vP0S3-08-7-00	Does not handle six INVITE transaction destinated to any user	CVE-2007-5583	DoS
Nokia N95 v12.0.013	Does not handle a CANCEL at an unexpected timing in an INVITE transaction	CVE-2007-6371	DoS

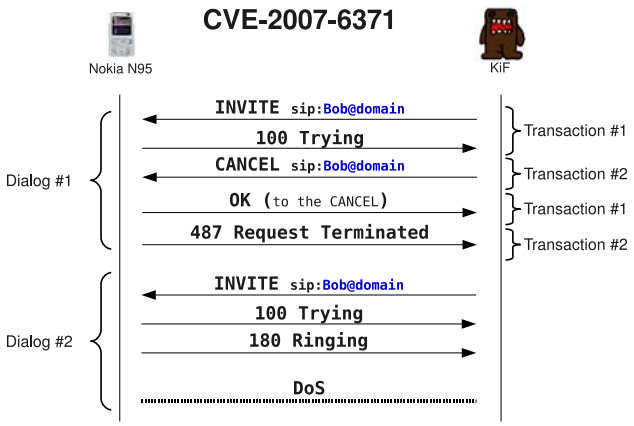


Fig. 3 Nokia N95 DoS attack

A second and more serious consequence is that this attack can be escalated by injecting JavaScript [11] tags to be executed by the administrator PC when she will perform simple management operations. In this case, a Cross-Site Scripting Attack (XSS) [11] is resulted, because malicious JavaScript can be stored into the database by the SQL injection. This malware gets executed on the browser when the administrator will check it—this is a very similar process to the log injection attacks known by the Web application security community. Similarly to the previous case, tools like Beef and XSS proxy can scan the internal network, deactivate firewalls and realize all the CSRF/XSRF specific attacks.

The main issue is that most current applications that deal with CDR data are not considering this type of threat. If the target system is not well secured, SQL injection can lead to system compromise because most database servers allow some interaction with the target OS [13].

This type of vulnerability is rather dangerous because few application (none of which we have tested) implement filtering on SIP headers. All applications do consider SIP related information to be sourced from a trusted origin and no security screening is performed. The mitigation should be proper input and output filtering whenever data is stored/read from another software component.

4.2 Remote eavesdropping vulnerabilities

A rather unexpected vulnerability was discovered by us in *CVE-2007-4498*. Several SIP messages sent to the affected device put the phone off-hook without ringing or making any other visual notification. The attacker is thus capable to remotely eavesdrop all the conversations performed at the remote location. Figure 4 shows the messages exchanged by the attack. The impact if this vulnerability goes beyond the simple eavesdropping of VoIP calls, because an entire room/location can be remotely monitored. This risk is major and should be considered when deploying any VoIP equipment. Although in the presented case, a software error was probably the cause, such backdoors left by a malicious entity/device manufacturer represent very serious and dangerous threats.

4.3 Weak cryptographic implementations

The authentication mechanism in SIP is a standard shared secret and challenge-response based one [12]. Nonces are generated by the server and submitted to an authenticating entity. The latter must use its shared key to compute a hash which is afterwards sent to the authenticator. This hash is computed on several values: SIP headers, nonces and random values. A received hash is validated by the server and checked to authenticate a client. For efficiency reasons, very few server implementations track the life-cycle of a valid

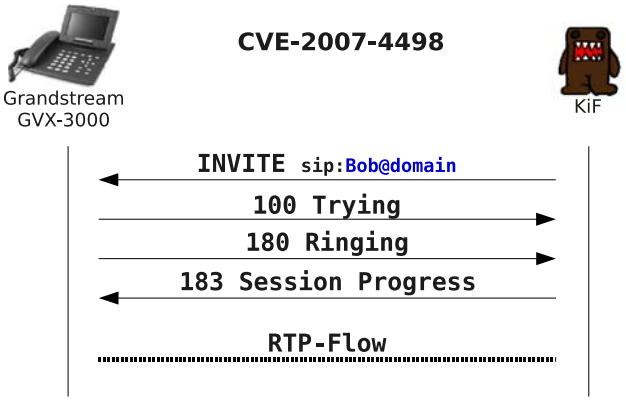


Fig. 4 Grandstream GXV-3000 remote eavesdrop

token. We have found at least two vulnerabilities *CVE-2007-5468 and CVE-2007-5469*, where intercepted tokens could be replayed. These vulnerabilities are not simple man in the middle attacks, since intercepted tokens were reusable for long time periods and they could be used for the authentication of any other call. Figure 5 shows the flow of messages for such attack. The impact of such a vulnerability is very high. Toll frauds and spoofing call identifiers are the straightforward consequences. The mitigation consists in trading off performance versus security and implementing efficient and secure cryptographic token management procedures.

5 Specification level vulnerabilities

Our main work consisted in searching for vulnerabilities in specific SIP implementations without considering the security of the SIP protocol per se. We were however surprised to discover during a complex fuzzing scenario the same anomaly (and apparent vulnerability) shared by all devices under test (Table 1). Under a more careful analysis, we did realize that in fact the SIP protocol itself has a major design vulnerability that makes toll fraud possible on any VoIP network [6]. The major issue is that a classical relay attack is possible by forcing a called party to issue a re-Invite operation. Due to the novelty and severity of it, we will detail the attack in the following:

An attacker issues a call to the victim, the victim answers it and later on, put the attacker on hold. To address this put on hold, an accomplice of the attacker may initiate another call. Once the attacker receives the re-INVITE specifying the "On hold", he/she will request the victim to authenticate. This last authentication may be use by the attacker to impersonate the victim at its own proxy.

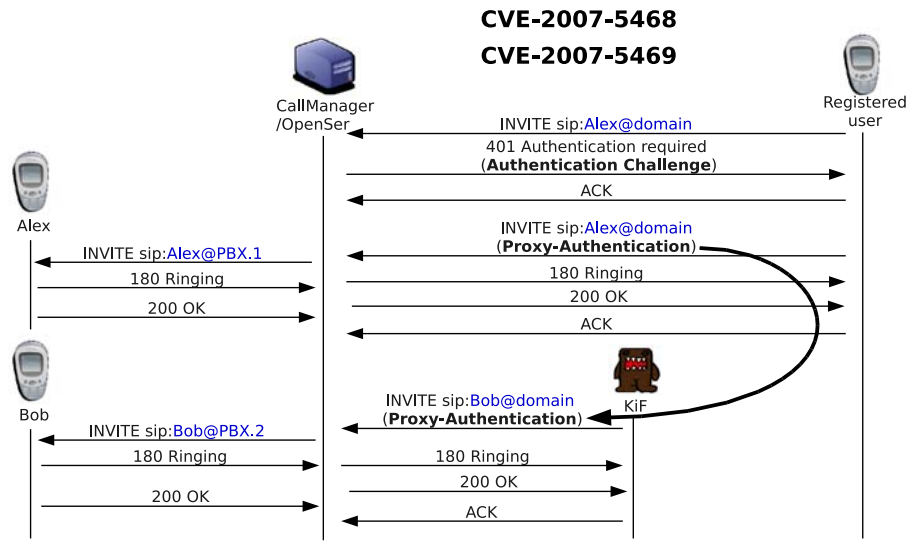
Notations:

- P is the proxy located at URL: proxy.org
- X is the attacker located at URL: attacker.lan.org
- V is the victim located at URL: victim.lan.org
- V is also registered with P under the username victim at proxy.org
- Y is the accomplice of X (it can be in fact X), but we use another notation for clarity sake

The described attack will show how X calls a toll fraud number 1-900-XXXX impersonating V.

1. X calls' directly V.
   "The route set MUST be set to the list of URIs in the Record-Route header field from the request...The remote target MUST be set to the URI from the Contact header field of the request." RFC 3261[14] Section 12.1.1 UAS calls

Fig. 5 Replay attack



```
X ----- INVITE victim.lan.org ----> V
    From : attacker at attacker.lan.org
    To: victim at victim.lan.org
    Contact: 1900-XXXX at proxy.org
    Record-Route: attacker.lan.org
```

2. The normal SIP processing

```
X <--------- 180 Ringing ---------- V
X <------------ 200 OK ------------ V
X <---------- Media Data ---------> V
```

3. The accomplice Y steps in and invites victim V, and then the victim decides to put X on hold
4. The victim, V, sends a re-INVITE to X (to put it on hold) "The UAC uses the remote target and route set to build the Request-URI and Route header field of the request." RFC 3261 [14] 12.2.1.1 Generating the Request (Requests within a Dialog)

```
X <-- INVITE 190XXXX at proxy.org -- V
    From: victim at victim.lan.org
    To : attacker at attacker.lan.org
```

5. X calls 1900-XXXX using the proxy P and the proxies asks X to authenticate using a Digest Access Authentication with nonce="Proxy-Nonce-T1" and realm ="proxy.org"
6. X request the victim to authenticate the re-INVITE from step 4 using the same Digest Access Authentication received in step 5

```
X ----- 401/407 Authenticate -----> V
   Digest: realm ="proxy.org",
       nonce="Proxy-Nonce-T1"
```

7. In this step the victim will do the work for X (Relay Attack)

```
X <-- INVITE 190XXXX at proxy.org -- V
   Digest: realm ="proxy.org",
       nonce="Proxy-Nonce-T1"
       username= "victim",
       uri="1900XXXX at proxy.org",
       response="the victim response"
```

8. X may reply now to the Proxy with the valid Digest Access Authentication computed by the victim. Note that the Digest itself it is a perfectly valid one.

6 Conclusions and future works

The quantitative conclusions after a long term work on searching vulnerabilities in the VoIP space are rather pessimistic. Feedback and support when contacting vendors remains highly unpredictable and poor. All tested devices have been found vulnerable. The scope of the detected vulnerabilities is very large. Trivial input validation vulnerabilities affecting highly sensitive communication materials are rather usual. More complex and protocol tracking related ones do also exist, though their discovery and exploitation is rather complex. The cause of these vulnerabilities is the weak software security life-cycle of their vendors. The integration of Web and VoIP technology is a Pandora's box comprising even more powerful and hidden dangers. Web specific attacks can be carried out over the SIP plane leading to potential devastating effects, like for instance the complete compromise of an internal network. This is possible since no application specific firewall today can easily interwork with several technologies and no proper guidelines for the secure interworking of Web and VoIP exist. The more structural cause is a missing VoIP specific threat model. The VOIPSA did develop a threat model [4] which however does not reflect the current state. Highly efficient Denial of Service attacks can be done with single-shot packets, remote eavesdropping goes beyond the simple call interception and the VoIP plane itself can be a major security threat to the overall IT infrastructure. Much remains to be done in the future, among which "Security Build in VoIP devices" remains the major among them. Changes in the software development cycles must be followed by an comprehensive security assessment and testing. Protocol fuzzing is one essential building block in this landscape, since no other additional approach can be used by independent security research. We have described in this paper our practical and hand-on experience in testing embedded SIP stack implementation. These tests were performed in order to validate our research on advanced security fuzzing techniques and the discovered vulnerabilities were properly and responsibly disclosed. Our future work will extend it by addressing additional protocols, case studies, implementations and formal approaches.

References

1. FreePBX: full-featured PBX web application. http://freepbx.org
2. MPack: Insight into MPACK Hacker kit. http://www.malwarehelp.org/news/article-6268.html/
3. The Asterisk PBX. http://www.asterisk.org/
4. The Voice over IP Security Alliance (VOIPSA). http://www.voipsa.org/Activities/taxonomy.php
5. trixbox: Asterisk-based IP-PBX. http://www.trixbox.com/
6. Abdelnur, H., State, R., Festor, O.: Security Advisory: "SIP Digest Access Authentication RELAY-ATTACK for Toll-Fraud". http://voipsa.org/pipermail/voipsec_voipsa.org/2007-November/002475.html
7. Abdelnur, H., State, R., Festor, O.: Security Advisory: "SQL injection in asterisk-addons and XSS injection in WWW application in Areski, FreePBX and Trixbox". http://voipsa.org/pipermail/voipsec_voipsa.org/2007-October/002466.html

8. Abdelnur, H., State, R., Festor, O.: KiF: a stateful SIP fuzzer. In: Proceedings of Principles, Systems and Applications of IP Telecommunications, IPTComm, pp. 47–56, New-York, NY, USA, July 2007. ACM Press, New York (2007)
9. Butti, L., Tinnes, J.: Discovering and exploiting 802.11 wireless vulnerabilities. J. Comput. Virol. **4**(1), 25–37 (2008)
10. Crocker, D.: Augmented BNF for Syntax Specifications: ABNF. Standards Track, November 1997
11. Fogie, S., Grossman, J., Hansen, R., Rager, A., Petkov, P.D.: XSS Exploits: Cross Site Scripting Attacks and Defense. Syngress (2007)
12. Johnston, A.B., Piscitello, D.M.: Understanding Voice over Ip Security. Artech (2006)
13. Litchfield, D., Anley, C., Heasman, J., Grindlay, B.: The Database Hacker's Handbook: Defending Database Servers. Wiley, New York (2005)
14. Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., Schooler, E.: SIP: Session Initiation Protocol. http://www.ietf.org/rfc/rfc3261.txt, June 2002
15. Sutton, M., Greene, A., Amini, P.: Fuzzing: Brute Force Vulnerability Discovery. Addison-Wesley Professional, Reading (2007)