

SECURITY SOLUTIONS

www.securitylab.net 304 Newbury St, Suite 300 Boston, MA 02115 (617) 275-4798

BLACKHAT BRIEFINGS 2005

The Art of SIP fuzzing and Vulnerabilities Found in VoIP

By Ejovi Nuwere & Mikko Varpiola

About the authors



Ejovi Nuwere

Ejovi Nuwere is the founder of SecurityLab Technologies. Nuwere gained media attention and international recognition for his highly publicized security audit of Japan's National ID system--JukiNet. Nuwere is the Chief Technology Officer of SecurityLab Technologies where he heads the companies VoIP security auditing group. He currently lives in Boston and is working on his second Practical Penetration book. Testing (O'Reilly).

Mikko Varpiola

Mikko Varpiola is the head of test tool development at Codenomicon Ltd. His specific area of expertise is in anomaly design - e.g. what to feed into software to make it fail. Before Codenomicon he worked as a researcher in the acclaimed PROTOS project at Oulu University Secure Programming Group (OUSPG). He is the author of the ASN.1 encoding anomalies first deployed in the widelypublicized PROTOS LDAP and SNMP test suites.



1. Proof of concept

2. Current state of the VoIP security

3. The art of SIP fuzzing

Proof of concept test setup



- Two user agents
- One infrastructure component
- Demonstrate the loss of availability
- Potential security implications of found bugs still under investigation
- Vendors have been notified



Current state...



Open-source vs. proprietary

- Large product companies are doing fairly well
- Some telcos and hardware vendors lacking?
- How to measure the differences between products?

Military and private usage

- Multilevel Precedence and Preemption (MLPP)
- Small businesses at risk
- Off shoring by large corporations

Current state continued...



• Progress since 2000?

- Lack of basic coding flaws (sorry no easy overflows)
- PROTOS (2001), discovered most basic bugs
- Some companies begin to have very mature threat modeling

Back to 1999...

- Some VoIP vendors have no concept of vulnerabilities (usual call the lawyers, downplay,...)
- Make it work mentality
- Closed network assumptions

Testing Approach



Defining fuzzing terminology

Products evaluated

- sipXphone (sip foundry stack)
- PartySIP (GNU SIPo stack)
- SIPset (vovida/vocal stack)
- linPhone (GNU SIPo stack)
- Commercial Brand X (unknown stack, proprietary?)

Doesn't look too promising!

- At least two critical bugs per product

The problems with SIP



Share and share alike

- Many are using the same flawed code base
- No one admits they are using the same code base

No update mechanisms for most products

- Hardware devices
- Consumer products

Writing parsers are inherently complex

- Ethereal (150+ vulnerabilities since 1999)

Don't forget the environment



- It is essential to understand the enviroment
- Some errors trigger in only certain environments and certain configurations
- In context of SIP just think of UDP vs.
 TCP as a transport
 - Stream vs. Datagram
 - alternate physical limitations for maximum message size
- Beyond the parser lies the application



The Art of SIP fuzzing



- What's all the fuzz about?
- Deciding what to fuzz
- Isolated bug fault model
- A systematic approach
- What ASCII (as in SIP) brings to the table
- Types of anomalies



What's all the fuzz about





- Bug symptoms usually located:
- Crashes
- Performance degradation
- Other unexpected behaviour

Deciding what to fuzz



- Decision need to relate to available protocols and surrounding environment
- Ideally test all open interfaces
- Environment
 - What are the open interfaces
 - History of identified protocols
 - Risk analysis

Protocol

- Only test the actual behaviour
- Check common sources for known vulnerabilities -> Improvise

WITH SIP TRY THESE:

- SIP REQUEST LINE
- SIP URIs in headers below
- Authorization headers
- Contact header
- CSeq
- From header
- Route header
- Record-Route header
- To header
- Via Header

Isolated bug fault model



YOU CAN'T TEST EVERYTHING AT SAME TIME – NEITHER YOU CAN DO EVERYTHING IN SAME MESSAGE/ELEMENT!



www.securitylab.net | 617-248-6440 | info@securitylab.net

A systematic approach



- 1. Identify sub structures (required and optional)
- 2. Identify data types of identified fields
- Anomalise fields one at the time with proper anomalies for data type
- 4. Or apply structural mutations



Fuzzing SIP



ASCII (as in SIP) allows various levels of freedom

- Human readable protocols tend to be harder to parse
- Binary vs. ASCII protocols
- It is easier to create huge amount of (redundant?) test cases with ASCII based protocols
- SDP and other content payloads a task of their own (may requre special injection arrangements)

Anomalies for ASCII based protocols



For each anomaly we present

- Examples up close and personal
- Them applied to SIP message

We cover:

- standard overflows (ascii, c-format strings, control/non-ascii,utf8)
- standard integers (negative, 'float', big)
- addresses (IPv4, IPv6, ISDN (tel uris))
- structural (repetitions (header, header element), underflows)
- protocol specifics (by closely observing the SIP & related specs)
- Why different lengths / values for each data type?
 - All rules of boundary value testing apply to fuzzing as well
 - Different software, different limits
 - Different routines likely get excercised with different strings

Standard overflows



- ASCII (alpha vs. Alphanumeric)
- C-format string
- Control character
- UTF8

16x 0x61 ; ('aaaaaaaaaaaaaaaaaaa') 1024x 0x62 2048x 0x34 '%s%s%s%s', '%n%a', '%99d', %.9999f' 128x 0x00, 512x 0x07, 1024x 0x7f, ...

Standard integer anomalies



- Cover the data range with presentative values
- Examine specification for enumerations

Contact: "user" <sip:us

Content-Type: applicati

Content-Length: -1

STANDARD: -1, 0, 1, 2, 4, 5, 6, 7, 15, 16, 32, 63, 64, 127, 128, 255,256,1023, 1024, 4095, 4096, ... FLOATS: 0.1, 0.9, -0.1, 0.0, -0.0, UNEXPECTED NUMERIC SYSTEMS: 000b, 0x01, 042364

Addresses



0.0.0/8	"This" Network	[RFC1700, page 4]	TWITE cinemana[2:1] SID/2 0
10.0.0/8	Private-Use Networks	[RFC1918]	INVIIE SID: USERG[?::1] SIP/2.0
14.0.0.0/8	Public-Data Networks	[RFC1700, page 181]	To: <sip:user@to.example.com></sip:user@to.example.com>
24.0.0.0/8	Cable Television Networks		From: "user" <sip:user@from.example< td=""></sip:user@from.example<>
39.0.0.0/8	Reserved but subject to allo	ation [RFC1797]	Via: SIP/2.0/UDP from.example.com:5
127.0.0.0/8	Loopback	[RFC1700, page 5]	Call-TD: s0c00000545i0t111090429066
128.0.0.0/16	Reserved but subject to allo	ation	Cantast: "wasy" (sin wasy from own
169.254.0.0/16	Link Local		Contact: "user" <sip:user@irom.exam< td=""></sip:user@irom.exam<>
172.16.0.0/12	Private-Use Networks	[RFC1918]	Content-Length: 177
191.255.0.0/16	Reserved but subject to allo	ation	Content-Type: application/sdp
192.0.0.0/24	Reserved but subject to allo	ation	
192.0.2.0/24	Test-Net		
192.88.99.0/24	6to4 Relay Anycast	[RFC3068]	"[ffff:ffff:ffff:ffff:ffff:ffff:ffff:ff
192.168.0.0/16	Private-Use Networks	[RFC1918]	"[0::0]"
198.18.0.0/15	Network Interconnect Device H	Benchmark Testing	"[2···1]"
223.255.255.0/24	Reserved but subject to allocation		
224.0.0.0/8	Various multicast		
240.0.0/4	Reserved for Future Use	[RFC1700, page 4]	
255.255.255.255	Broadcast		
	"FF01.0.0.0.0.0.0.1"	# All Nodes Add	ress [RFC2373]

"-1111" "10.10.101"" "%s.%x.%n.%d"	"FF01:0:0:0:0:0:0:1" #	All Nodes Address	[RFC2373]
	"FF01:0:0:0:0:0:0:2" #	All Routers Address	[RFC2373]
	"FF02:0:0:0:0:0:0:1" #	All Nodes Address	[RFC2373]
	"FF02:0:0:0:0:0:0:2" #	All Routers Address	[RFC2373]
	"FF02:0:0:0:0:0:0:3" #	Unassigned	[JBP]
	"FF02:0:0:0:0:0:0:4" #	DVMRP Routers	[RFC1075,JBP]
	"FF02:0:0:0:0:0:0:5" #	OSPFIGP	[RFC2328,Moy]
	"FF02:0:0:0:0:0:0:6" #	OSPFIGP Designated Routers	[RFC2328,Moy]

Structural anomalies



Repetitions

- Header
- Sub elements

Underflows

Unexpected data

Content-Type: application/sdp CSeq: 7038 INVITE a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 a15 a16 a Max-Forwards: 70

INVITE sip:user@to.example.com SIP/2.0\r
To: <sip:user@to.example.com>\r
From: "user" <sip:user@from.example.com:5060>;tag=00031912\r
Via: SIP/2.0/UDP from.example.com:5060;branch=z9hG4bK31912t1110908159733\r
Call-ID: s0c00031912i0t1110908159733@from.example.com\r
Contact: "user" <sip:user@from.example.com;transport=udp>\r
Cont

INVITE sip:user@to.example.com SIP/2.0
To: <sip:user@to.example.com>
From: "user" <sip:user@from.example.com:5060>;tag=00001889
Via: SIP/2.0/UDP from.example.com:5060;branch=z9hG4bK1889t1110904451140
Accept: application/sdp, text/html

Protocol specific anomalies



- SIP Tokens as in RFC3261
- SIP line continuations as in RFC3261
- URI escapes as in RFC2616/RFC1945
- Embedded BASE64 encoding of RFC2617 headers
- UTF8 (see ttp://www.cl.cam.ac.uk/~mgk25/unicode.html)
- Other SIP specific escapings
- MIME multipart bodies
- You name it!

Protocol specifics continued



INVITE http:user@to.example.com SIP/2.0 To: <sip:user@to.example.com> From: "user" <sip:user@from.example.com:5060>;tag=00000154 Via: SIP/2.0/UDP from.example.com:5060;branch=z9hG4bK154t1110904245845 Call-ID: s0c00000154i0t1110904245845@from.example.com Contact: "user" <sip:user@from.example.com;transport=udp> Content-Length: 177 Content-Type: application/sdp INVITE; sip:user@to.example.com SIP/2.0 CSeq: 155 INVITE To: <sip:user@to.example.com> Max-Forwards: 70 From: "user" <sip:user@from.example.com To: <sip:user@to.example.com> Via: SIP/2.0/UDP from.example.com:5060; From: "Displayname" <sip:%25%32%35%25%33%36%25%33%31@to.e: Call-ID: s0c0000065i0t1110904235150@fr Via: SIP/2.0/UDP from.example.com:5060;branch=z9hG4bK8708 Contact: "user" <sip:user@from.example. Call-ID: s0c00008708i0t1110905304013@from.example.com Contact: "user" <sip:user@from.example.com;transport=udp> Content-Length: 177 Content-Length: 177 Content-Type: application/sdp

Is that all about anomalies?



"Thrill to the excitement of the chase! Stalk bugs with care, methodology, and reason. Build traps for them..... [Beizer]"

"Testers! Break that software (as you must) and drive it to the ultimate - but don't enjoy the programmer's pain. [Beizer]"

"The tester in you must be suspicious, uncompromising, hostile, and compulsively obsessed with destroying, utterly destroying, the programmer's software. The tester in you is your Mister Hyde ... [Beizer]"

Conclusions



- VoIP is going prime time lets fix it before its too late!!!
- Find out what stacks your vendors are using and how they are testign them!
- Its not only the signaling there is voice and management among others to be worried about as well
- Beyond presented fundamental problems there are other cans of worms to be opened:
 - Tapping, session hijacking, etc....

Questions?



Ejovi Nuwere http://www.securitylab.net/research/ ejovi[AT]securitylab.net

www.securitylab.net | 617-248-6440 | info@securitylab.net