INTERSTATE: A Stateful Protocol Fuzzer for SIP

Thoulfekar Alrahem, Alex Chen, Nick DiGiussepe, Jefferey Gee, Shang-Pin Hsiao, Sean Mattox, Taejoon Park, Albert Tam, Ian G. Harris

Department of Computer Science
University of California Irvine
Irvine, CA 92697 USA
harris@ics.uci.edu

Marcel Carlsson
Fortconsult
Tranevej 16-18
2400 Copenhagen NV Denmark
mc@fortconsult.net
Fuzzing Basics

• Transmit a sequence of messages to a server, attempting to “break” it
• Apply “fuzzing functions” to message fields to reveal vulnerabilities

Typical Fuzzing Functions
➢ **Buffer Overflow** - Make a field very long to force buffer overflow
➢ **Command Injection** - Insert shell metacharacters to see if string is passed to a shell
➢ **SQL Injection** - Insert SQL reserved word to see if string is used to build an SQL query

INVITE sip:marXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXconi@radio.org SIP/2.0
Via: SIP/2.0/UDP lab.test.org:5060;branch=ziuh2w
Max-Forwards: 70
To: G. Marconi <sip:Marconi@radio.org>
From: Nikola Tesla <sip:n.tesla@test.org>;tag=98767
Call-ID: 123456789@lab.test.org
Cseq: 1 INVITE
Session Initiation Protocol (SIP)

- Used to start, end, and modify communication sessions between VOIP phones
- SIP does not transfer media (audio/video)

User Agent Client (UAC)
- Initiates call
- Sends Request Messages

User Agent Server (UAS)
- Receives call requests
- Send Response Messages

- We do not consider other SIP entities, proxies, registrar servers, etc.
- We are fuzzing the UAS, fuzzer is a client
Previous Work, SIP Fuzzers

• SNOOZE Fuzzer


  • Protocol state machine is used, XML-based description

  • Fuzzing scenario defines message sequence, what to fields to fuzz, what fuzzing primitives to use

  • Fuzzing scenarios must be developed manually
Previous Work, SIP Fuzzers

• PROTOS Suite
  • Free version: http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip
  • Industrial version: http://www.codenomicon.com
  • Predefined test suite, 4527 test cases
  • Fuzz the INVITE message, teardown with CANCEL/ACK messages
  • Detected vulnerabilities in several SIP implementations (8 of 9)
INTERSTATE Fuzzer, Contributions

1. Automatic exploration of server state machine
   - Input sequences are generated (messages, etc.) to perform a random walk of the state machine

2. Evaluation of response messages
   - Responses received from server are checked for correctness
   - Allows the detection of more subtle failures
   - Needed to accurately maintain current state of UAS

3. Control server GUI during fuzzing
   - GUI control needed to fully explore state space (ie. accepting a phone call)
INTERSTATE Fuzzer System

**Protocol Description** - State machine describing the protocol

**Test Sequence Generator** - Selects paths in the state machine and generates inputs (messages, timeouts, GUI) to explore the paths

**Response Analyzer** - Verifies correctness of response messages. Supports synchronization between fuzzer and UAS
Protocol Description

- State machine describes the behavior of the UAS
- We only consider the following requests: INVITE, CANCEL, BYE, ACK
- Edges are not included in this picture for clarity
- Edges are labeled with Inputs and Outputs
- Inputs are Messages (R:), Timeouts (T:), or GUI (U:)
Message Inputs

- Fuzzer generates the message required to traverse selected edge
- Dialog state is generated for INVITE and used for all other messages in dialog

• Messages are fuzzed with a given probability
• The following fuzzing functions are used:
  ➢ **Repeat String** - Increase string length by repeating it to force buffer overflow
  ➢ **Command Injection** - Insert shell metacharacters to see if string is passed to a shell
Timer and GUI Inputs

- Some UAS state transitions depend on timeouts
- Some UAS state transitions depend on local user inputs
  - Accepting and declining a call
- Control of UAS GUI is needed to fully explore state machine
- X11::GUITest Toolkit
  - http://sourceforge.net/projects/x11guitest
Fuzz Generation Algorithm

\( \text{curr\_state} = \text{`start'}; \)
while () {
    \( e = \text{select\_outgoing\_edge(\text{curr\_state});} \)
    \( \text{generate\_trigger}(e); \)
    \( r = \text{get\_response\_message();} \)
    if (!\text{correct\_response}(r)) then
        exit(error detected);
    else
        \( \text{curr\_state} = e.\text{successor\_state}; \)
} 

1. Select outgoing edge of UAS state machine
2. Generate input to trigger edge
3. Repeat until error is detected
• Used INTERSTATE to fuzz KPHONE, an open source SIP phone
• Revealed a timing vulnerability which causes a crash
• After a phone call is accepted, KPHONE loads necessary codecs
• Crash occurs if a BYE message is received during that time (<1sec)
Fuzzer Result Information

- Vulnerability detected in 6 seconds wall clock time
- 1.3 GHz AMD Athalon, 512 MB RAM, Debian Linux
- 8 state machine edges traversed before vulnerability detected

Iteration 1:
  - Edge 1: start -> invite
  - Edge 2: invite -> ring
  - Edge 3: ring -> start

Iteration 2:
  - Edge 4: start -> invite
  - Edge 5: invite -> start

Iteration 3:
  - Edge 6: start -> invite
  - Edge 7: invite -> ring
  - Edge 8: ring -> OK (crash)
Conclusions

• Fuzzer automatically explores UAS state machine
• Verifies response messages for correctness
• Controls UAS GUI to enable full state space exploration

Future Work

• Test more open source soft phones
• Debug the phones to identify the source of the vulnerabilities
• Examine hard phones, circumvent keypad interface

http://testlab.ics.uci.edu/interstate

Get the source code!