SIPping your network

Humberto J. Abdelnur Radu State Olivier Festor
{Humberto.Abdelnur, Radu.State, Olivier.Festor}@loria.fr

MADYNES
INRIA Nancy-Grand Est
http://madynes.loria.fr

February 16, 2008

Outline

About us

Who we are?

Humberto J. Abdelnur

- Ph.D student supervised by Radu and Olivier
- Fuzzing and Fingerprinting
- `http://www.loria.fr/~abdelnur`

Radu State

- Ph.D senior researcher
- Network and Service Management and VoIP Security Monitoring and Assessment

Olivier Festor

- Ph.D research director
- Distributed network, security and service management
- `http://www.loria.fr/~festor/`

Where we work?

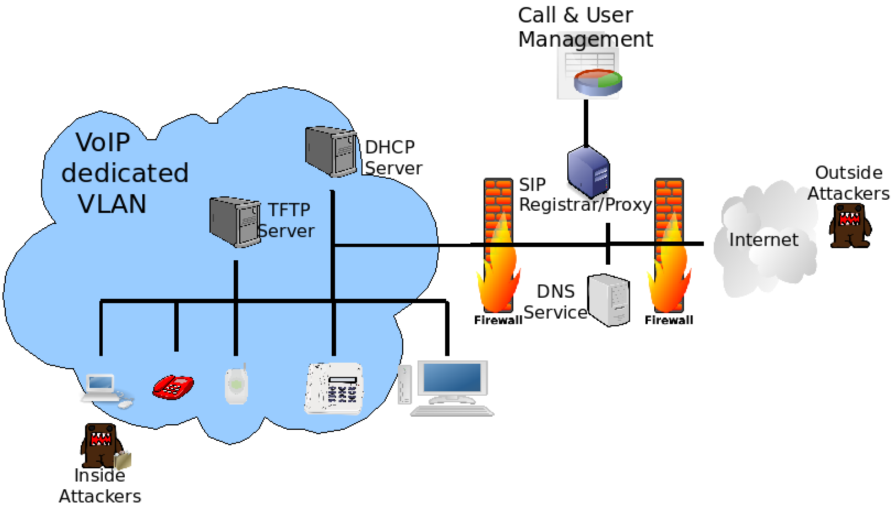
Madynes team

- `http://madynes.loria.fr`

INRIA Nancy-Grand Est, FRANCE

- `http://www.loria.fr`
- `http://www.inria.fr`

VoIP Deployment Layout



VoIPSA[1] VoIP Threat Taxonomy

- Social threats (e.g. misrepresentation of entities, theft of services, unwanted contacts)
- Eavesdropping; Interception and Modification (e.g. rerouting, alteration, hijacking)
- Denial of Service (e.g. flooding, network services DoS, DDoS, malformed protocol messages, fake teardown of session)
- Service Abuse (e.g bill bypassing, hijacking)
- Physical access (e.g. social engineering attacks)
- Interruption of services (e.g. loss of power, resource exhaustion, latency).

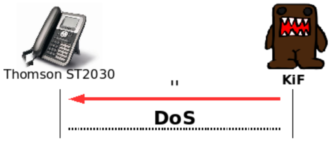
[1] VoIP Security Alliance. `http://voipsa.org`

Rethinking the threats

- Why to kill a fly with a hammer?
- Why limit to sniff network traffic if you can remote-eavesdrop
- Operational toll-fraud on VoIP networks
- VoIP is only the cherry on the cake - Owning the internal network only with VoIP
- Exploiting weaknesses in standardized protocols (SIP)
- The list may continue ...

“What if you are alone and dial 911 and no one answers?”[2]

- When you have nothing to say ...
- One message, just an empty packet can do it
- Affected device Thomson ST2030 v1.52.1

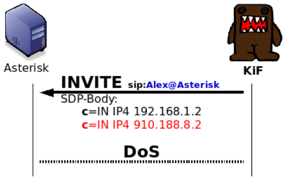


Vulnerability by KiF
CVE-2007-4753

[2]Live Free or Die Hard

Easy starting (DoS)

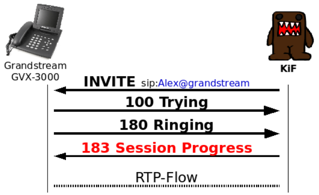
- One INVITE message (even from an anonymous user)
- SDP contains 2 connection headers
- One is an invalid IP address
- All services of the PBX go down
- Affected product Asterisk 1.2.16, 1.4.1 and older



Vulnerability by KiF
CVE-2007-1561

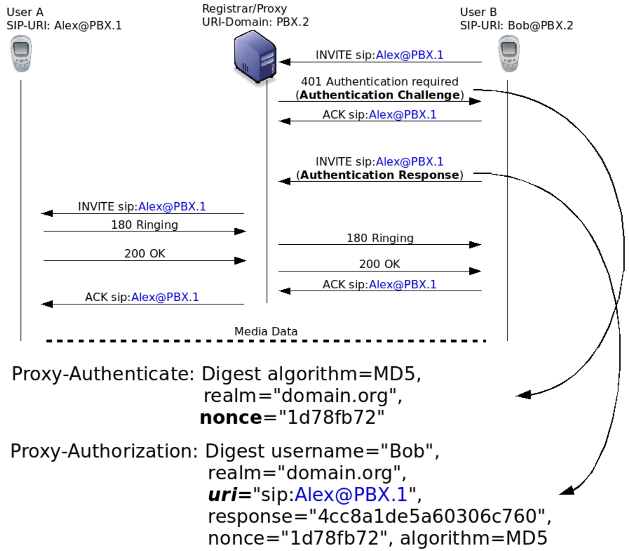
Beyond CALEA / Big Brother dreams

- INVITE an entity but ... reply yourself
- Remote entity accept the call without asking
- Eavesdrops the conversation taken in the room
- **Required stateful fuzzing**



Vulnerability by KiF
CVE-2007-4498

SIP Authentication Background[3]

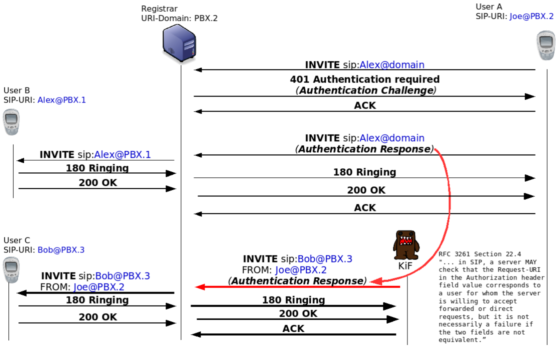


```
Proxy-Authenticate: Digest algorithm=MD5,
                    realm="domain.org",
                    nonce="1d78fb72"

Proxy-Authorization: Digest username="Bob",
                     realm="domain.org",
                     uri="sip:Alex@PBX.1",
                     response="4cc8a1de5a60306c760",
                     nonce="1d78fb72", algorithm=MD5
```

When Crypto is not enough

- Digest Authentication is cryptographically sound but developers ...
- Affected devices
  - Cisco CallManager CVE-2007-5468
  - OpenSer v1.2.2 CVE-2007-5469
- Impact
  - Toll-fraud
  - Call-ID spoofing

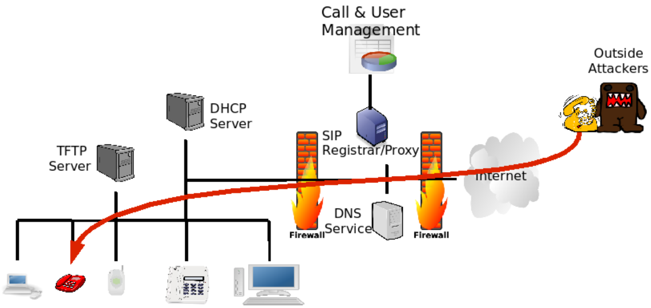


RFC 3261 Section 22.4 "... in SIP, a server MAY check that the Request-URI in the Authorization header field value corresponds to a user for whom the server is willing to accept forwarded or direct requests, but it is not necessarily a failure if the two fields are not equivalent."

- Allows **"Replay"** Attacks but ... to any other entity
- Digest-URI not checked to be the same as Request-URI

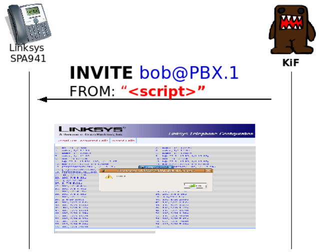
Why VoIP insecurity is really BAD?

- Can VoIP insecurity lead to compromise my network?
- Can I own the internal network just from a regular phone call?



When web2.0 meets SIP

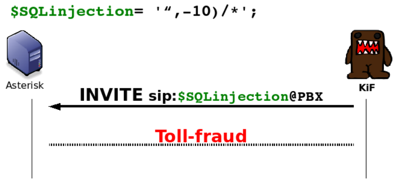
- XSS SIP attacks via VoIP phones
- Extremly dangerous because users connect from the internal network
- Many VoIP devices have integrated Webservers...
- Easily integrated with tools like Beef, AttackAPI, XSSProxy, JIKTO...
- Affected product: Linksys SPA-941 firmware v.5.1.5



Vulnerability by KiF
CVE-2007-5411

The missing ingredient: SQL

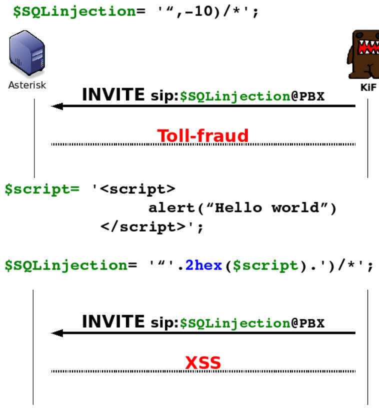
- SQL Injections over SIP
  - SQL tables used for CDR
  - Unescaped inputs
  - Asterisk addons



Vulnerability by KiF
CVE-2007-54881

The missing ingredient: SQL

- SQL Injections over SIP
  - SQL tables used for CDR
  - Unescaped inputs
  - Asterisk addons
- Got one SQL injection?
  Have one XSS for free!
  - Unescaped database inputs
  - FreePBX, trixbox
- XSS via SQL injections
  through SIP



```
$SQLinjection= '",-10)/*';
```

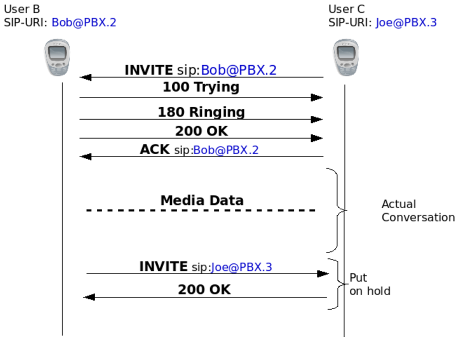
```
$script= '<script>
              alert("Hello world")
          </script>';

$SQLinjection= '"'.2hex($script).')/*';
```

Vulnerability by KiF
CVE-2007-54881

Bunch of Features

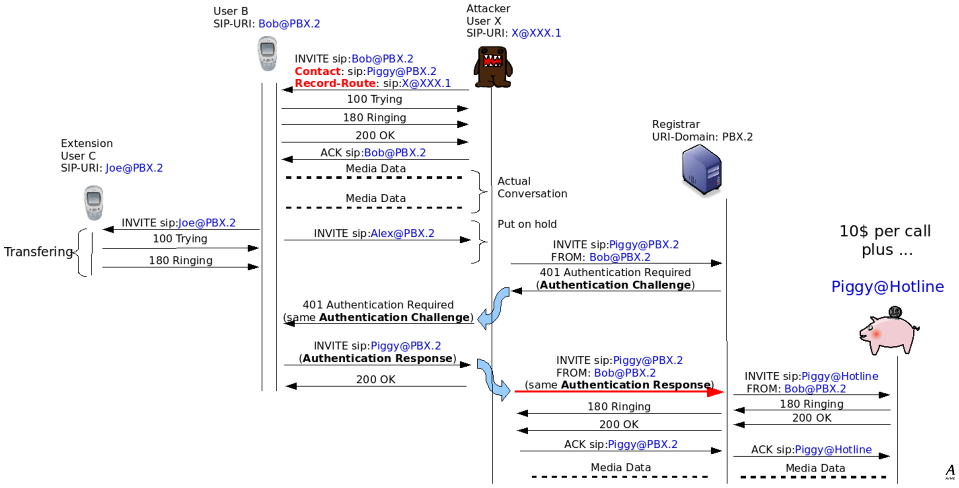
- How re-INVITEs work



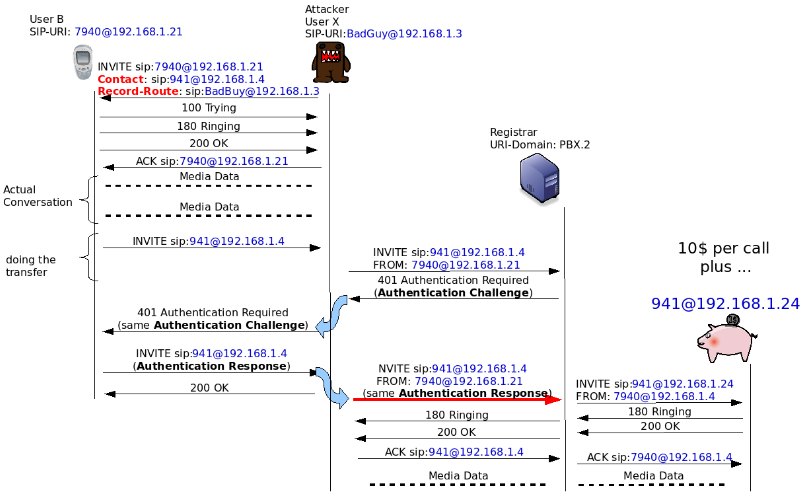
- We can ask to authenticate re-INVITEs?

SIP design flaw?

- We may use such authentication at will :)



The Demo as it is


User B
SIP-URI: 7940@192.168.1.21
Attacker
User X
SIP-URI:BadGuy@192.168.1.3
Registrar
URI-Domain: PBX.2
INVITE sip:7940@192.168.1.21
Contact: sip:941@192.168.1.4
Record-Route: sip:BadBuy@192.168.1.3
100 Trying
180 Ringing
200 OK
ACK sip:7940@192.168.1.21
Media Data
Actual Conversation
Media Data
INVITE sip:941@192.168.1.4
doing the transfer
INVITE sip:941@192.168.1.4
FROM: 7940@192.168.1.21
401 Authentication Required
(Authentication Challenge)
401 Authentication Required
(same Authentication Challenge)
INVITE sip:941@192.168.1.4
(Authentication Response)
200 OK
NVITE sip:941@192.168.1.4
FROM: 7940@192.168.1.21
(same Authentication Response)
180 Ringing
200 OK
ACK sip:941@192.168.1.4
Media Data
10$ per call
plus ...
941@192.168.1.24
INVITE sip:941@192.168.1.24
FROM: 7940@192.168.1.4
180 Ringing
200 OK
ACK sip:7940@192.168.1.4
Media Data

Outline

3. Fuzzing
   - How to find bugs?
   - Syntax fuzzing
   - Stateful fuzzing
   - Evaluation impact

Fuzzing ... one of many way

"Thus, fuzz testing can only be regarded as a bug-finding tool rather than an assurance of quality"[4]

- Emerged as a branch of Software Testing
- Important topic for **Development Cycle/Independent Assessment**
- Based in input data validation
  - Random or invalid characters
  - Malicious data (e.g. string formatters)
- Functional verification is marginal
- Main objective is to find possible potential vulnerabilities

[4] `http://en.wikipedia.org/wiki/Fuzz_testing`

General limitations

- Requires more specification as more precise it gets
- Limited data generation
- Hard to estimate what will be the generated output/expected answer
- Success evaluation depends only in crashed or NOT-crashed
- Unavailable to test specific states of the target (i.e. stateless)
- Learning is not considered
- Unable to decide when to stop
    - Time of testing
    - Quantity of tests or some new metrics?

What to fuzz?

Syntax fuzzing

- Invalid messages may reveal vulnerabilities
- Consider which item of the message should be fuzzed
- Headers or input values may be fuzzed
- Which value should be the one to replace
- The new value may or may not be syntactically correct

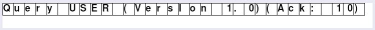
Stateful fuzzing

- Unexpected messages may reveal vulnerabilities
- Decide what type of message to send
- Decide when to send the next message

Fuzzer bases

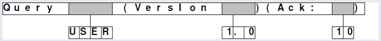
Random fuzzers

- Easy to launch
- Non protocol aware mutations
- Just for mutations, not always useful



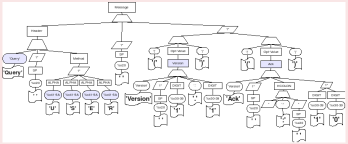
Block based

- Protocol aware messages
- Limited set of variables depending in the blocks definition
- Requires manual description of which are the blocks



Grammar based

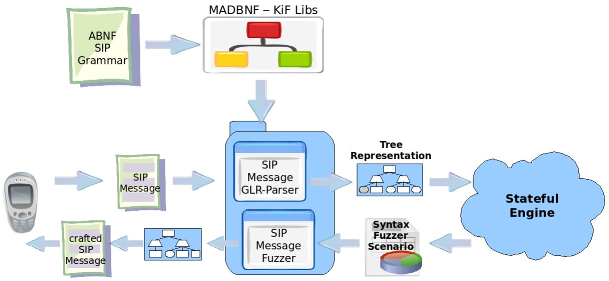
- Protocol aware messages
- Fine grained block based
- Specificity based on the grammar
- Requires detailed grammar as input



Create input grammars or block definitions = tedious job

Making things easier

- Each protocol has its own grammar specification (e.g ABNF grammars as defined in RFC 2234). Why not reuse it?
- Full and precise description of the Protocol Syntax
- Generic approach, allows Parsing & Fuzzing to any Rule of any Grammar

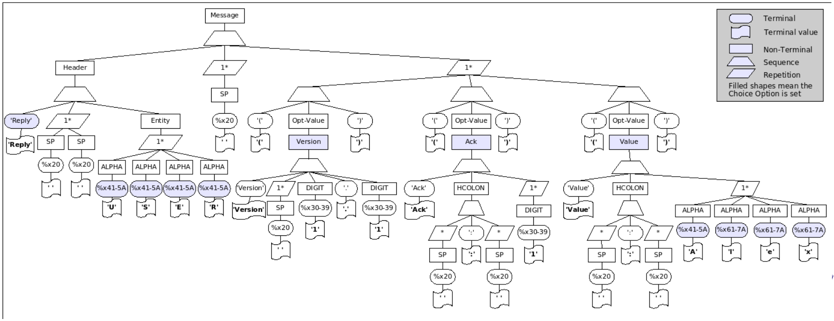


Grammar inference

- Infer rules from a Context-Free Grammar
  (the use of an ABNF provides a complete knowledge of the messages syntax)
- Admits any grammar to create new fuzzers
  (i.e. genericity)
- Allows choosing the fields to fuzz
  (i.e. specificity to generate the crafted message)

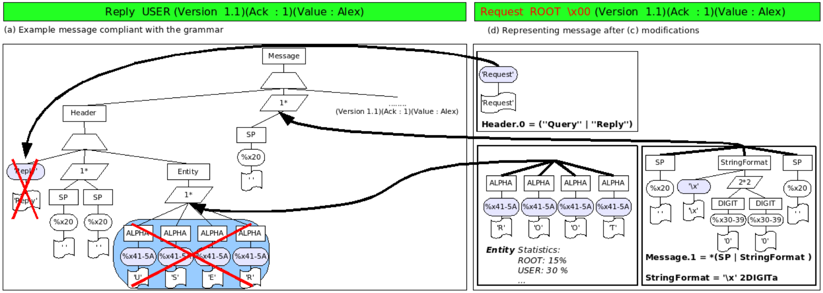
```
Message   =  Header 1*SP 1*( "(" Opt-Value ")" )
Header    =  ("Query" / "Reply") 1*SP Entity
Opt-Value =  (Ack / Value / Version)
Entity    =  1*ALPHA
Ack       =  "Ack" HCOLON 1*DIGIT
Value     =  "Value" HCOLON 1*ALPHA
Version   =  "Version" 1*SP DIGIT "." DIGIT
ALPHA     =  %x41-5A / %x61-7A          ; A-Z / a-z
DIGIT     =  %x30-39                    ; 0-9
HCOLON    =  *SP ":" *SP
SP        =  %x20                       ; space
```

Reply USER (Version 1.1)(Ack : 1)(Value : Alex)



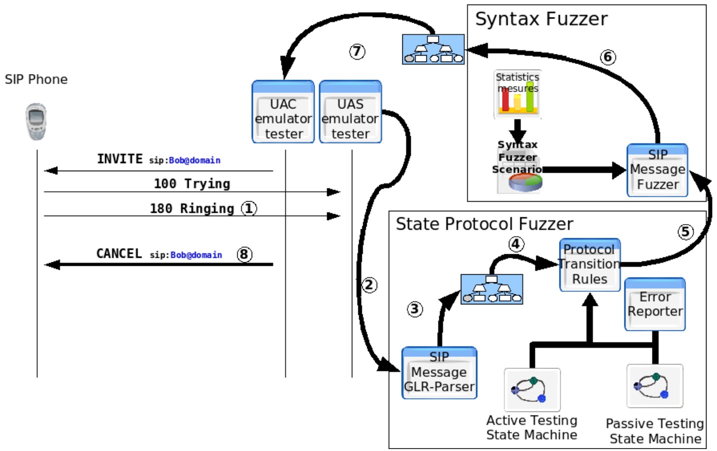
Syntax modifications

- Any grammar rule may be generated (i.e. generation from scratch)
- Any existing reduction may be replaced (i.e. mutation or merging)
- Statistic measures may influence the reduction (i.e. learning from the past)
- New rules can be defined on the fly (i.e. evolving rules)
- Semantic computation may be applied from other nodes (e.g. checksum computations)


Reply USER (Version 1.1)(Ack : 1)(Value : Alex)
Request ROOT \x00 (Version 1.1)(Ack : 1)(Value : Alex)
Message
Header
1*
(Version 1.1)(Ack : 1)(Value : Alex)
SP
%x20
'Reply'
Entity
ALPHA
%x41-5A
'U' 'S' 'E' 'R'
'Request'
Header.0 = ("Query" | "Reply")
'R' 'O' 'O' 'T'
Entity Statistics:
ROOT: 15%
USER: 30 %
...
StringFormat
'\x'
2*2
DIGIT
%x30-39
'0'
Message.1 = *(SP | StringFormat)
StringFormat = '\x' 2DIGITa

(a) Example message compliant with the grammar
(d) Representing message after (c) modifications
(b) Infered structure from the Message in (a)
(c) Structure modifications to the Message in (b)

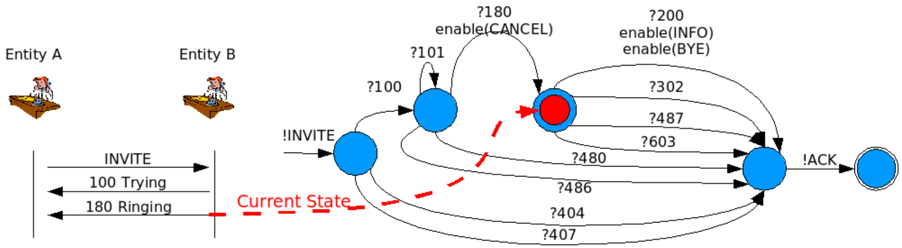
KiF: quick framework



Behavioral testing

Passive Testing

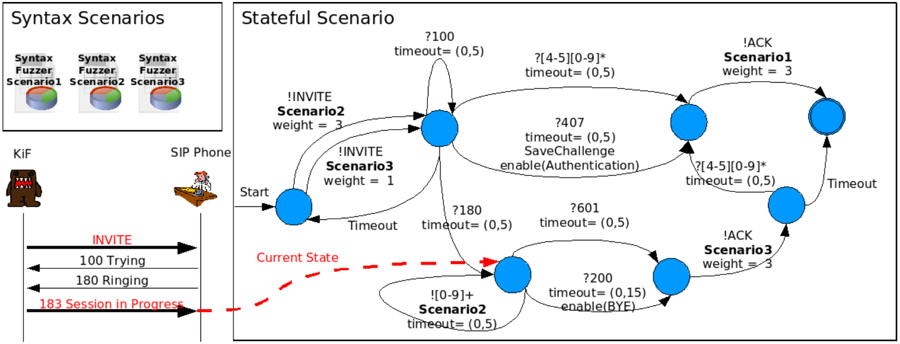
- Collect traces under normal conditions to deduces normal behavior
- Just observes the current traffic
- Infers current state of the unit under test
- Detects abnormal events



Behavioral testing

Active Testing

- Leads the target into a specific state
- Specify which action must be taken at each step
- Event-Driven Probabilistic Finite Automata based Scenarios


Syntax Scenarios
Syntax Fuzzer Scenario1
Syntax Fuzzer Scenario2
Syntax Fuzzer Scenario3
KiF
SIP Phone
INVITE
100 Trying
180 Ringing
183 Session in Progress
Stateful Scenario
Start
!INVITE Scenario2 weight = 3
!INVITE Scenario3 weight = 1
?100 timeout= (0,5)
?[4-5][0-9]* timeout= (0,5)
?407 timeout= (0,5) SaveChallenge enable(Authentication)
!ACK Scenario1 weight = 3
?[4-5][0-9]* timeout= (0,5)
Timeout
Timeout
?180 timeout= (0,5)
?601 timeout= (0,5)
!ACK Scenario3 weight = 3
Current State
![0-9]+ Scenario2 timeout= (0,5)
?200 timeout= (0,15) enable(BYE)

Reporting errors

- If the reply messages are syntactically incorrect
- The type of transition does not match any of the possible ones from the Passive State Machine
- When a message other than the expected one in the scenario occurs (i.e. when the scenario is trying to avoid the normal protocol flow, e.g. for registering)
- And when the device is not responding anymore

About KiF

What you need to launch KiF?

- Understand what you are trying to fuzz
- Python and SIP knowledge required

What KiF does not do!

- Click & launch ...
- Identify the exact problem and create a PoC

Why to use KiF

- Precision and specificity
- Dynamicity, results may be always different
- Adaptability
- Stateful

Demo/Questions?

How can I get KiF?

- KiF source is accessible under conditions
  1. Follow the instructions at `http://kif.gforge.inria.fr/`
  2. Fill the license
  3. Ground mail the License to us