An overview of Stateful Fuzzing

Seyed Andarzian  
seyedbehnam.andarzian@ru.nl  
Radboud University

Cristian Daniele  
cristian.daniele@ru.nl  
Radboud University

Erik Poll  
erik.poll@ru.nl  
Radboud University
What is fuzzing?
Fuzzing a stateless system
Fuzzing a stateful system

System Under Test (e.g. FTP)
Fuzzing a stateful system

Two meanings of grammar:
1. Grammar of the individual messages
2. Grammar of the sequence of messages (state machine)
Fuzzers for stateful systems

Seven relevant categories for fuzzer of stateful systems:

1. Evolutionary tools
2. Grammar-Based tools
3. Evolutionary Grammar-Based Tools
4. Grammar Learner Tools
5. Evolutionary Grammar-Learner Tools
6. Machine Learning-Based Tools
7. Man-in-the-middle Based Tools

All the tools use some common components, although some of these have something specific.
General components

CRAFTER

SYSTEM UNDER TEST

ANOMALY DETECTOR
1. Evolutionary tools

Examples

❖ IJON [1]
❖ SNPS fuzzer [2]
❖ Chen et al. [3]
❖ FFUZZ [4]
1. Evolutionary tools

- Need the source code
- Need human interaction
- Use of different kind of coverage
2. Grammar-based tools

- **GRAMMAR (PROVIDED BY THE USER)**
- **CRAFTER**
- **SYSTEM UNDER TEST**
- **ANOMALY DETECTOR**

**Examples**

- Peach [5]
- SNOOZE [6]
- PROTOS [7]
- Sulley [8]
- BooFuzz [9]
- Fuzzowski [10]
2. Grammar-based tools

- Require a complete grammar of the protocol
- Bugs may be overlooked
- Users can specify the message field to fuzz
- No source code required

```
<xml version="1.0" encoding="UTF-8"/>
<Peach xmlns="http://phd.org/2006/Peach" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="peach.xsd">
  <Include ms="default" src="file:defaults.xml"/>

  <DataModel name="WOFF">
    <Blob name="WOFFHeader" length="44"/>
    <Blob name="TableDirectoryEntry" length="28" maxOccurs="100"/>
    <Blob name="FontTables"/>
  </DataModel>

  <StateModel name="State" initialState="Initial">
    <State name="Initial">
      <Action type="output">
        <DataModel ref="WOFF"/>
        <!-- Peach will use this file as a starting point for generating fuzzed files. It will therefore need to figure out how this file can be interpreted based on the DataModel specified above (ref=WOFF).-->
        <DataFile name="/tmp/SomeRealFont.woff"/>
      </Action>
      <Action type="close"/>
    </State>
  </StateModel>

  <Test name="TheTest">
    <StateModel ref="State"/>
    <Publisher class="file.FileWriter">
      <!-- Peach will write each generated fuzzed file to this path. -->
      <Param name="fileName" value="/tmp/fuzz/font.woff"/>
    </Publisher>
    <Test>
      <Run name="DefaultRun">
        <Test ref="TheTest"/>
        <!-- Configure a logger to store collected information -->
        <Logger class="logger.FileSystem">
          <Param name="path" value="/tmp/peach.log"/>
        </Logger>
      </Run>
    </Test>
  </Test>
</Peach>
```

Peach data model of WOFF file format
3. Evolutionary grammar-based tools

Examples

- RESTler
- SPFuzz
3. Evolutionary grammar-based tools

❖ This (inevitably) may lead to an enormous number of combinations

❖ Feedback mechanism

- Category limits
- Category strengths

- Zero knowledge
- Some knowledge
- Full knowledge
4. Grammar learner tools

GRAMMAR -> CRAFTER -> SYSTEM UNDER TEST -> ANOMALY DETECTOR

GRAMMAR EXTRACTOR <- SET OF TRACES

Examples

❖ Pulsar [13]
❖ Hsu et al. [14]
4. Grammar learner tools

- Category limits
  - A comprehensiveness and completeness of the traces is required

- Category strengths
  - No protocol knowledge required
  - No source code required

Overview of Pulsar
5. Evolutionary grammar learner tools

- SGPFuzzer
- nyx-net
- AFLnet
- StateAFL
- LearnLib
- EPF
- Doupé et al.
- Böhme et al.

Examples

- SGPFuzzer
- nyx-net
- AFLnet
- StateAFL
- LearnLib
- EPF
- Doupé et al.
- Böhme et al.
5. Evolutionary grammar learner tools

- Need the source code
- Human interaction
- No protocol knowledge required
- Feedback mechanism

Overview of AFLNet

Seyed Andarzian, Cristian Daniele and Erik Poll
6. Machine learning-based tools

- SeqFuzzer [23]
- GANFuzz [24]
- Fan et al. [25]
6. Machine learning-based tools

Category limits
- Cannot significantly change the order of the messages
- Need many traces

Category strengths
- No protocol knowledge required
- No source code required

GAN to learn the grammar of protocol messages
7. Man-in-the-middle tools

CLIENT (OR SERVER) → CRAFTER → SYSTEM UNDER TEST → ANOMALY DETECTOR

Examples

- AutoFuzz [26]
- SECFuzz [27]
- Ramsauer’s [28]
7. Man-in-the-middle tools

- Category limits
  - Inability to modify the order of the messages

- Category strengths
  - No protocol knowledge required
  - No source code required

AutoFuzz Proxy Model
Ideas used by some tools

- FORK STRATEGY
- DIFFERENT MODEL EXTRACTION ALGORITHMS
- CODE ANNOTATION
- DIFFERENT CRAFT FUNCTIONS
Which tool is best?

It depends! What can I use?

- Source Code
- Grammar of the System
- Traces
- Live Communication
Possible cases study

- FTP
- TLS/SSL
- 104 (electricity grid)
- OCPP (electric cars)
- DICOM (health application)
- ModBus
- OPC-UA
- COM/ DCOM
- ZeroMQ
- Bluetooth
- MQTT
- IMAP
- 5G
- ...
Future work

❖ Why not use L* to extract the state model?

❖ Why not use AFLNet just to extract the state model?

❖ Is it possible to combine different strategies/tools?

❖ Is it possible to realise a solid benchmark, maybe using ProFuzzBench?
## Conclusion

<table>
<thead>
<tr>
<th>TOOL</th>
<th>Type of the tool</th>
<th>Evolution guided by</th>
<th>Limitations</th>
<th>Techniques used</th>
<th>Tools it is based on</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNPS fuzzer [2]</td>
<td>Evolutionary</td>
<td>Response &amp; Code</td>
<td>Initial seeds are provided by users</td>
<td>message chain queue, snapshot</td>
<td>AFL</td>
<td>Traces</td>
</tr>
<tr>
<td>FFUZZ [4]</td>
<td>Evolutionary</td>
<td>Response &amp; Code</td>
<td>Require human Interaction</td>
<td>Different management of the socket, initial seeds built by observing a real communication</td>
<td>AFL, AFLNet</td>
<td>Traces</td>
</tr>
<tr>
<td>Peach [5]</td>
<td>Grammar based</td>
<td>NA</td>
<td>Cannot use runtime information</td>
<td>Peach Pit (xml file including DataModel and State Model)</td>
<td>-</td>
<td>Grammar</td>
</tr>
<tr>
<td>PROTOS [7]</td>
<td>Grammar based</td>
<td>NA</td>
<td>Cannot use runtime information</td>
<td>Block based approach to build the request</td>
<td>-</td>
<td>Grammar</td>
</tr>
<tr>
<td>Sulley [8]</td>
<td>Grammar based</td>
<td>NA</td>
<td>Cannot use runtime information</td>
<td>-</td>
<td>Sulley</td>
<td>Grammar</td>
</tr>
<tr>
<td>BooFuzz [9]</td>
<td>Grammar based</td>
<td>NA</td>
<td>Cannot use runtime information</td>
<td>-</td>
<td>Sulley</td>
<td>Grammar</td>
</tr>
<tr>
<td>Fuzzowski [10]</td>
<td>Grammar based</td>
<td>NA</td>
<td>Cannot use runtime information</td>
<td>-</td>
<td>-</td>
<td>Grammar</td>
</tr>
<tr>
<td>SPFuzz [12]</td>
<td>Evolutionary Grammar Based</td>
<td>Code</td>
<td>-</td>
<td>Three-level mutation strategy (head, content and sequence)</td>
<td>AFL, BooFuzz</td>
<td>Grammar</td>
</tr>
<tr>
<td>Pulstar [13]</td>
<td>Grammar Learner</td>
<td>NA</td>
<td>Only mutates a single input from a sequence, grammar of the message required mutates a single input</td>
<td>Extracts markov chain from a sample of traces</td>
<td>PRISMA</td>
<td>Traces</td>
</tr>
<tr>
<td>Hisu et al. [14]</td>
<td>Grammar Learner</td>
<td>NA</td>
<td>Only mutates a single input from a sequence, grammar of the message required mutates a single input</td>
<td>Passive Synthesizer with Partial FSM Reduction, four level mutation strategy</td>
<td>-</td>
<td>Traces</td>
</tr>
<tr>
<td>LearnLib [19]</td>
<td>Evolutionary Grammar Learner</td>
<td>Response</td>
<td>Not a proper fuzzer</td>
<td>-</td>
<td>-</td>
<td>L*</td>
</tr>
<tr>
<td>EPF [20]</td>
<td>Evolutionary Grammar Learner</td>
<td>Response &amp; Code</td>
<td>The response oracle is only able to detect a drop in connectivity</td>
<td>-</td>
<td>AFL++ &amp; Fuzzowski</td>
<td>Traces &amp; Code</td>
</tr>
<tr>
<td>Enemy of the State [21]</td>
<td>Evolutionary Grammar Learner</td>
<td>Response</td>
<td>Can not infer the grammar of the message</td>
<td>Observe web page response to devise the state model</td>
<td>-</td>
<td>Traces</td>
</tr>
<tr>
<td>SeqFuzzer [23]</td>
<td>ML based</td>
<td>NA</td>
<td>Cannot fuzz the order of the messages</td>
<td>seq2seq model</td>
<td>seq2Seq model</td>
<td>-</td>
</tr>
<tr>
<td>EnFuzz [24]</td>
<td>Activity based</td>
<td>NA</td>
<td>Cannot fuzz the order of the messages</td>
<td>Seq-Can algorithm</td>
<td>Seq-Can algorithm</td>
<td>-</td>
</tr>
<tr>
<td>Machine Learning for Black-Box Fuzzing of Network Protocols [25]</td>
<td>ML based</td>
<td>NA</td>
<td>Cannot fuzz the order of the messages</td>
<td>-</td>
<td>-</td>
<td>Traces</td>
</tr>
<tr>
<td>AutoFuzz [26]</td>
<td>Man in the middle based</td>
<td>NA</td>
<td>Cannot fuzz the order of the messages</td>
<td>Construct and minimize FSA &amp; Construct Generic Message Sequences</td>
<td>-</td>
<td>Traces</td>
</tr>
<tr>
<td>SECFFUZZ: Fuzz-testing Security Protocols [27]</td>
<td>Man in the middle based</td>
<td>Memory</td>
<td>Cannot fuzz the order of the messages</td>
<td>Insert a well formed message at random position in the sequence</td>
<td>Purify, Valgrind</td>
<td>Traces</td>
</tr>
<tr>
<td>Black-Box Live Protocol Fuzzing [28]</td>
<td>Man in the middle based</td>
<td>NA</td>
<td>Cannot fuzz the order of the messages</td>
<td>XML templates</td>
<td>-</td>
<td>Traces</td>
</tr>
</tbody>
</table>

Seyed Andarzian, Cristian Daniele, and Erik Poll

Stateful Fuzzing — Slide 26 of 28
Fuzzing has been proved a valid security testing technique
Stateful fuzzing is much more difficult than stateless fuzzing
Fuzzing is an active research field
Plenty of fuzzers have been developed in the last years
References


