StateFuzzVis:

Identifying Blockers in Network Protocol Fuzzing

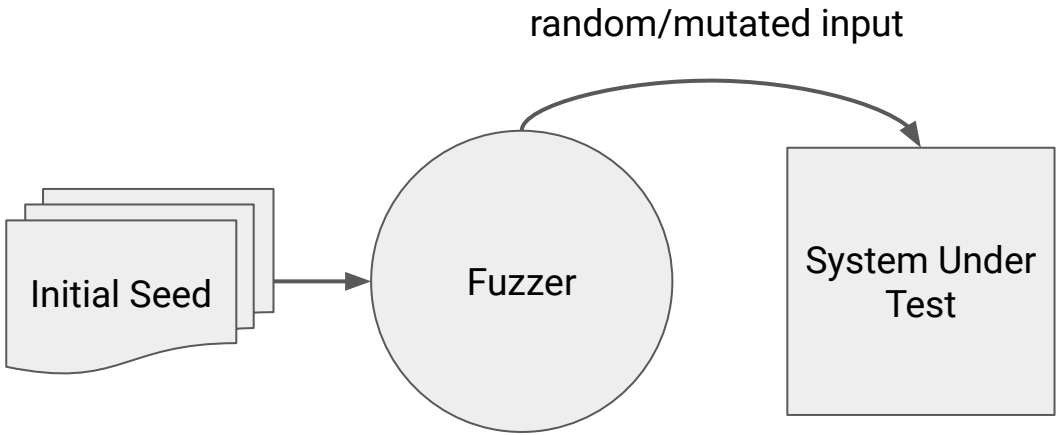
Steve Gustaman
stevegustaman@kaist.ac.kr

Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique

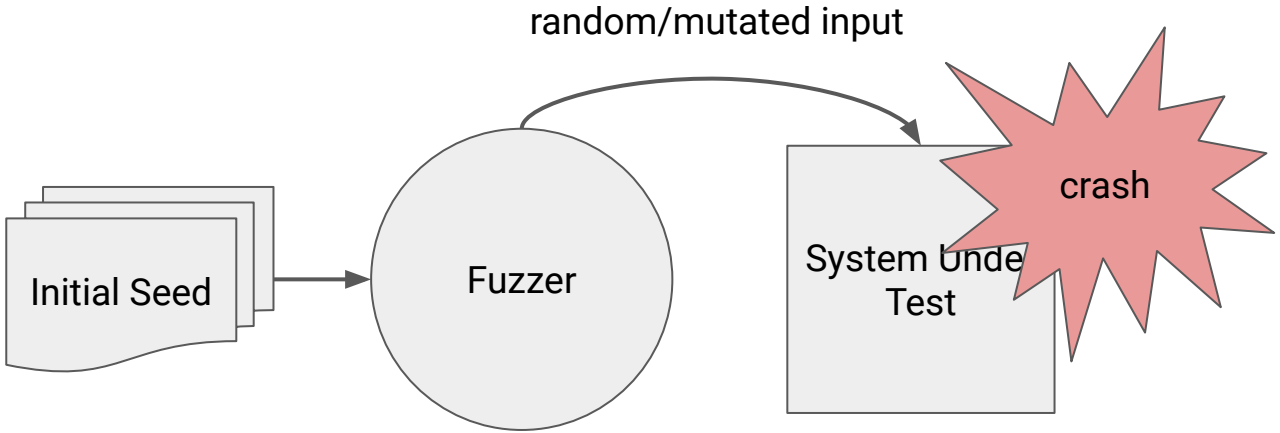
Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique



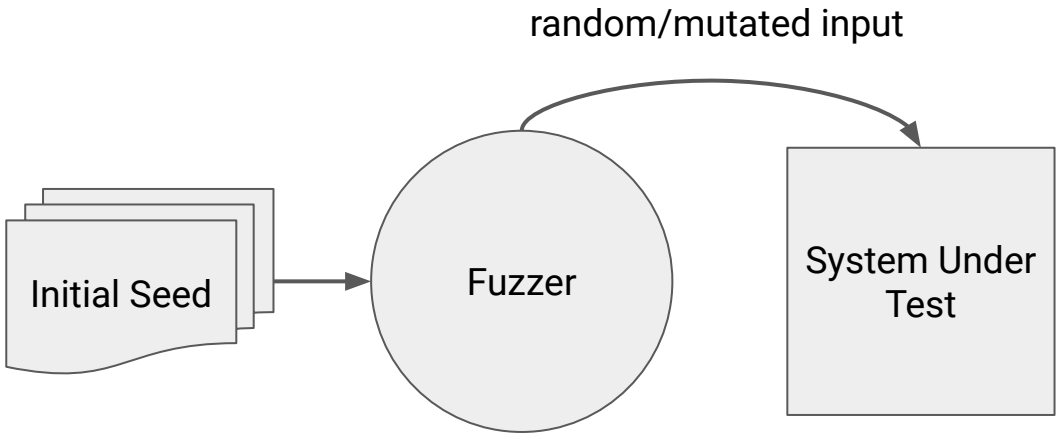
Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique



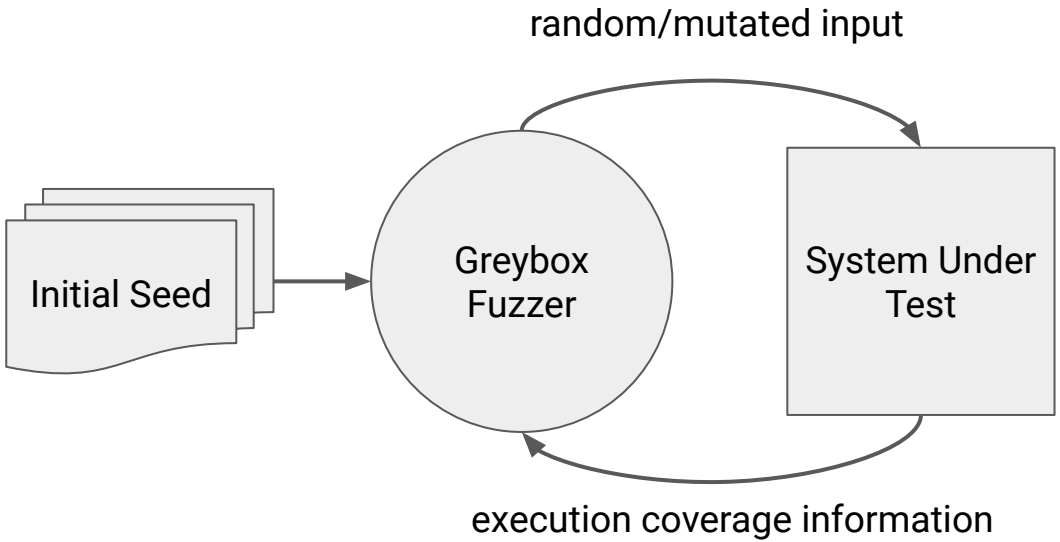
Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzing uses **coverage information** to guide its fuzzing process



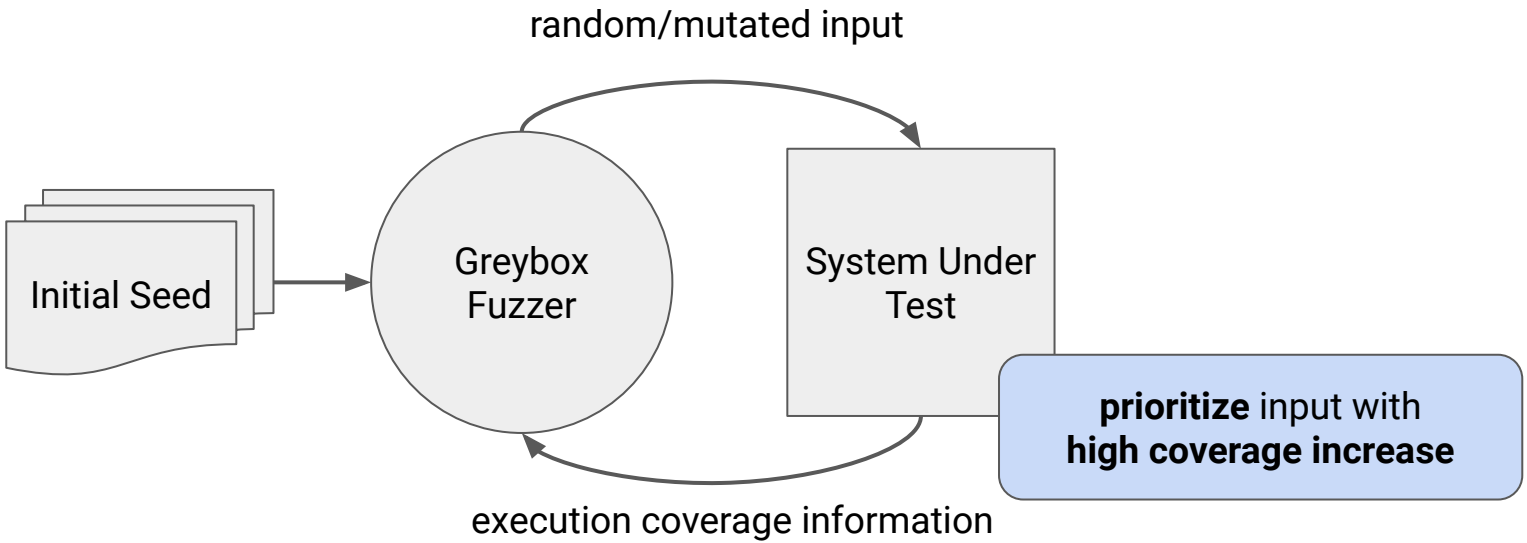
Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzing uses **coverage information** to guide its fuzzing process



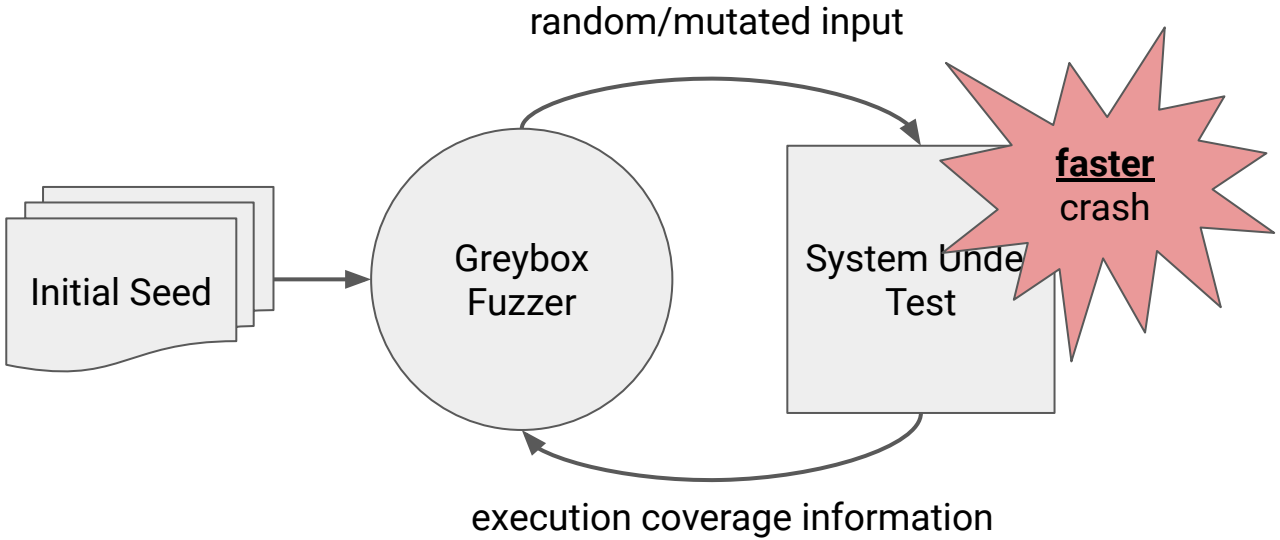
Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzing uses **coverage information** to guide its fuzzing process



Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzing uses **coverage information** to guide its fuzzing process

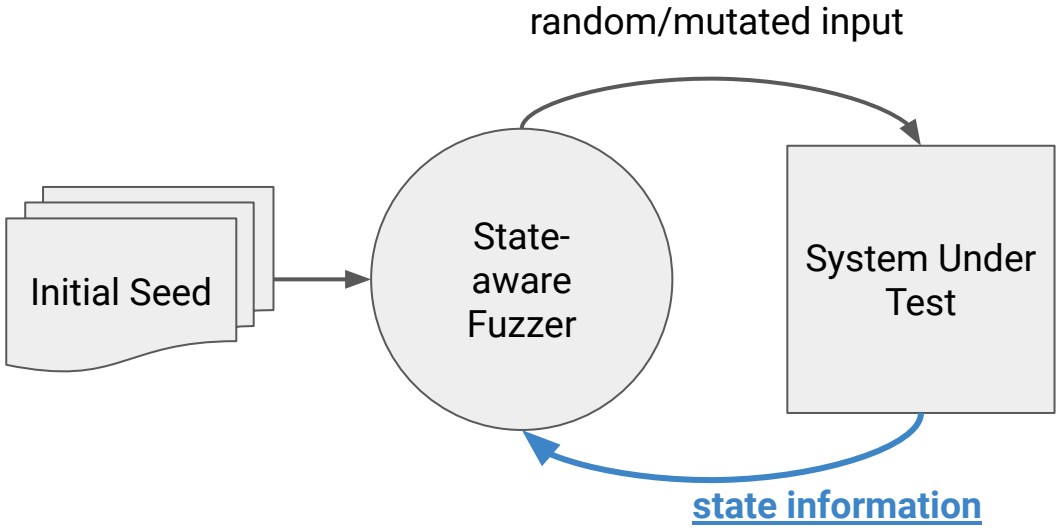


Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process

Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process



Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

Motivation

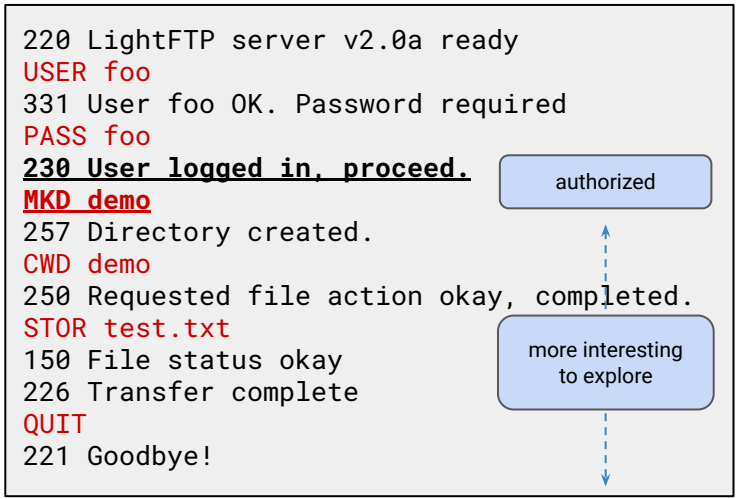
- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

authorized

Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process



Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different **blockers**

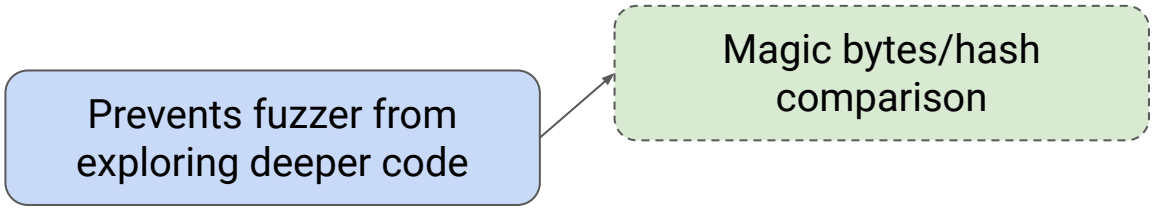
Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different **blockers**

Prevents fuzzer from exploring deeper code

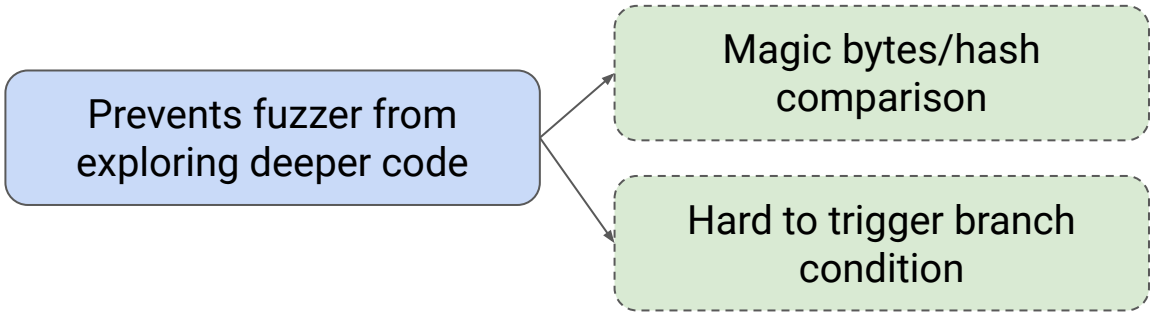
Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different **blockers**



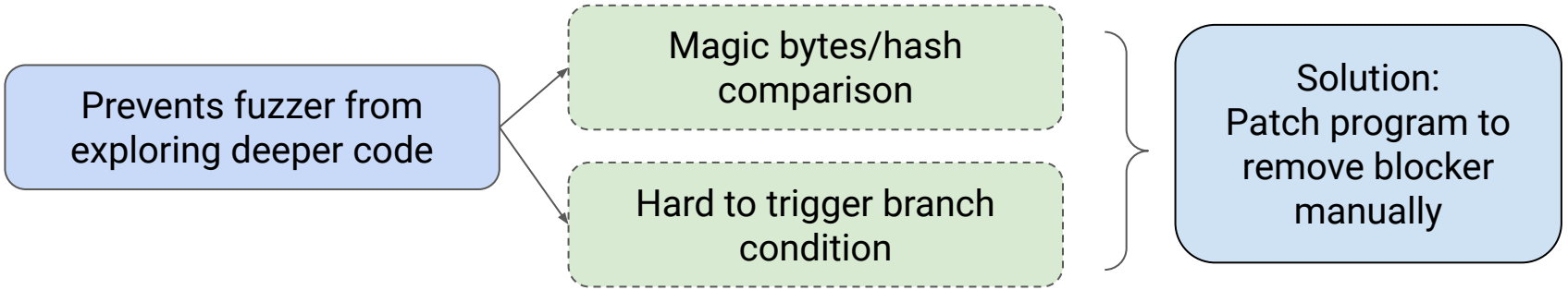
Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different **blockers**



Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different **blockers**



Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different blockers
- Different network protocol implementations have different **blockers**

Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different blockers
- Different network protocol implementations have different **blockers**
  - **Blockers are critical to fuzzing performance**

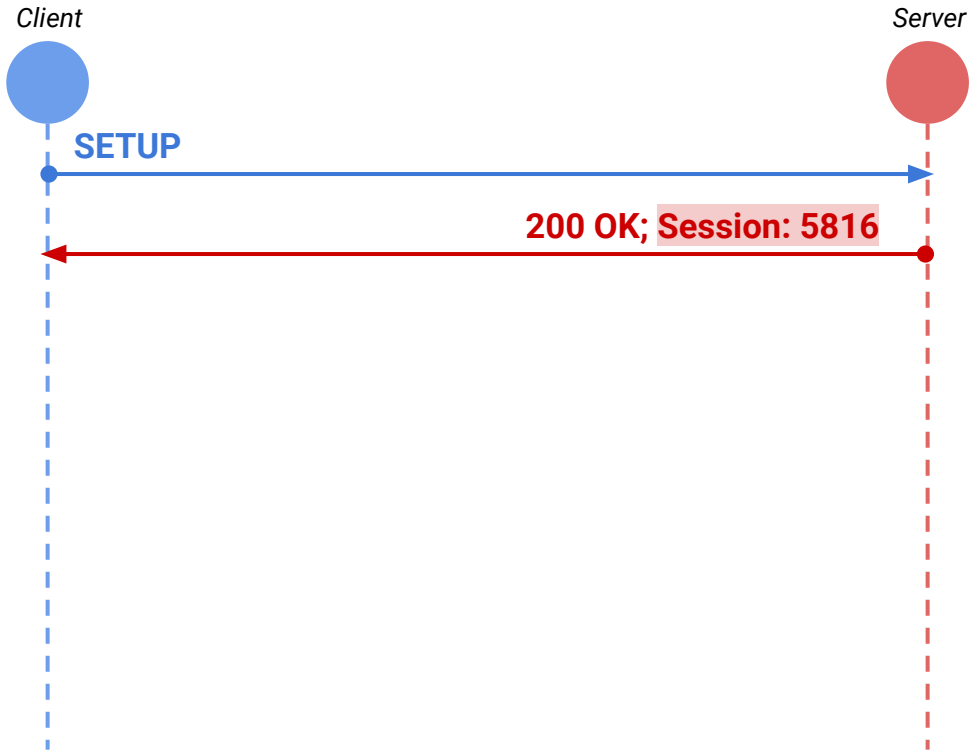
Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different blockers
- Different network protocol implementations have different **blockers**
  - **Blockers are critical to fuzzing performance**

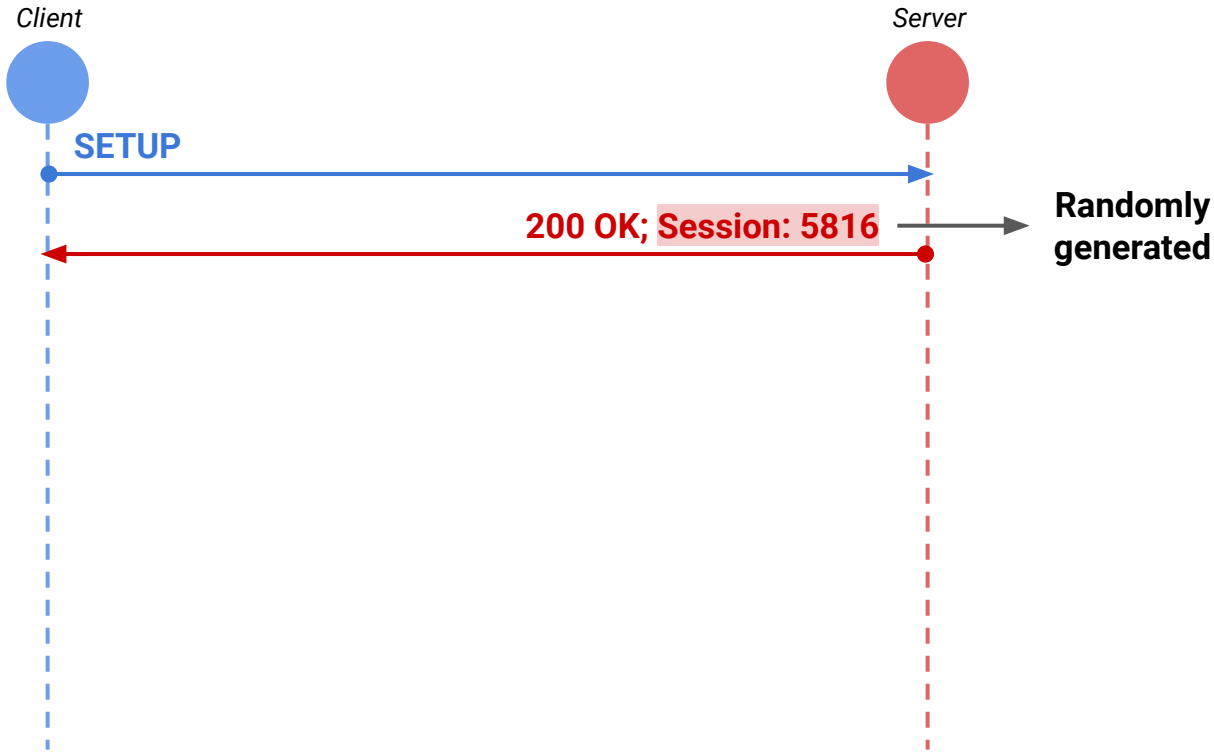
Prevents fuzzer from exploring
deeper code and **states**

Blockers example - RTSP Client-Server

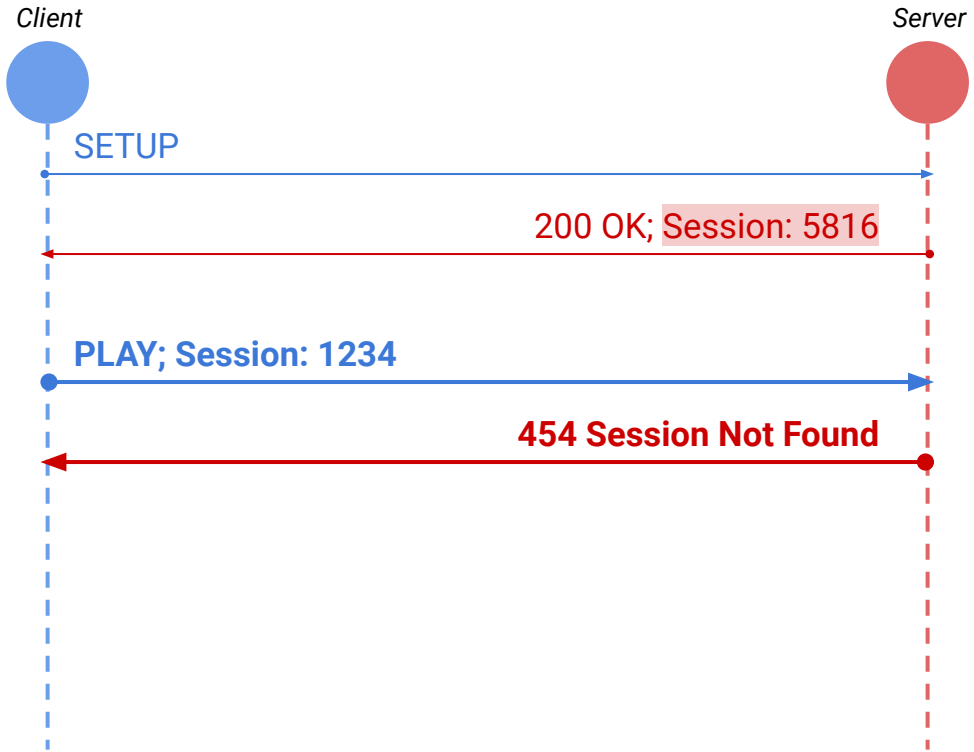
Blockers example - RTSP Client-Server



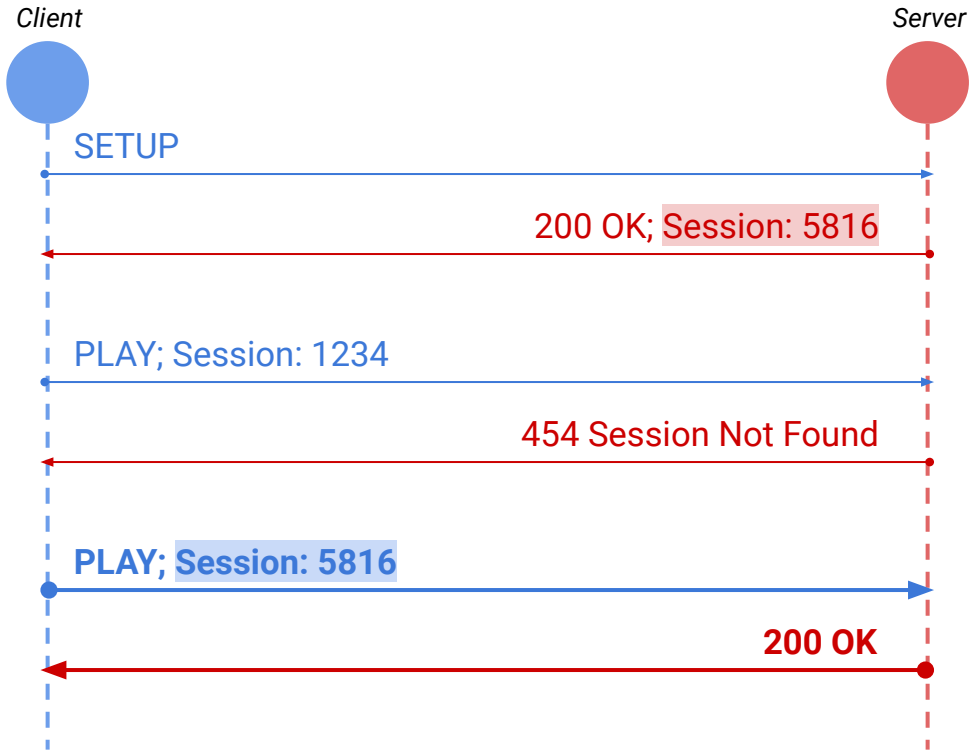
Blockers example - RTSP Client-Server



Blockers example - RTSP Client-Server

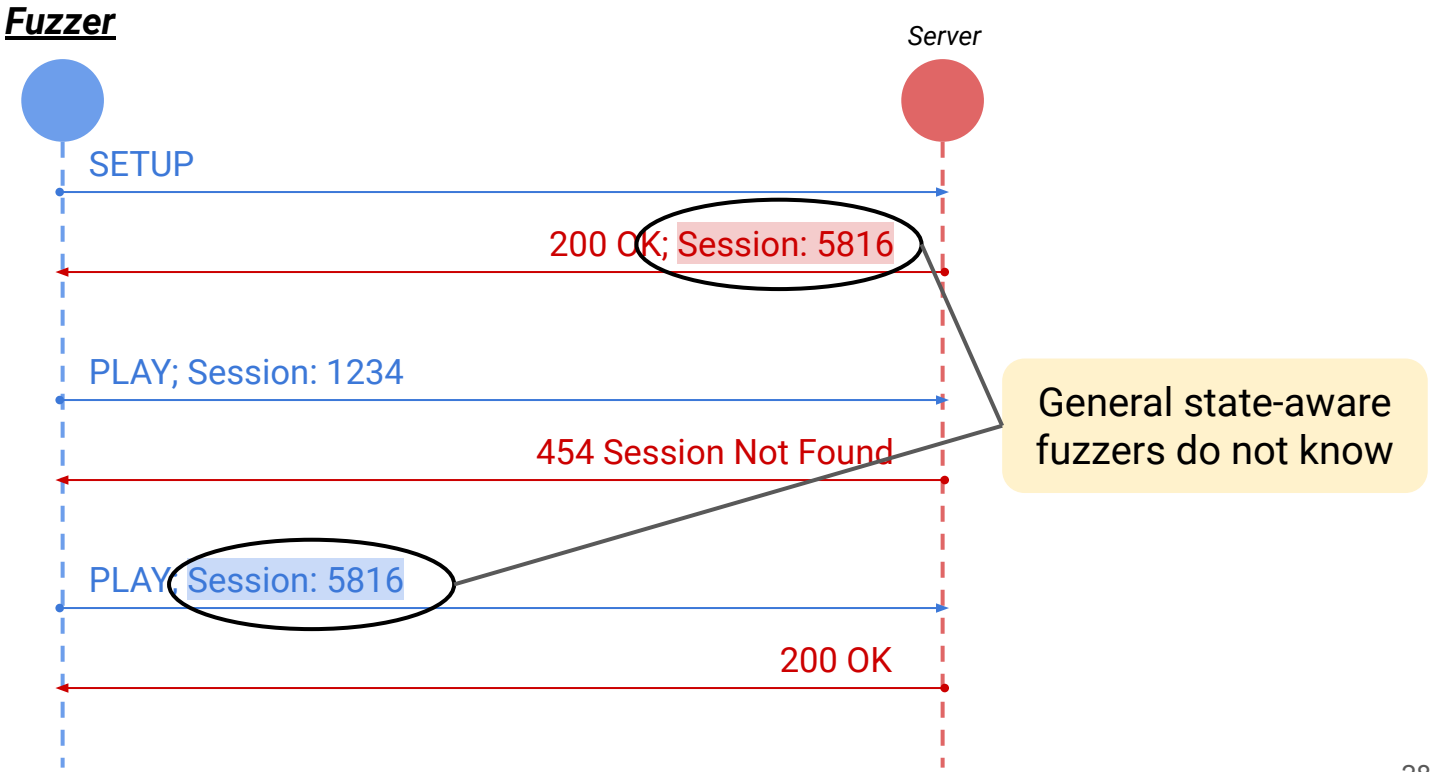


Blockers example - RTSP Client-Server



Blockers example - RTSP Client-Server

Blockers example - RTSP Client-Server



Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different blockers
- Different network protocol implementations have different **blockers**
  - **Blockers are critical to fuzzing performance**

Prevents fuzzer from exploring
deeper code and **states**

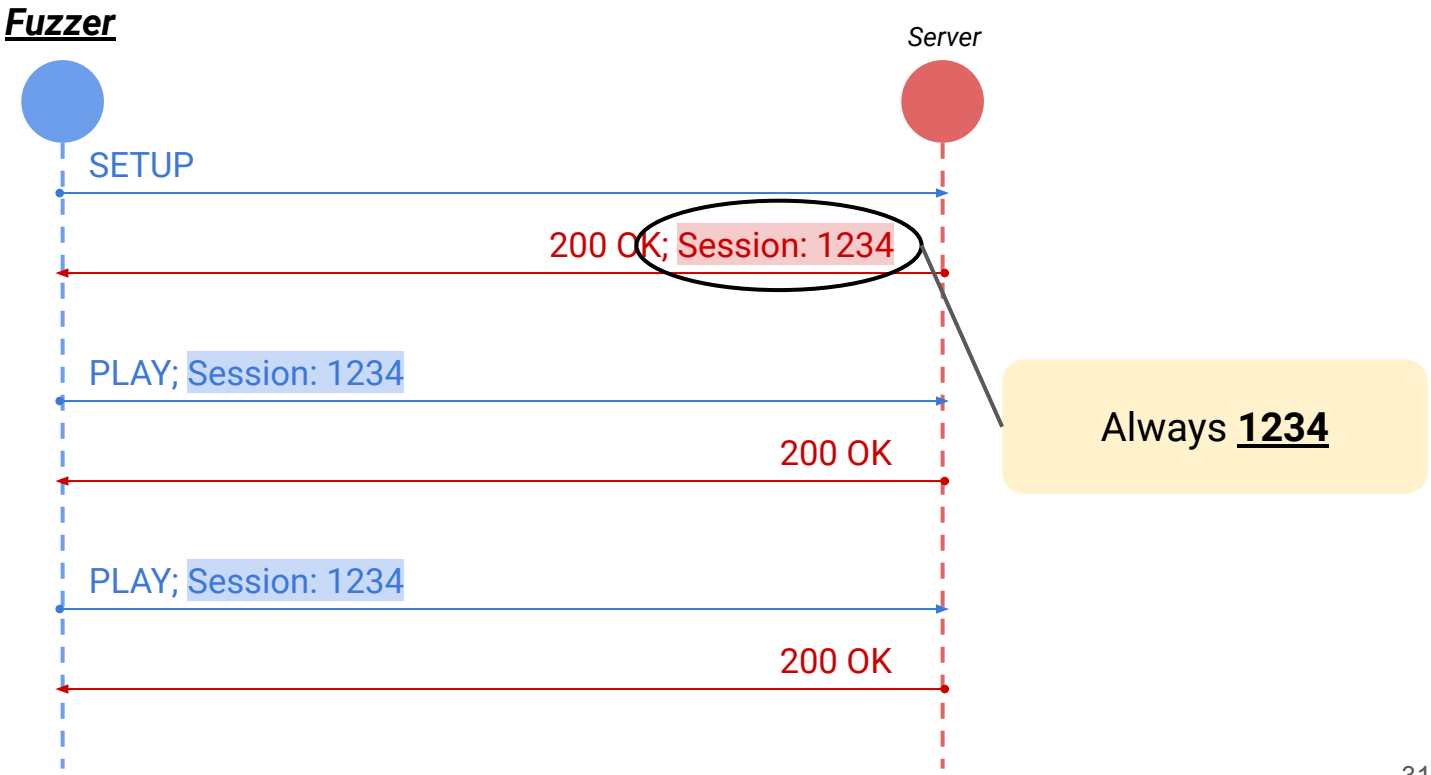
Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different blockers
- Different network protocol implementations have different **blockers**
  - **Blockers are critical to fuzzing performance**

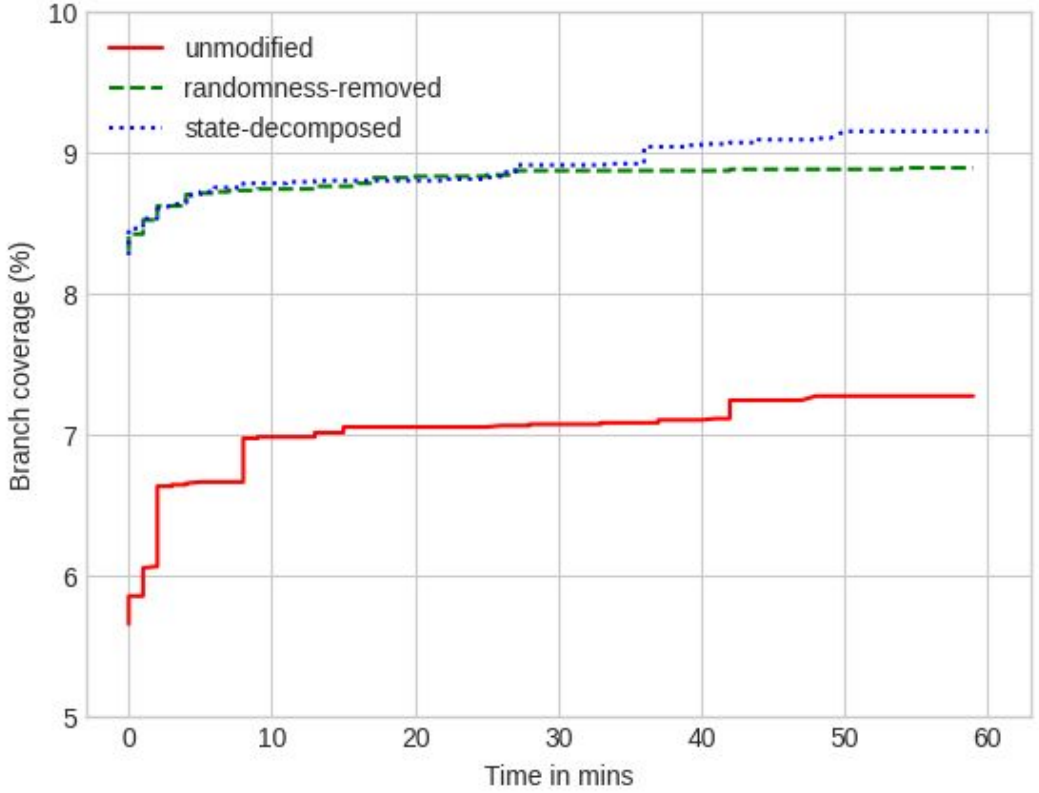
Prevents fuzzer from exploring **deeper code** and **states**

Solution:
Patch **server** to remove blocker manually

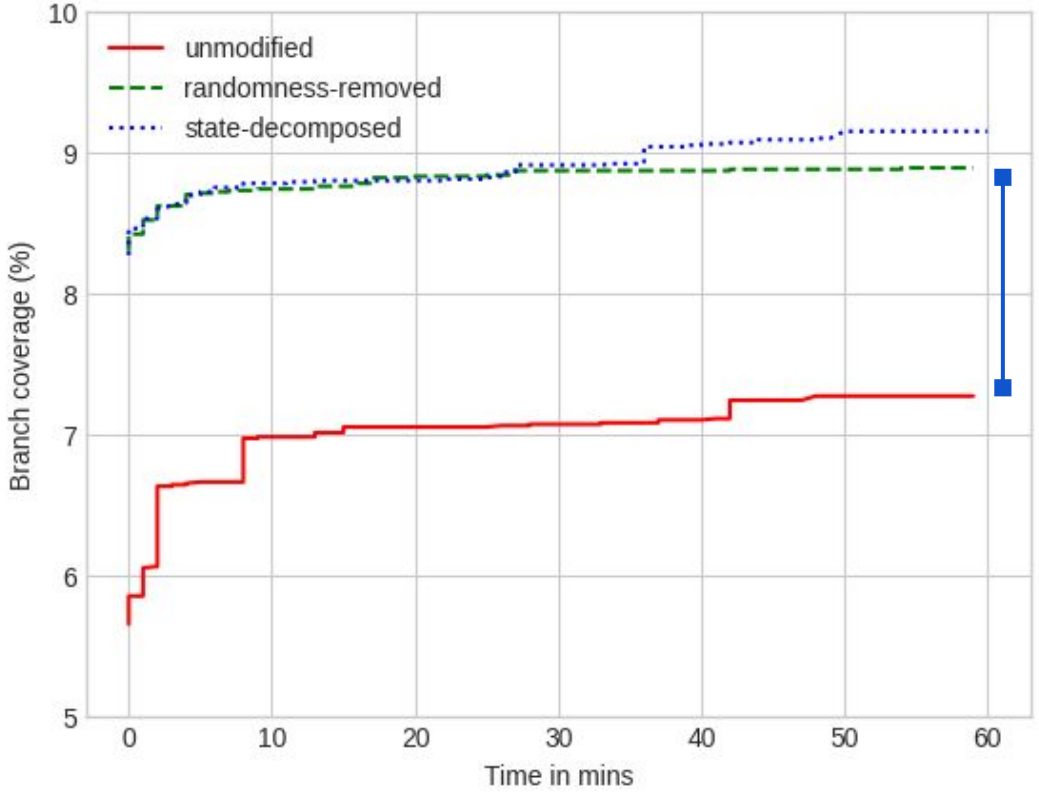
Blockers example - RTSP Client-Server



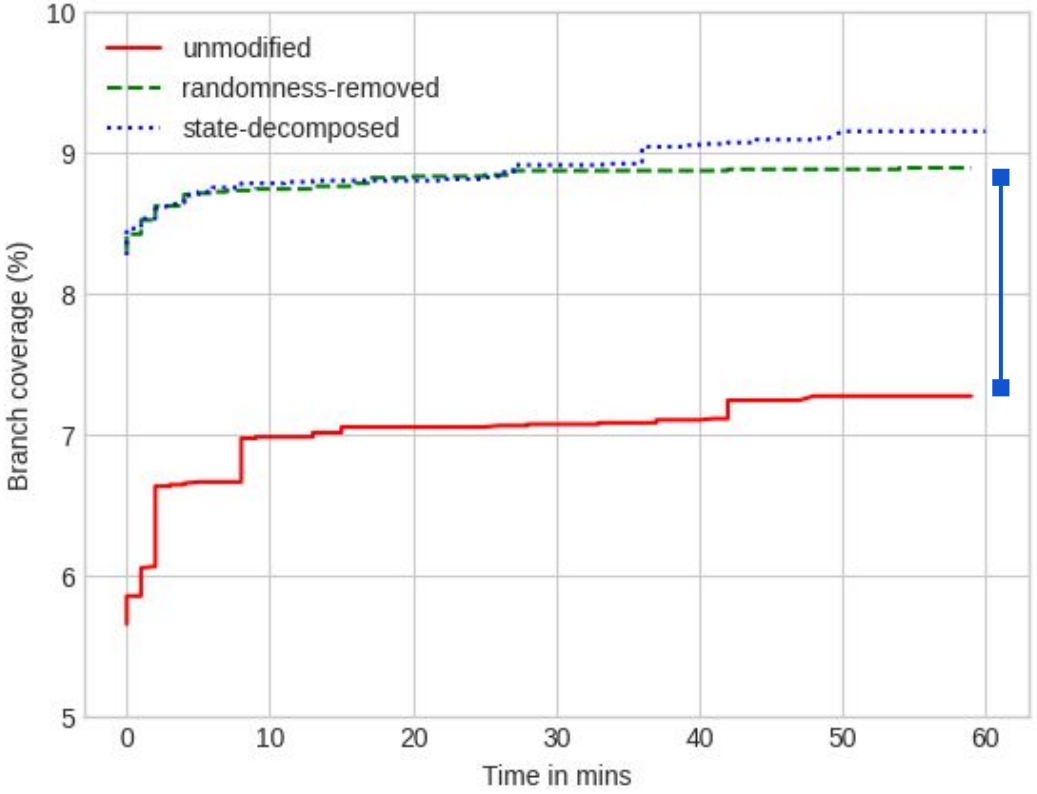
Motivation


unmodified
randomness-removed
state-decomposed
Branch coverage (%)
Time in mins
10
9
8
7
6
5
0
10
20
30
40
50
60

Motivation


unmodified
randomness-removed
state-decomposed
Branch coverage (%)
Time in mins
10
9
8
7
6
5
0
10
20
30
40
50
60

Motivation



removing blocker
=
26% coverage increase

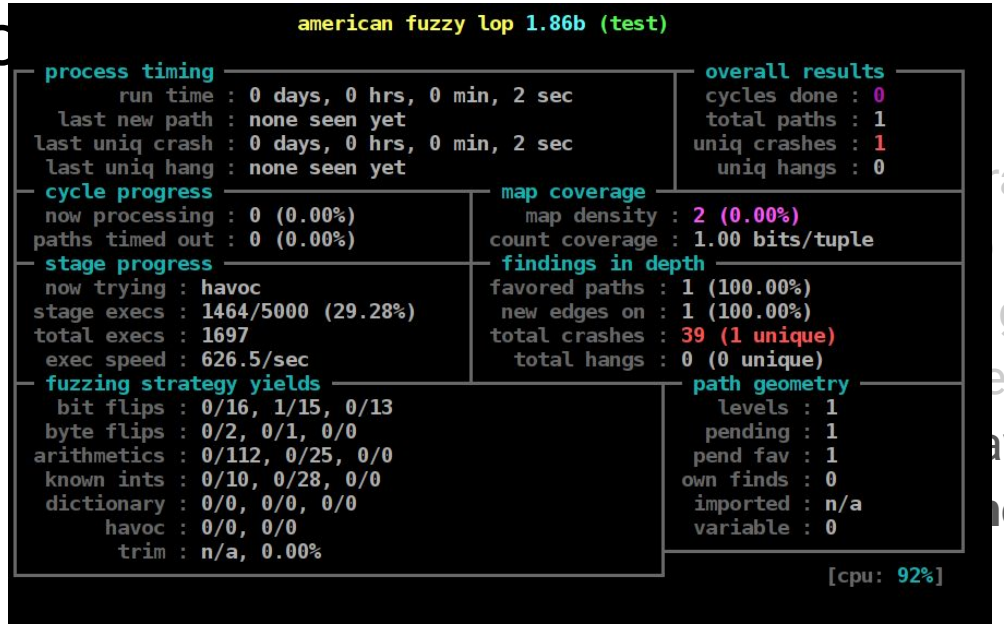
Motivation

- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different blockers
- Different network protocol implementations have different **blockers**
  - **Blockers are critical to fuzzing performance**

Motivation

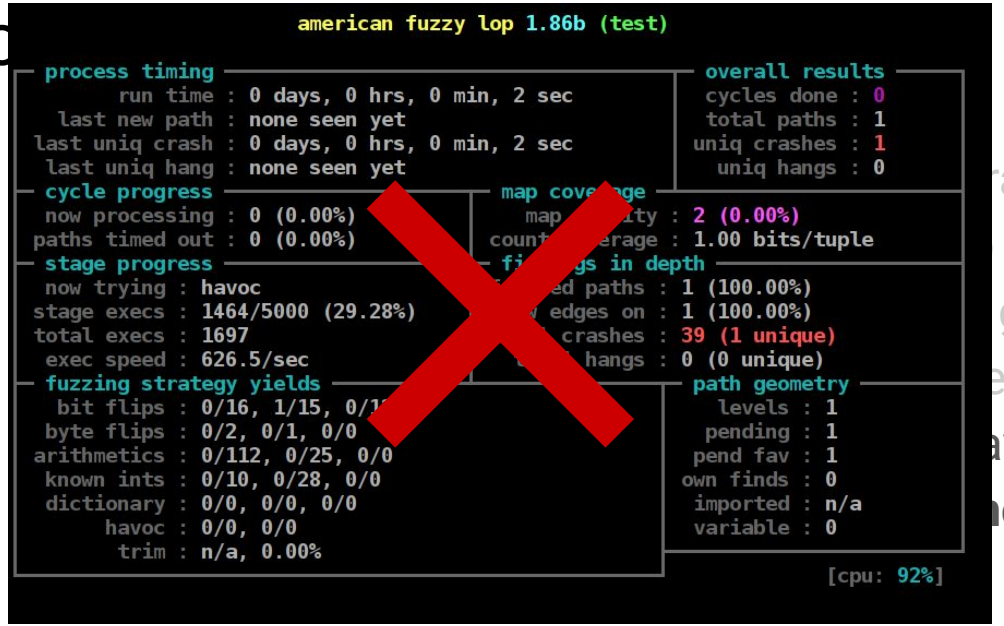
- **Fuzzing** is an **effective** and **widely-used** vulnerability detection technique
- Greybox fuzzers uses **coverage information** to guide its fuzzing process
- State-aware fuzzers uses **state information** to guide its fuzzing process
- Different target programs have different blockers
- Different network protocol implementations have different **blockers**
  - **Blockers are critical to fuzzing performance**
  - **Blockers can be hard to notice**

Mo



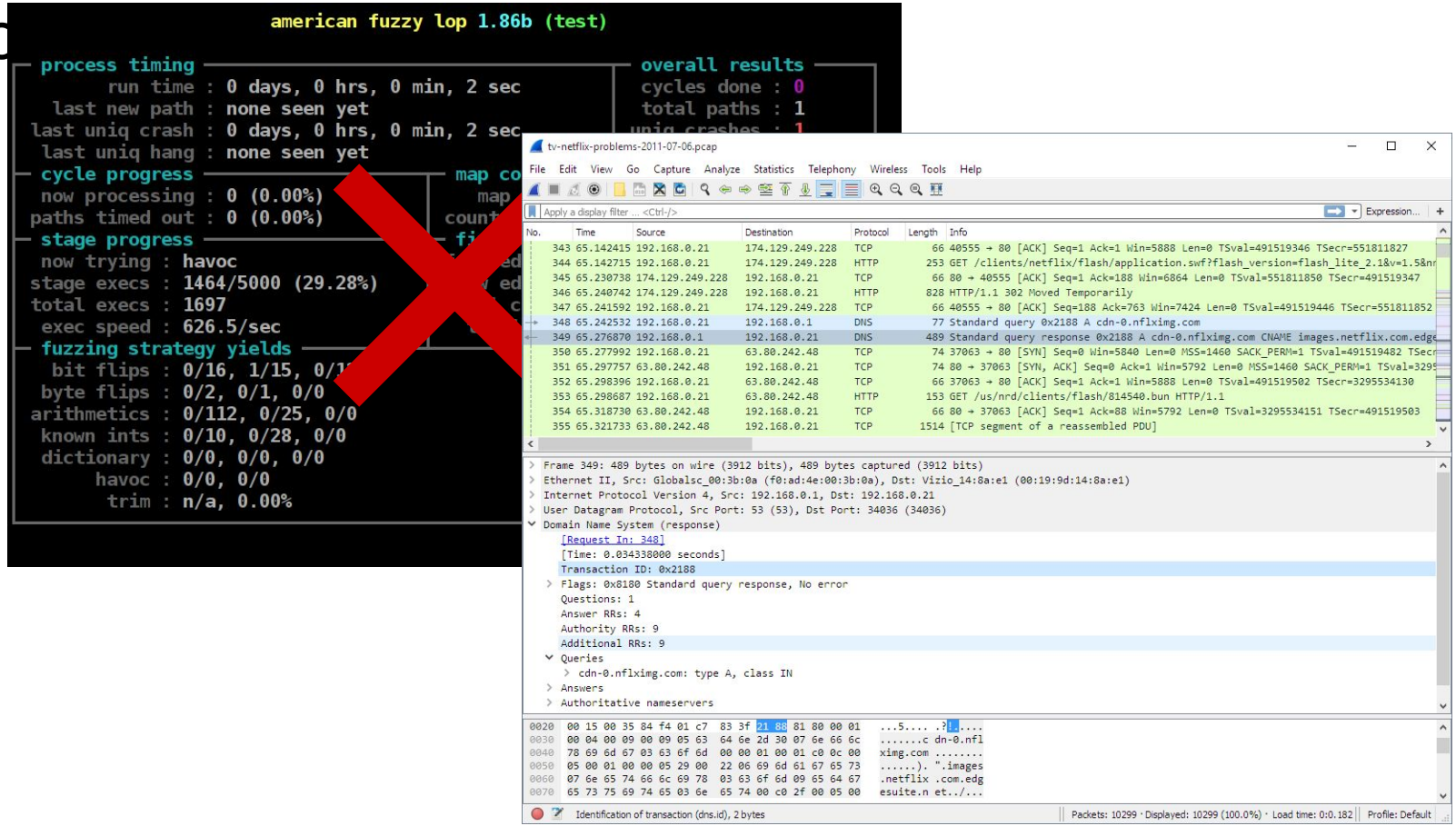
- rability detection technique
- guide its fuzzing process
- guide its fuzzing process
- ers
- ave different **blockers**
- **nce**

Mo

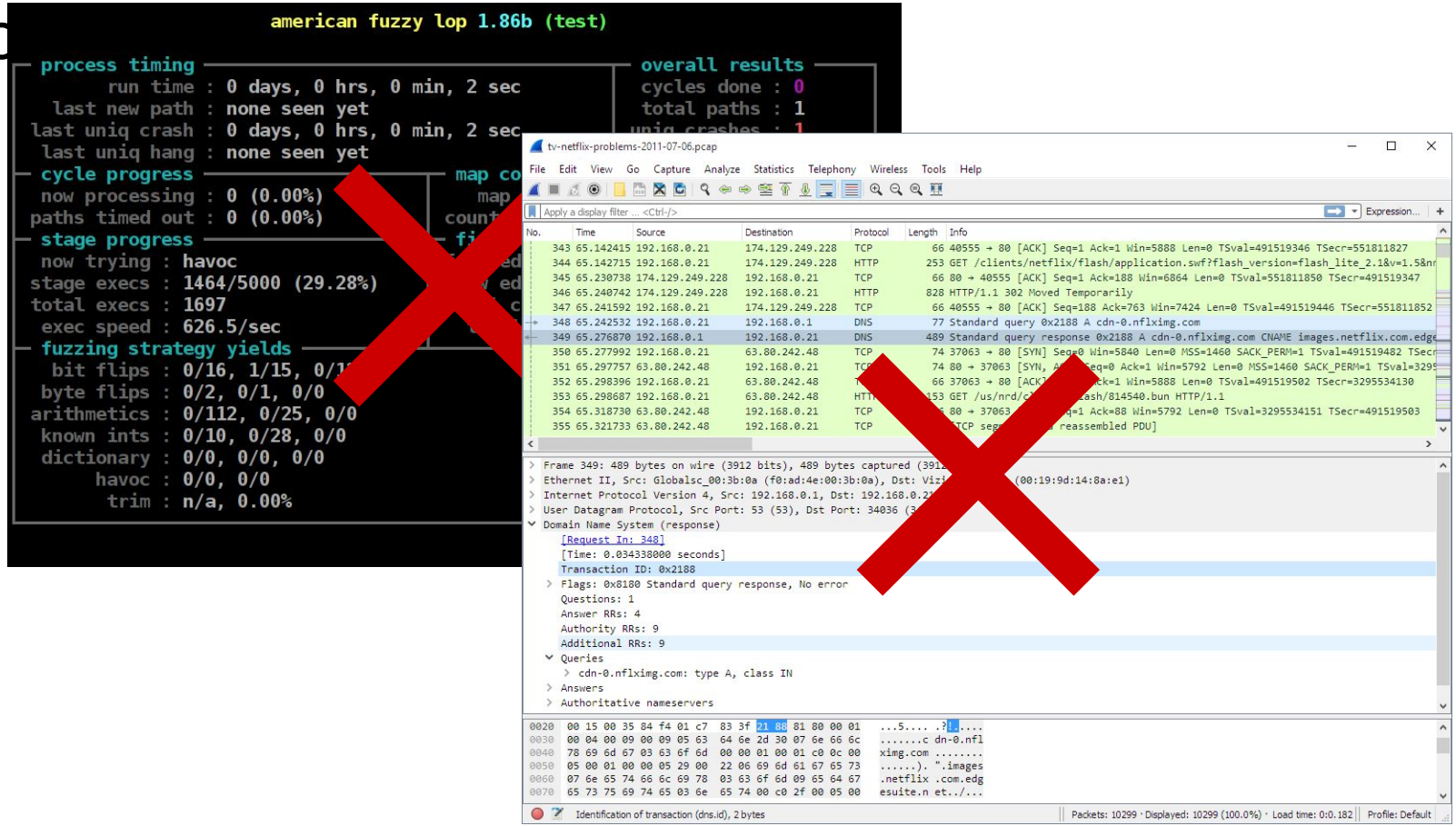


- rability detection technique
- guide its fuzzing process
- guide its fuzzing process
- ers
- ave different **blockers**
- **nce**

Mo


american fuzzy lop 1.86b (test)
process timing
run time : 0 days, 0 hrs, 0 min, 2 sec
last new path : none seen yet
last uniq crash : 0 days, 0 hrs, 0 min, 2 sec
last uniq hang : none seen yet
overall results
cycles done : 0
total paths : 1
cycle progress
now processing : 0 (0.00%)
paths timed out : 0 (0.00%)
stage progress
now trying : havoc
stage execs : 1464/5000 (29.28%)
total execs : 1697
exec speed : 626.5/sec
fuzzing strategy yields
bit flips : 0/16, 1/15, 0/1
byte flips : 0/2, 0/1, 0/0
arithmetics : 0/112, 0/25, 0/0
known ints : 0/10, 0/28, 0/0
dictionary : 0/0, 0/0, 0/0
havoc : 0/0, 0/0
trim : n/a, 0.00%
tv-netflix-problems-2011-07-06.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
343 65.142415 192.168.0.21 174.129.249.228 TCP 66 40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
344 65.142715 192.168.0.21 174.129.249.228 HTTP 253 GET /clients/netflix/flash/application.swf?flash_version=flash_lite_2.1&v=1.5&nr
345 65.230738 174.129.249.228 192.168.0.21 TCP 66 80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
346 65.240742 174.129.249.228 192.168.0.21 HTTP 828 HTTP/1.1 302 Moved Temporarily
347 65.241592 192.168.0.21 174.129.249.228 TCP 66 40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
348 65.242532 192.168.0.21 192.168.0.1 DNS 77 Standard query 0x2188 A cdn-0.nflximg.com
349 65.276870 192.168.0.1 192.168.0.21 DNS 489 Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
350 65.277992 192.168.0.21 63.80.242.48 TCP 74 37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
351 65.297757 63.80.242.48 192.168.0.21 TCP 74 80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
352 65.298396 192.168.0.21 63.80.242.48 TCP 66 37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
353 65.298687 192.168.0.21 63.80.242.48 HTTP 153 GET /us/nrd/clients/flash/814540.bun HTTP/1.1
354 65.318730 63.80.242.48 192.168.0.21 TCP 66 80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
355 65.321733 63.80.242.48 192.168.0.21 TCP 1514 [TCP segment of a reassembled PDU]
Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
Domain Name System (response)
[Request In: 348]
[Time: 0.034338000 seconds]
Transaction ID: 0x2188
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 4
Authority RRs: 9
Additional RRs: 9
Queries
cdn-0.nflximg.com: type A, class IN
Answers
Authoritative nameservers
0020 00 15 00 35 84 f4 01 c7 83 3f 21 88 81 80 00 01 ...5..... .?!.....
0030 00 04 00 09 00 09 05 63 64 6e 2d 30 07 6e 66 6c ....c dn-0.nfl
0040 78 69 6d 67 03 63 6f 6d 00 00 01 00 01 c0 0c 00 ximg.com ....
0050 05 00 01 00 00 05 29 00 22 06 69 6d 61 67 65 73 ......). ".images
0060 07 6e 65 74 66 6c 69 78 03 63 6f 6d 09 65 64 67 .netflix .com.edg
0070 65 73 75 69 74 65 03 6e 65 74 00 c0 2f 00 05 00 esuite.n et../...
Identification of transaction (dns.id), 2 bytes
Packets: 10299 · Displayed: 10299 (100.0%) · Load time: 0:0.182
Profile: Default

Mo


american fuzzy lop 1.86b (test)
process timing
run time : 0 days, 0 hrs, 0 min, 2 sec
last new path : none seen yet
last uniq crash : 0 days, 0 hrs, 0 min, 2 sec
last uniq hang : none seen yet
cycle progress
now processing : 0 (0.00%)
paths timed out : 0 (0.00%)
stage progress
now trying : havoc
stage execs : 1464/5000 (29.28%)
total execs : 1697
exec speed : 626.5/sec
fuzzing strategy yields
bit flips : 0/16, 1/15, 0/1
byte flips : 0/2, 0/1, 0/0
arithmetics : 0/112, 0/25, 0/0
known ints : 0/10, 0/28, 0/0
dictionary : 0/0, 0/0, 0/0
havoc : 0/0, 0/0
trim : n/a, 0.00%
overall results
cycles done : 0
total paths : 1
tv-netflix-problems-2011-07-06.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length Info
343 65.142415 192.168.0.21 174.129.249.228 TCP 66 40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
344 65.142715 192.168.0.21 174.129.249.228 HTTP 253 GET /clients/netflix/flash/application.swf?flash_version=flash_lite_2.1&v=1.5&nr
345 65.230738 174.129.249.228 192.168.0.21 TCP 66 80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
346 65.240742 174.129.249.228 192.168.0.21 HTTP 828 HTTP/1.1 302 Moved Temporarily
347 65.241592 192.168.0.21 174.129.249.228 TCP 66 40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
348 65.242532 192.168.0.21 192.168.0.1 DNS 77 Standard query 0x2188 A cdn-0.nflximg.com
349 65.276870 192.168.0.1 192.168.0.21 DNS 489 Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
350 65.277992 192.168.0.21 63.80.242.48 TCP 74 37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
351 65.297757 63.80.242.48 192.168.0.21 TCP 74 80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
352 65.298396 192.168.0.21 63.80.242.48 TCP 66 37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
353 65.298687 192.168.0.21 63.80.242.48 HTTP 153 GET /us/nrd/clients/flash/814540.bun HTTP/1.1
354 65.318730 63.80.242.48 192.168.0.21 TCP 80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
355 65.321733 63.80.242.48 192.168.0.21 TCP [TCP segment of a reassembled PDU]
Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912
Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizi (00:19:9d:14:8a:e1)
Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34
Domain Name System (response)
[Request In: 348]
[Time: 0.034338000 seconds]
Transaction ID: 0x2188
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 4
Authority RRs: 9
Additional RRs: 9
Queries
cdn-0.nflximg.com: type A, class IN
Answers
Authoritative nameservers
0020 00 15 00 35 84 f4 01 c7 83 3f 21 88 81 80 00 01 ...5..... .?!.....
0030 00 04 00 09 00 09 05 63 64 6e 2d 30 07 6e 66 6c ....c dn-0.nfl
0040 78 69 6d 67 03 63 6f 6d 00 00 01 00 01 c0 0c 00 ximg.com ....
0050 05 00 01 00 00 05 29 00 22 06 69 6d 61 67 65 73 ......). ".images
0060 07 6e 65 74 66 6c 69 78 03 63 6f 6d 09 65 64 67 .netflix .com.edg
0070 65 73 75 69 74 65 03 6e 65 74 00 c0 2f 00 05 00 esuite.n et../...
Identification of transaction (dns.id), 2 bytes
Packets: 10299 · Displayed: 10299 (100.0%) · Load time: 0:0.182
Profile: Default

StateFuzzVis

- **Visualizer** to help identify blockers in network fuzzing

StateFuzzVis

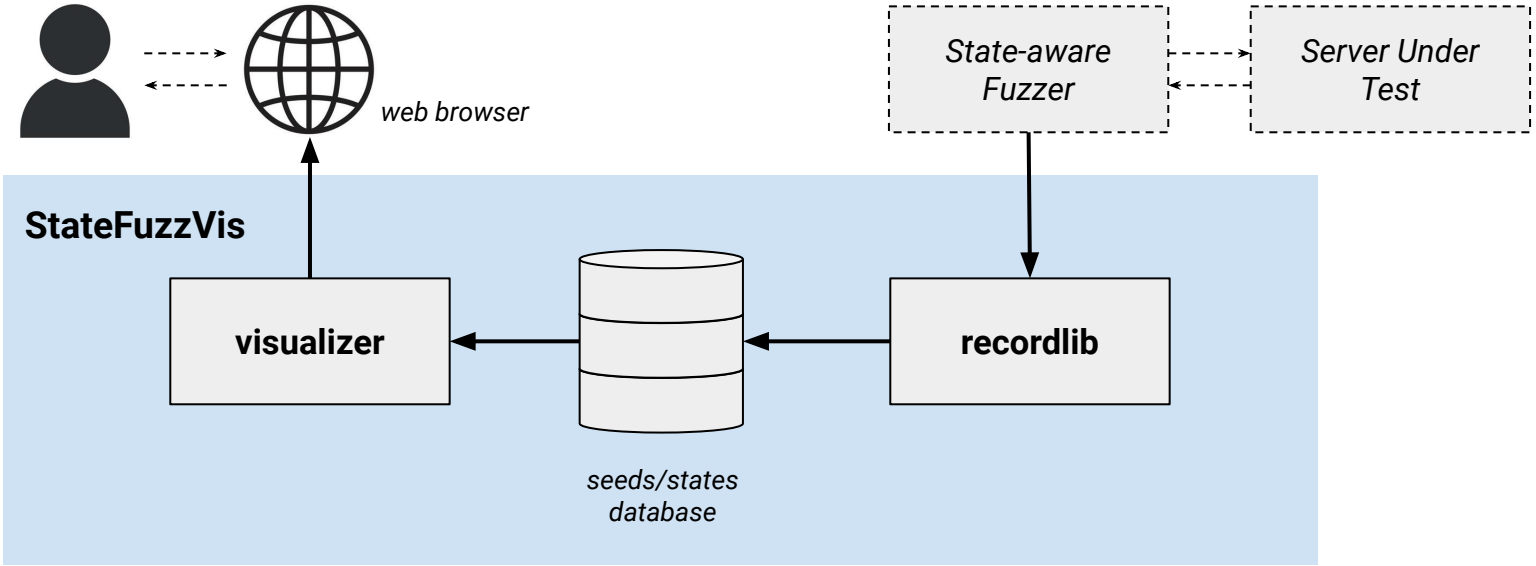
- **Visualizer** to help identify blockers in network fuzzing
- Designed to be **compatible** with **SOTA state-aware network fuzzers**

StateFuzzVis

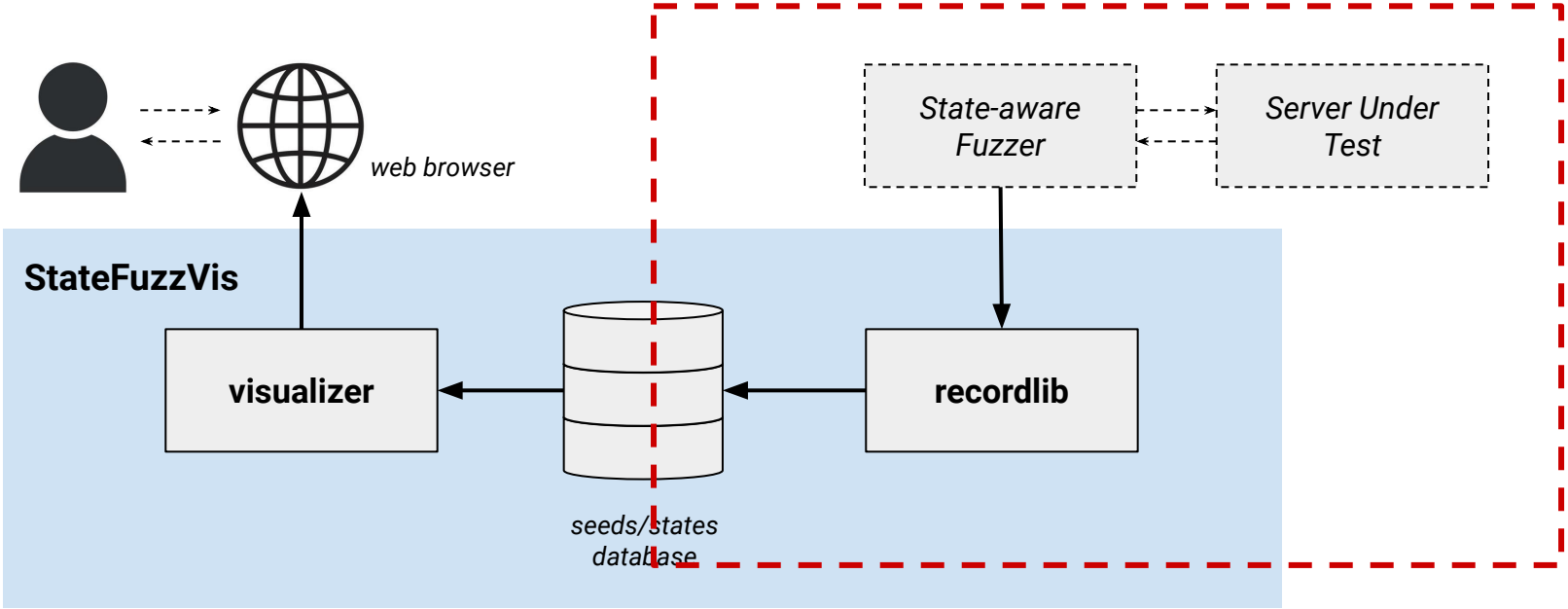
- **Visualizer** to help identify blockers in network fuzzing
- Designed to be **compatible** with **SOTA state-aware network fuzzers**
- Visualizes **state machine** of state-aware fuzzing **on-the-fly**

StateFuzzVis Demo

Technical Details



Technical Details



Technical Details

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

Technical Details

State identification

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

Technical Details

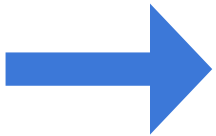
State identification AFLnet

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

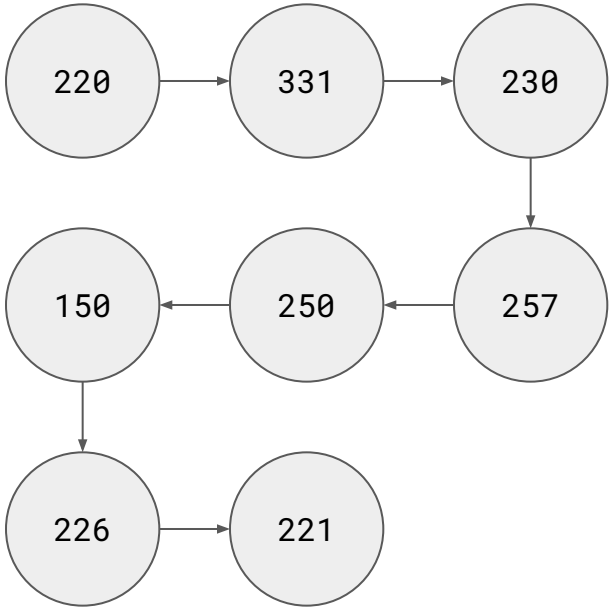
Technical Details

State identification AFLnet

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```



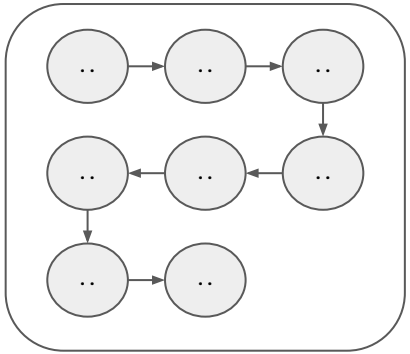
State exercised information



Technical Details

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

Seed info from fuzzer

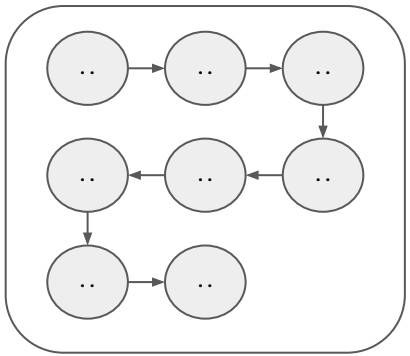


State info from fuzzer

Technical Details

```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

Seed info from fuzzer



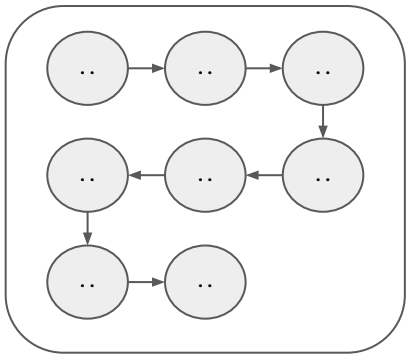
State info from fuzzer

recordlib (.c API)

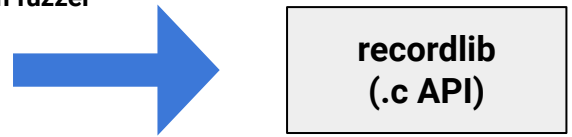
Technical Details

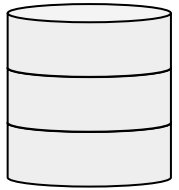
```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

Seed info from fuzzer



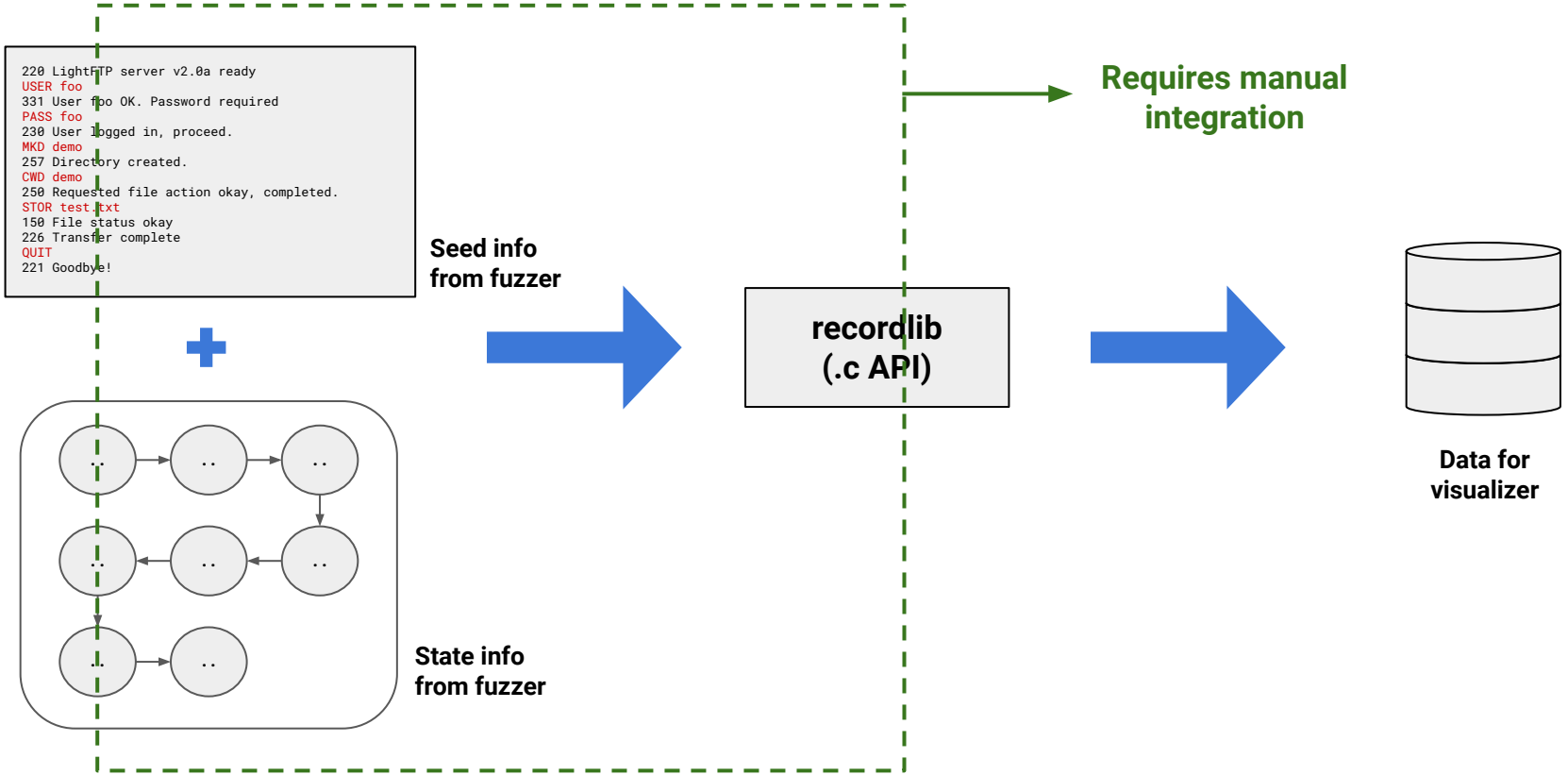
State info from fuzzer





Data for visualizer

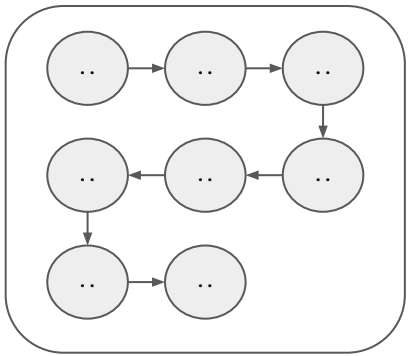
Technical Details



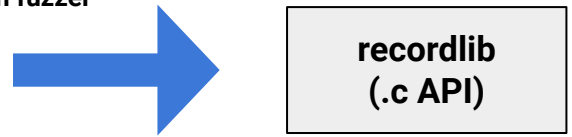
Technical Details

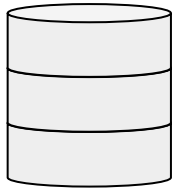
```
220 LightFTP server v2.0a ready
USER foo
331 User foo OK. Password required
PASS foo
230 User logged in, proceed.
MKD demo
257 Directory created.
CWD demo
250 Requested file action okay, completed.
STOR test.txt
150 File status okay
226 Transfer complete
QUIT
221 Goodbye!
```

Seed info from fuzzer



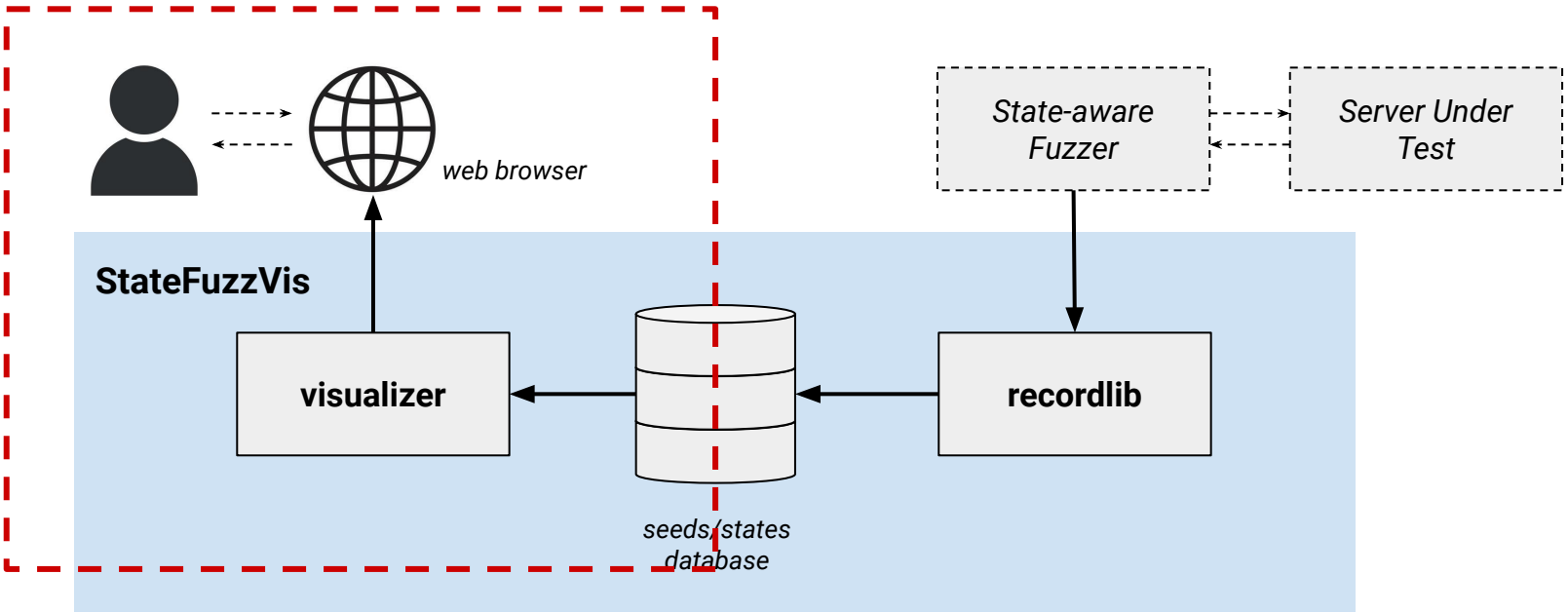
State info from fuzzer



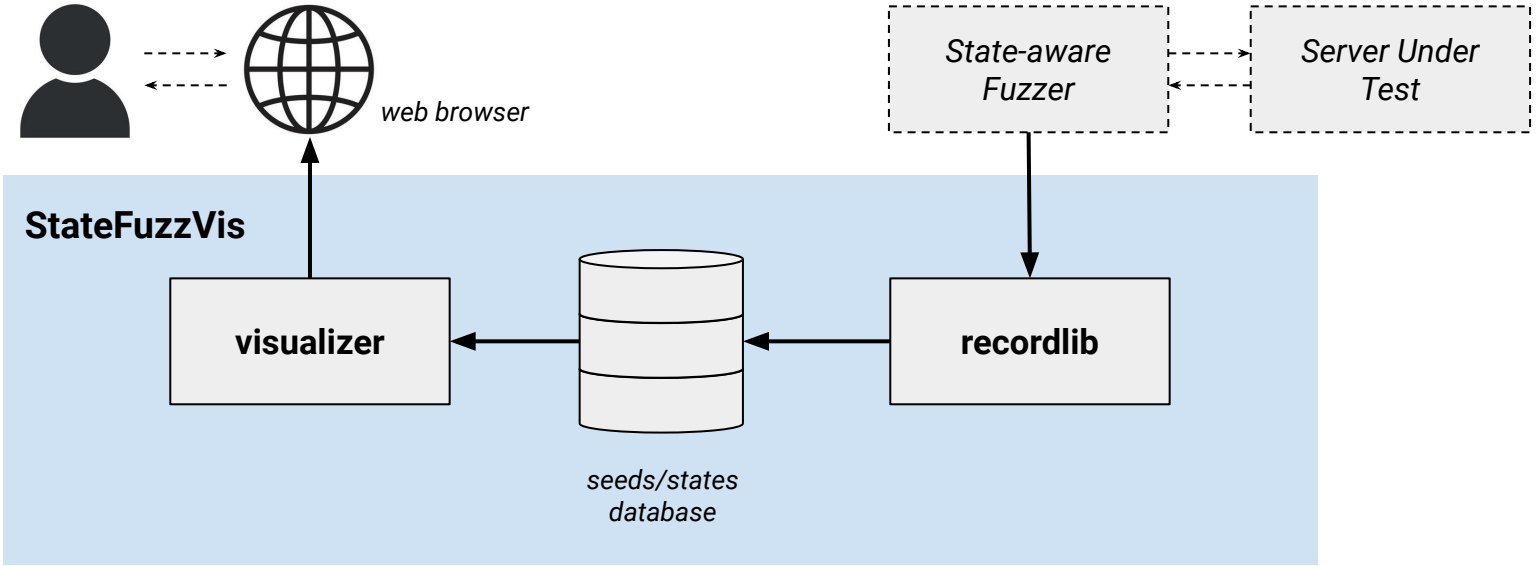


Data for visualizer

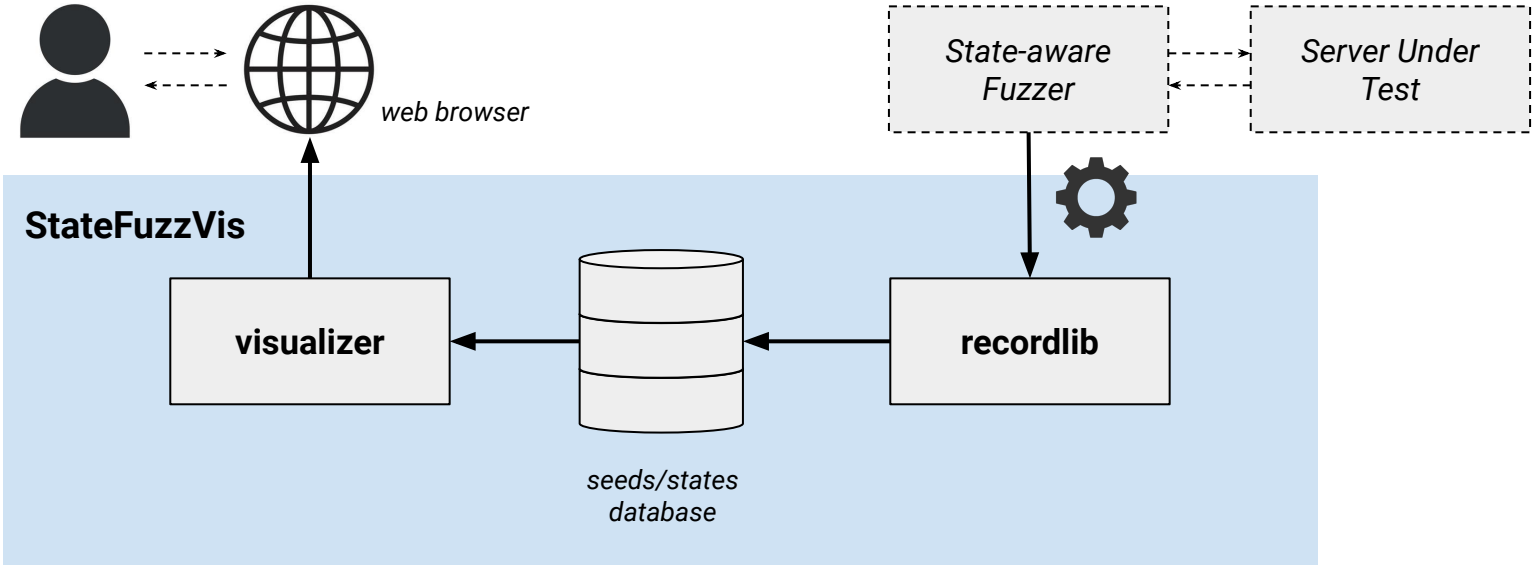
Technical Details



Technical Details



Technical Details



Evaluation (Ease of Integration)

- RQ: How hard is it to integrate **StateFuzzVis** to SOTA state-aware network fuzzers?

Evaluation (Ease of Integration)

- For an amateur fuzzer developer:

Evaluation (Ease of Integration)

- For an amateur fuzzer developer:
  - Integrate **StateFuzzVis** with:
    - AFLnet
    - StateAFL
    - SGFuzz

Evaluation (Ease of Integration)

- For an amateur fuzzer developer:
  - Integrate **StateFuzzVis** with:
    - AFLnet
    - StateAFL

    } ***AFL-based***

    - SGFuzz

    } ***LibFuzzer-based***

Evaluation (Ease of Integration)

- For an amateur fuzzer developer:
  - Integrate **StateFuzzVis** with:
    - AFLnet
    - StateAFL
    - SGFuzz

- Takes <2hr

Evaluation (Ease of Integration)

- For an amateur fuzzer developer:
  - Integrate **StateFuzzVis** with:
    - AFLnet
    - StateAFL
    - SGFuzz

- Takes <2hr
- Needs **66**, **87**, and **17** LOCs
  - Mostly data type reformatting logic

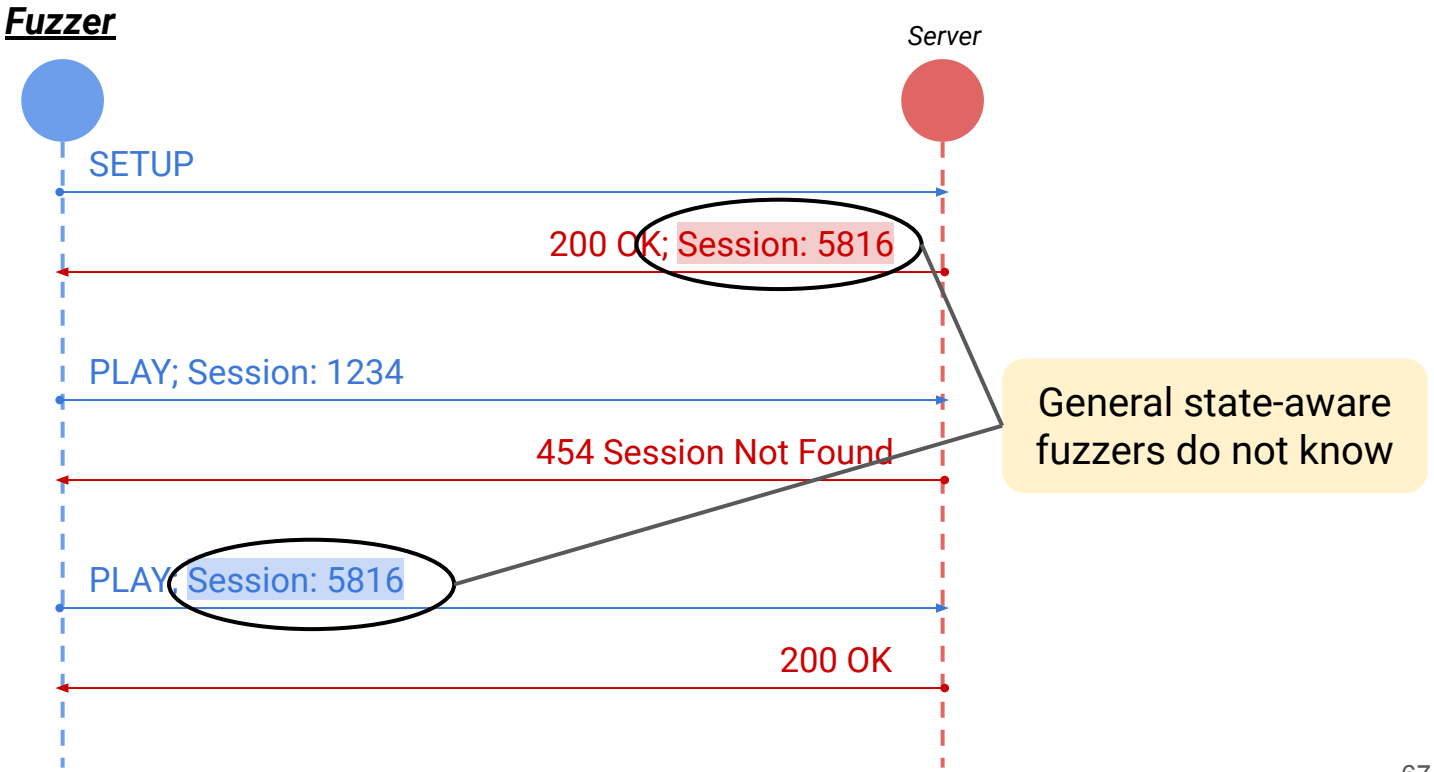
Evaluation (Usefulness)

- RQ: Can **StateFuzzVis** help identify fuzz blockers?

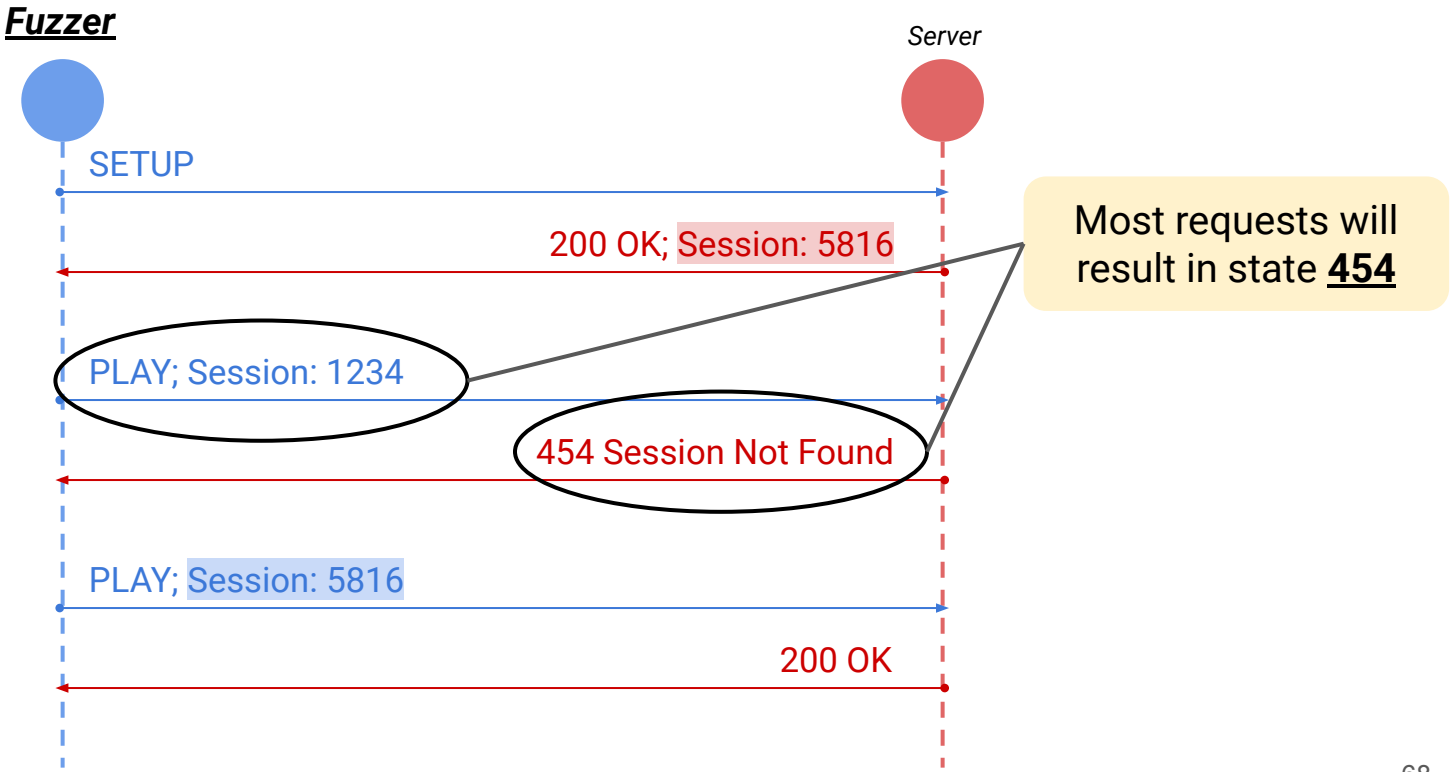
Evaluation (Usefulness)

- Fuzz blocker: Randomly generated session dependency

Blocker (Randomly Generated Session Dependency)

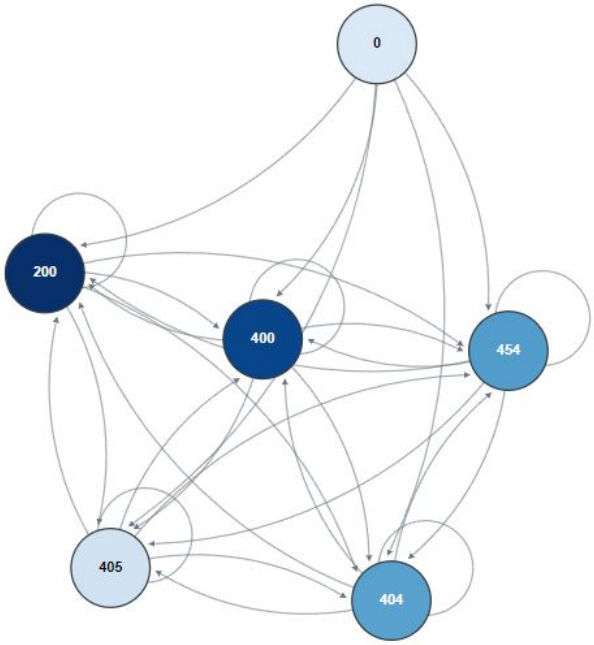


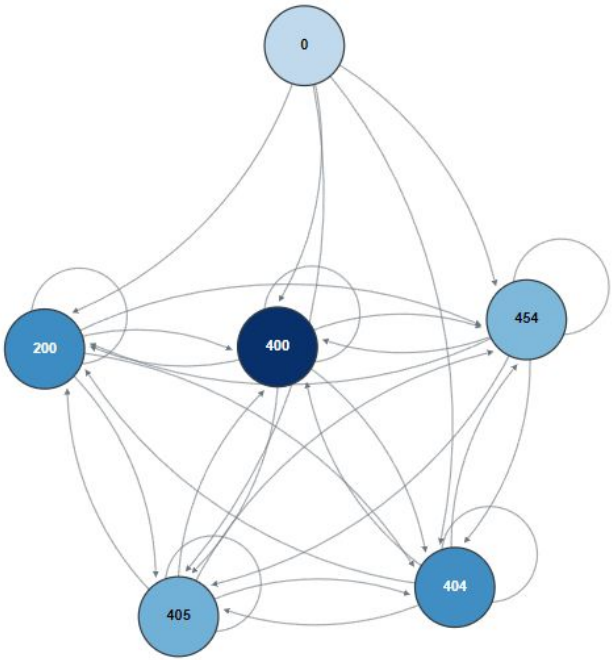
Blocker (Randomly Generated Session Dependency)



Evaluation (Usefulness)

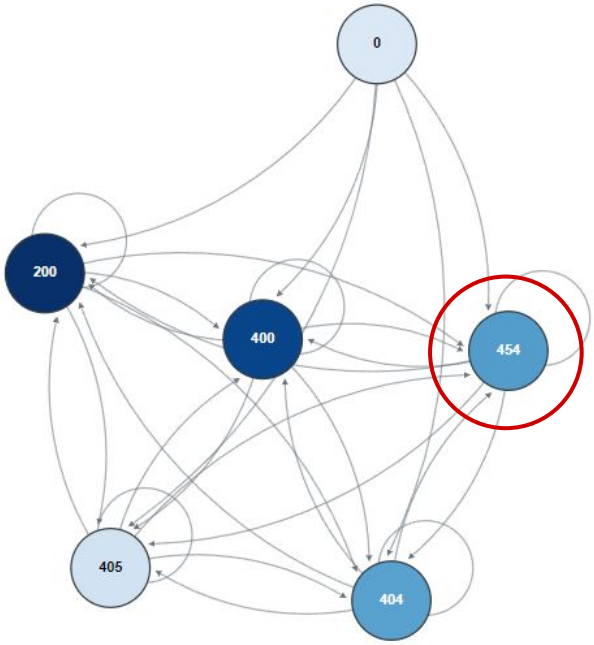
- Fuzz blocker: Randomly generated session dependency

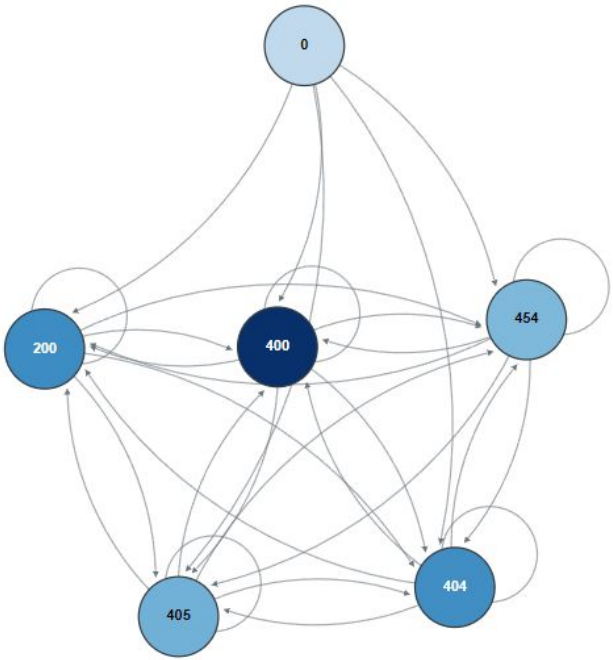




Evaluation (Usefulness)

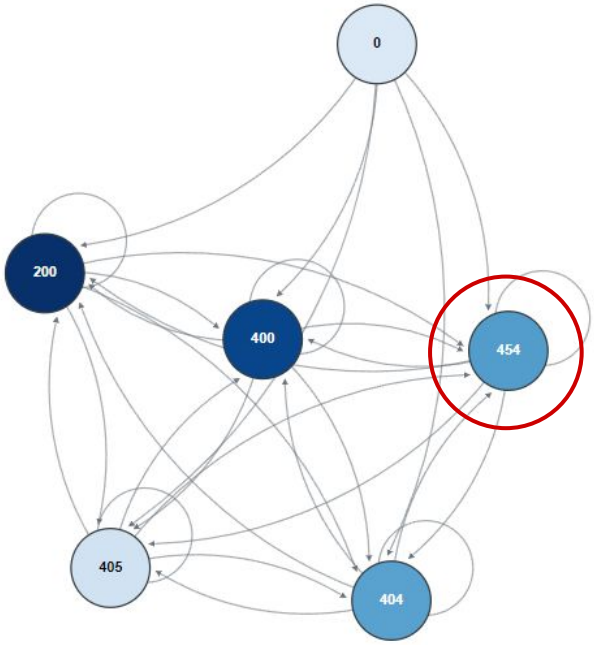
- Fuzz blocker: Randomly generated session dependency

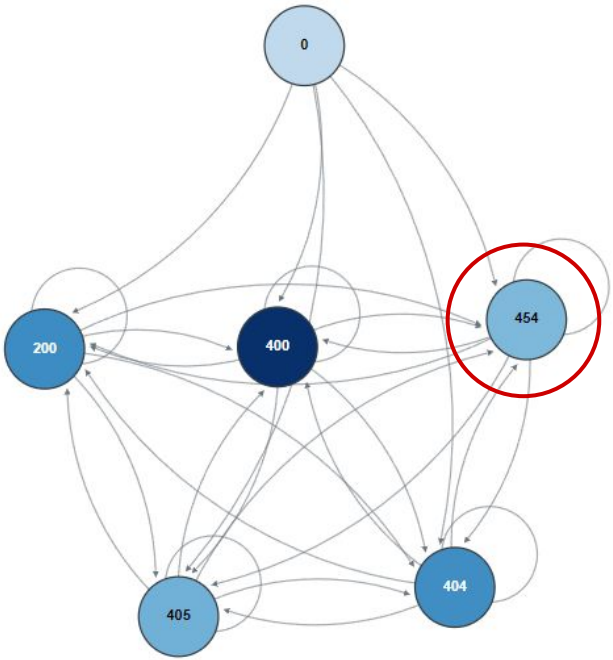




Evaluation (Usefulness)

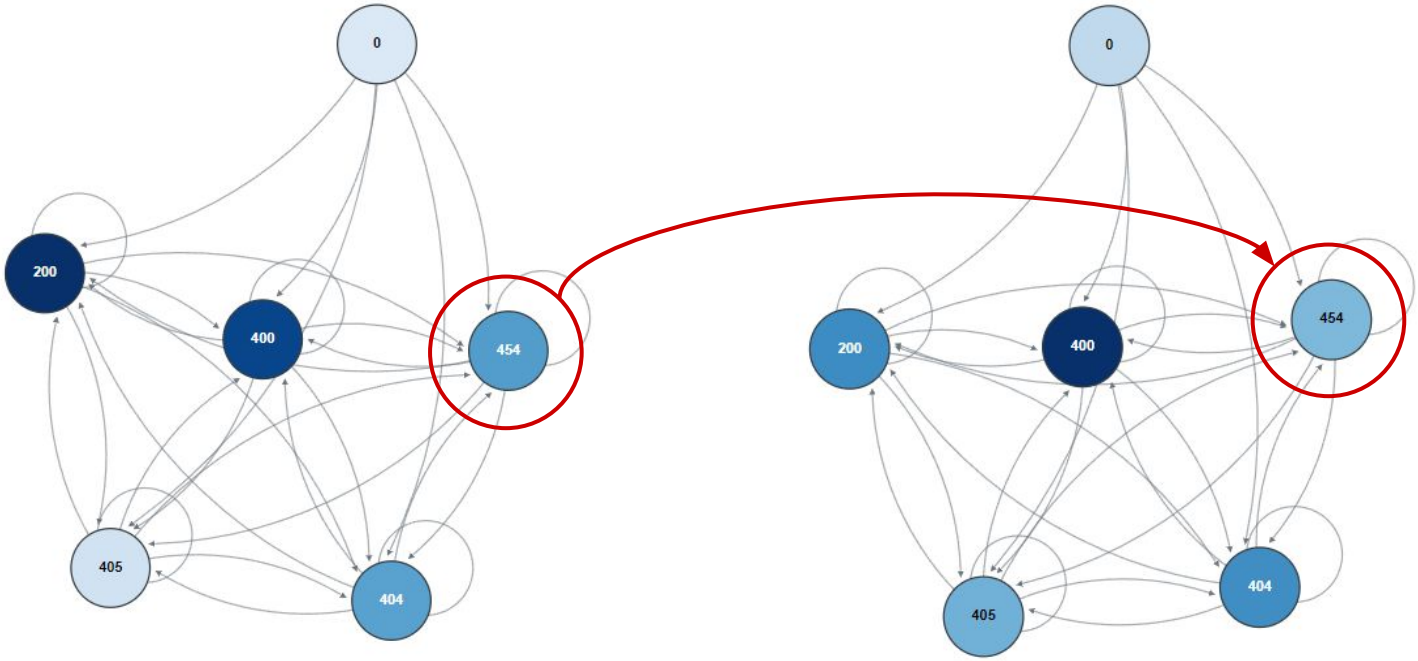
- Fuzz blocker: Randomly generated session dependency





Evaluation (Usefulness)

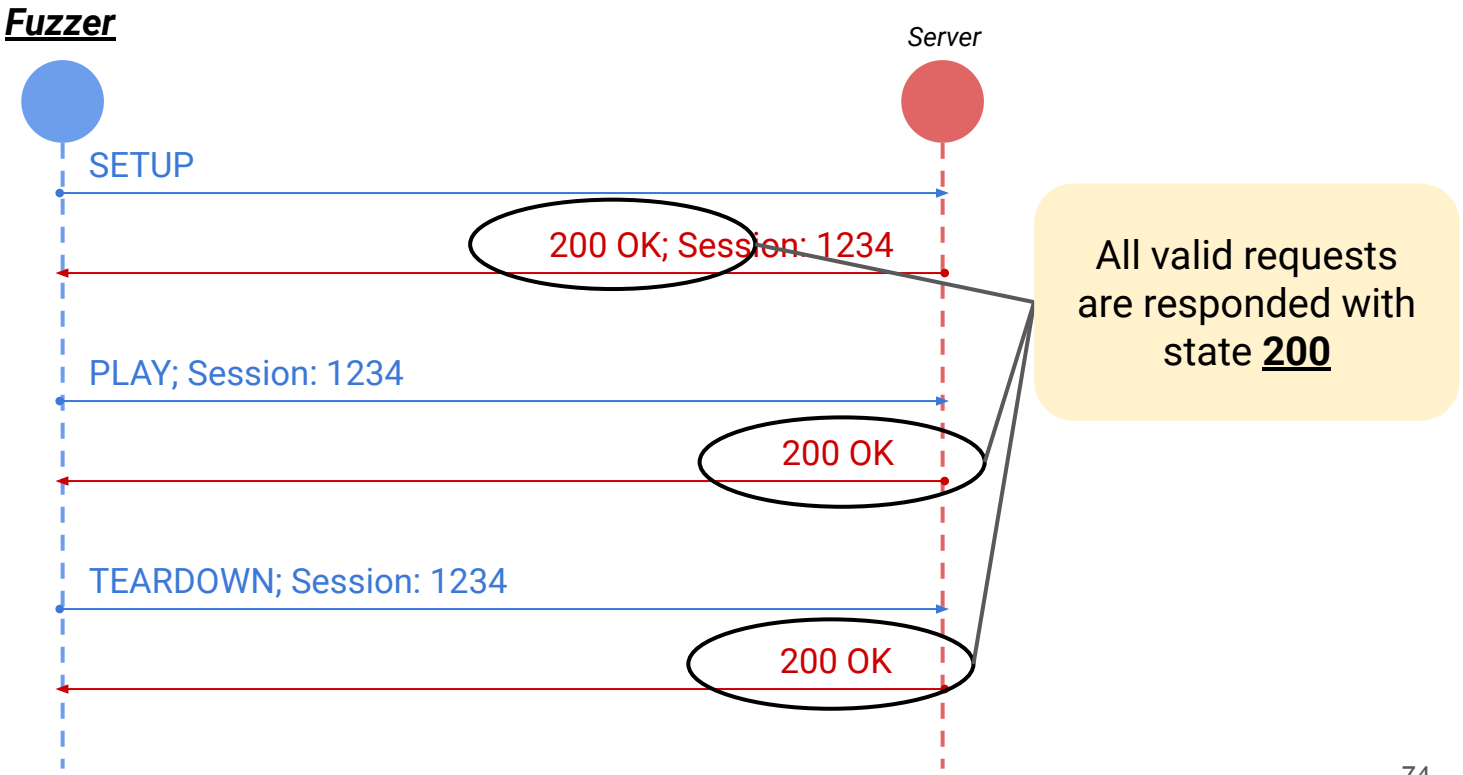
- Fuzz blocker: Randomly generated session dependency



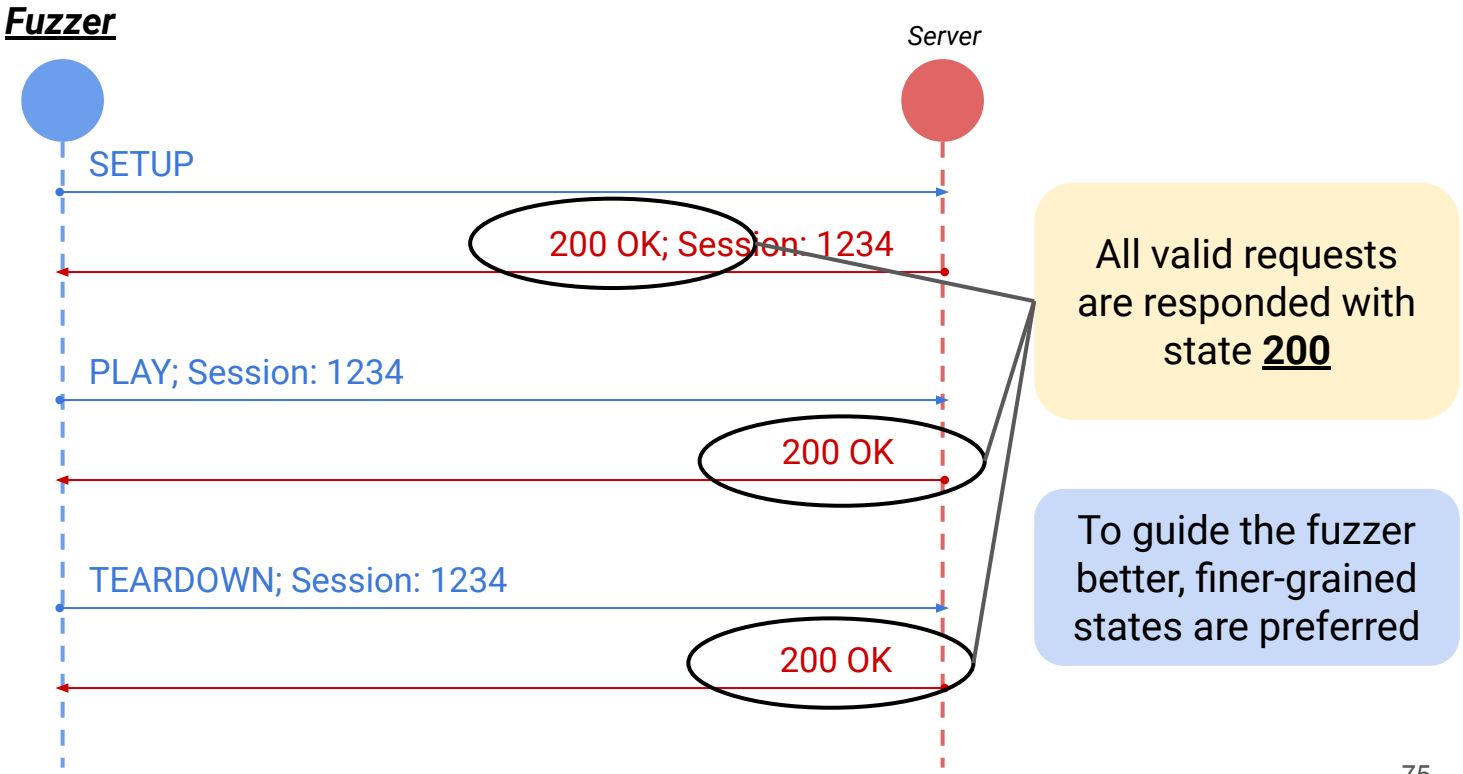
Evaluation (Usefulness)

- Fuzz blocker: One large state for all valid requests

Blocker (One large state for all valid requests)

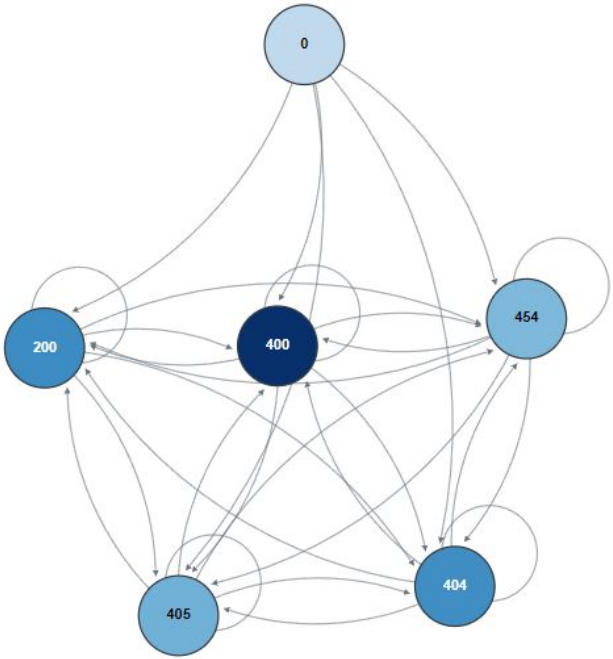


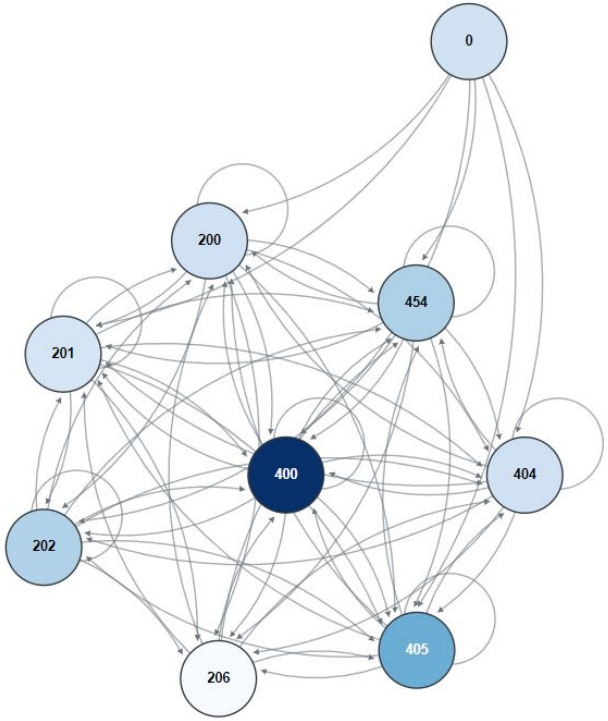
Blocker (One large state for all valid requests)



Evaluation (Usefulness)

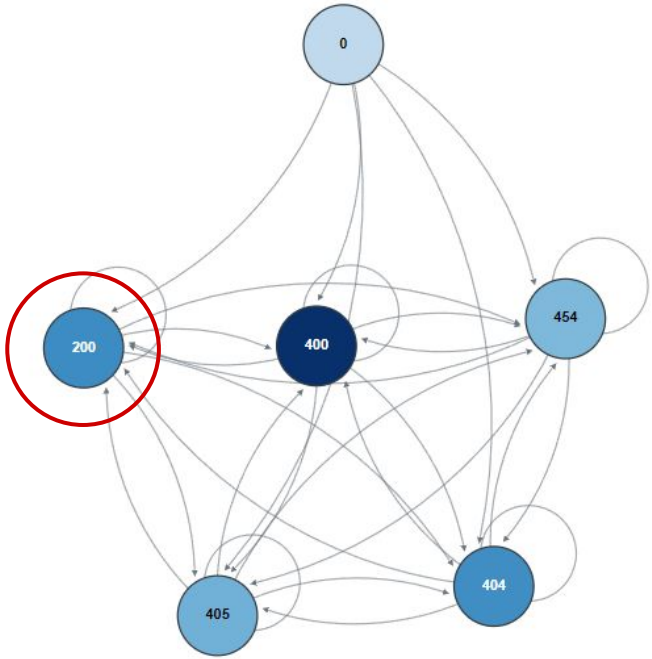
- Fuzz blocker: One large state for all valid requests

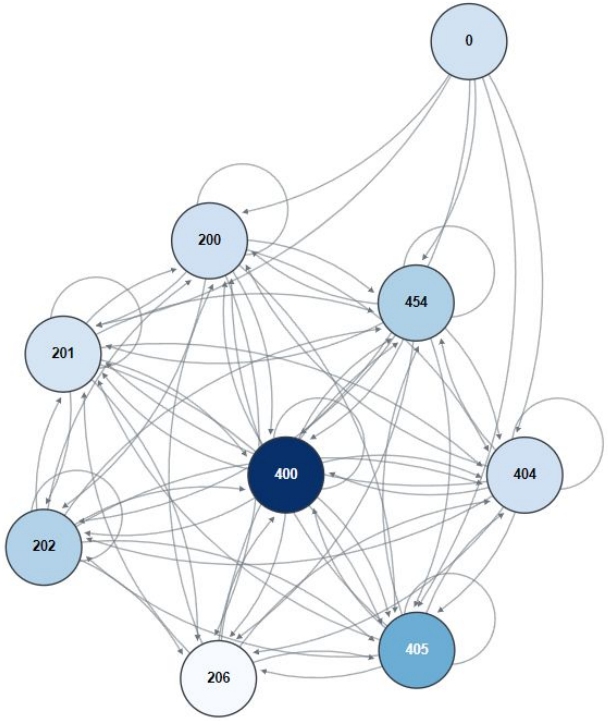




Evaluation (Usefulness)

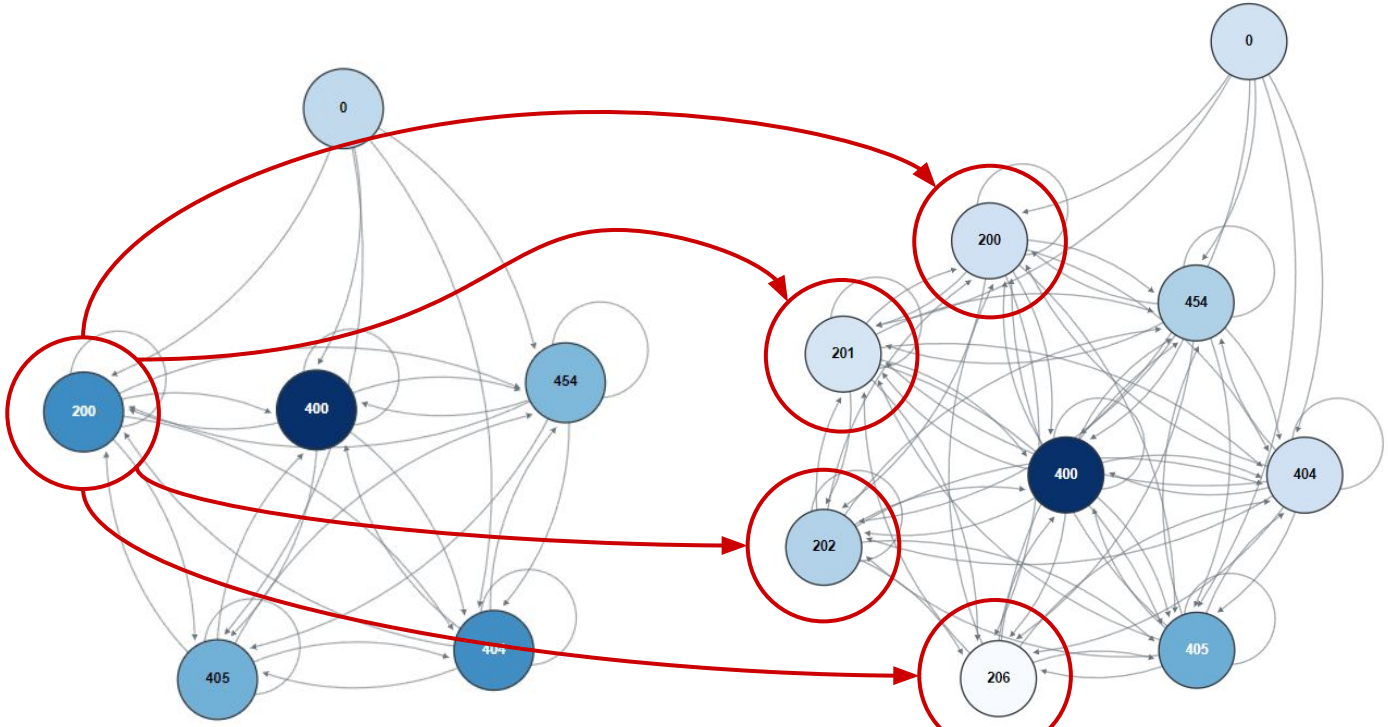
- Fuzz blocker: One large state for all valid requests





Evaluation (Usefulness)

- Fuzz blocker: One large state for all valid requests



Conclusion

- We propose **StateFuzzVis**, a visualizer for state-aware fuzzers
- **StateFuzzVis** was effortlessly integrated to SOTA protocol fuzzers
- **StateFuzzVis** helped identify fuzzing blockers to improve the effectiveness of protocol fuzzing

- Code available: https://github.com/fraglantia/StateFuzzVis

Future works

- Inspect commonalities of network fuzzing blockers
  - Automatic blocker detection/removal
- User-assisted fuzzing
- Thorough evaluation of SOTA stateful fuzzers

Thank you

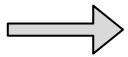
Steve Gustaman
stevegustaman@kaist.ac.kr

Real World Scenario

- In its repo tutorial, AFLNet patches target program manually to decompose states for Live555 RTSP Server*

- State **200** is too large

```
SETUP       -> 200 OK
PLAY        -> 200 OK
TEARDOWN    -> 200 OK
```



```
SETUP       -> 201 OK
PLAY        -> 202 OK
TEARDOWN    -> 206 OK
```

*) https://github.com/aflnet/aflnet/blob/master/tutorials/live555/ceeb4f4_states_decomposed.patch