CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

AmpFuzz: Fuzzing for Amplification DDoS Vulnerabilities

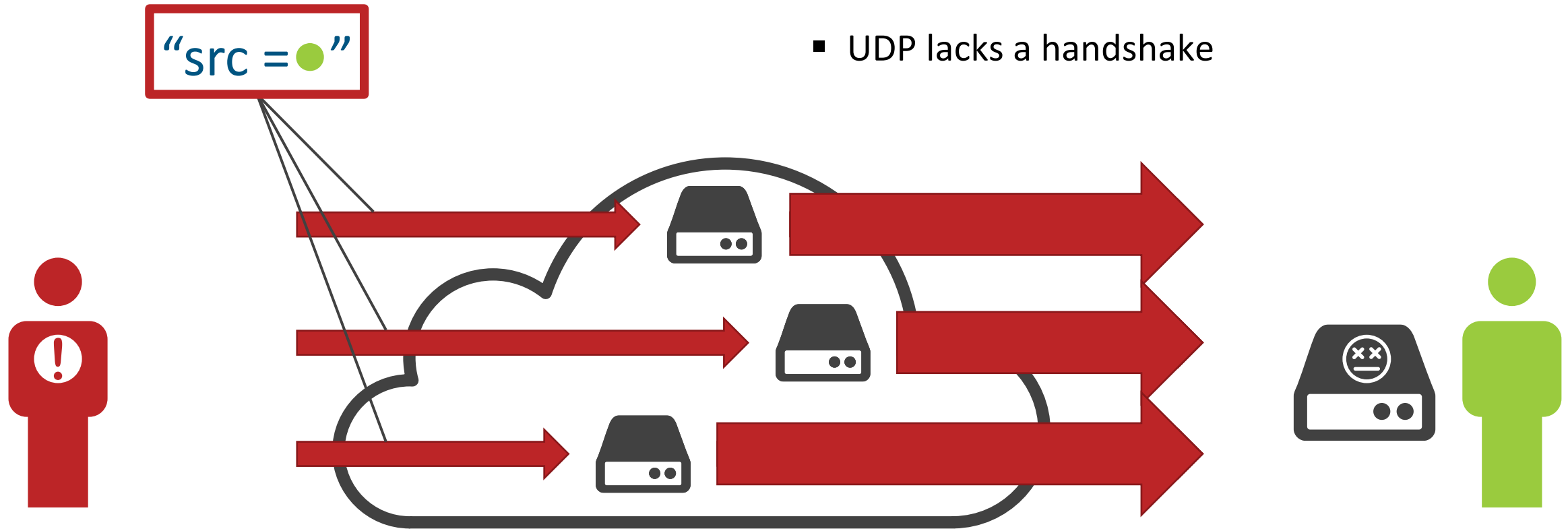
Johannes Krupp
CrowdStrike

Ilya Grishchenko
University of California, Santa Barbara

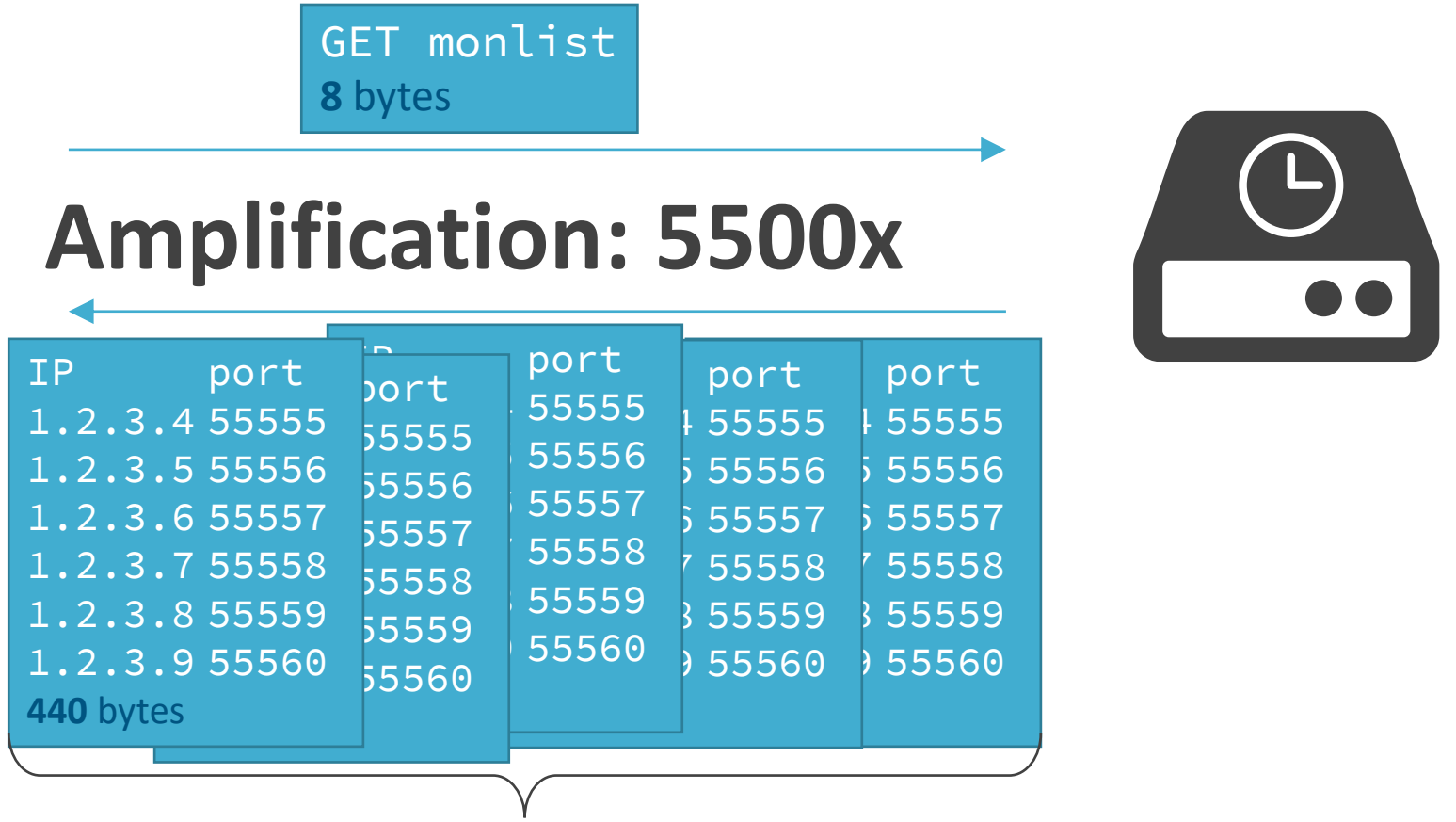
Christian Rossow
CISPA Helmholtz Center for Information Security

Amplification Distributed Denial-of-Service (DDoS)

- IP header non-authenticated
- UDP lacks a handshake

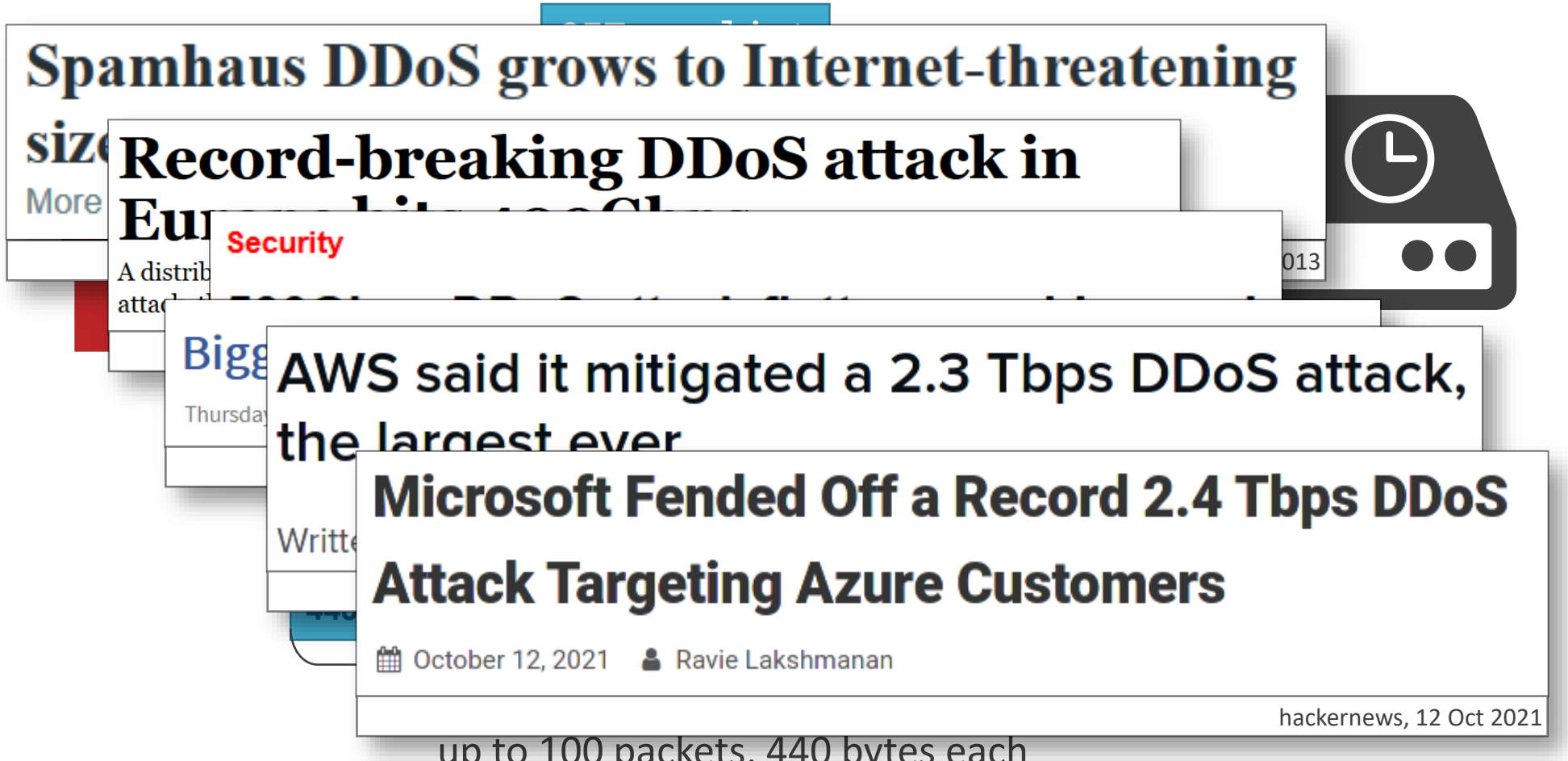


NTP Amplification



up to 100 packets, 440 bytes each
= up to **44.000 bytes**

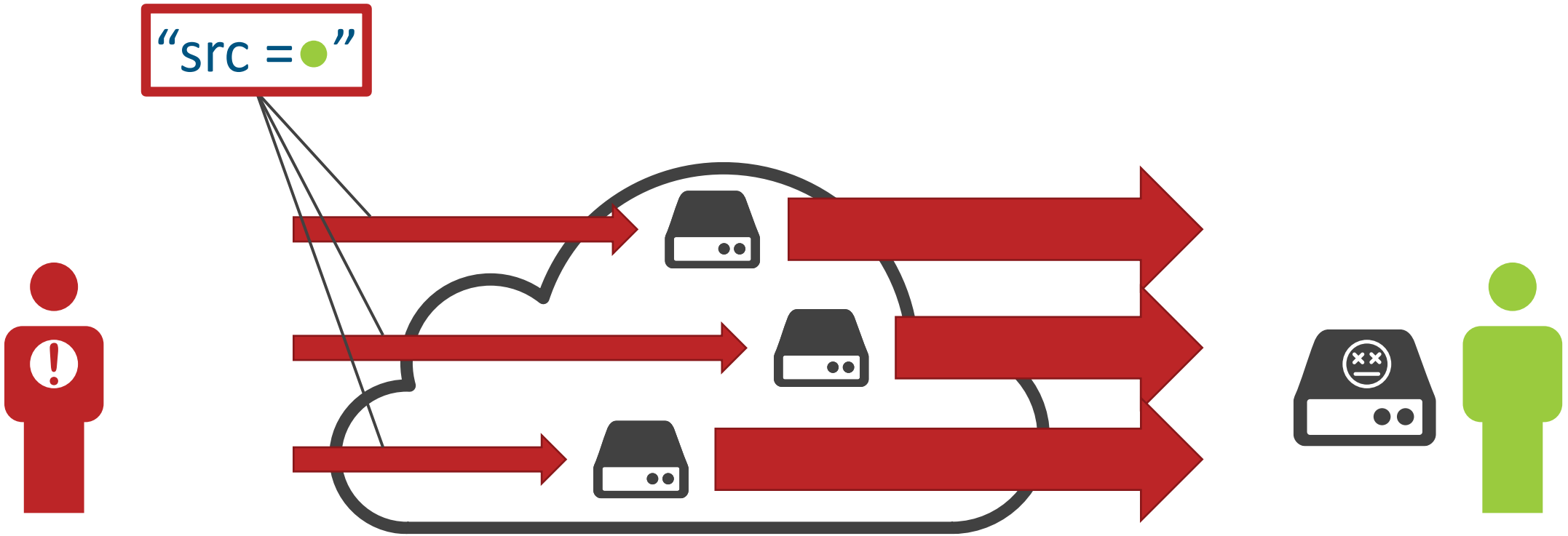
NTP Amplification



up to 100 packets, 440 bytes each

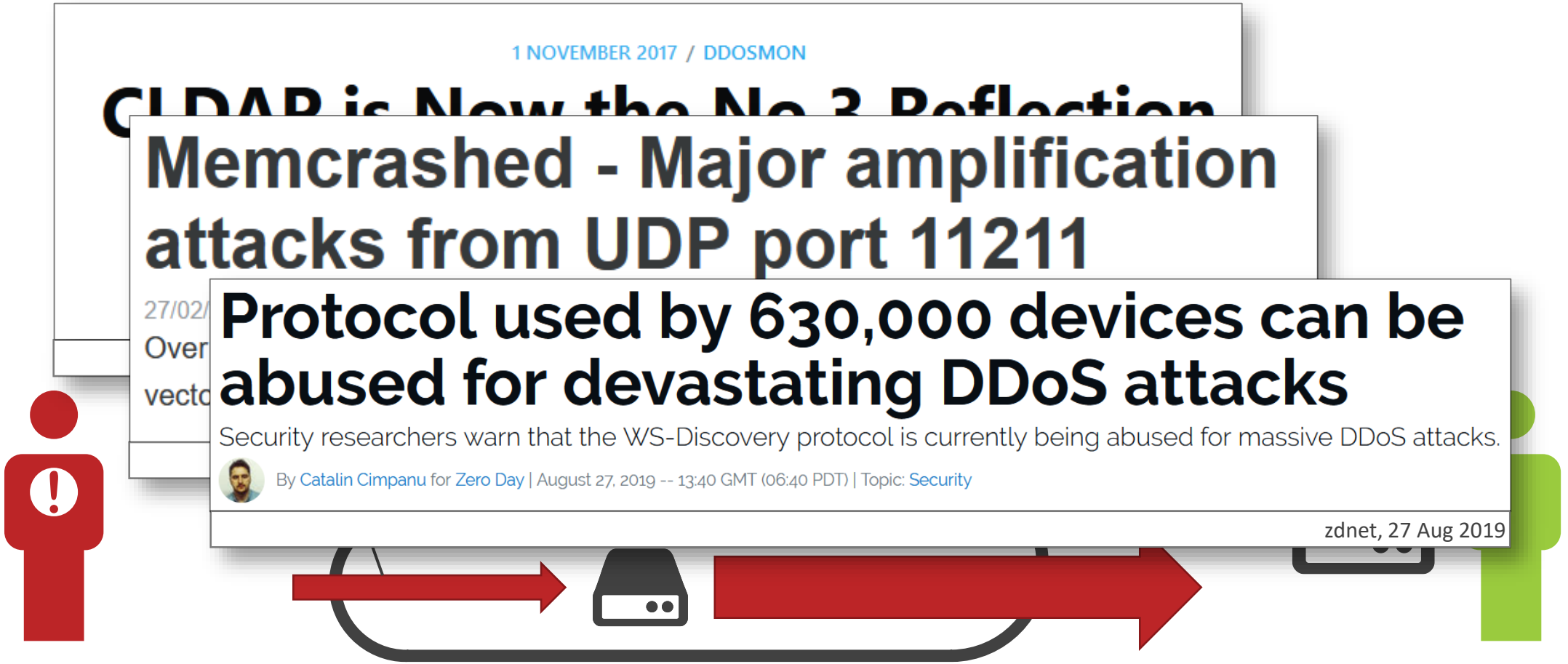
= up to **44.000 bytes**

Amplification DDoS – Challenges



- Large amplification potential
- New vulnerabilities discovered regularly

Amplification DDoS – Challenges



- Large amplification potential
- New vulnerabilities discovered regularly

Amplification DDoS – Challenges

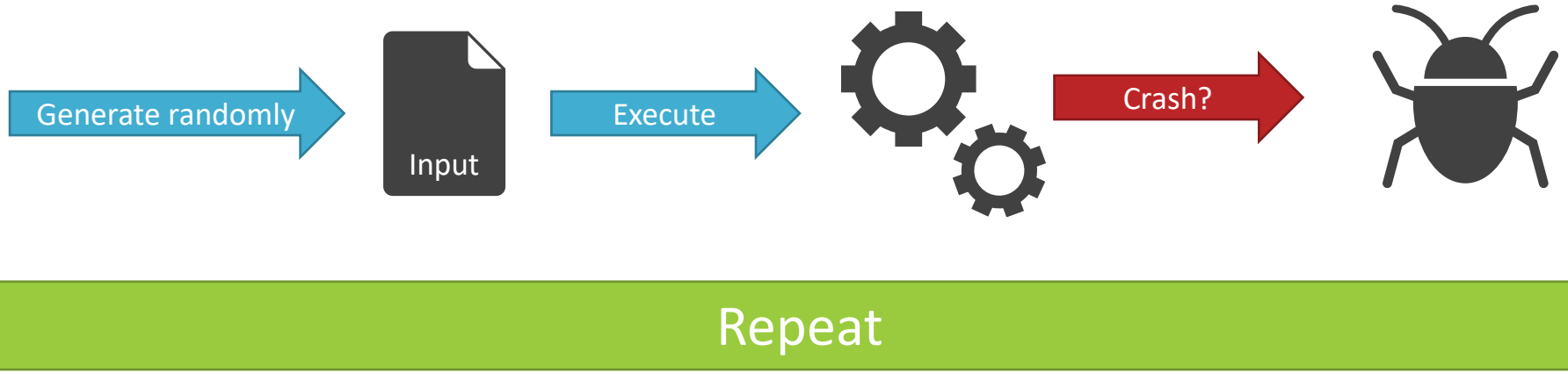
Known vectors found manually or post-mortem

Can we find vulnerable protocols systematically and proactively?

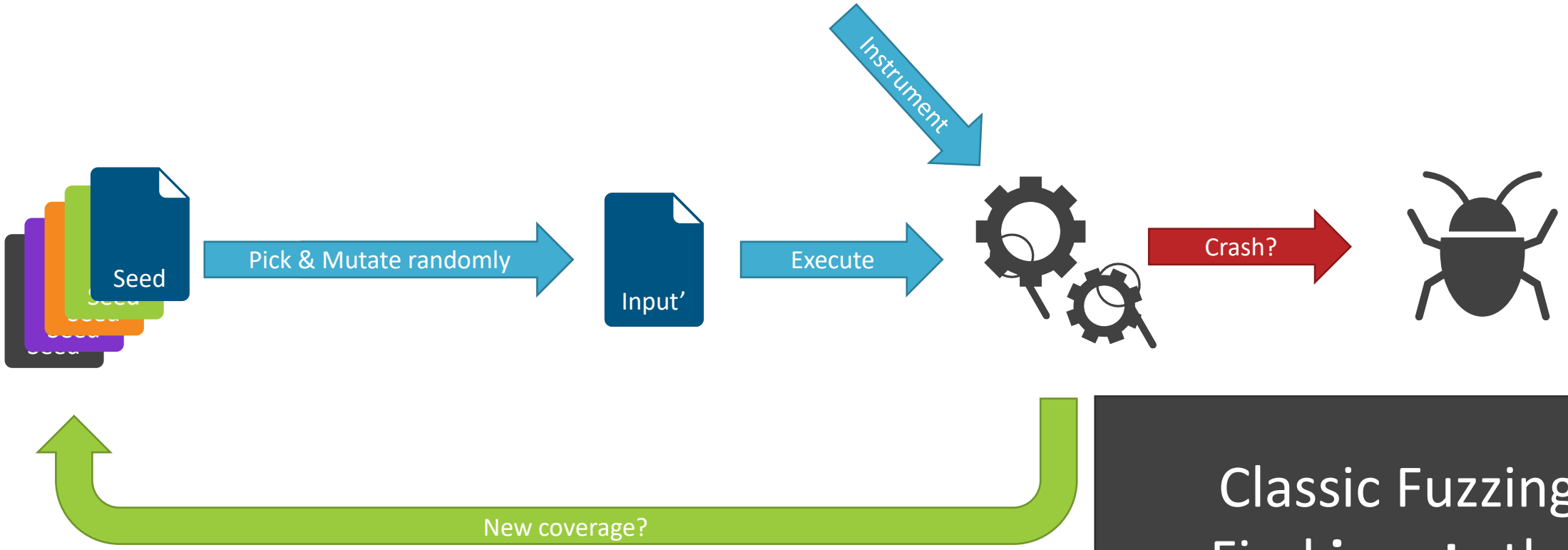
-> Let‘s try fuzzing!

- New vulnerabilities discovered regularly

Fuzzing

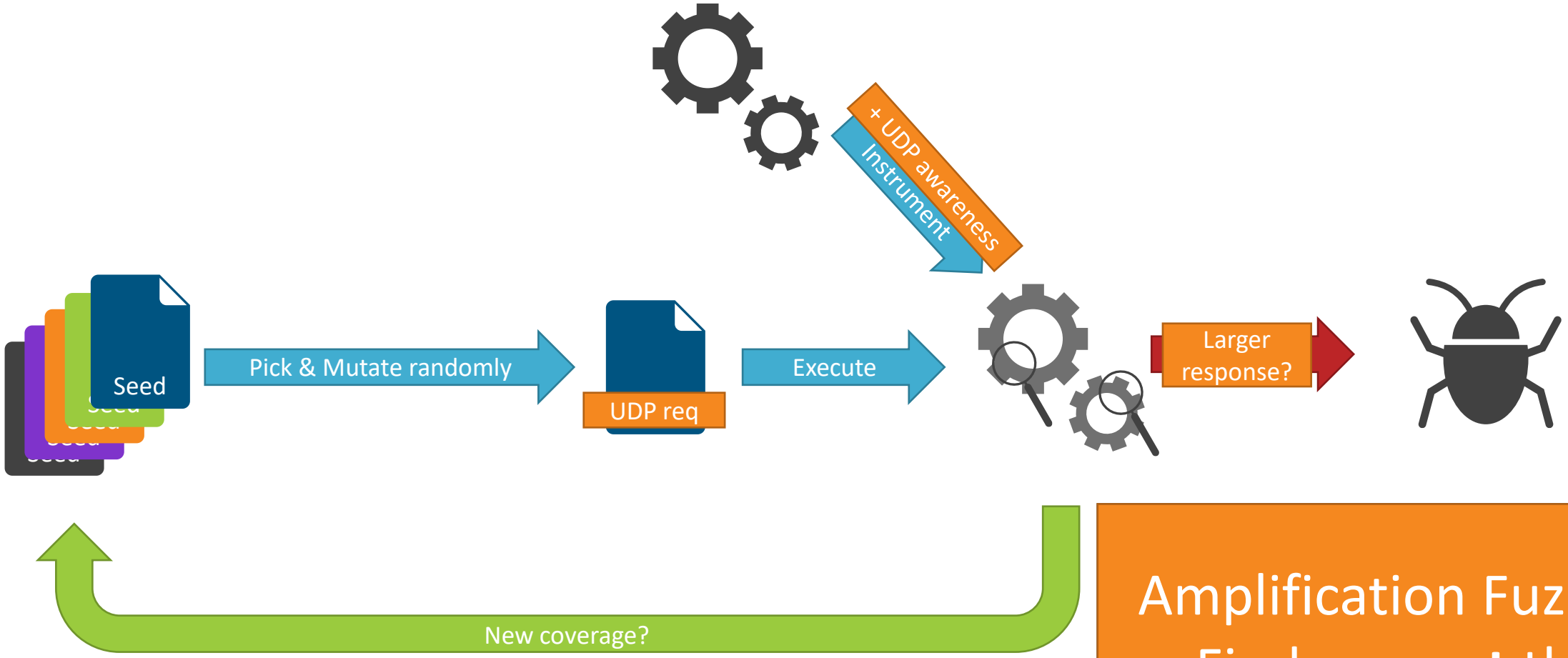


Coverage-Guided Mutation-Based Greybox Fuzzing



Classic Fuzzing:
Find **inputs** that
crash the program

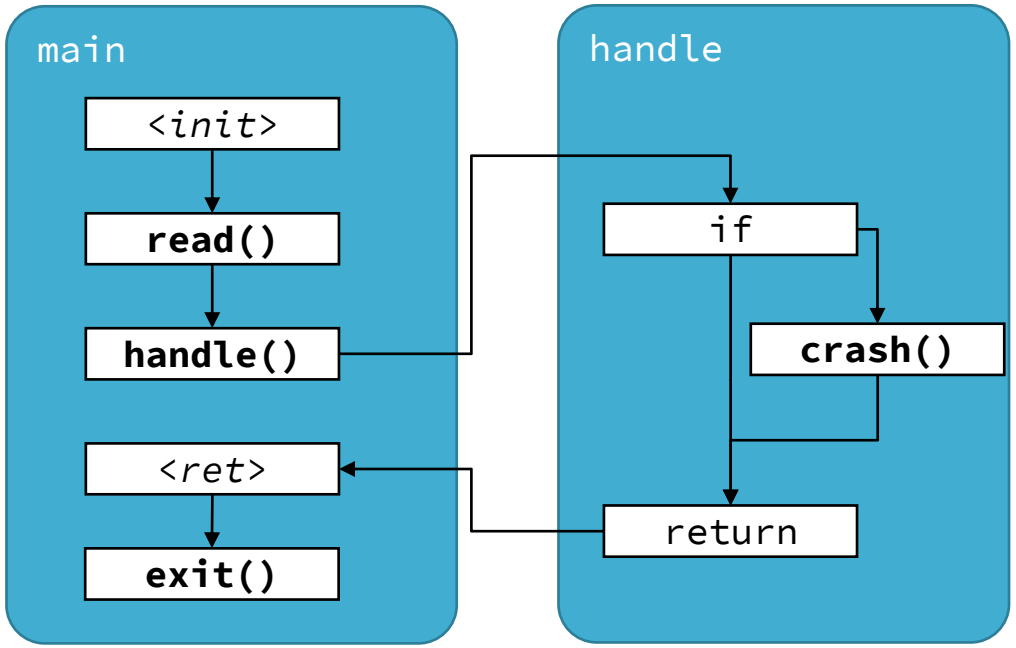
AmpFuzz – Fuzzing for Amplification Vectors



Amplification Fuzzing:
Find **request** that
excite large responses

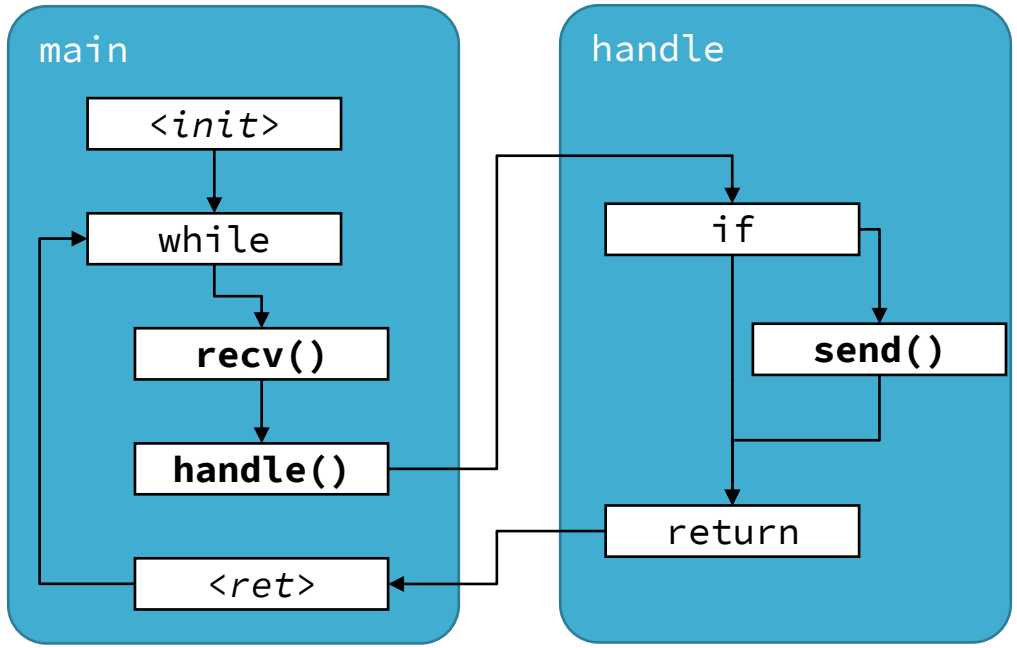
UDP Awareness

Classic Fuzzing Target



- Input from file/stdin
- One-shot program, terminates after processing

AmpFuzz Target

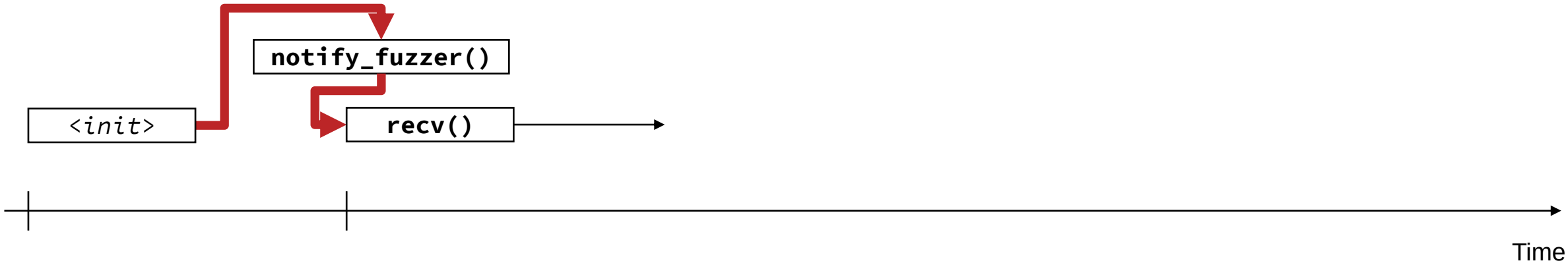


- Input from socket
- Long-running server, recv-handle-loop

Timing sensitive

Does not terminate

UDP Awareness – Begin of Request Processing



Too early:
Not listening,
UDP request
dropped

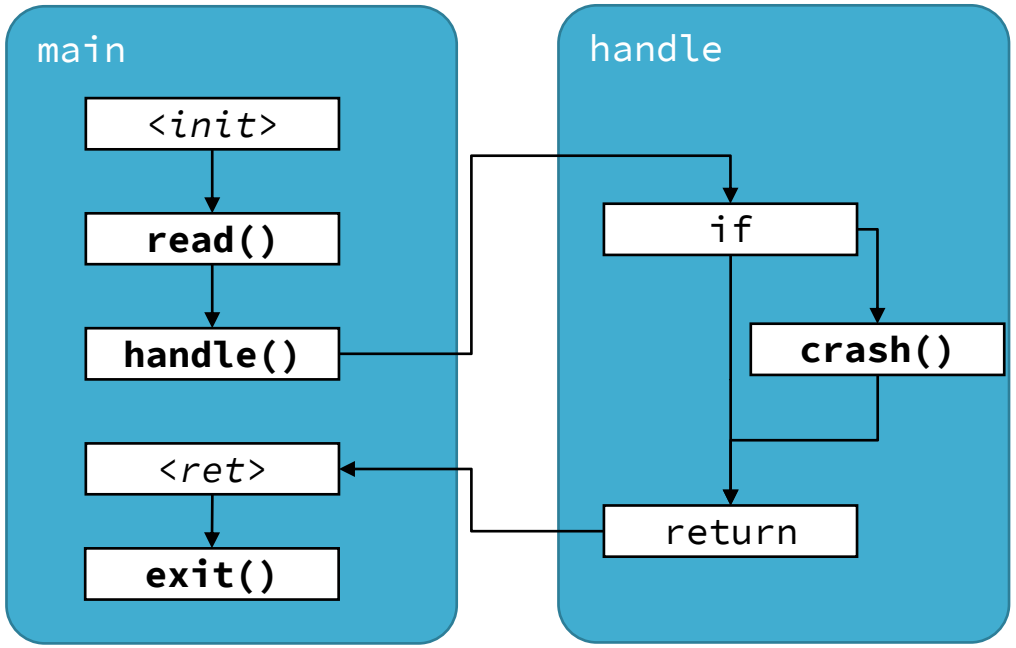
Too late:
Time wasted,
Low throughput

Ideal timing

Let target notify the fuzzer

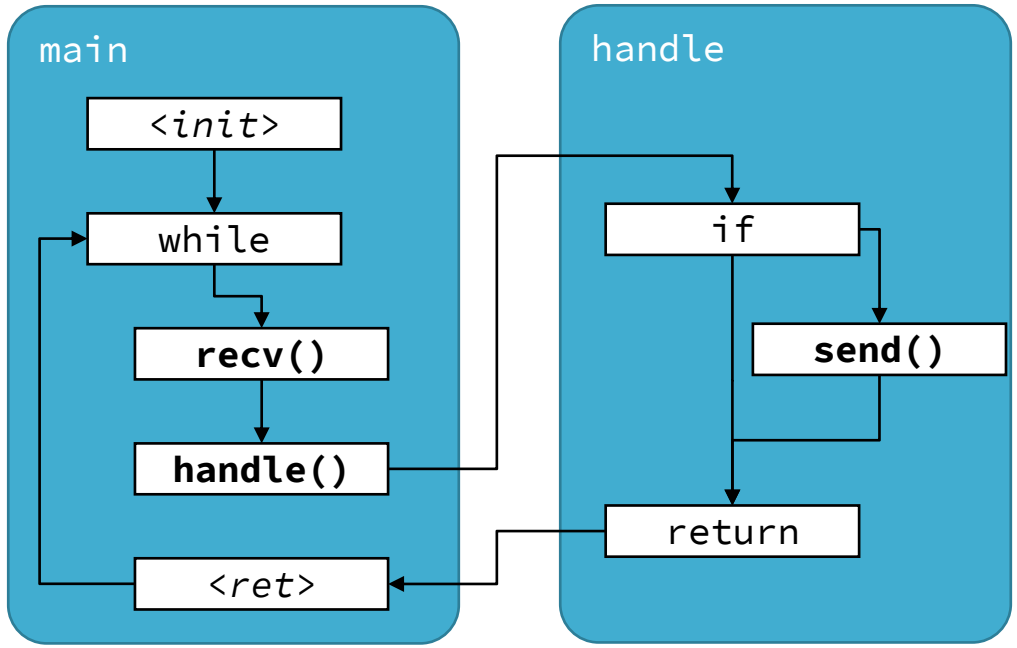
UDP Awareness

Classic Fuzzing Target



- Input from file/stdin
- One-shot program, terminates after processing

AmpFuzz Target

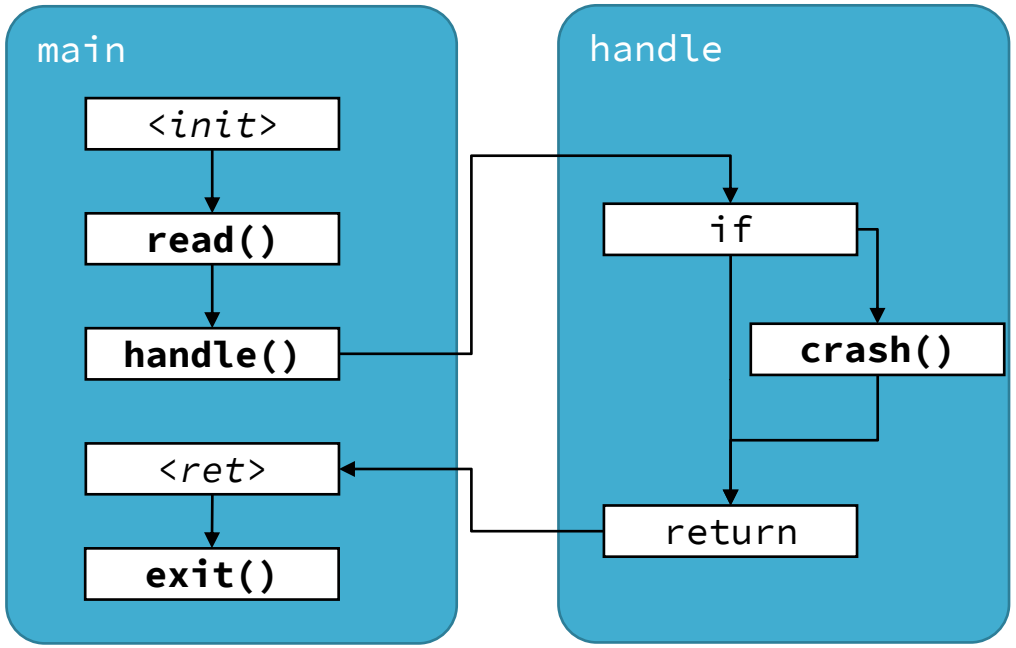


- Input from socket
- Long-running server, recv-handle-loop

Does not terminate

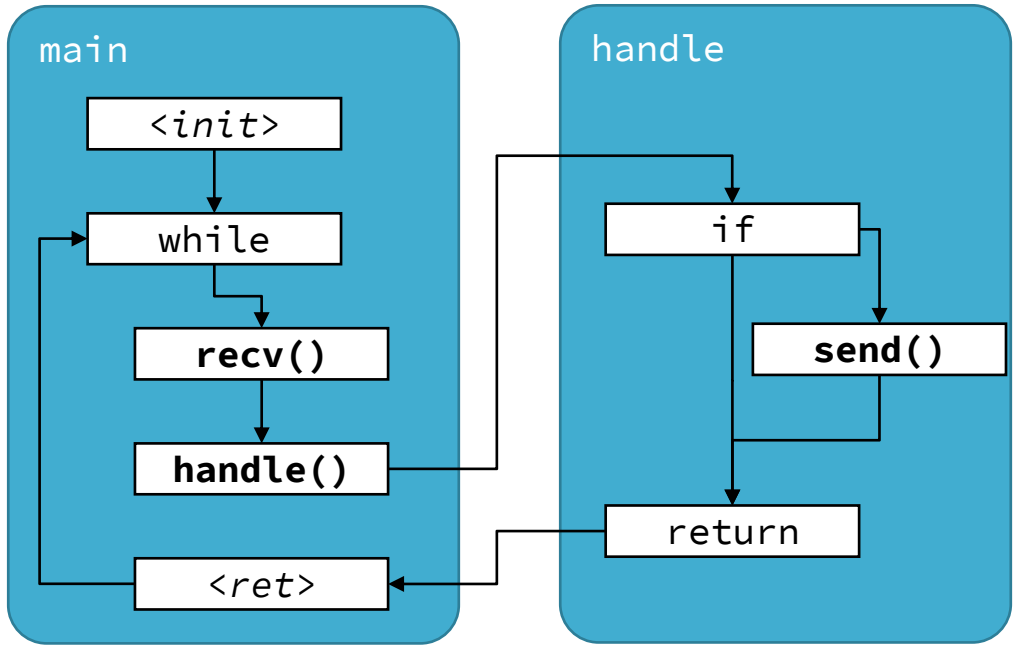
UDP Awareness

Classic Fuzzing Target



- Input from file/stdin
- One-shot program, terminates after processing

AmpFuzz Target



- Input from socket

Transform into one-shot program?

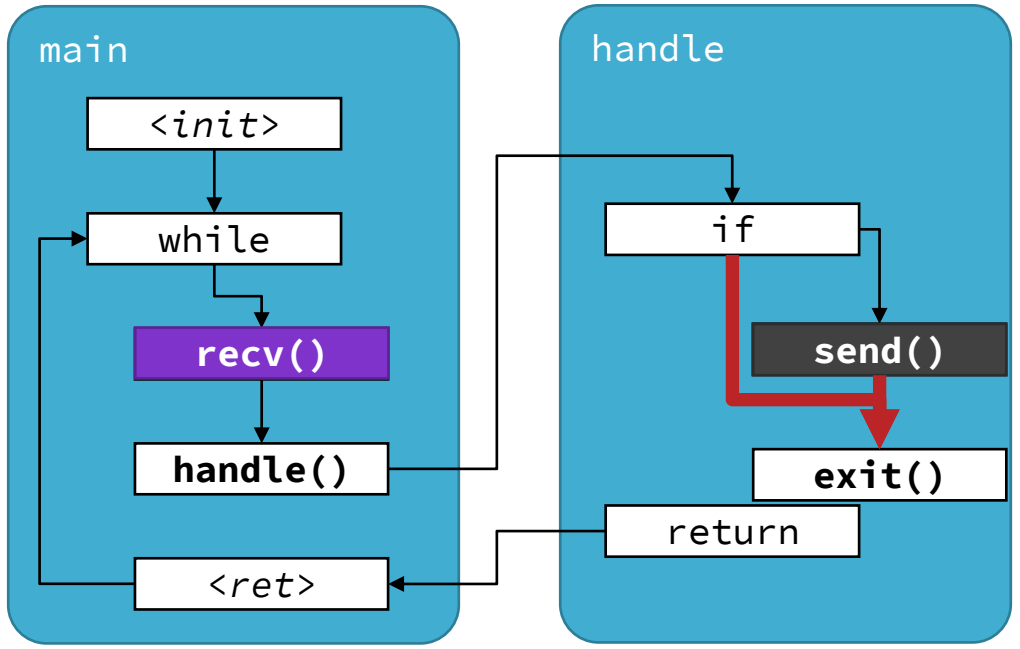
UDP Awareness – End of Request Processing

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Transform into one-shot program?

1. Identify **source** functions
2. Identify **sink** functions
3. Find edges from a **source** that **cannot** reach a **sink** without passing another **source** first
4. Redirect those edges

AmpFuzz Target

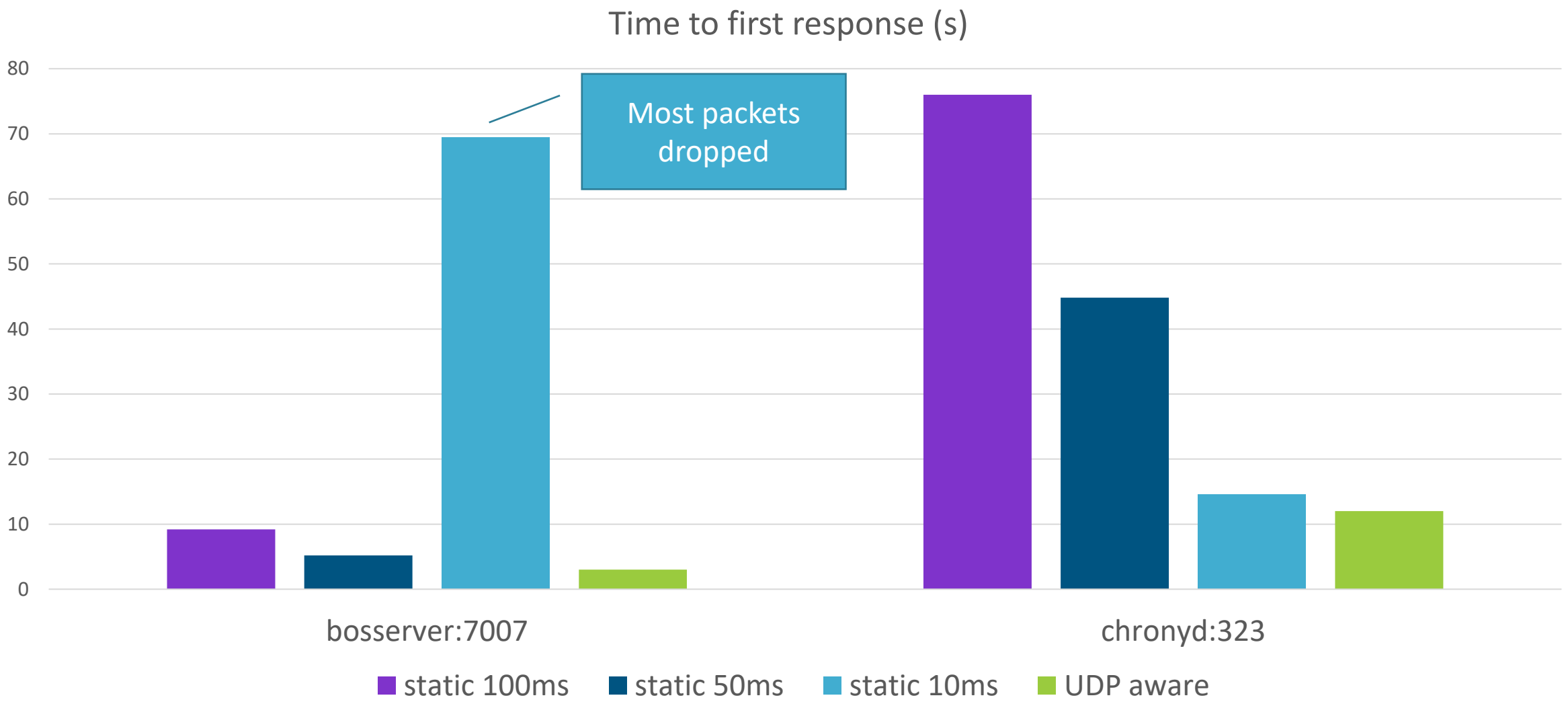


- Input from socket ✓
- Long-running server, recv-handle-loop ✓

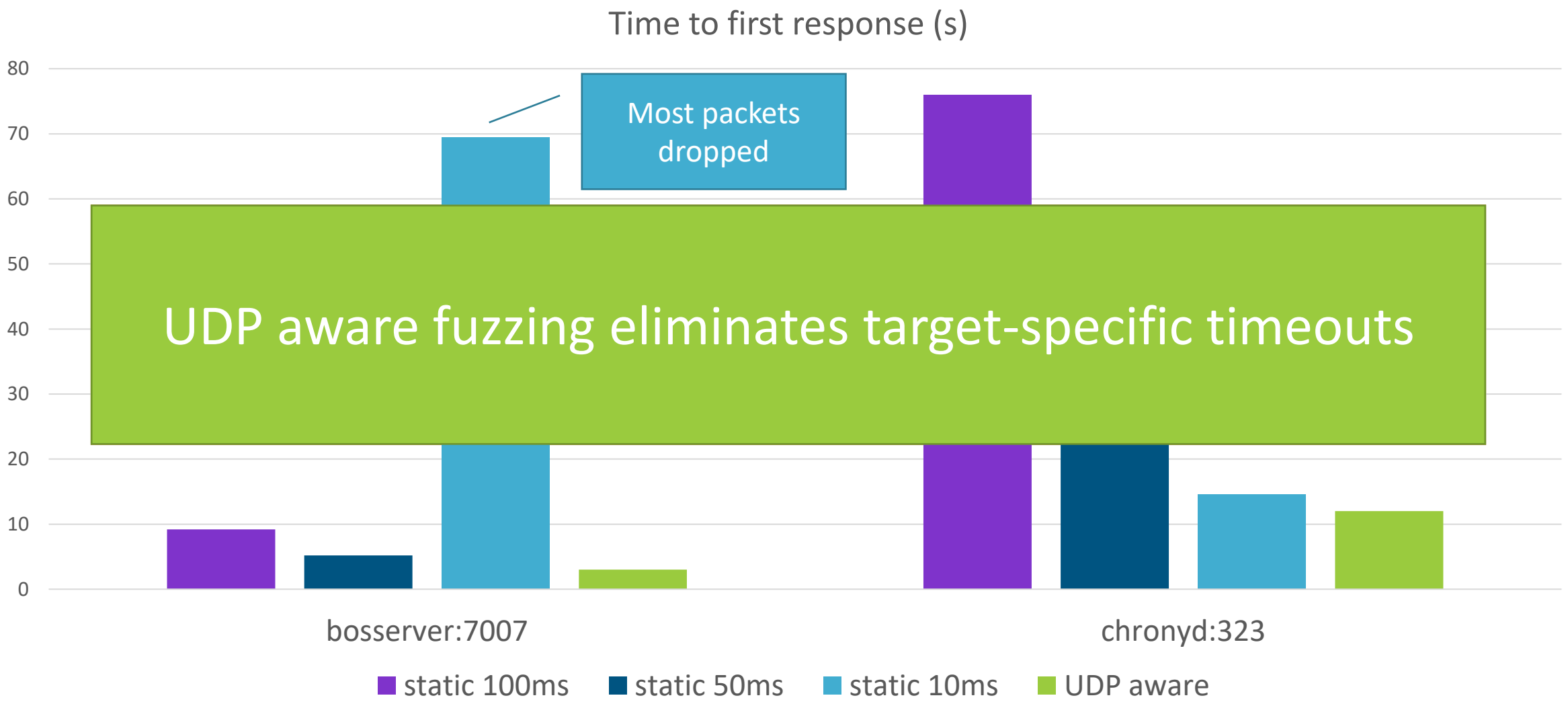
UDP Awareness vs. Static Timeouts

Why not just fixed timeouts?

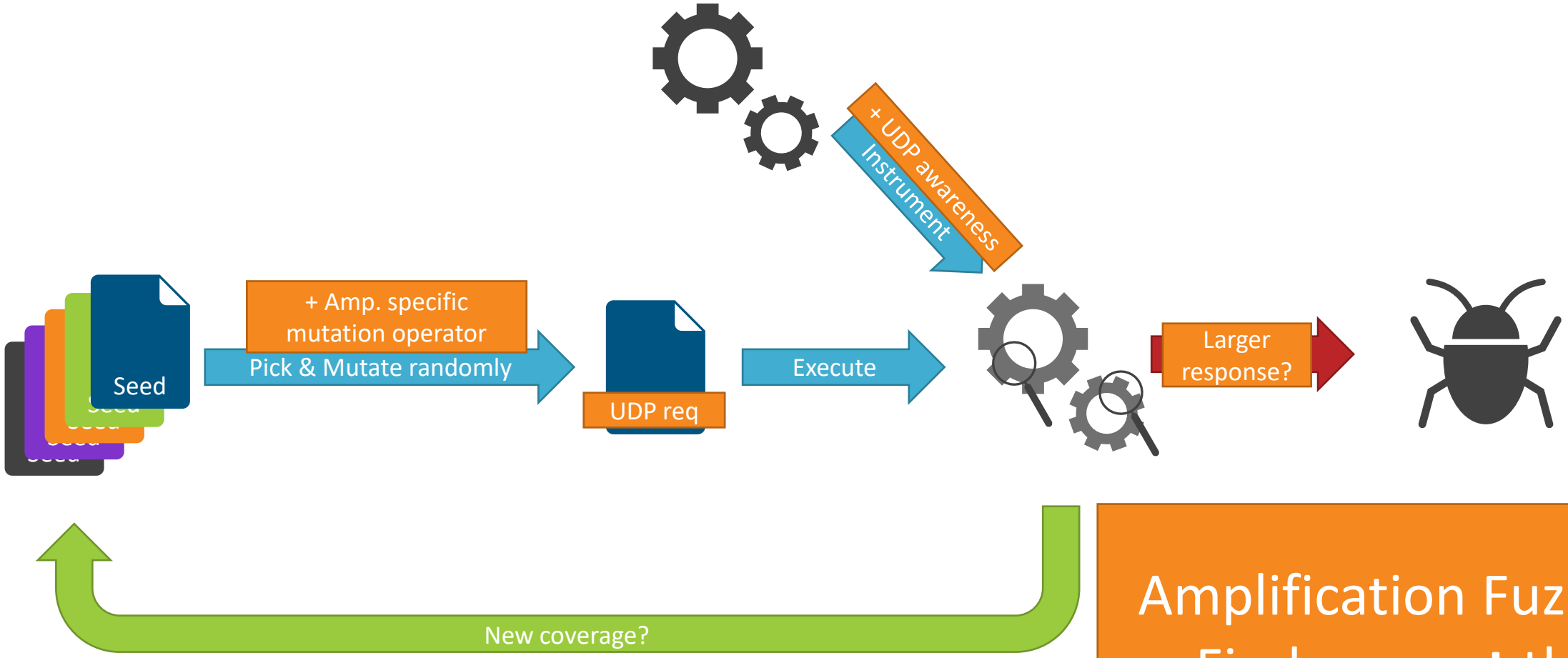
UDP Awareness vs. Static Timeouts



UDP Awareness vs. Static Timeouts



AmpFuzz – Fuzzing for Amplification Vectors



Amplification Fuzzing: Find **request** that **excite large** responses

Amplification-Specific Mutation Operator

Example: RX_DEBUG amplification

- Protocol-compliant request:
  - 36 bytes -> 312 bytes **8.67x amp**

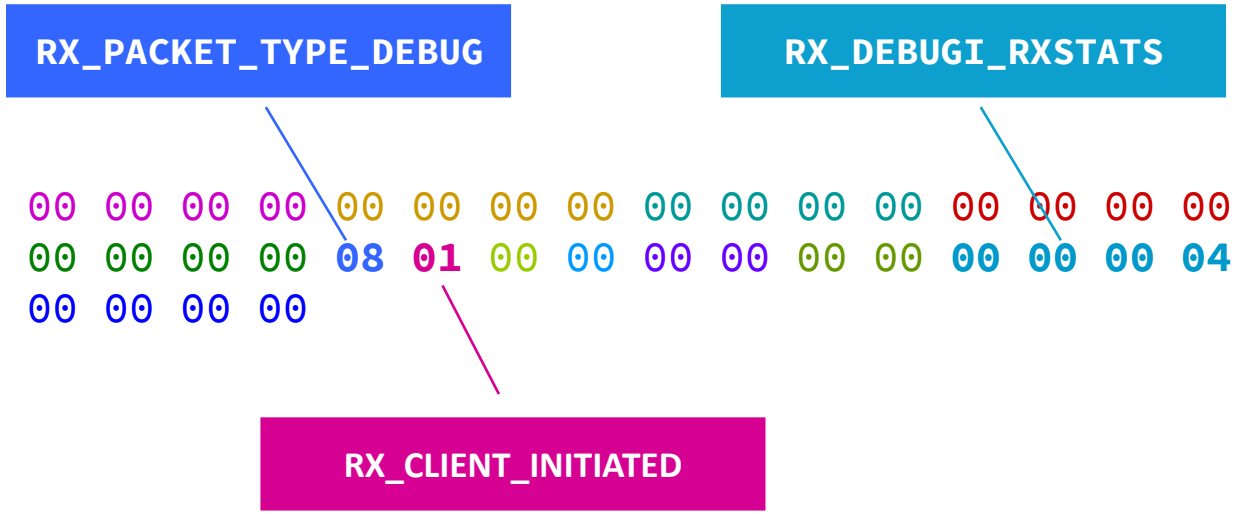
```
struct rx_header { //HEADER  (28 bytes)
  afs_uint32 epoch;
  afs_uint32 cid;
  afs_uint32 callNumber;
  afs_uint32 seq;
  afs_uint32 serial;
  u_char type;
  u_char flags;
  u_char userStatus;
  u_char securityIndex;
  u_short serviceId;
  u_short spare;
};

struct rx_debugIn { //PAYLOAD (8 bytes)
  afs_int32 type;
  afs_int32 index;
};
```

Amplification-Specific Mutation Operator

- Few fields relevant
- Request length often not checked

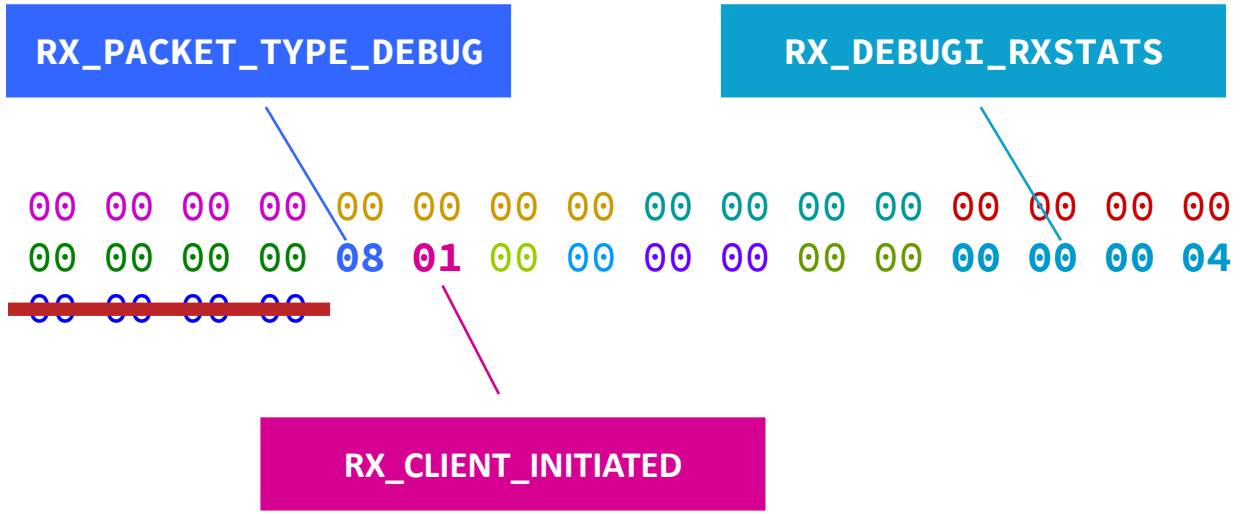
Example: RX_DEBUG amplification

- Protocol-compliant request:
  - 36 bytes -> 312 bytes **8.67x amp**



Amplification-Specific Mutation Operator

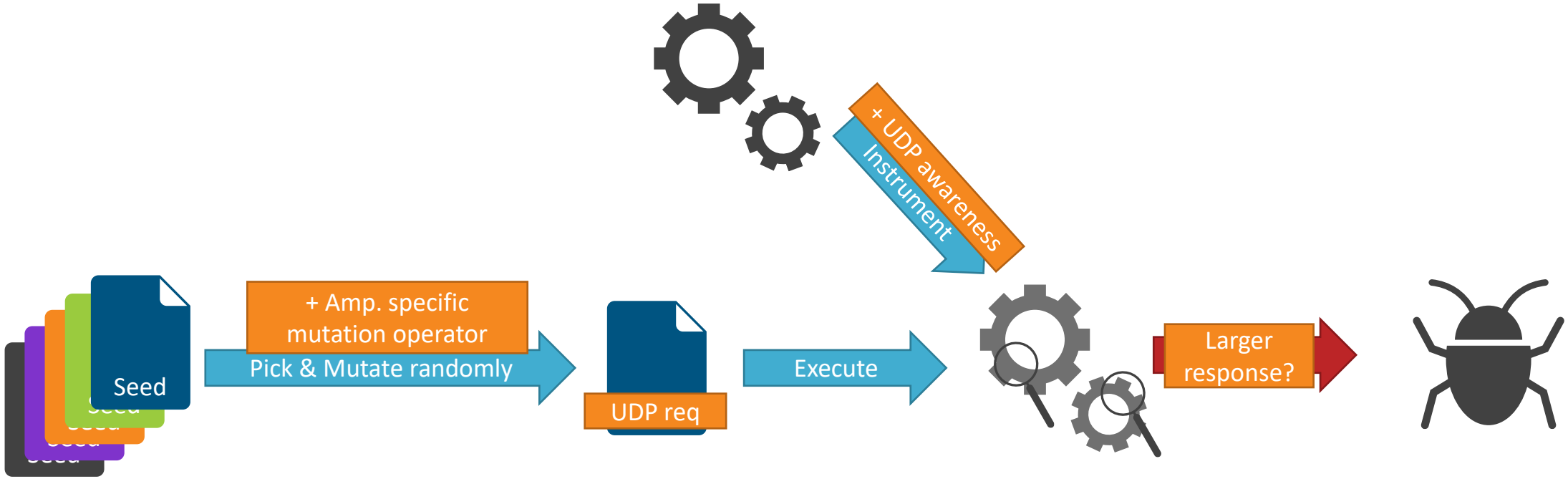
- Few fields relevant
- Request length often not checked



Example: RX_DEBUG amplification

- Protocol-compliant request:
  - 36 bytes -> 312 bytes **8.67x amp**
- Truncated request:
  - 32 bytes -> 312 bytes **9.75x amp**

AmpFuzz – Fuzzing for Amplification Vectors



Amplification Fuzzing: Find **request** that **excite large** responses

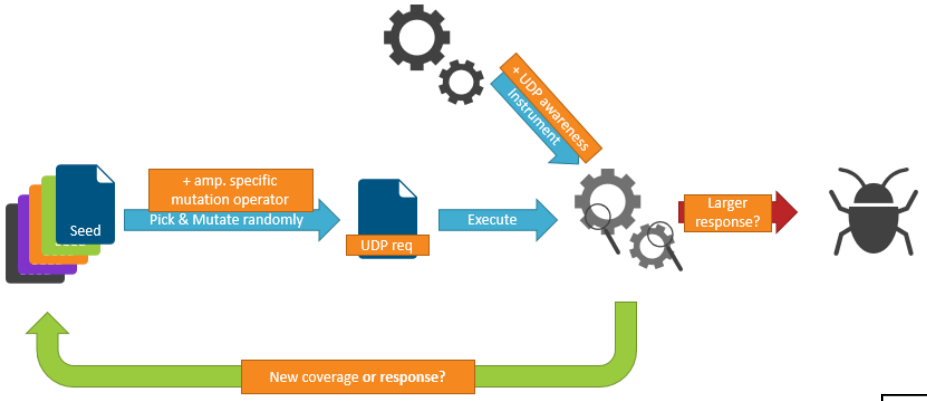
Evaluation

- 28 UDP open-source network services, 5x 24h runs

Target	Amp. Factor (L2)	# Vectors	
atftpd	3.63	47	NEW
kadmind	2.91	92	NEW
memcached	32.45	33	
ntpd	7.47	20	
ntpd (ntpsec)	7.28	10	
bosserver	4.59	212	NEW
talkd	1.09	1	NEW
tftpd	1.14	22	
xl2tpd	3.52	10	NEW

All vulnerabilities have been responsibly disclosed to the respective project maintainers

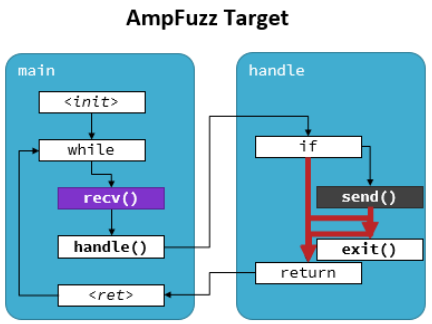
AmpFuzz - Summary



@JohKrupp

Transform into one-shot program?

1. Identify source functions
2. Identify sink functions
3. Find edges from a source that cannot reach a sink without passing another source first
4. Redirect those edges



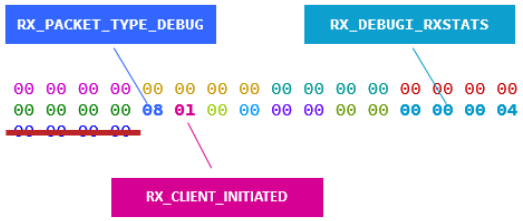
ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

cispa/ampfuzz

- Few fields relevant
- Request length often not checked

Example: RX_DEBUG amplification



- Protocol-compliant request:
  - 36 bytes -> 312 bytes **8.67x amp**
- Truncated request:
  - 32 bytes -> 312 bytes **9.75x amp**

Target	Amp. Factor (L2)	# Vectors	
atftpd	3.63	47	NEW
kadmind	2.91	92	NEW
memcached	32.45	33	
ntpd	7.47	20	
ntpd (ntpsec)	7.28	10	
bosserver	4.59	212	NEW
talkd	1.09	1	NEW
tftpd	1.14	22	
xl2tpd	3.52	10	NEW

All vulnerabilities have been responsibly disclosed to the respective project maintainers