



CISPA

HELMHOLTZ CENTER FOR
INFORMATION SECURITY

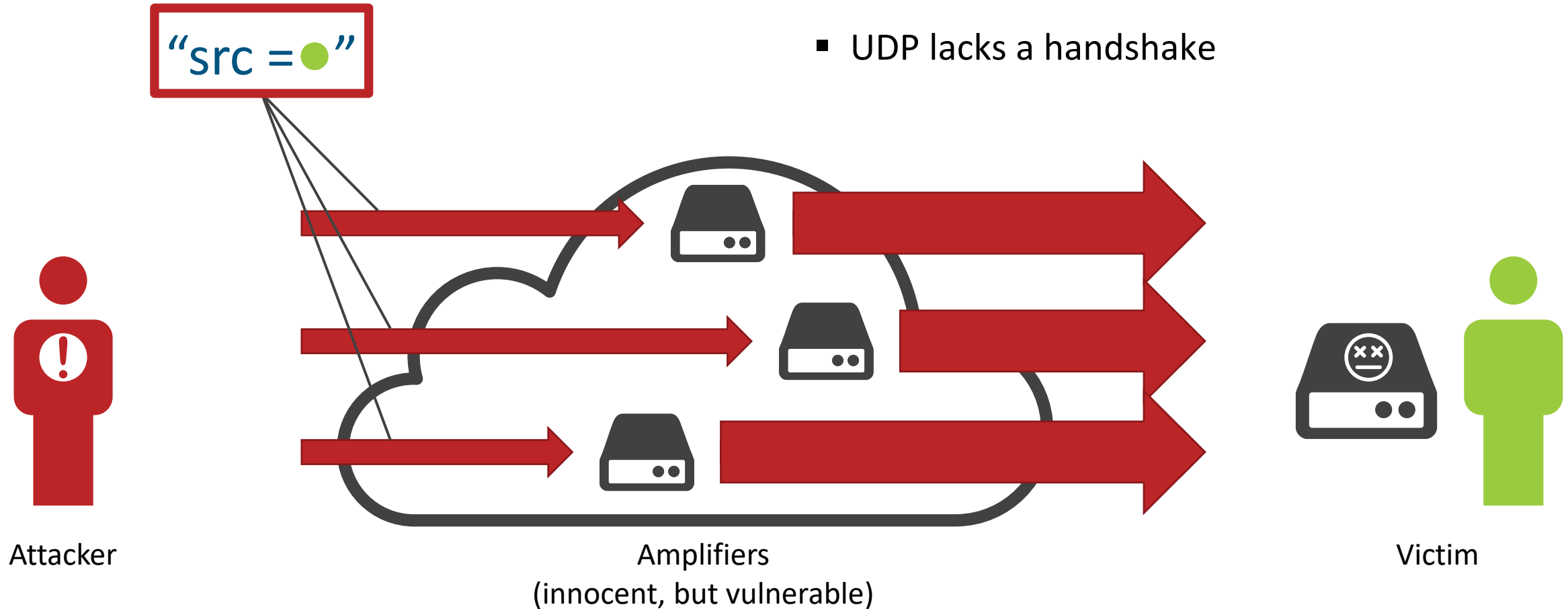
AmpFuzz: Fuzzing for Amplification DDoS Vulnerabilities

Johannes Krupp
CrowdStrike

Ilya Grishchenko
*University of California,
Santa Barbara*

Christian Rossow
*CISPA Helmholtz Center
for Information Security*

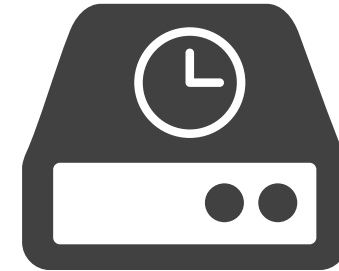
Amplification Distributed Denial-of-Service (DDoS)





GET monlist
8 bytes

Amplification: 5500x



IP	port	port	port	port
1.2.3.4	55555	55555	55555	55555
1.2.3.5	55556	55556	55556	55556
1.2.3.6	55557	55557	55557	55557
1.2.3.7	55558	55558	55558	55558
1.2.3.8	55559	55559	55559	55559
1.2.3.9	55560	55560	55560	55560

440 bytes

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

Spamhaus DDoS grows to Internet-threatening

Record-breaking DDoS attack in

Europe

A distrib
attack

Security

013

Bigg

Thursda

**AWS said it mitigated a 2.3 Tbps DDoS attack,
the largest ever**

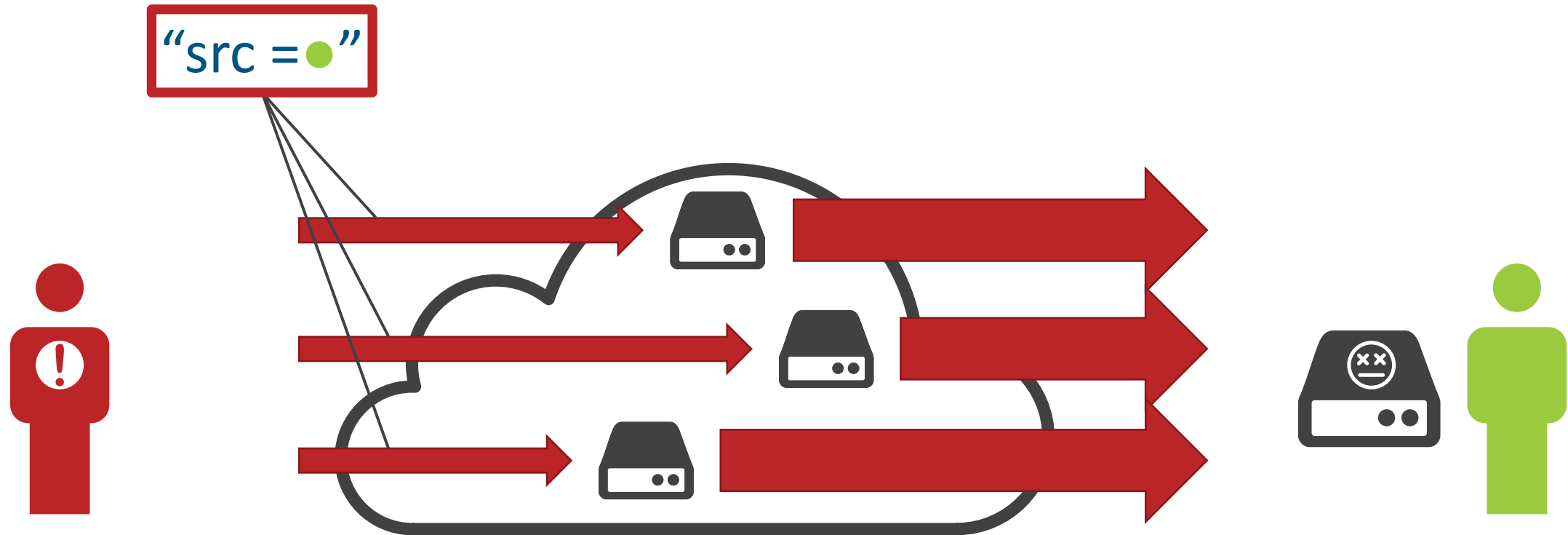
**Microsoft Fended Off a Record 2.4 Tbps DDoS
Attack Targeting Azure Customers**

Writte

📅 October 12, 2021 👤 Ravie Lakshmanan

hackernews, 12 Oct 2021

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



- Large amplification potential
- New vulnerabilities discovered regularly

1 NOVEMBER 2017 / DDOSMON

GLDAP is Now the No. 2 Reflection

Memcrashed - Major amplification attacks from UDP port 11211

27/10/2019
Over
vectors

Protocol used by 630,000 devices can be abused for devastating DDoS attacks

Security researchers warn that the WS-Discovery protocol is currently being abused for massive DDoS attacks.



By Catalin Cimpanu for Zero Day | August 27, 2019 -- 13:40 GMT (06:40 PDT) | Topic: Security

zdnet, 27 Aug 2019



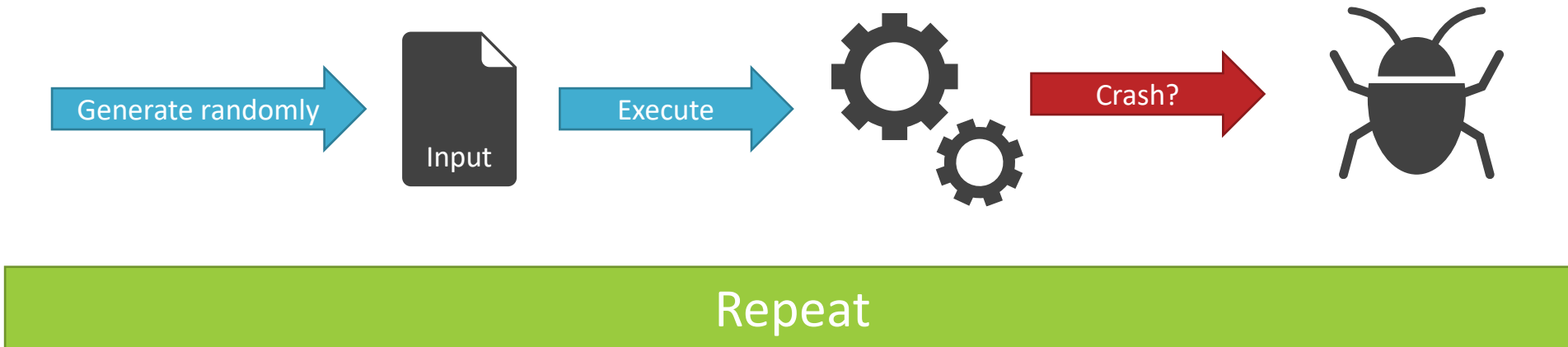
- Large amplification potential
- New vulnerabilities discovered regularly

Known vectors found manually or post-mortem

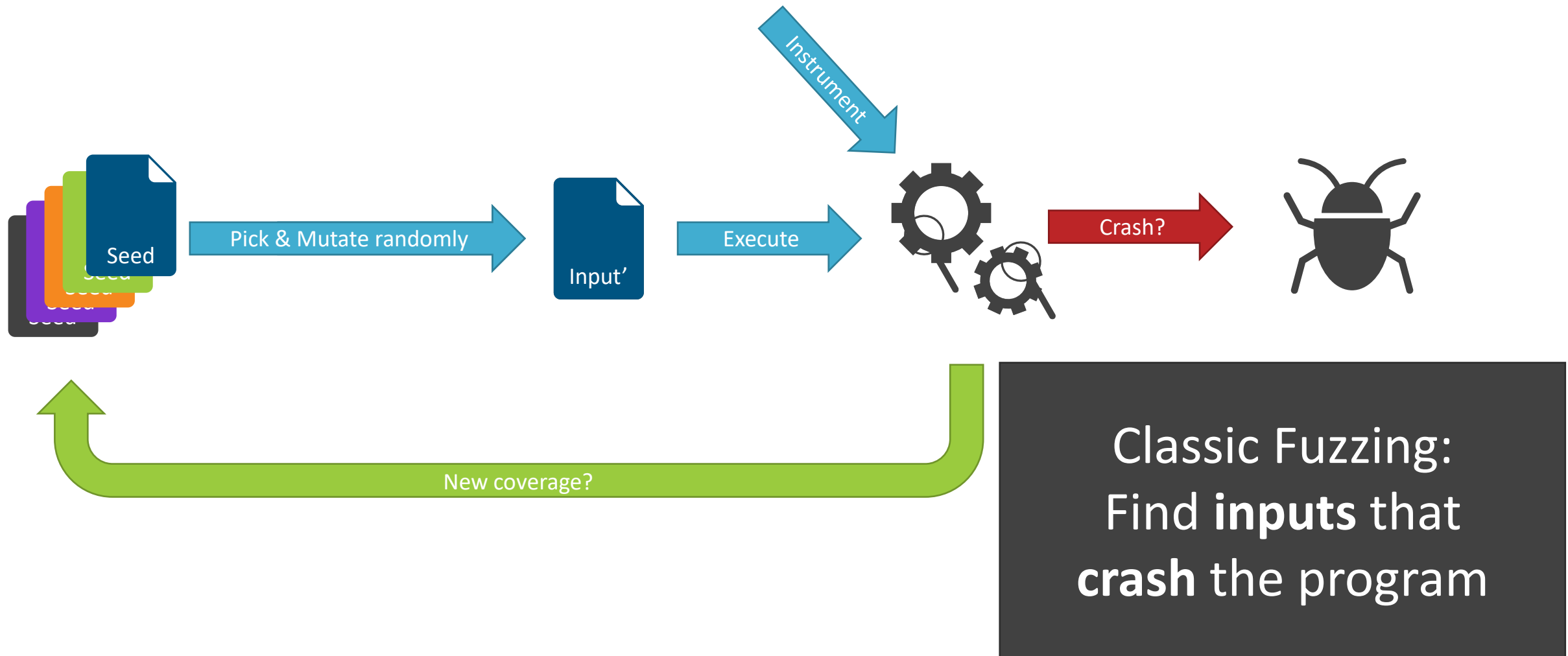
Can we find vulnerable protocols
systematically and proactively?

-> Let's try fuzzing!

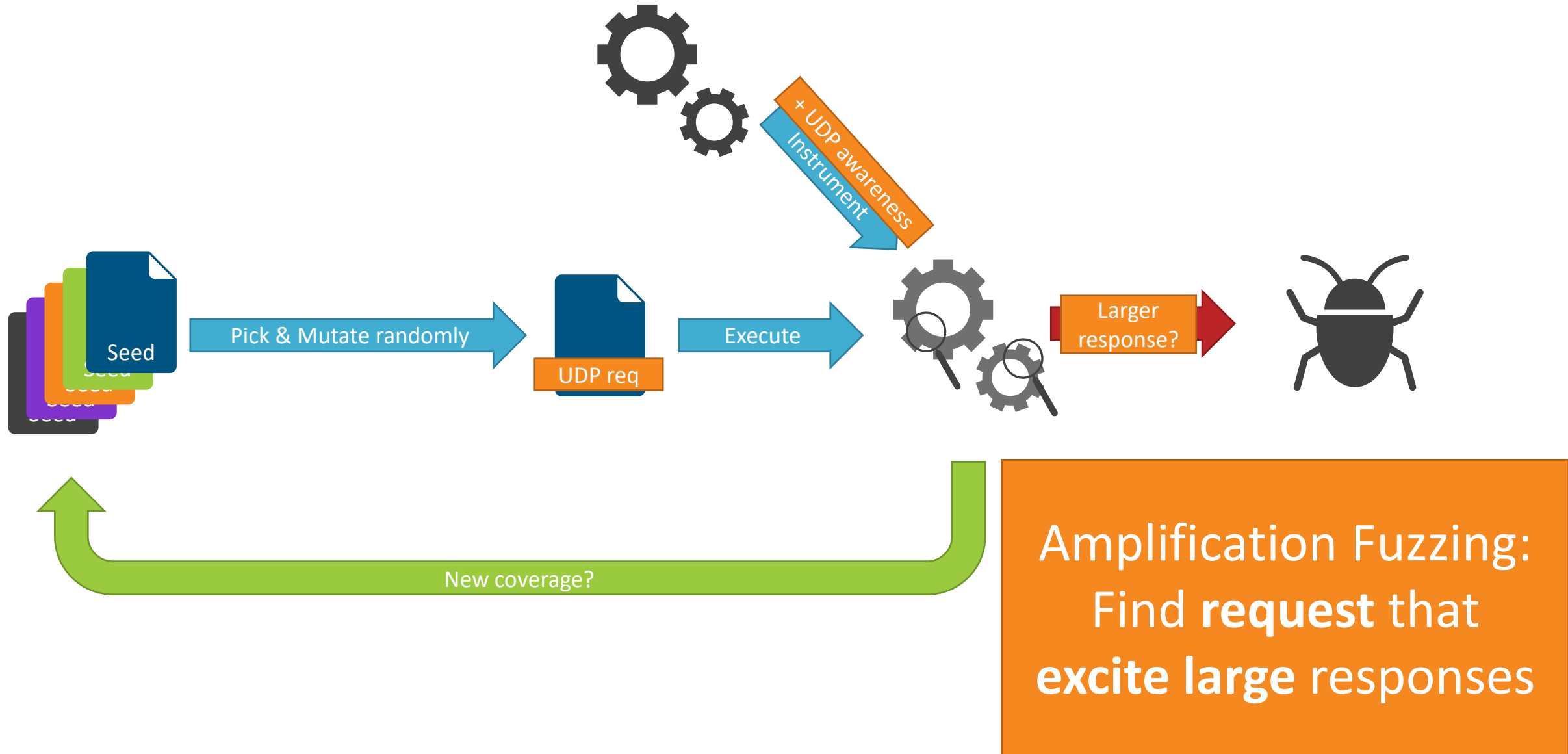
- New vulnerabilities discovered regularly



Coverage-Guided Mutation-Based Greybox Fuzzing

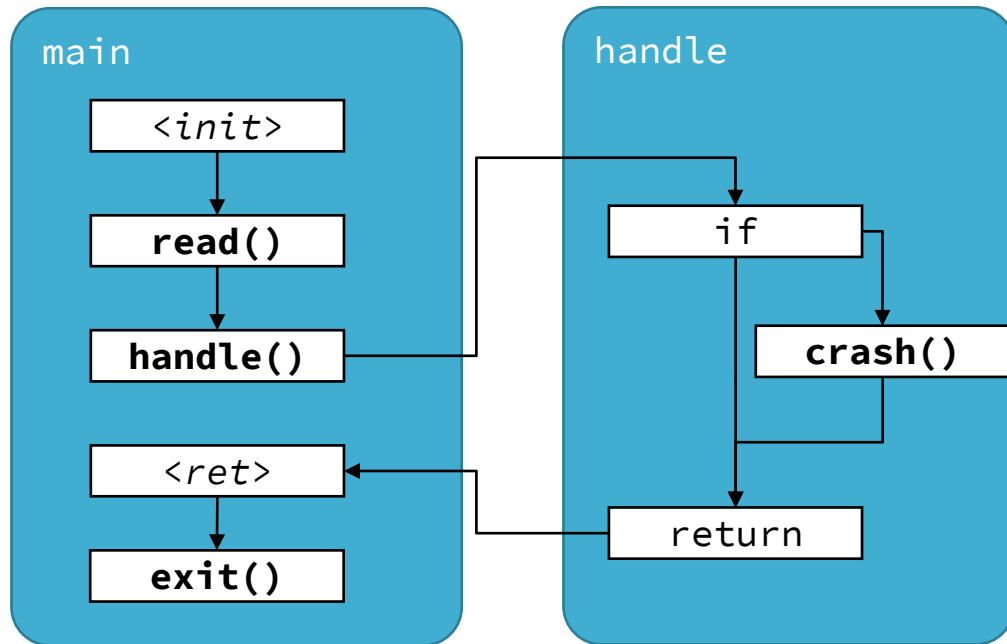


AmpFuzz – Fuzzing for Amplification Vectors



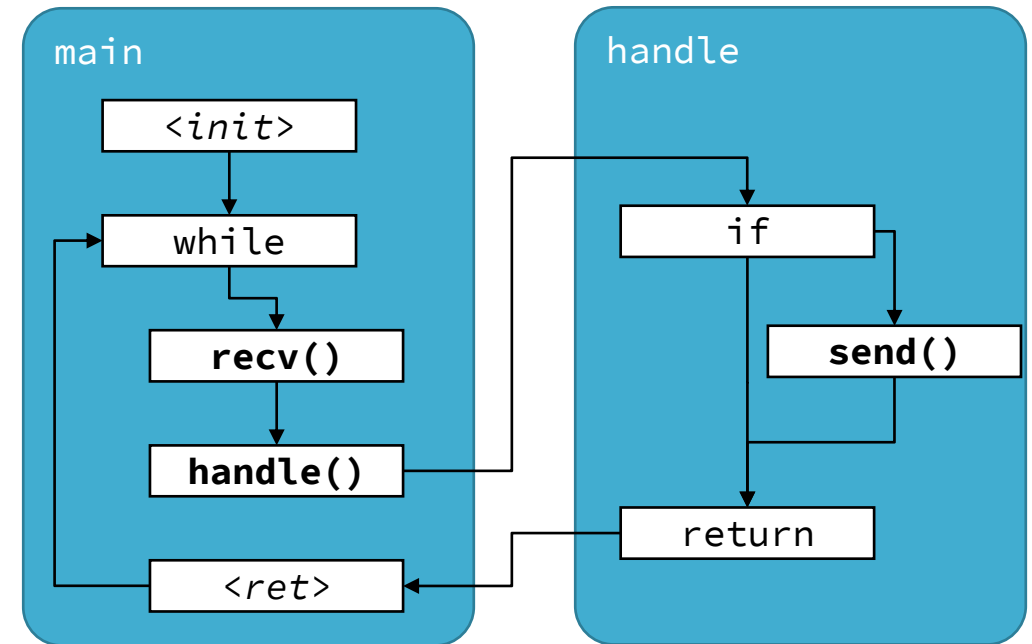
Amplification Fuzzing:
Find request that
excite large responses

Classic Fuzzing Target



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

AmpFuzz Target

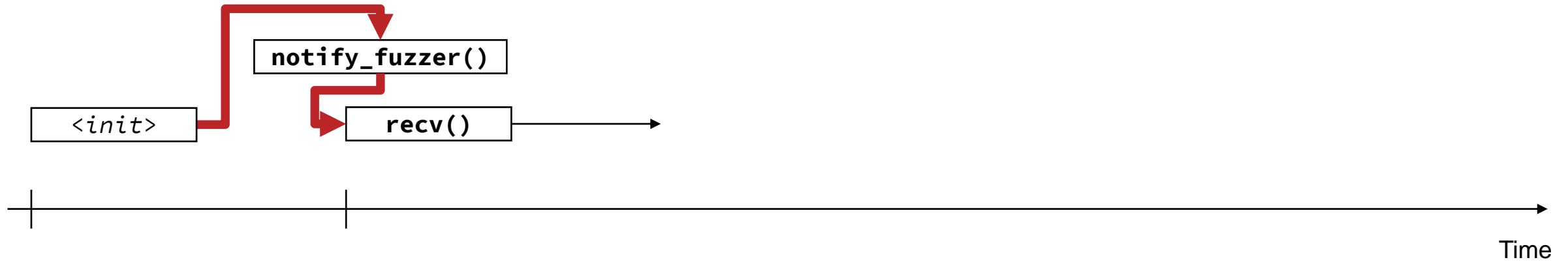


- Input from socket
- Long-running server, recv-handle-loop

Timing sensitive

Does not terminate

UDP Awareness – Begin of Request Processing



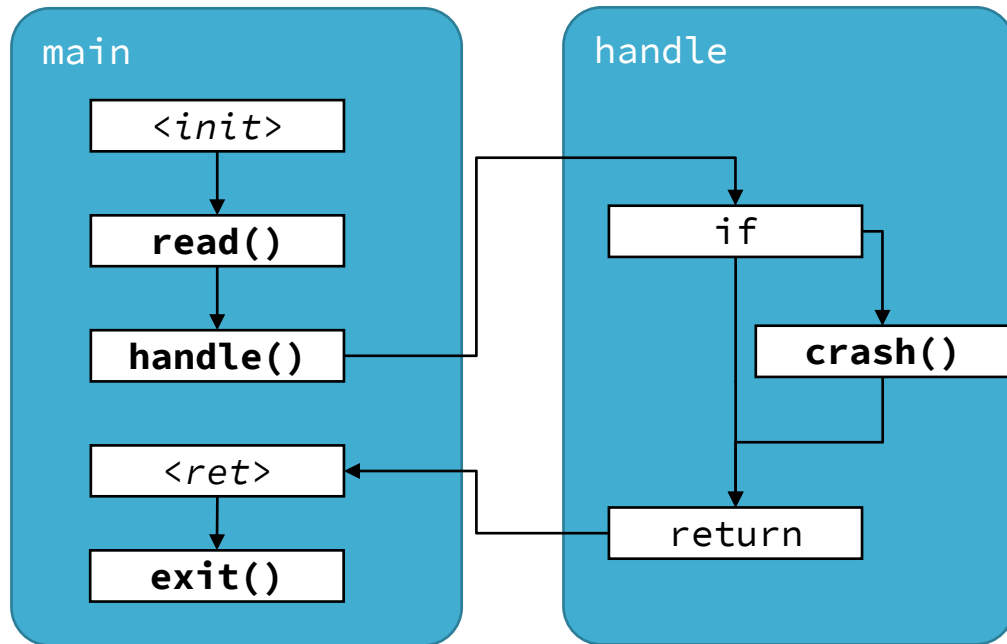
Too early:
Not listening,
UDP request
dropped

Too late:
Time wasted,
Low throughput

Ideal timing

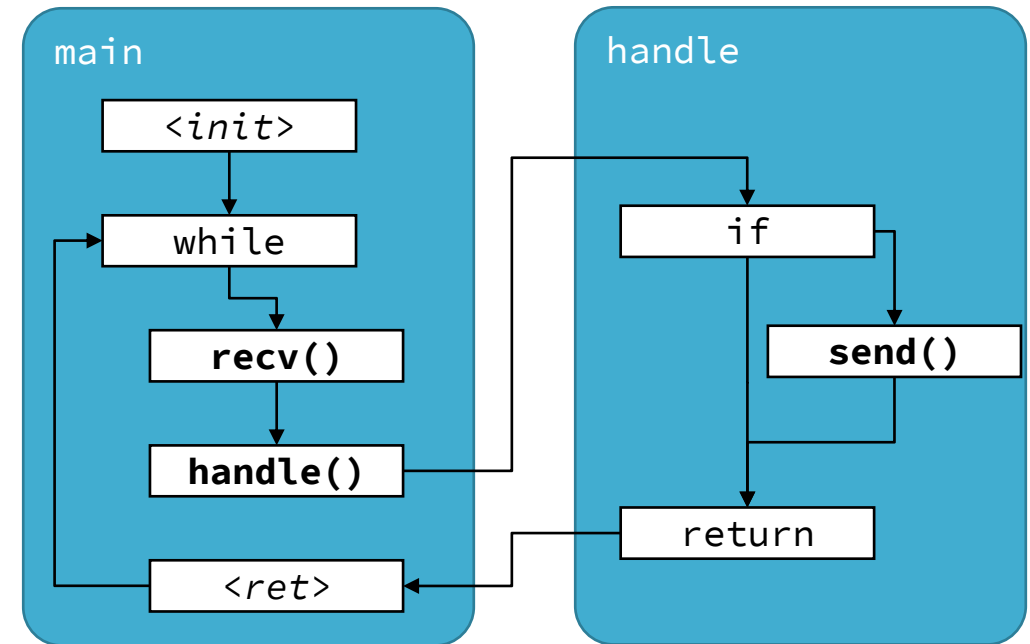
Let target notify the fuzzer

Classic Fuzzing Target



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

AmpFuzz Target

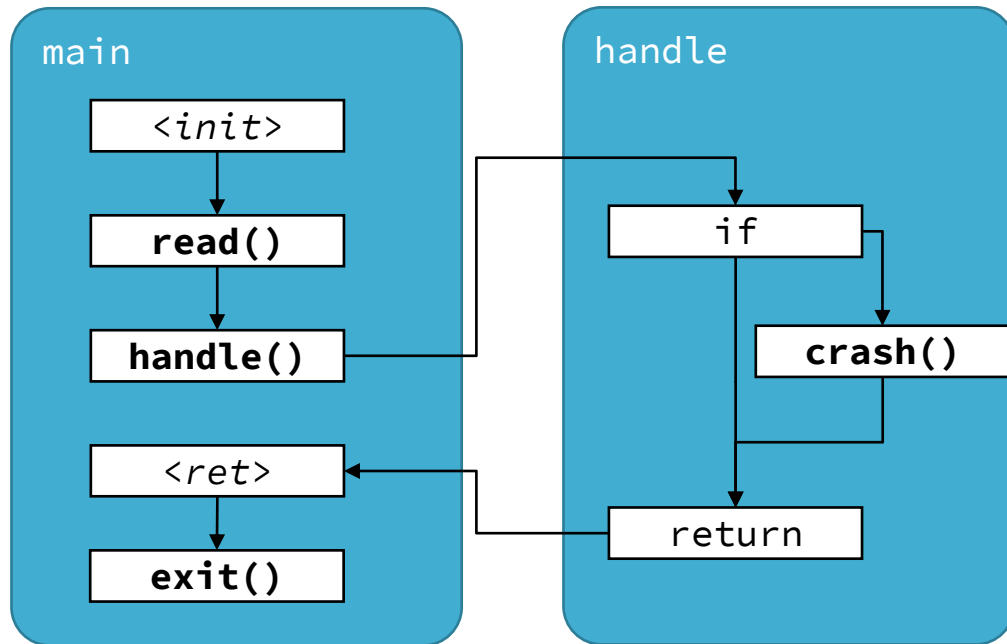


- Input from socket
- Long-running server, recv-handle-loop



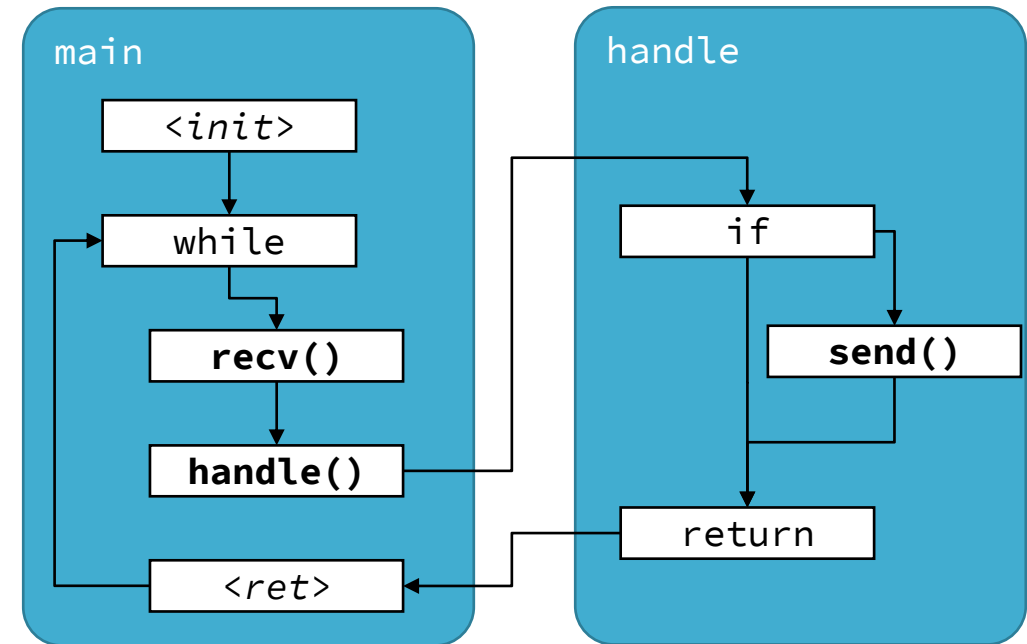
Does not terminate

Classic Fuzzing Target



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

AmpFuzz Target



- Input from socket

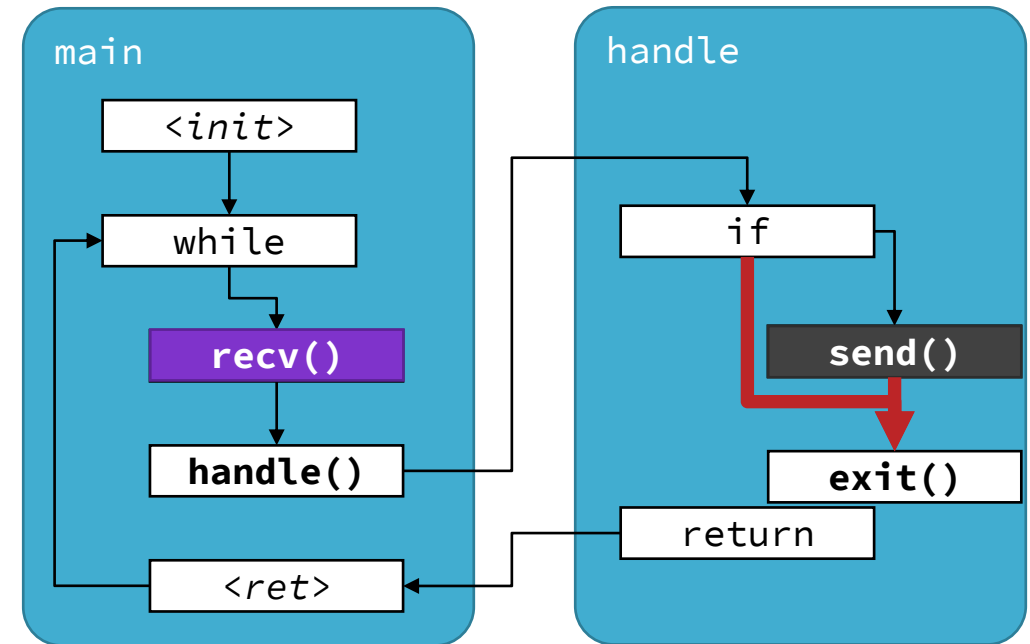


Transform into one-shot program?

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



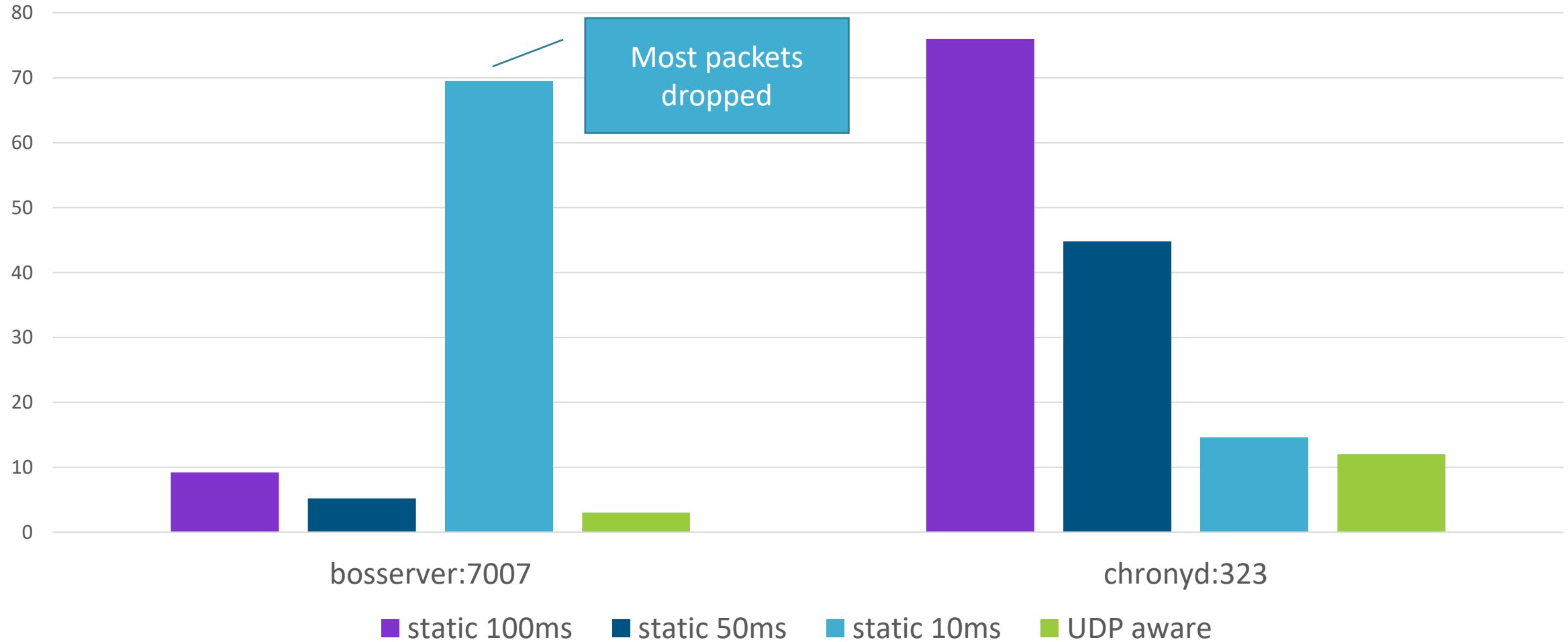
- Input from socket
- Long-running server, recv-handle-loop



Why not just fixed timeouts?

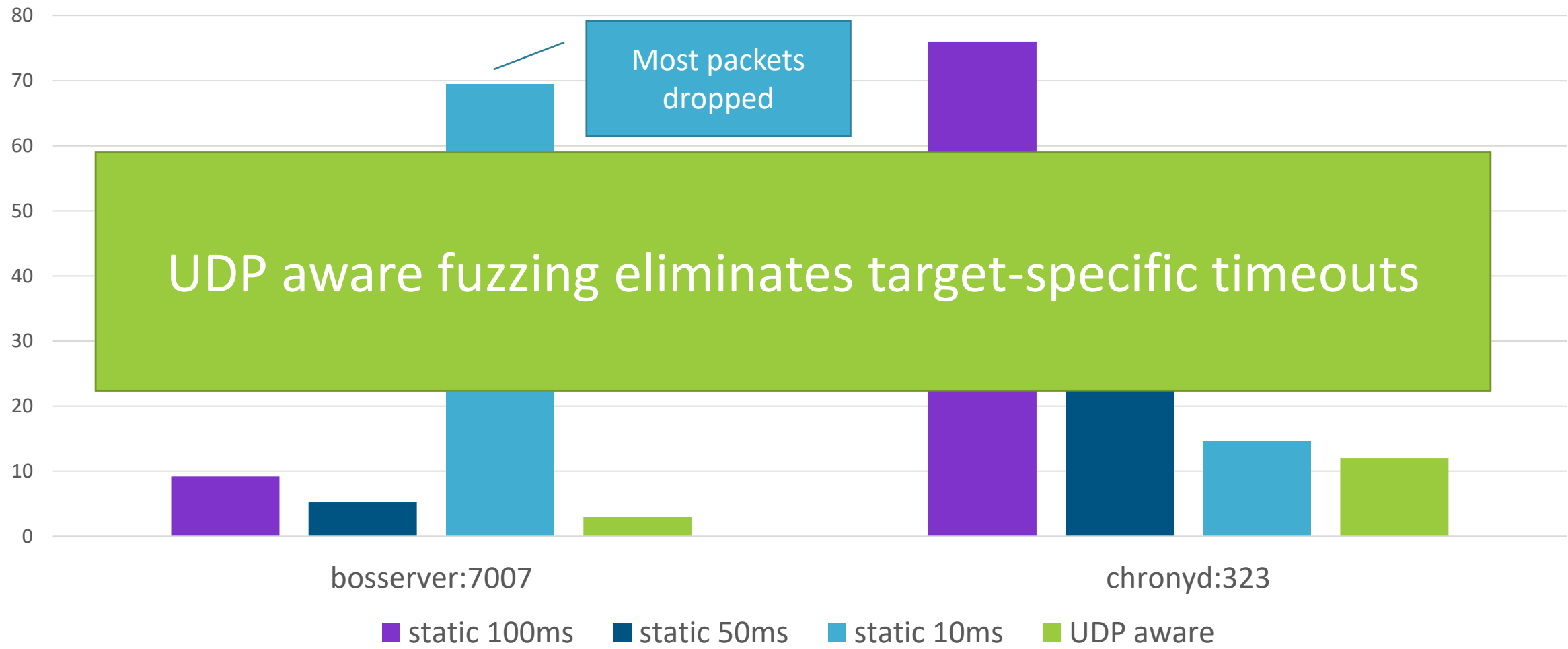
UDP Awareness vs. Static Timeouts

Time to first response (s)

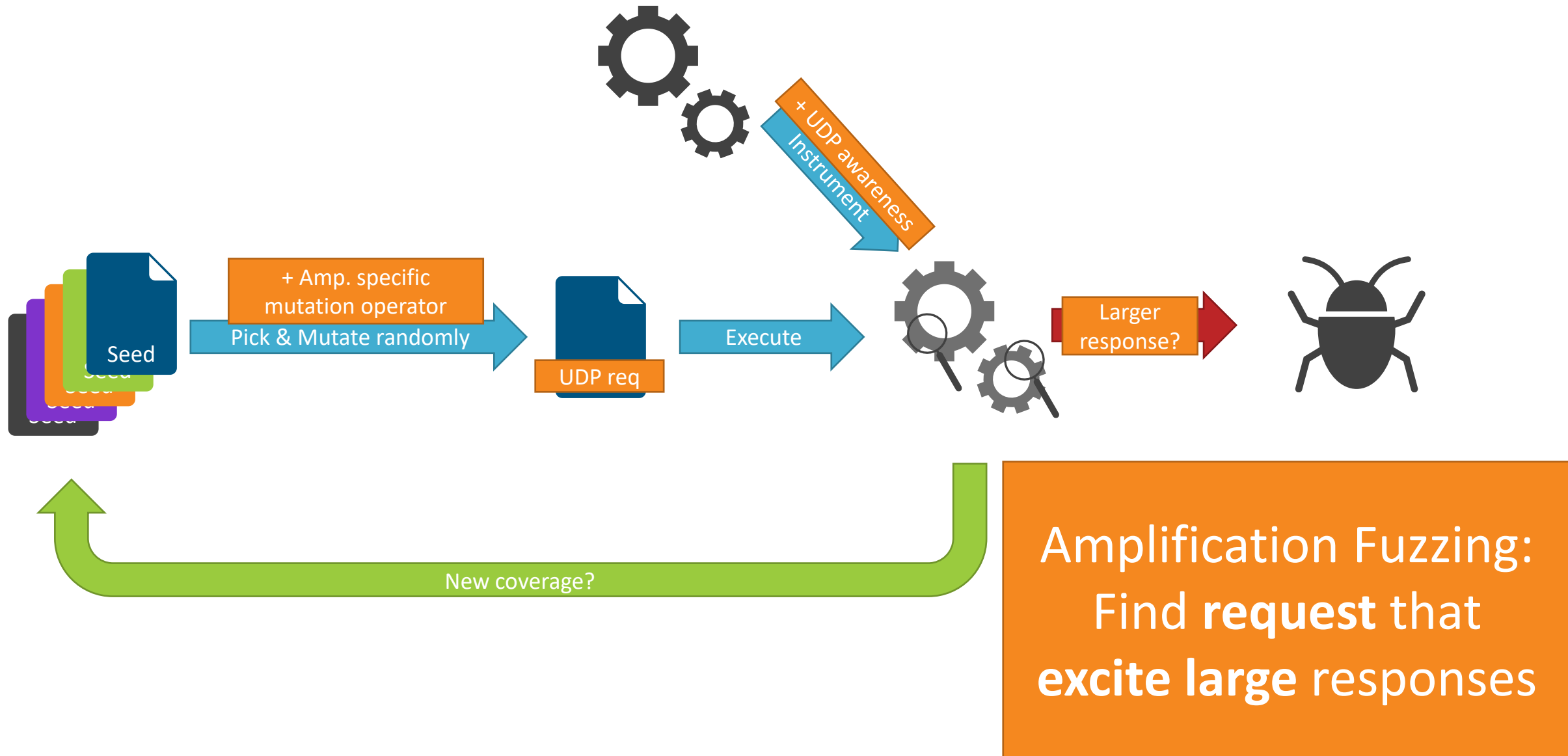


UDP Awareness vs. Static Timeouts

Time to first response (s)



AmpFuzz – Fuzzing for Amplification Vectors



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;
};

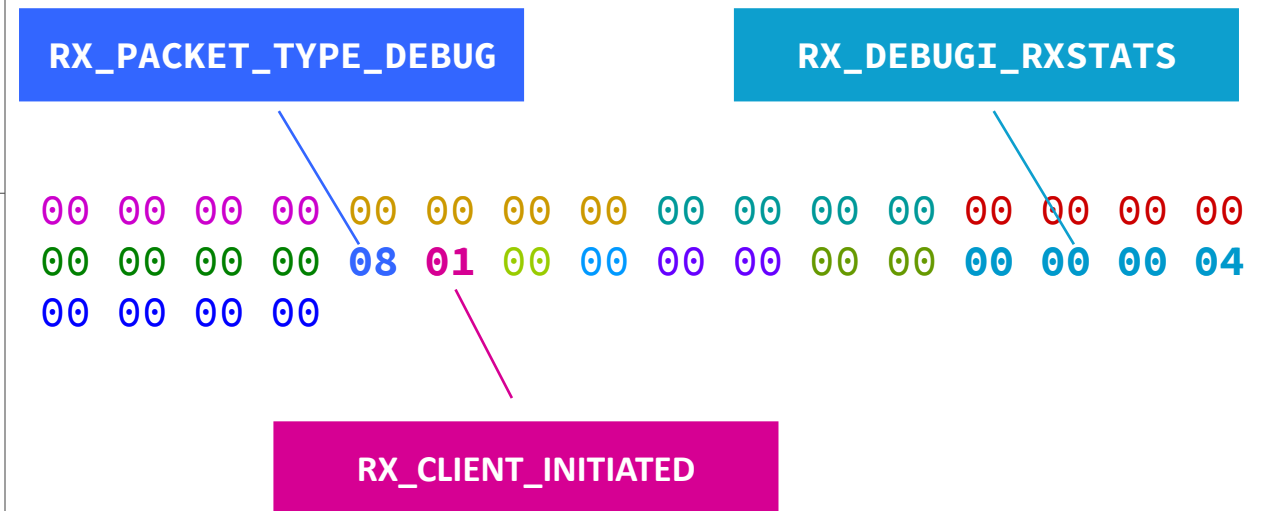
struct rx_debugIn { //PAYLOAD (8 bytes)
    afs_int32 type;
    afs_int32 index;
};
```

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**

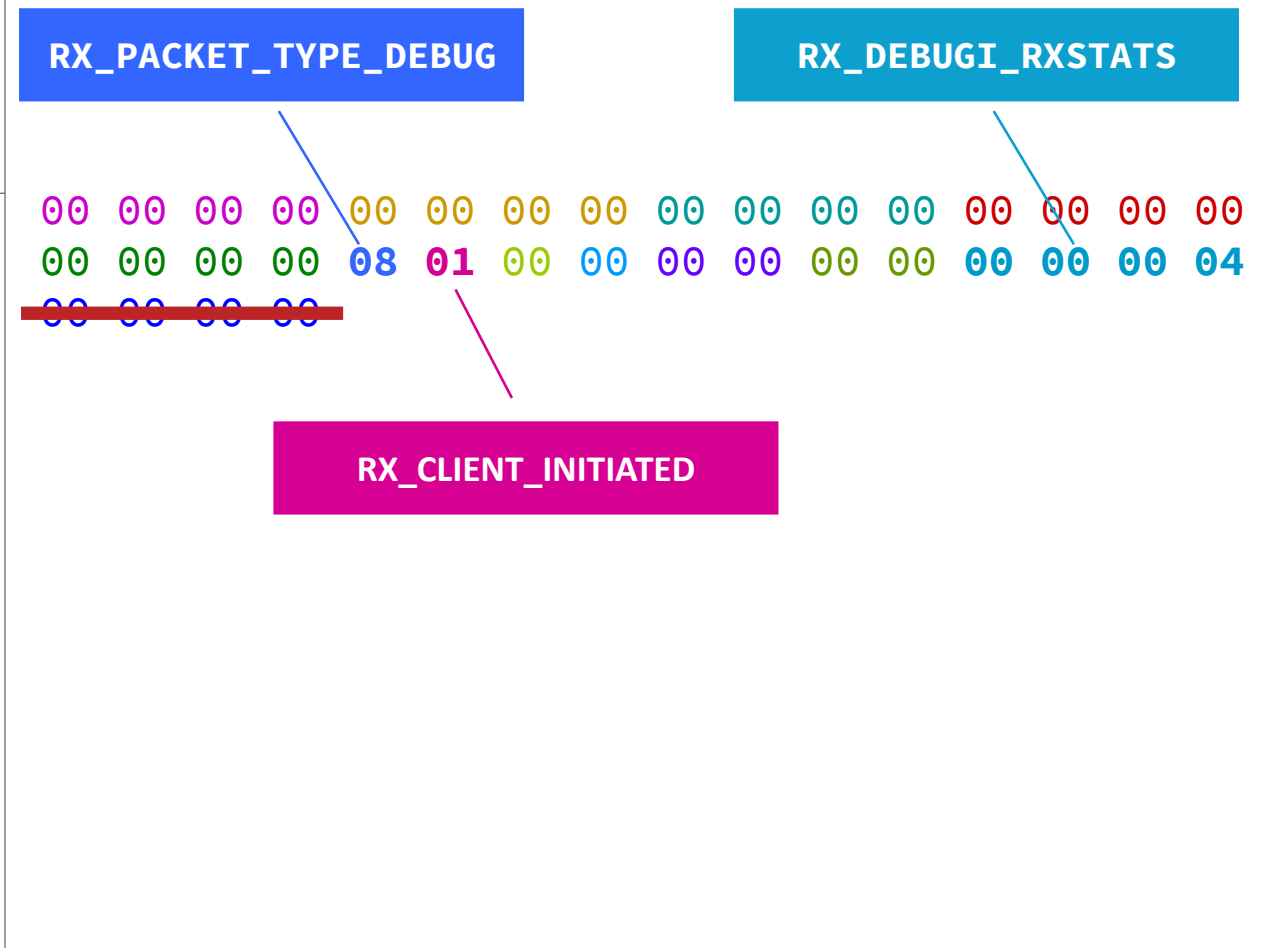


Amplification-Specific Mutation Operator

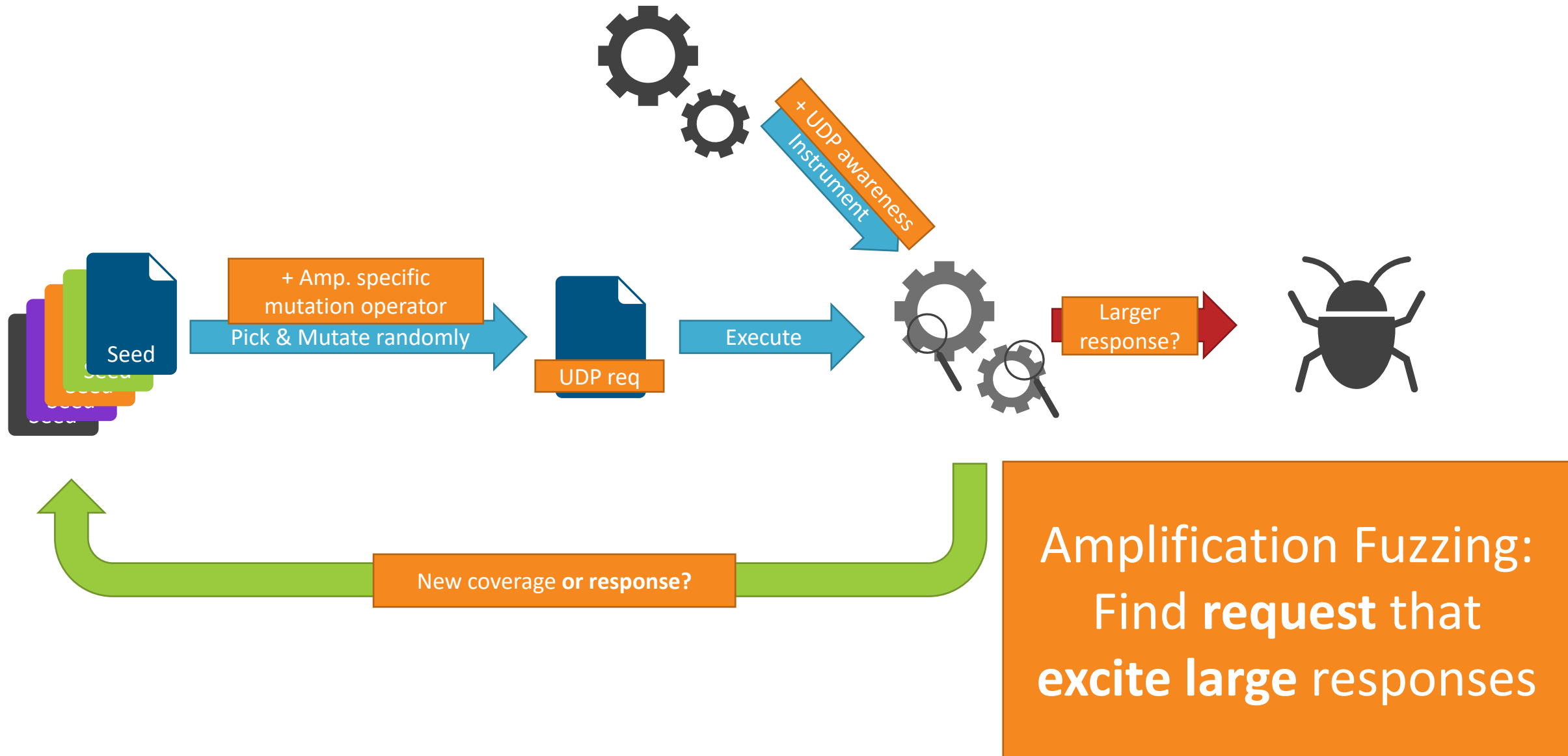
- Few fields relevant
- Request length often not checked

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



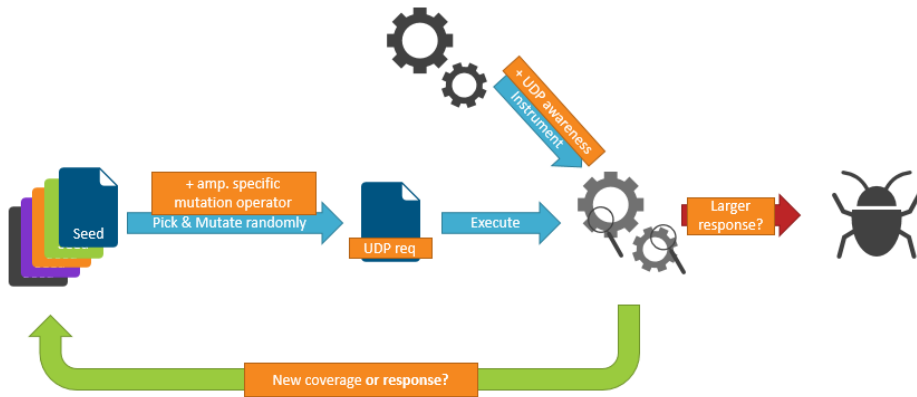
Amplification Fuzzing:
Find request that
excite large responses

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

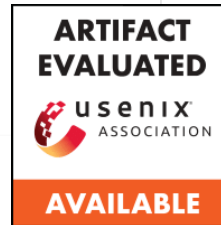
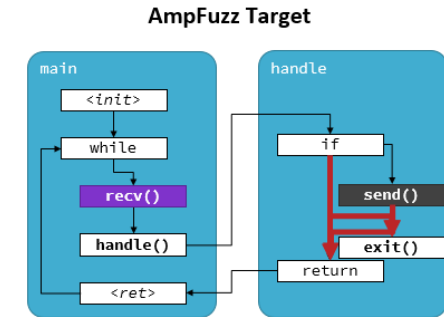
Target	Amp. Factor (L2)	# Vectors	
atftpd	3.63	47	NEW
kadmind	2.91	92	NEW
memcached	32.45	33	
ntpd	7.47	20	
ntpd (ntpsec)	7.28	10	
bosserv	4.59	212	NEW
talkd	1.09	1	NEW
tftpd	1.14	22	
xl2tpd	3.52	10	NEW

All vulnerabilities have been responsibly disclosed to the respective project maintainers

AmpFuzz - Summary



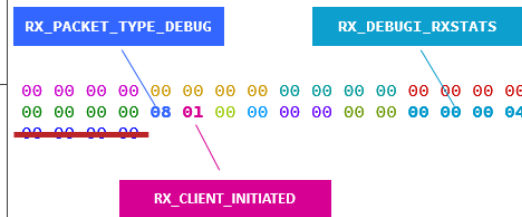
- Transform into one-shot program?
1. Identify **source** functions
 2. Identify **sink** functions
 3. Find edges from a **source** that **cannot** reach a **sink** without passing another **source** first
 4. Redirect those edges



- 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	
bossserver	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