A P4-Based Content-Aware Approach to Mitigate Slow HTTP POST Attacks

EuroP4 - December 9, 2022

Chih-Yu Hsieh,
Hong-Yen Chen,
Shan-Hsiang Shen,
Chen-Hsiang Hung,
Tsung-Nan Lin

國立臺灣大學
National Taiwan University

臺科大
TAIWAN TECH

Agenda

- Introduction
- Proposed Method
- Experiments and Results
- Conclusion

Slow HTTP DDoS Attacks

Slow HTTP DDoS attacks disturb services by occupying server threads with

- HTTP headers: slowloris / slow header
- HTTP body: slow POST / slow body / RUDY

Sending body simulates **realistic file upload**

```
POST /posts HTTP/1.1                                  start-line
Host: 10.0.1.1                                        headers
User-Agent: Mozilla/4.0
Content-Length: 7
Content-Type: application/x-www-form-urlencoded

foo=bar                                               body
```

HTTP request example

	slowloris	slow POST
segment	HTTP header	HTTP body
expected size	small	large

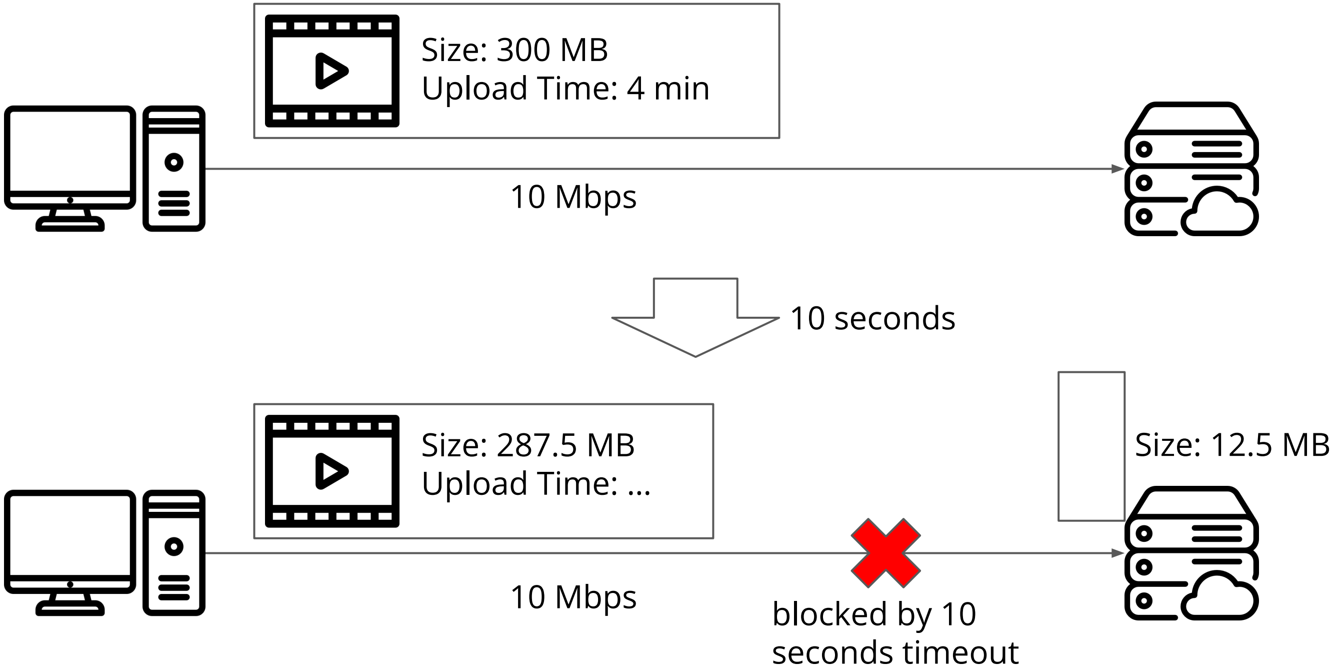
Challenge of Detection

- How to distinguish attackers from clients correctly in various network activities?
  - Viewing websites
  - Uploading photos / videos
  - Filling forms
  - Slow HTTP attack
- Existing works
  - timeout methods [1-3]
  - credibility method [4]
- **False positives** make legitimate users suffer from denial-of-service

[1] J. Park, K. Iwai, H. Tanaka, and T. Kurokawa, “Analysis of slow read dos attack and countermeasures on web servers,” International Journal of Cyber-Security and Digital Forensics, vol. 4, no. 2, pp. 339–353, 2015.
[2] T. Hirakawa, K. Ogura, B. B. Bista, and T. Takata, “A defense method against distributed slow http dos attack,” in 2016 19th International Conference on Network-Based Information Systems (NBiS), 2016, pp. 152–158.
[3] K. Hong, Y. Kim, H. Choi, and J. Park, “Sdn-assisted slow http ddos attack defense method,” IEEE Communications Letters, vol. 22, no. 4, pp. 688–691, 2018.
[4] Y.-C. Wang and R.-X. Ye, “Credibility-based countermeasure against slow http dos attacks by using sdn,” in 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), 2021, pp. 0890–0895.

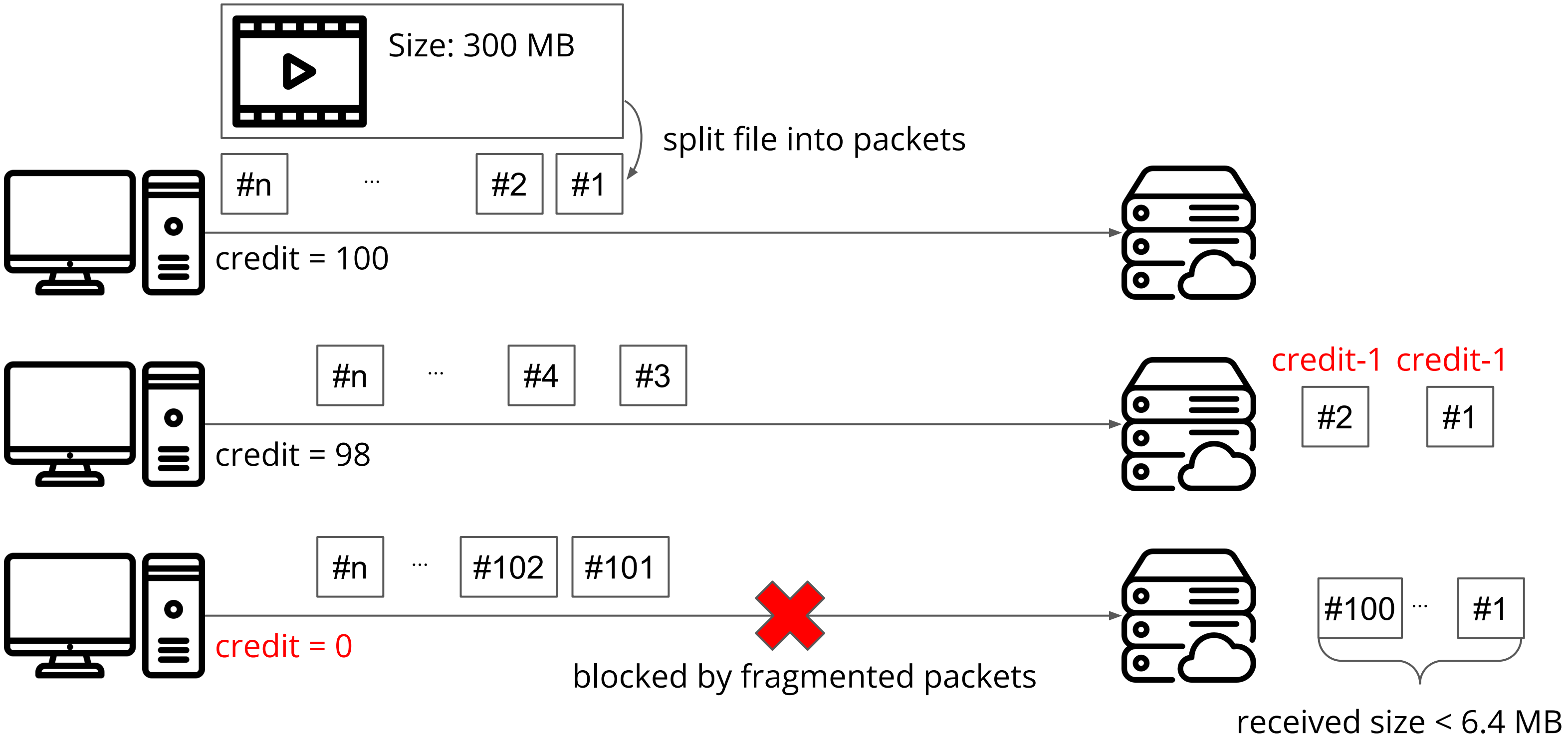
Timeout-based Defense Mechanism

Files are corrupt because the user cannot finish uploading within the timeout.



Credibility-based Defense Mechanism

The file to upload cannot be completed within the specified number of packets



Contribution

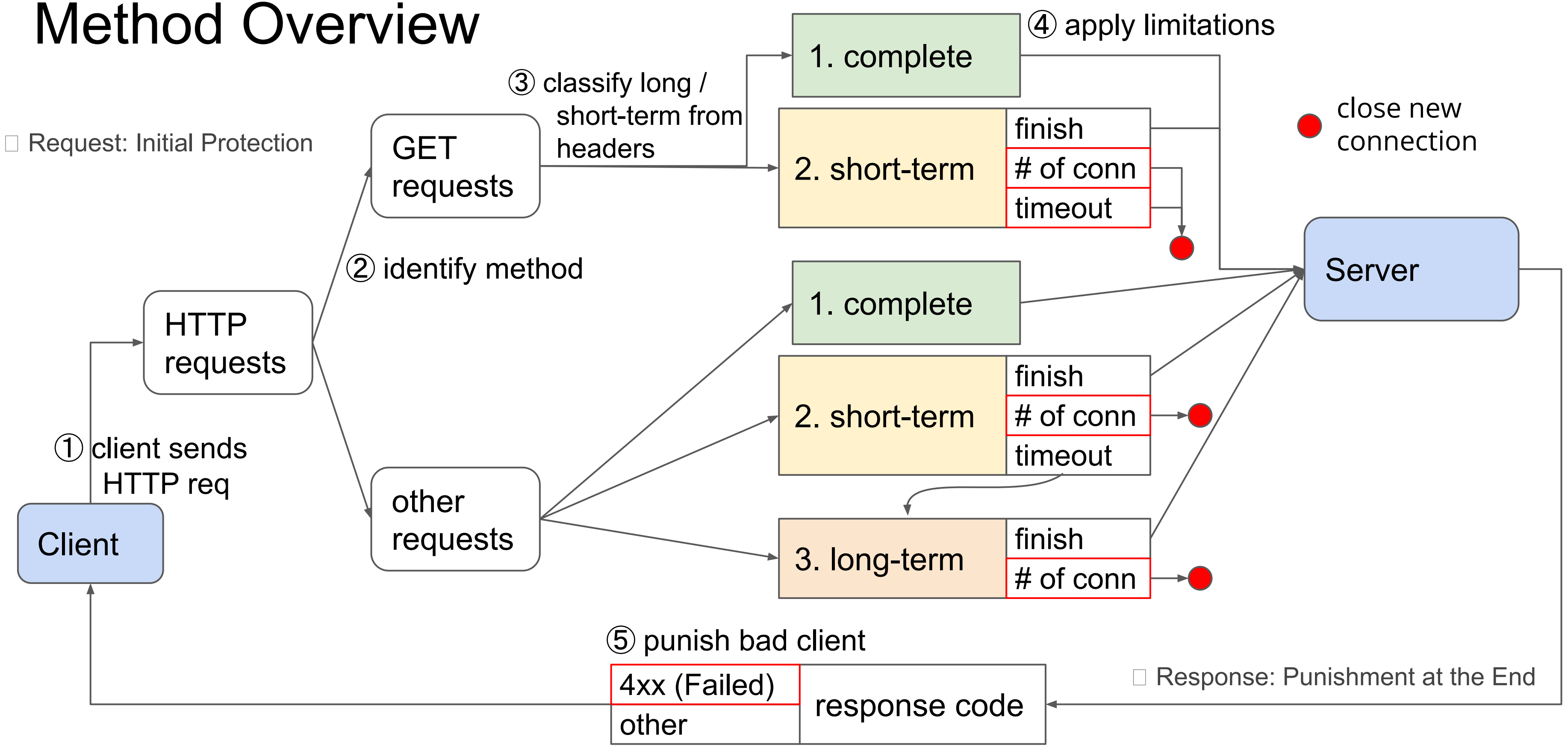
Our proposed method, RASP, is an open source[1], P4-based **content-aware** countermeasure.

- **High accuracy**: overcome the false positive issue by utilizing HTTP information
- **Scalable** deployment by P4
  - Application-layer headers processing is distributed to switches
  - Quantifies network usage savings
- Demonstrates the ability of P4 to parse variable-length header fields

---

1 https://github.com/doraeric/p4-rasp

Method Overview



Initial Protection

- Limitation per client per category
  - complete: none
  - short-term: number of requests is 8, connection time < 10 seconds
  - long-term: number of requests is 4
- Close excess connections and keep old ones.

  The user needs to finish old requests first.

Punishment at the End

- HTTP status code can indicate whether a request is successful.
  - **2xx**: the backend processes the request without error
  - **4xx**: the request failed due to client error (malformed / invaild request)
- Punishment is to decrease the number of allowed connections.

```
HTTP/1.1 200 OK
Server: Apache/2.4.25 (Debian)
Content-Type: text/html
...
```

A good HTTP response.

```
HTTP/1.1 400 Bad Request
Server: Apache/2.4.25 (Debian)
Content-Type: text/html
...
```

A bad HTTP response.

Implementation

- Control plane
  - manage connection state
- Data plane
  - **parse HTTP** headers
  - manage the number of open connections with **register**
  - report to controller with **digest** messages

```
struct headers_t {
    char_header_t[200] http_buffer;
}
```

Experiments - Simulation Scenario

We simulate different usage scenarios to verify the robustness of RASP:

1. short GET: slow client **viewing websites** under a slow header attack
2. long non-GET: clients **uploading** several **photos** under a slow POST attack
3. short non-GET: clients **uploading GPS** locations under a slow POST attack.

We investigate

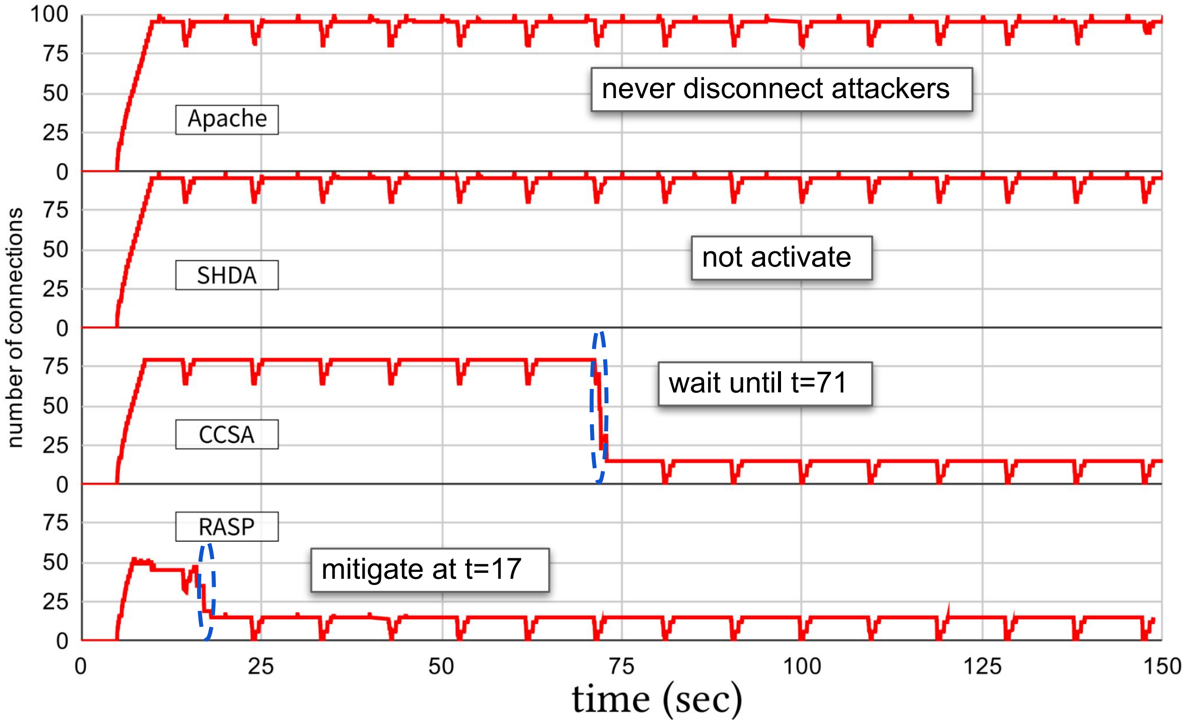
- the number of successful requests the clients send
- reduction in network usage by adopting P4

Experiment with BMv2

1. Slow Header Attack

Short-term GET clients under slow header attack

Our proposed RASP mitigates attacks **earlier** by sending TCP RST.



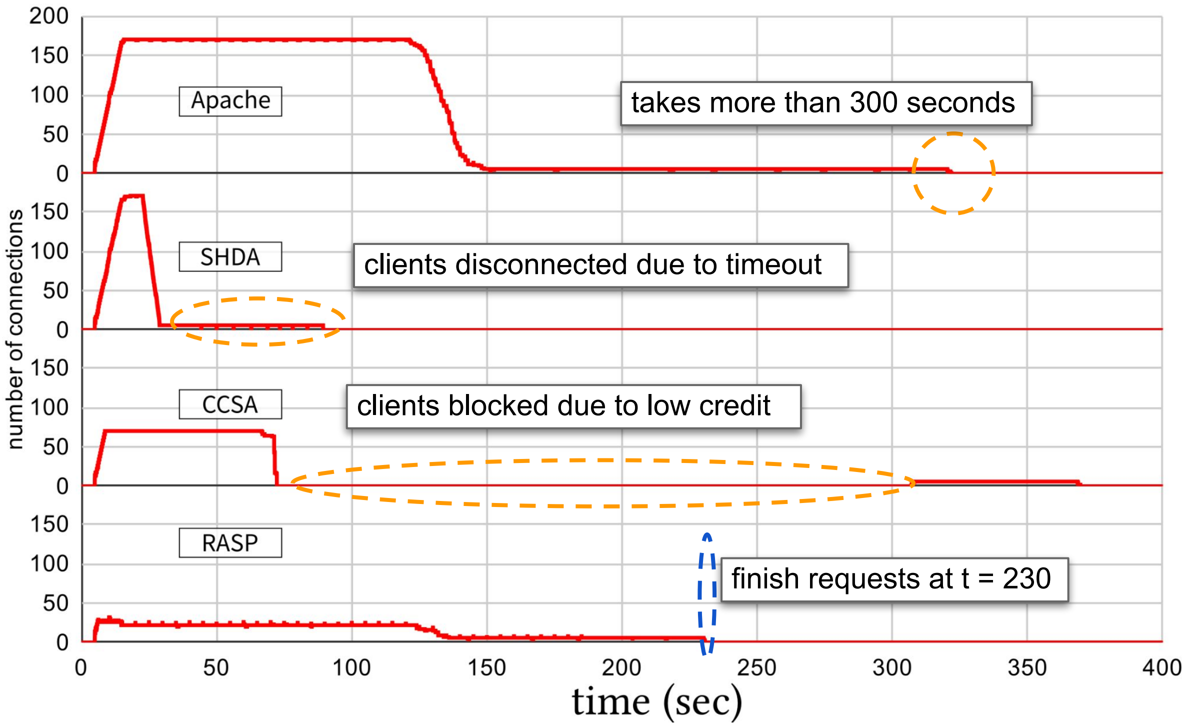
2. POST Photos

Long-term non-GET clients under slow POST attacks

RASP **correctly** completes all client requests in time

Table. Received files by backend

Method	receive bytes	complete files
SHDA	43.3 MB	0
CCSA	1.7 MB	0
RASP	**129 MB**	**60**



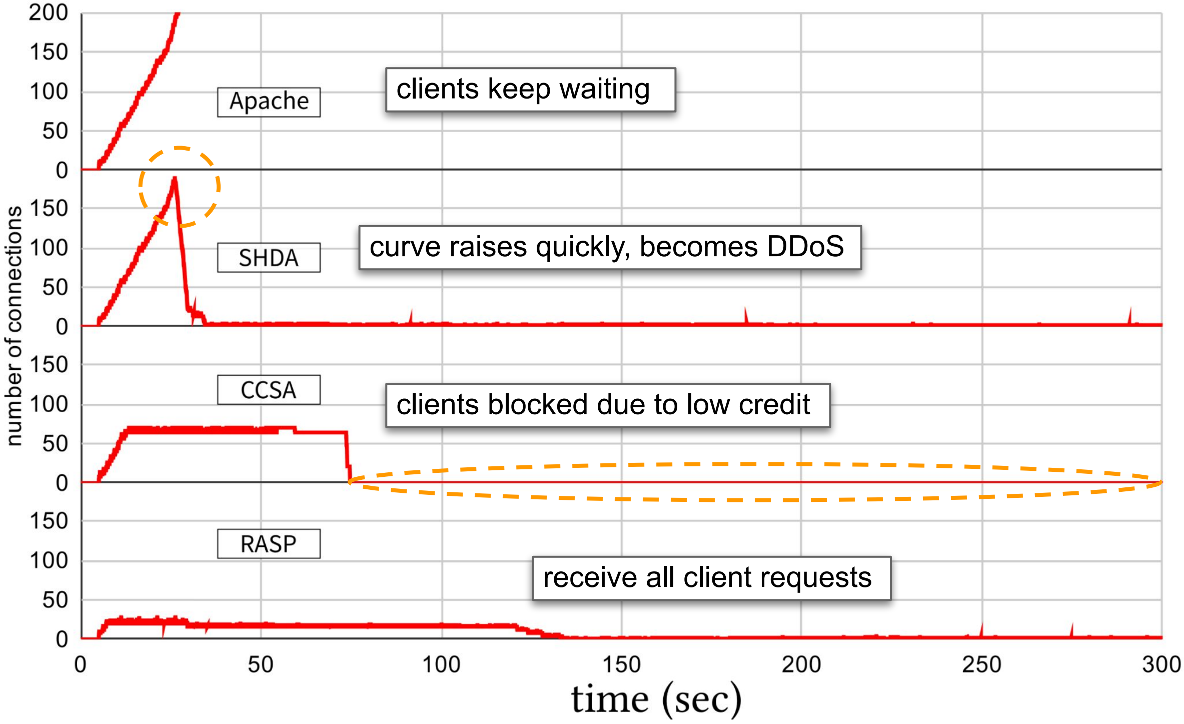
3. Upload GPS Locations

Short-term non-GET clients under slow POST attacks

RASP **correctly** protects clients from DDoS attacks.

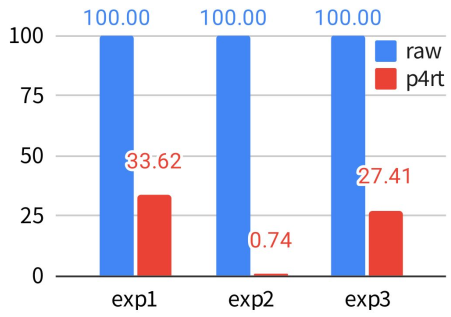
Table. Received requests by backend

Method	# of req	success
SHDA	1782	99%
CCSA	300	16.7%
RASP	**1800**	**100%**

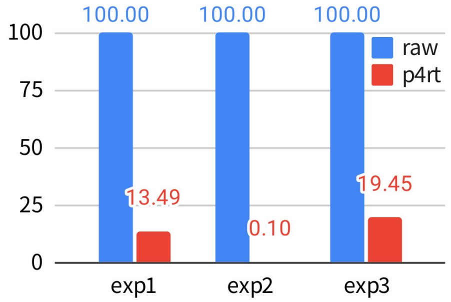


Network Usage

- Send smaller digests messages than raw packets
  - Raw: between switches and **clients**, including attackers
  - P4RT: degest messages between switches and **controller**
- Digest message (P4) compared to raw packets (OpenFlow)
  - Number of packets -> approximately **30%**
  - Number of bytes -> **20%**
- Larger HTTP body benefit more (exp2, 0.74% / 0.1%)



(a) Packets sent in ratio (%).



(b) Bytes sent in ratio (%).

Conclusion

- We propose RASP, a defense mechanism against slow HTTP POST DDoS attacks. RASP utilizes new information from **application-layer headers** to implement more delicate control.
- RASP achieves more **accurate** detection than that in previous work under realistic simulations.
- It is implemented on the highly **programmable P4**, which provides potential for future development. Other plaintext-based protocols like HTTP, may be applied in similar approaches.

Thank you!

Chih-Yu Hsieh
r09921a17@ntu.edu.tw

github.com/doraeric/p4-rasp