UNIVERSITY OF WATERLOO

ERICSSON

A Security Orchestration System for CDN Edge Servers

ELAHEH JALALPOUR
MILAD GHAZNAVI
DANIEL MIGAULT
STERE PREDA
MAKAN POURZANDI
RAOUF BOUTABA

Outline

Introduction

Edge Server Security Orchestration

Implementation

Evaluation

Conclusion

Introduction

CONTENT DELIVERY NETWORK (CDN)

CONTENT DELIVERY PROCEDURE

ATTACKS AGAINST CDN EDGE-SERVERS

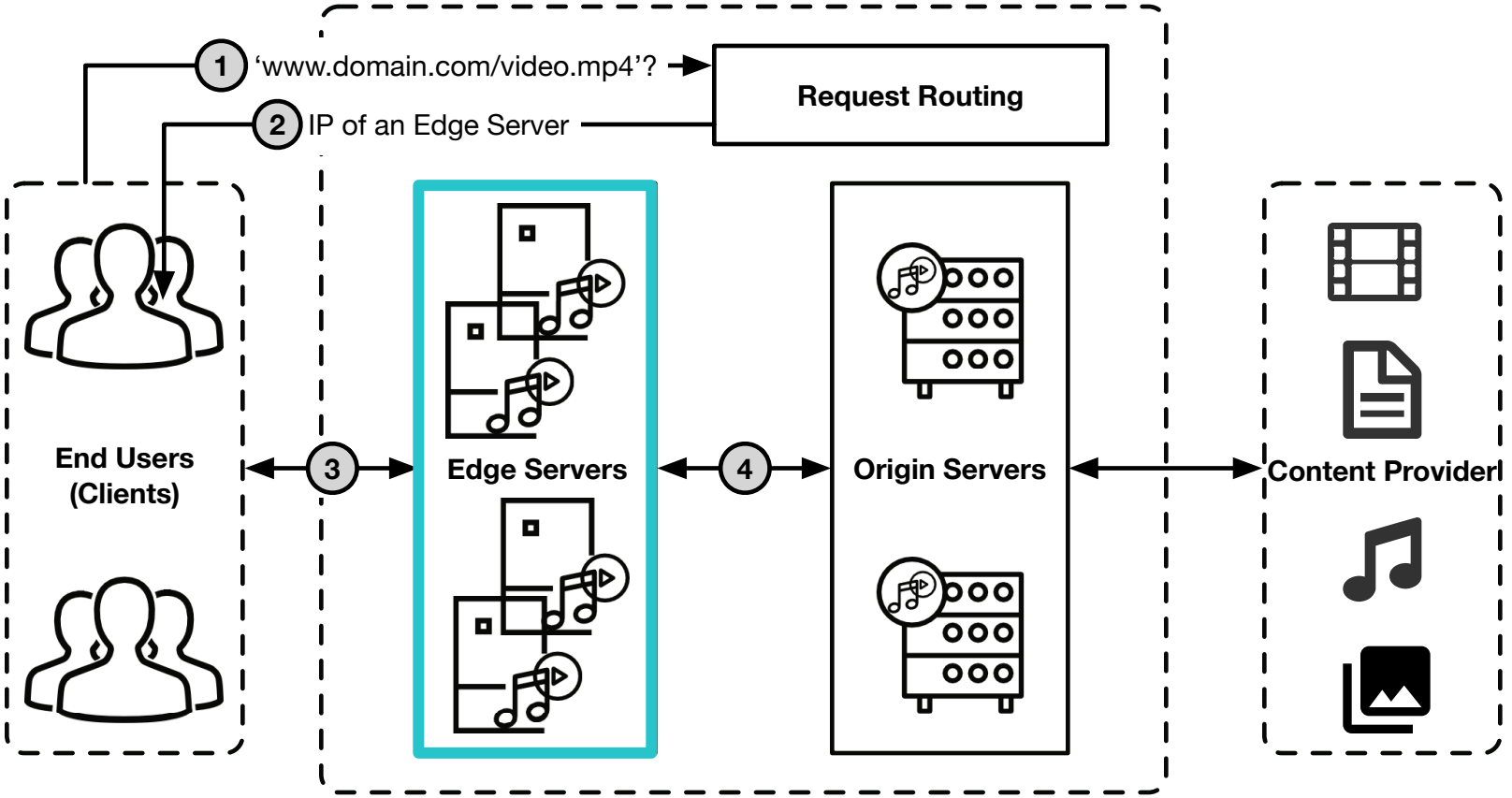
CURRENT DEFENSE MECHANISMS

Content Delivery Network

Content Delivery Network (CDNs) play a critical role in delivering digital content

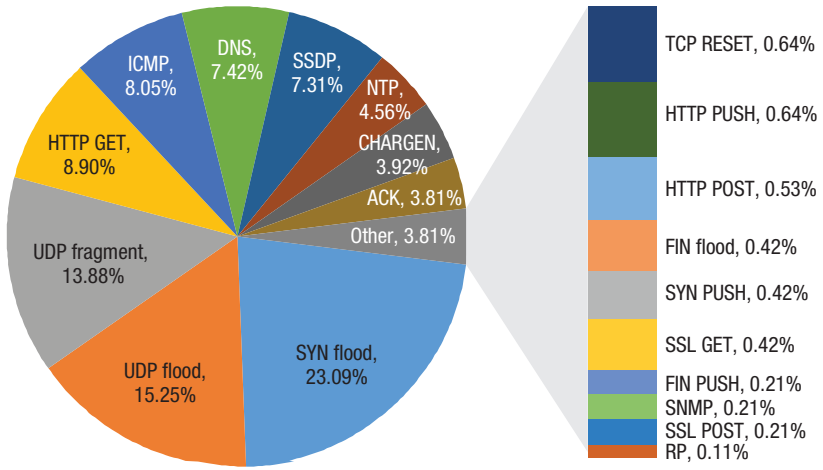
- Open-Connect carries Netflix's content (35.2 of all the traffic across North America)
- Akamai CDN daily delivers more than 30 Tbps of traffic

Content Delivery Procedure

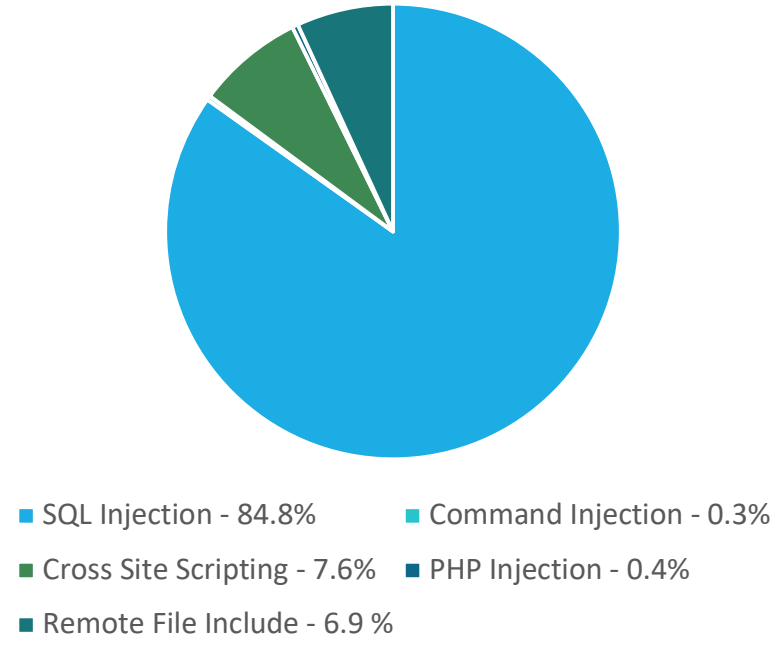


Attacks against CDN Edge Servers

DENIAL OF SERVICE ATTACKS



APPLICATION LAYER ATTACKS



[1] Gillman, D., Lin, Y., Maggs, B. and Sitaraman, R.K., 2015. Protecting Websites from Attack with Secure Delivery Networks. *Computer*, *48*(4), pp.26-34.

Current Defense Mechanisms

Hardware Security Functions
- Vertically integrated
- Not Elastic/flexible

Scrubbing Centers
- Redirection latency
- Proprietary mechanisms

Existing Software Defined Solutions
- Not-automated deployment
- Exclusive to DDoS

Edge Server Security Orchestration

OUR APPROACH

ARCHITECTURE

Our Approach

Virtual Security Functions
- Virtual functions running on commodity hardware
- Elastic/flexible

In-house Mitigation
- No redirection latency
- Custom mechanisms

Our Software Defined Solutions
- Automated deployment
- Wide range of attacks

Hardware Security Functions
- Vertically integrated
- Not elastic/flexible

Scrubbing Centers
- Redirection latency
- Proprietary mechanisms

Existing Software Defined Solutions
- Not-automated deployment
- Exclusive to DDoS

Our Approach

Deploying security services on edge servers

Dynamic and automatic deployment of security services

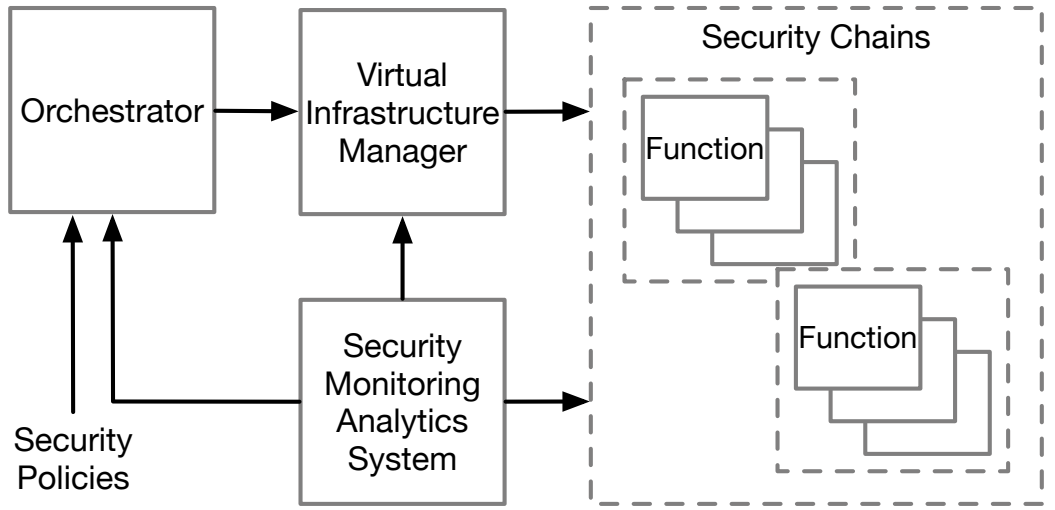
Security services realized through **service function chaining**

Architecture

Orchestrator
- Enforcing high-level policies
- Reacting to environment states and attacks

Virtual Infrastructure Managers
- Creating, updating, querying, and deleting security chains

Security Monitoring Analytics System
- Monitoring and analyzing the collected data
- Feeding the orchestrator with alerts



Implementation

ORCHESTRATOR

VIRTUAL INFRASTRUCTURE MANAGERS

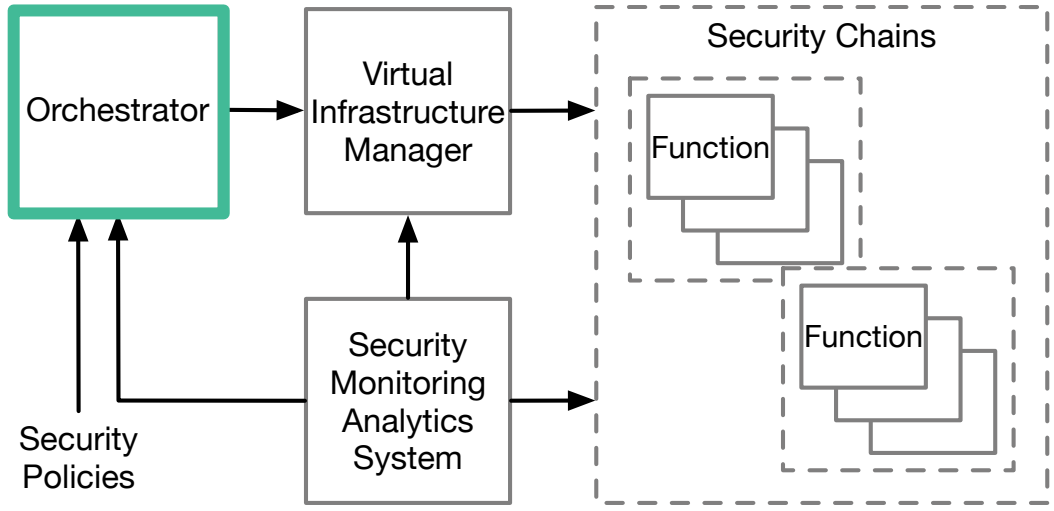
SECURITY MONITORING ANALYTICS SYSTEM

Orchestrator

Adopting $\mathcal{L}_{active}$ language

Event Condition Action paradigm

- Event
  - Security alerts generated by SMAS
  - Internal events
- Condition
  - Time related
  - Service related
  - Traffic related
- Action
  - Creating, deleting, modifying a chain



Orchestrator – Continue

SECURITY POLICIES

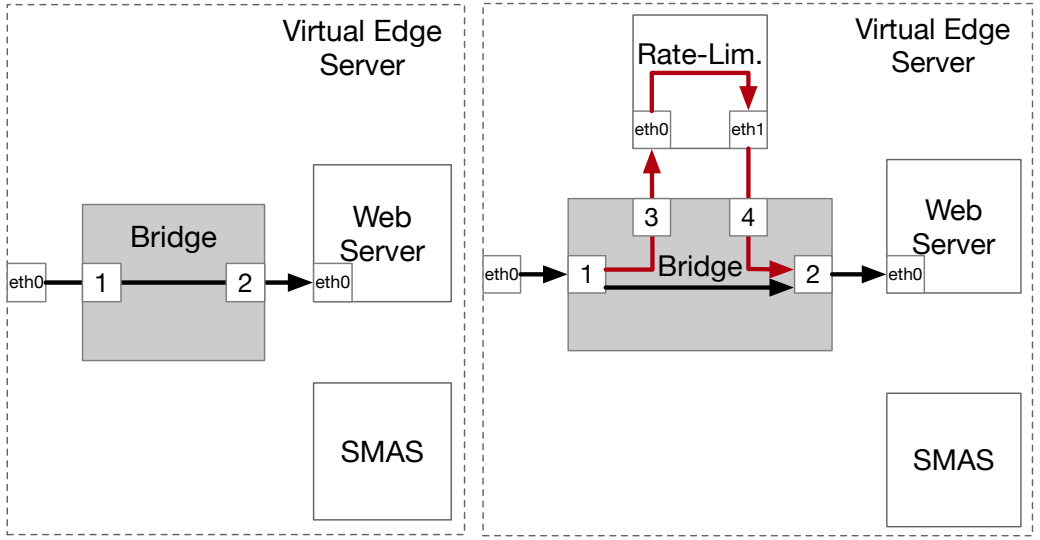
Deploy rate limiting chain in reaction to event *high_rate* happened and there are no rate limiting chain

Trigger event *lim* if the rate limiting chain is deployed

Run `rate_limit.sh` in response to event *lim*

$$
\begin{aligned}
&high\_rate \textbf{ initiates } create\_chain(r: \\
&\quad <\text{“not src net 129.97.124.0/24”}, 1, 2>, \\
&\quad \{f\text{:}Rate\text{-}limit\}) \\
&\quad \textbf{if not } chain(r) \\
&lim \textbf{ after } create\_chain(r) \\
&\quad \textbf{if true} \\
&lim \textbf{ initiates } run(f, \text{“}rate\_limit.sh\text{”}) \\
&\quad \textbf{if true}
\end{aligned}
$$

ENFORCED SECURITY CHAIN

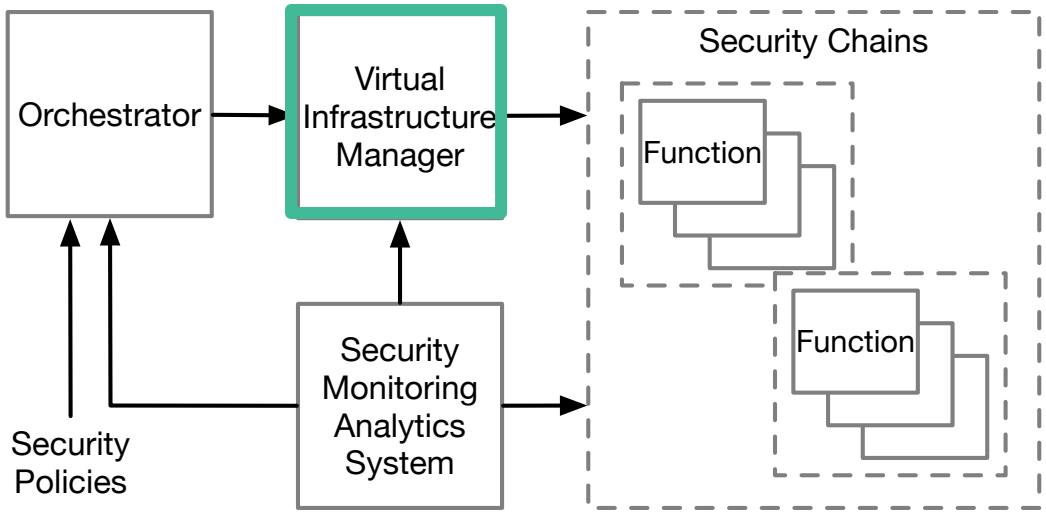


Virtual Infrastructure Manager

Docker
- Containers as service functions

Network Service Header (NSH)
- IETF standard for service function chaining
- Realizing service function paths
- Supporting carrying metadata

Open Virtual Switch
- Software based switching



Virtual Infrastructure Manager – Cont.

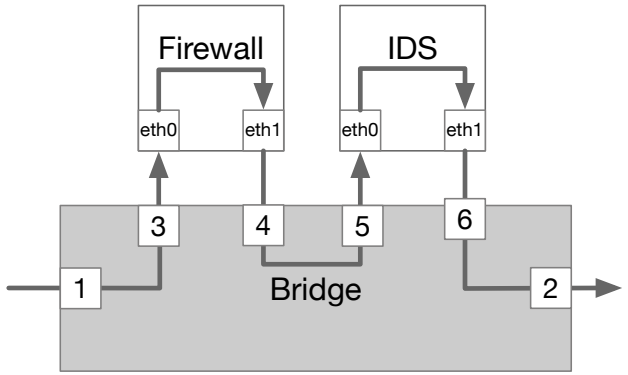
VIM APIs

```
def create_chain(chain_sp)
def delete_chain(chain_name)
def insert(chain_name, func_sp)
def delete(chain_name, func_name)
def run(func_name,cmd)
def chains()
def chain(chain_name)
def chain_functions(chain_name)
def functions()
def function(func_name)
def steered(bpf,chain_name)
```

Chain Specification

```
{
  "chain_name": "ch",
  "ingress": "1",
  "egress": "2",
  "classification_rules": "ip",
  "functions": [
    {
      "function_image": "Firewall",
      "function_name": "firewall",
      "nsh_aware": false
    },
    {
      "function_image": "IDS",
      "function_name": "ids",
      "nsh_aware": false
    }
  ]
}
```

Deployed Chain

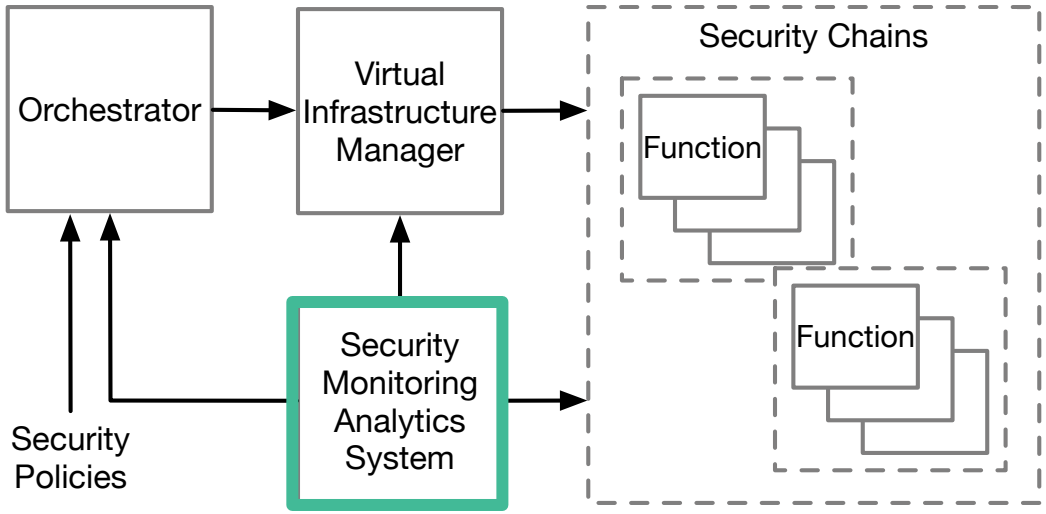


Security Monitoring Analytics System

Periodic resource monitoring
- Network-bandwidth
- CPU
- Memory
- Storage

Event generation
- Based on predefined thresholds

Standard Linux commands
- `/proc/stat`
- `free`
- `iostat`



Evaluation

ENVIRONMENT SETUP

PERFORMANCE EVALUATION

RESPONSIVENESS

DYNAMIC SECURITY SERVICE

Environment Setup

A cluster of servers
- 16 GB RAM
- 8-cores 3.30 GHz Xeon CPU
- 10 Gbps NIC

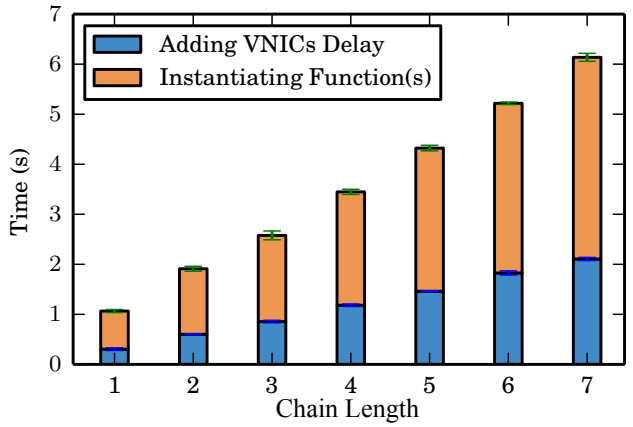
Device under test
- Hosting security chains
- Hosting an active daemon of our system

Traffic sink
- `iperf` server
- `Apache` Web Server

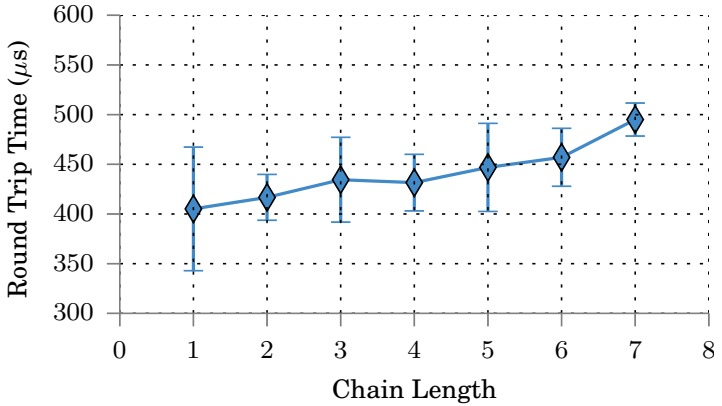
Traffic generator
- `iperf` client
- `HTTPERF`

Performance evaluation

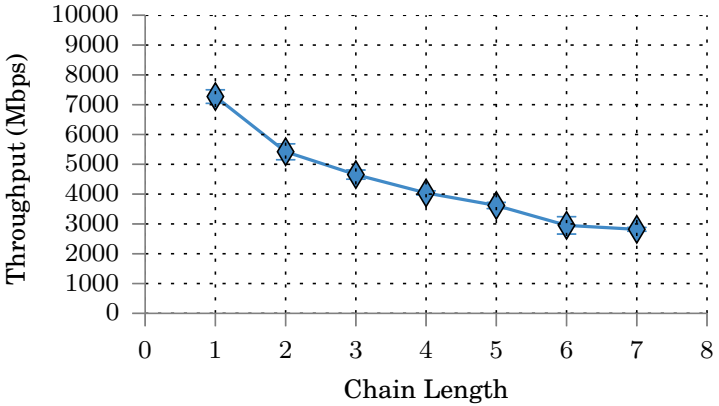
Chain Deployment Time



Traffic Round Trip Time



Throughput vs. Chain Length



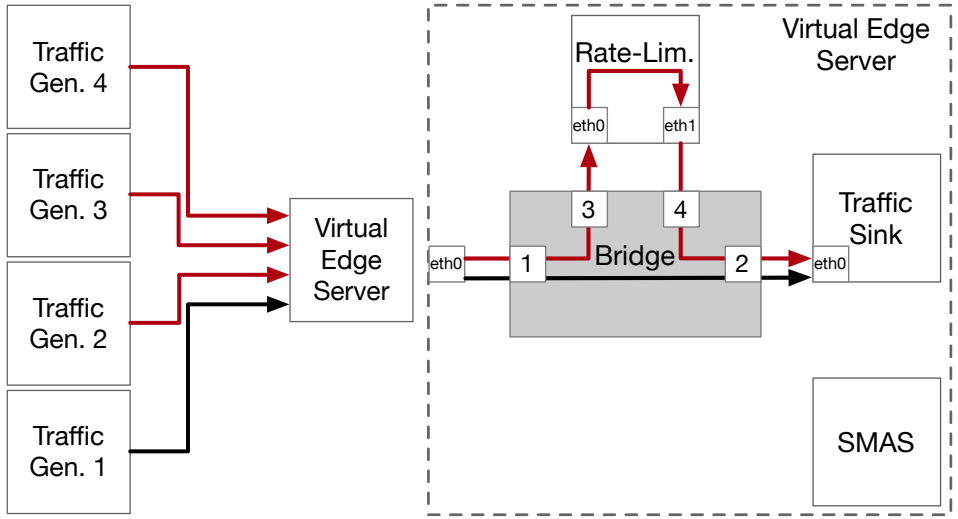
Responsiveness

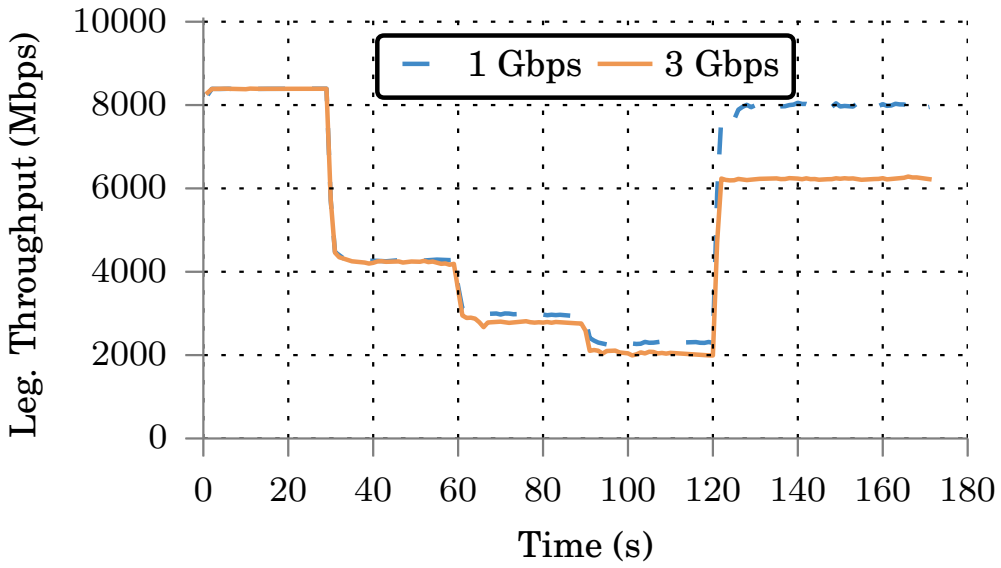
Traffic

- TCP flooding attacks

Defense

- Network layer rate-limiting





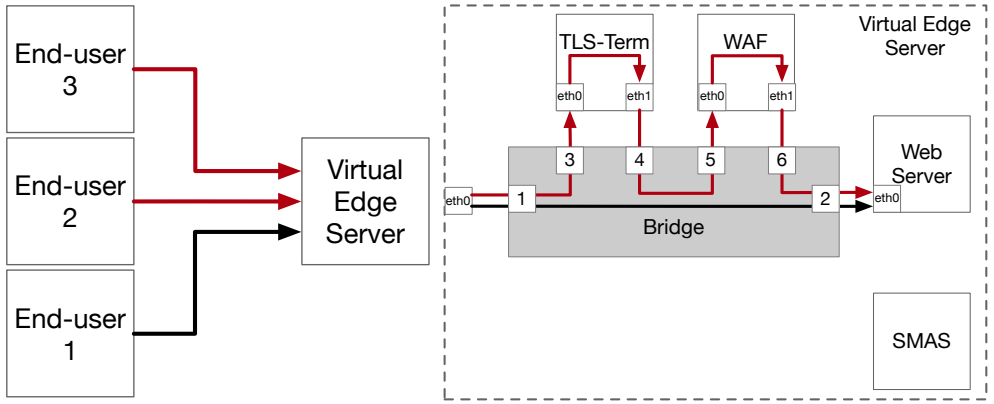
Dynamic Security Service

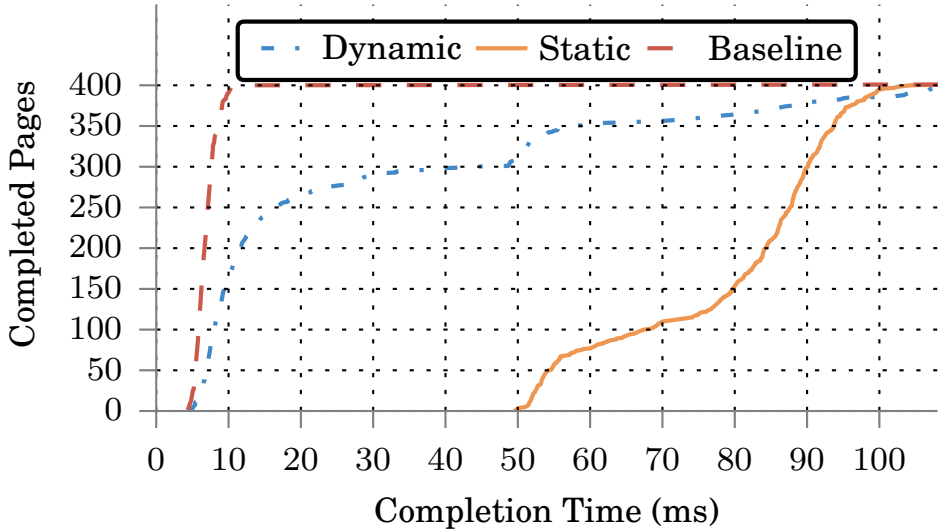
Traffic

- 300 legitimate requests
- 100 suspicious requests

Defense

- Application layer mitigation





Conclusion

SUMMARY

FUTURE WORK

Summary

Software defined security orchestration for CDN edge-servers

Governed by high-level policies

Dynamic and automatic security function chaining

Future Work

NSH compatible SFs

- Passing the metadata between functions

Ensuring security policy consistency through formal verifications

- Free of conflicting rules

Reduce the signaling overhead in the Orchestration process

- Delegation of part of the SF chain management

Q&A