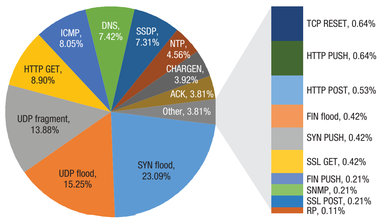
Dynamic Security Orchestration for CDN Edge-Servers

Elaheh Jalalpour, Milad Ghaznavi, Daniel Migault, Stere Preda, Makan Pourzandi, and Raouf Boutaba
David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada
Ericsson Research, Montreal, QC, Canada

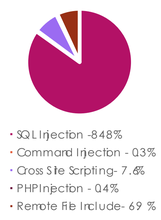
Introduction and Motivation

- Content Delivery Networks (CDNs) provide high QoE in delivering digital content
- CDNs cache content in *edge-servers* in the vicinity of end-users
- Attacks against CDN edge-servers deteriorate QoE
- Limitations of current defense mechanisms
  - Hardware security functions
    - Expensive, not elastic, not flexible
  - Scrubbing centers
    - Redirection latency, proprietary mechanisms
  - Current software solutions
    - Massive changes in infrastructure, not-automated deployment, exclusive to DDoS

Denial of Service Attacks
Distribution by Type



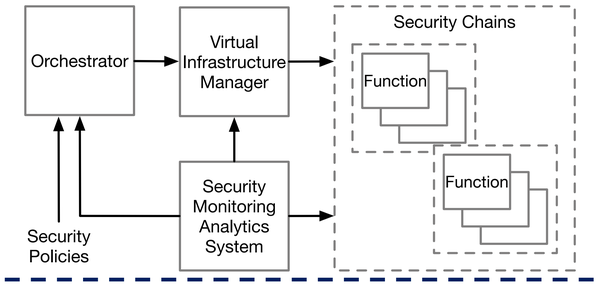
Application Layer Attacks
Distribution by Type



- SQLInjection -848%
- Command Injection - 03%
- Cross Site Scripting- 7.4%
- PHPInjection - 04%
- Remote File Include- 69 %

Dynamic Security Orchestration

- **Automatic** and **dynamic** deployment and **flexible** modification of security services
  - Instantiating and modifying **security chains** in reaction to **events**
- Inspecting only **suspicious** traffic by security services
  - **Classifying** traffic at the beginning of the chains



Demo 1: Network Layer Rate Limiting

Context

- **Attack**
  - TCP flooding attacks
- **Impacts**
  - Exhausting bandwidth resources
  - Making service unavailable
- **Defense**
  - Per IP traffic rate-limiting

Security Policy

too_many_con **initiates** $create\_chain(r:$
<"not src net 129.97.124.0/24", 1, 2>,
$\{f{:}Rate\text{-}Lim.\})$
if not $chain(r)$
lim **after** $create\_chain(r)$
if true
lim **initiates** $run(f, \text{“}rate\_limit.sh\text{”})$
if true

Demo Setup

- A cluster of machines
  - 16~GB RAM
  - 8-cores 3.30~GHz Xeon CPU
  - 10~Gbps NIC.
- Device under test
  - Hosting security chains
  - Hosting our system
- Traffic generators: `iperf client`
- Traffic sink: `iperf server`

Demo 2: Application Layer Rate Limiting

Context

- **Attack**
  - HTTP flooding attacks
- **Impacts**
  - Exhausting processing resources
  - Degrading quality of experience
- **Defense**
  - Per IP, per request rate-limiting

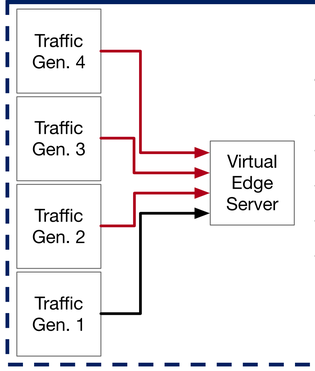
Security Policy

$suspicious\_ip$ **initiates** $create\_chain(l:$
"not src net 99.231.0.0/16", 1, 2,
$\{t : TLS\text{-}Term, w : WAF\}$)
if not $chain(l)$

Demo Setup

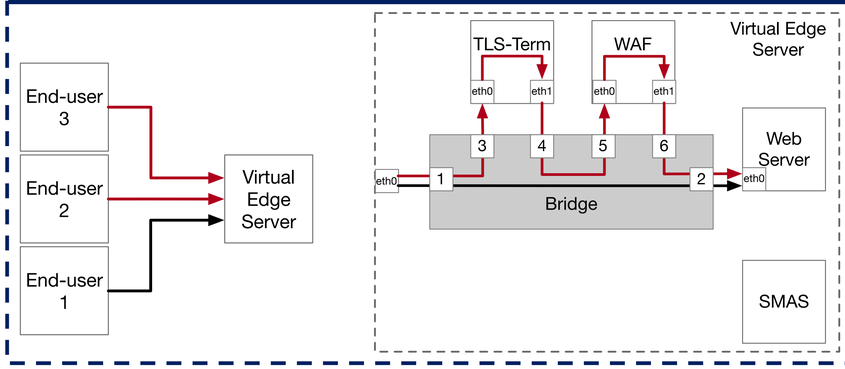
- A cluster of machines
  - 16~GB RAM
  - 8-cores 3.30~GHz Xeon CPU
  - 10~Gbps NIC.
- Device under test
  - Hosting security chains
  - Hosting our system
- Traffic generators: `VLC`, `curl`
- Traffic sink: `Apache server`

Attack Emulation



Stage	Flooding traffic share	Active traffic generators
1	0%	Traffic Gen. 1
2	50%	Traffic Gen. 1 and 2
3	66.6%	Traffic Gen. 1, 2, and 3
4	75%	Traffic Gen. 1, 2, 3, and 4

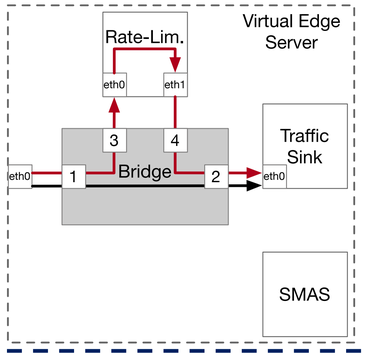
Attack Emulation and Mitigation Chain

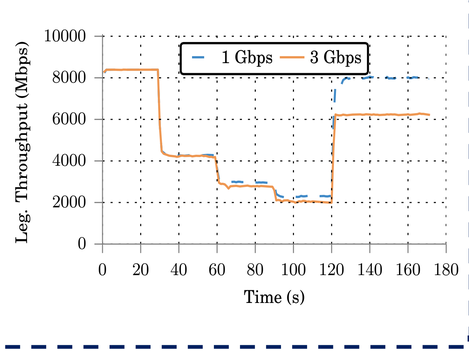


Architecture Components

- **Orchestrator**
  - Programed by policies scripted in an adapted version of $\mathcal{L}_{active}$ language
  - Reacts to environment states and threats
- **Virtual Infrastructure Manager (VIM)**
  - Manages the resources of a virtual edge-server
  - Provides a north-bound API for Orchestrator and SMAS
- **Security Monitoring Analytics System (SMAS)**
  - Monitors and analyzes the collected data
  - Feeds the orchestrator with alerts that may trigger security actions

Mitigation Chain





Conclusion

- We demonstrated a configurable security system that protects CDN edge-servers
- This system behavior is governed by high-level policies
- The deployment of security function chains is dynamic and automatic
- We illustrated how our system can be flexibly programmed to mitigate real-world threats
- In first demonstration, our system mitigates a network layer flooding attack
  - Deploying a chain of a rate-limiting function recovering legitimate traffic
- In second demonstration, an application layer abusive behavior is rate-limited
  - Deploying a chain of a TLS termination and a WAF to rate-limit abusive requests