Temporal CDN-Convex Lens
A CDN-Assisted Practical Pulsing DDoS Attack

Run Guo, Jianjun Chen, Yihang Wang, Keran Mu, Baojun Liu, Xiang Li, Chao Zhang, Haixin Duan, Jianping Wu

Tsinghua University
奇安信
ZGC Lab
Outlines

• Background
• Attacks
• Mitigations
• Conclusion
A warm-up wargame

**Limited Attack Capability**
- 1 artilleries
- 1 shell per minute per unit

**Defense Capability**
- Blast Resistance
  - < 5 shells within 1 minute

1 shells “at max” within 1 minute
if all artilleries fire at the same time

Mission Impossible?
Multiple Round Simultaneous Impact (MRSI)

- **MRSI** is when a single gun fires multiple shells so all arrive at the same target simultaneously.
- A variation of military tactic “**Time on Target (TOT)**” in World War I.
Advantages of MRSI

**Efficiency**
- Attacker
  - just fire the shells slowly
- Victim
  - receive all shells instantly

**Stealth**
- Observe one of the attacker’s artilleries alone, the rate of fire is pretty **low**
- The alarm won’t be triggered

**Prime Target**

“Moments to go down, hours to recover”
DoS a target with a **limited** bandwidth?
Previous Attack: Abusing DNS Infrastructure [1]

Attacker → DNS queries with IP Source Spoofing → Global Open DNS Servers → Reflected DNS responses → Target

<table>
<thead>
<tr>
<th>Properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectories</td>
<td>≥ Thousands of Open DNS</td>
</tr>
<tr>
<td>Flight time of payload</td>
<td>≤ 700 milliseconds</td>
</tr>
<tr>
<td>Bandwidth Concentration Ratio</td>
<td>≈ 14</td>
</tr>
</tbody>
</table>

Our Work: CDN-Convex Lens Attack

HTTP Requests to globally-distributed CDN edge Servers → CDN Edge Servers → CDN-Forwarded HTTP Requests → Target

<table>
<thead>
<tr>
<th>Properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trajectories</td>
<td>≥ Millions of CDN edge servers</td>
</tr>
<tr>
<td>Flight time of payload</td>
<td>≥ 5,400,000 milliseconds</td>
</tr>
<tr>
<td>Bandwidth Concentration Ratio</td>
<td>≥ 1000</td>
</tr>
</tbody>
</table>
What is a Content Delivery Network (CDN)?

- **Globally Distributed**: a large volume of servers on Internet backbone
- **Cache then Forward**: act as the Reverse Proxy to the website
- **Proximity Service**: redirect the user’s request to the nearest server
- **DDoS Protection**: off-load traffic from botnet-based DDoS attack
What is a Pulse Wave DDoS attack?

- **Efficiency**: Periodical Saturation of Bottleneck Resources
- **Stealthy**: High-rate, short-lived bursts
- **Unusual on Internet**
  - Require a botnet
  - Botnet is preferably used to launch simple flooding attack
Outlines

- Background
- **Attacks**
- Mitigations
- Conclusion
Concept of CDN-Convex Lens Attack

Attacker

CDN Global Edge Servers

Victim

Send requests sequentially and slowly

Requests converged as a pulse wave

Low Bandwidth

High Concentrated Bandwidth
Attack Steps

- **Step I**: CDN Node Harvest
- **Step II**: Configure CDN to Point to the Victim
- **Step III**: Measure the flight time (latency)
- **Step IV**: Bypass the cache mechanism
- **Step V**: Send the requests on time
Step I: CDN Node Harvest

- collect IP addresses of global CDN edge servers by
  - Internet-wide **scanning** / **fingerprinting**

<table>
<thead>
<tr>
<th>CDN</th>
<th>Fingerprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloudfront</td>
<td>X-Amz-Cf-Id</td>
</tr>
<tr>
<td>Cloudflare</td>
<td>Cf-Cache-Status</td>
</tr>
<tr>
<td>FrontDoor</td>
<td>X-Azure-Ref</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
CDN Edge Servers can be abused by the Attacker

- Tons of edge servers can be abused by the attacker
  - CDN edge servers are allowed to forward HTTP requests with a valid host header

<table>
<thead>
<tr>
<th>Host Header</th>
<th>Origin Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.com</td>
<td>1.2.3.4:8080</td>
</tr>
<tr>
<td>b.com</td>
<td>origin-b.com</td>
</tr>
</tbody>
</table>

CDN Internal Forwarding Tables
Step II: Configure CDN to Point to the Victim

- **Register** CDN services, then **config** the victim website as a origin server.

<table>
<thead>
<tr>
<th>Host Header</th>
<th>Origin Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>attacker.com</td>
<td>1.2.3.4:8080</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

GET / HTTP/1.1
Host: attacker.com

CDN Global Edge Servers

Victim
1.2.3.4:8080
CDN Lacks of Origin Ownership Verification

- CDN **lacks of ownership verification** for the Origin Server
- CDN can be configured to fetch resource from any IP and any port

User

```
GET / HTTP/1.1
Host: a.com
```

<table>
<thead>
<tr>
<th>Host Header</th>
<th>Origin Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.com</td>
<td>1.2.3.4:8080</td>
</tr>
<tr>
<td>b.com</td>
<td>origin-b.com</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CDN Internal Forwarding Tables

CDN Global Edge Servers

1.2.3.4:8080

origin-b.com
Step III: Measure the flight time (latency)

- **Measure latencies** of CDN forwarding paths and filter stable ones.

![Diagram showing an attacker, CDN Global Edge Servers, and a victim.]
Step IV: Bypass CDN cache mechanism

- Craft request to **bypass CDN cache** and saturate the bottleneck resources

```
POST /bottleneck.php?nonce HTTP/1.1
Host: victim.com
Cache-control: no-cache
```
Step V: Send the requests on time

- Send low rate of the HTTP requests in accord with path latencies
The Pulsing-Wave is Coming!

- Requests **converged** as a **high-rate, short-lived** pulse burst to saturate target.
Result of the Basic CDN-Convex Attack

**Core Concept**
Use native path latency to arrange all HTTP requests

Max Bandwidth Concentration Ratio ~ 6

<table>
<thead>
<tr>
<th>CDN</th>
<th>Akamai</th>
<th>Azure</th>
<th><strong>CloudFront</strong></th>
<th>Cloudflare</th>
<th>Fastly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth Concentration Ratio</td>
<td>5.46</td>
<td>4.66</td>
<td><strong>6.42</strong></td>
<td>3.73</td>
<td>1.49</td>
</tr>
</tbody>
</table>
Why did we obtain a low concentration ratio?

- **Concentration ratio is limited by**
  - diversity of path latencies
  - the longest forwarding time (latency) of CDN global paths

- **Challenges**
  - How can we **Enlarge / Control** the forwarding time to allow more requests being buffered in CDN global paths?
Our Attacks: Exploit CDN Features to \textit{Enlarge} / \textit{Control} the forwarding time

\textbf{Attack \#1} Extend \textit{CDN Paths}: Cascaded CDN Convex Attack

\textbf{Attack \#2} Extend \textit{DNS Resolving} time: DNS-holdon Convex Attack

Incomplete packets being \textit{buffered} at CDN servers for a period of time

\textbf{Attack \#3} \textbf{IP-Fragmentation} Convex Attack

\textbf{Attack \#4} \textbf{Request-Pending} Convex Attack
Attack #1 Extend **CDN Paths**: Cascaded CDN Convex Attack

**Core Concept**

**Chain** more CDNs to **enlarge** the flight time

Max Bandwidth Concentration Ratio

~ 9
**Attack #2** Extend **DNS Resolving time**: DNS-holdon Convex Attack

**Core Concept**
Use **DNS query** by edge servers to control flight time

Max Bandwidth Concentration Ratio ~ 17
**Core Concept**
Use incomplete fragmented IP packages to control flight time

Max Bandwidth Concentration Ratio ~ 140

---

<table>
<thead>
<tr>
<th>CDN</th>
<th>Akamai</th>
<th>Azure</th>
<th>CloudFront</th>
<th>Cloudflare</th>
<th>Fastly</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Fragmentation Timeout</td>
<td>~30s</td>
<td>~30s</td>
<td>~30s</td>
<td>~15s</td>
<td>~10s</td>
</tr>
<tr>
<td>Bandwidth Concentration Ratio</td>
<td><strong>142.23</strong></td>
<td>118.35</td>
<td>72.62</td>
<td>48.66</td>
<td>21.63</td>
</tr>
</tbody>
</table>
Core Concept

Use **incomplete HTTP requests** to control flight time

Max Bandwidth Concentration Ratio

~ 4800

<table>
<thead>
<tr>
<th>CDN</th>
<th>Akamai</th>
<th>Azure</th>
<th>CloudFront</th>
<th>Cloudflare</th>
<th>Fastly</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP Forwarding Timeout</td>
<td>~ 16s</td>
<td>~ 1600s</td>
<td>~ 12s</td>
<td>≥ 3600s</td>
<td>~ 16s</td>
</tr>
<tr>
<td>Bandwidth Concentration Ratio</td>
<td>1426.38</td>
<td>4842.69</td>
<td>31.3</td>
<td>1786.37</td>
<td>988.48</td>
</tr>
</tbody>
</table>
Experiment Setup

- **Only 32** edge servers were used
- **Only 16MB × 32 = 512MB** data were sent
- **No impact** on other websites
  - the victim website is under our control
- Attacker Outbound-Bandwidth: ~7Mbps
- Victim Inbound-Bandwidth: ~100Mbps
  - Limited by the cloud provider (100Mbps at max)

Demo

The targeted website server is directly **out of service**

“Out of memory: Killed process apache2”
Outlines

• Background
• Attacks
• Mitigations
• Conclusion
Mitigations

• For CDN
  • Validate the ownership of customer-supplied origin configuration
  • Stop CDN being abused to attack 3rd party targets
  • Can still attack websites hosted on CDN
  • Fast forwarding of requests (#enhancement 4)
    • Forward on each byte of received request
  • Standardizing a unified head field to expose client IP
    • Filter or limit attacking traffic based on client IP

• For Victim
  • limit the request rate from the same client IP
Outlines

• Background
• Attacks
• Mitigations
• Conclusion
Conclusion

- We present a novel **the CDN-Convex attack** which uses CDN-Introduced delay distribution to launch a pulsing DDoS attack against any 3rd party TCP service
- **4 novel enhancement** for the impact from 2 aspects
  - Increasing network pathways (Cascaded CDN)
  - Controlling network latency (DNS-Holdon, IP-Fragmentation, HTTP-Holdon)
- **Bandwidth Concentration Ratio ≥ 1000**
Thank you for listening!

Q & A