

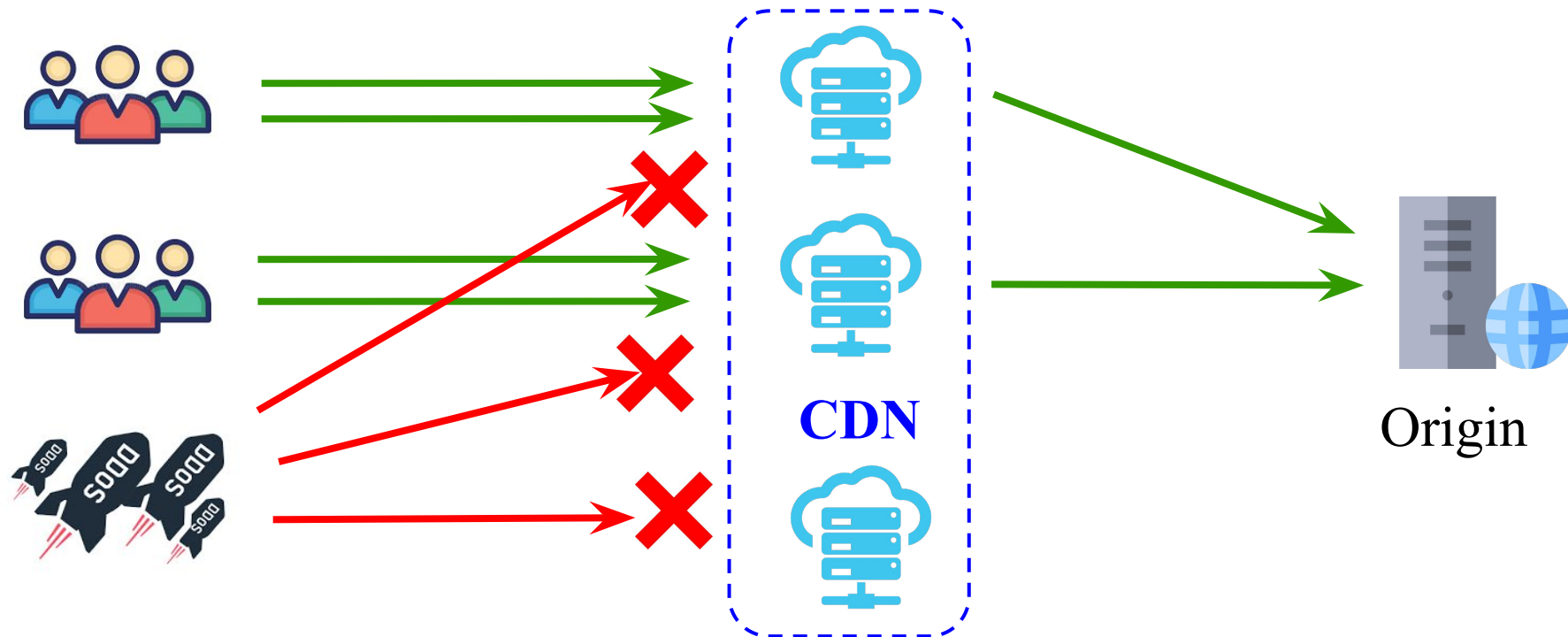
CDN Judo : Breaking the CDN DoS Protection with Itself

Run Guo, Weizhong Li, Baojun Liu, Shuang Hao,
Jia Zhang, Haixin Duan, Kaiwen Shen, Jianjun Chen, Ying Liu

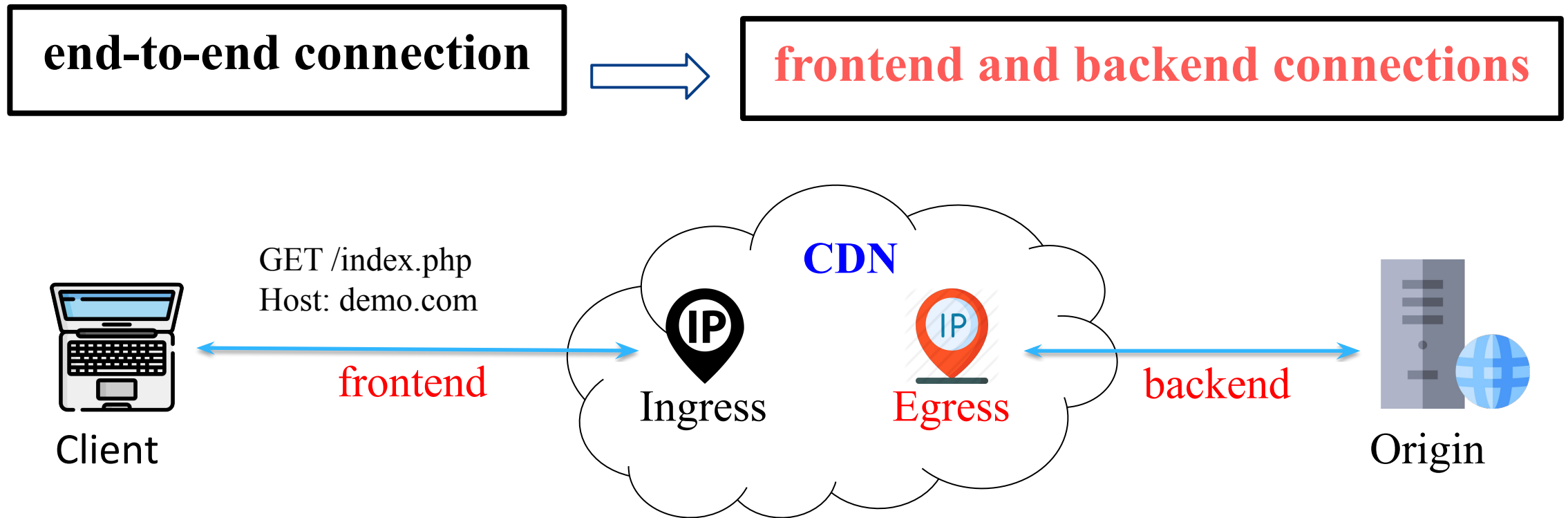


Content Delivery Network

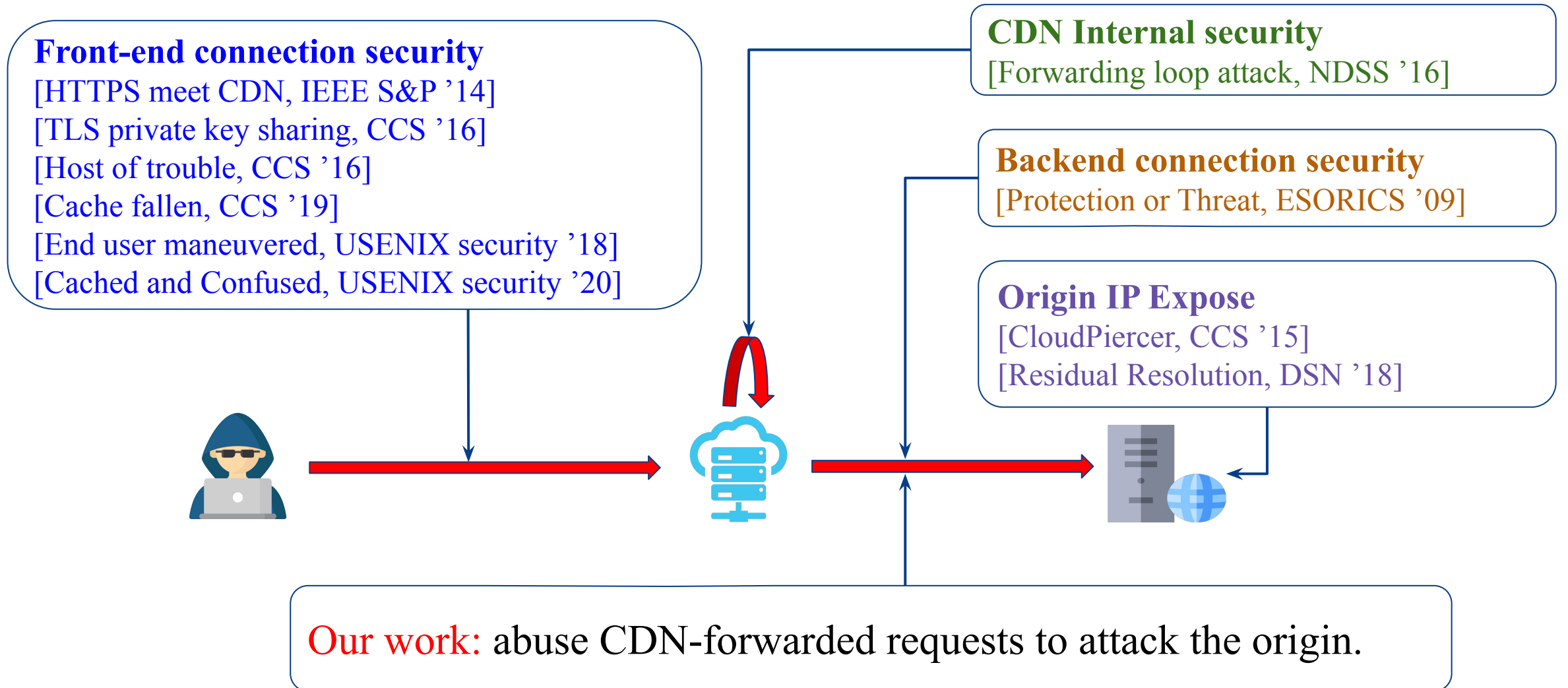
- ❖ Infrastructure for access acceleration and DDoS defence
 - 14.85% of top 1M, 38.98% of top 10K websites [Your Remnant Tells Secret, DSN '18]
 - We find CDN itself can be abuse to break its DDoS protection



CDN Forwarding Process



Previous Works



Our Work

- ❖ Exploiting request-forwarding flaws

Attack-1	HTTP/2 bandwidth amplification attack
Attack-2	Pre-POST slow HTTP attack
Attack-3	Egress IP blocking attack

- ❖ Performed realworld evaluations on six vendors



Request-forwarding Features Could be Abused

Attack-1

Protocol converting: HTTP/2 -> HTTP/1.1

Maximum compatibility

Attack-2

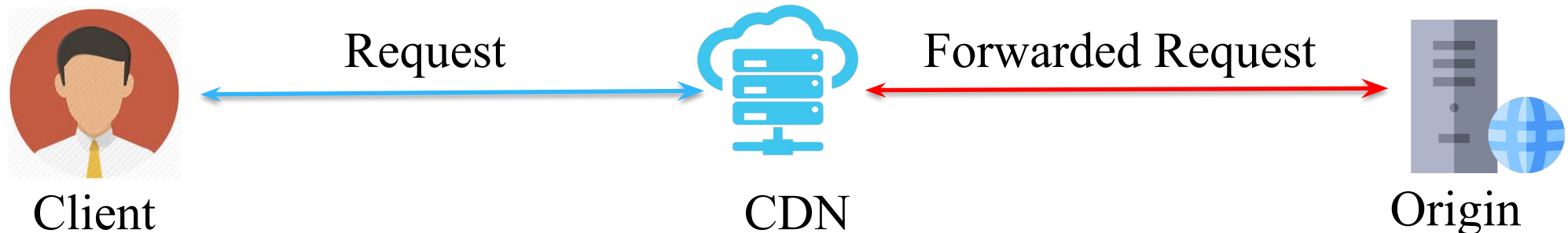
Pre-POST forwarding: speed up request-forwarding

Access Acceleration

Attack-3

Origin shield: reduce backend connections

Traffic offloading



Attack-1

HTTP/2 Bandwidth Amplification Attack

HTTP/2 Protocol

- ❖ RFC7540, released in 2015

- **Binary message framing**

- **Streams and multiplexing**

Multiple HTTP requests and responses can be transferred in the same TCP connection in parallel and staggered.

- **HPACK: header compression**

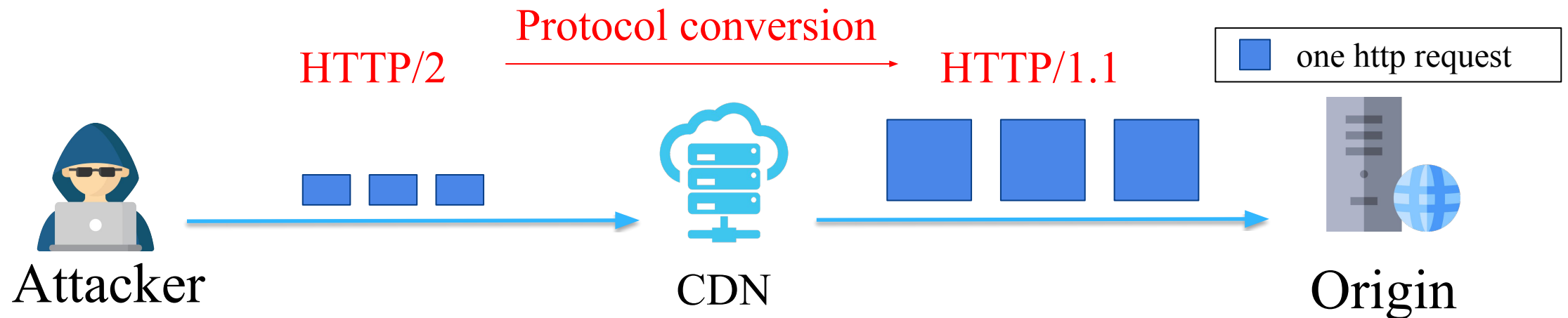
Avoid repeatedly transferring headers that do not change frequently.

- ❑ *Deployment: Over 43.2% of Alexa top 1M websites (w3techs.com, 12 Feb 2020)*

HTTP/2-1.1 Amplification on CDN

❖ Our study

- Identify that HTTP/2-1.1 conversion of CDN will cause amplification attack.
- Improve the attack with the feature of Huffman encoding.
- Real-world measurement and evaluation.



- ❑ [HTTP/2 Tsunami Attack, EST '17]
Show an amplification attack in HTTP/2-1.1 proxies built with Nginx and Nhttp2.

CDN Vendors Claim to Support HTTP/2

- ❖ HTTP/2 is supported by most major CDNs
- ❖ The backend connection still uses HTTP/1.1

	CloudFront	Cloudflare	CDNSun	Fastly	KeyCDN	MaxCDN
Frontend Connection	Default on Configurable	Default on	Default on	Default off Configurable	Default on	Default on Configurable
Backend Connection	Still use HTTP/1.1					

Attack-1.1: Using HPACK Static Table

- ❖ Referencing an indexed static table of common header fields to encode request headers.

Raw Request

```
:method: GET
:path: /
:authority: demo.com
:scheme: https
```

54 Bytes

Static Table

1	:authority	
2	:method	GET
3	:method	POST
4	:path	/
...
7	:scheme	https
...
61	www-authenticate	

Encoded Data

```
82 84 41 86 90 b4 9d 72
1e 9f 87
```

11 Bytes

2

4

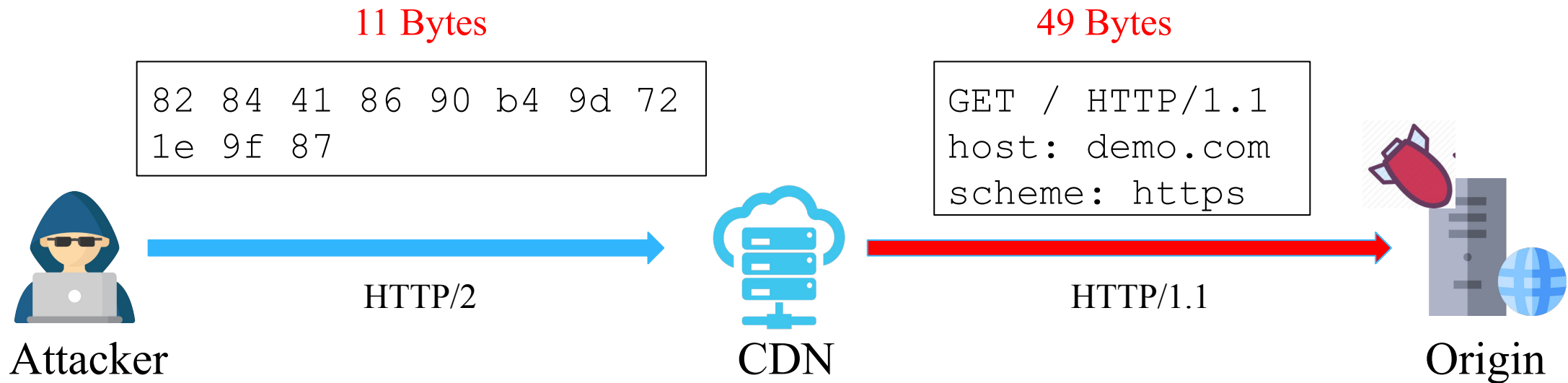
1

7

H("demo.com")

Attack-1.1: Using HPACK Static Table

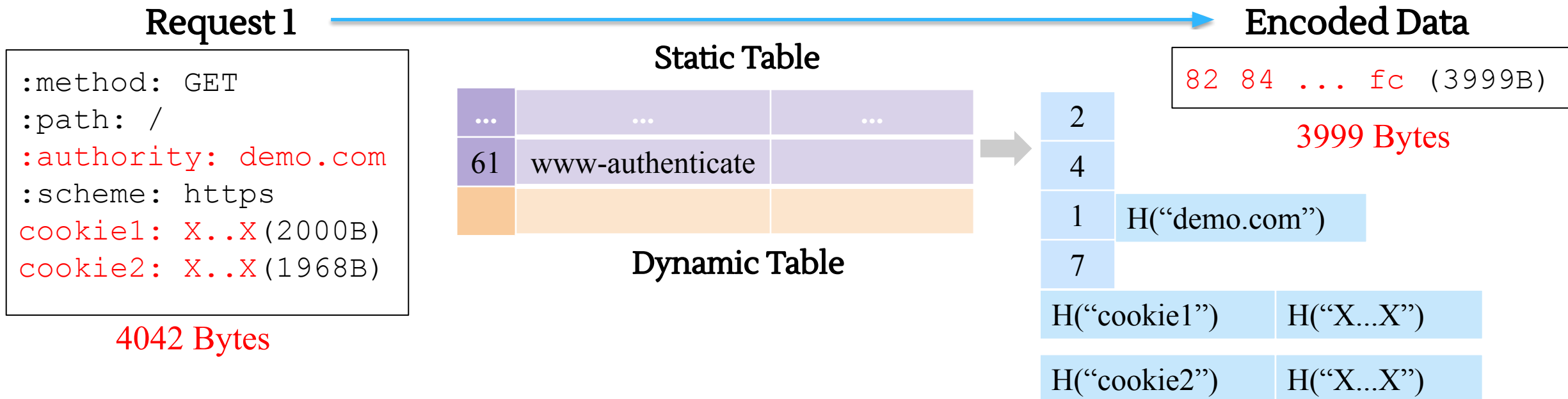
- ❖ HTTP/2-1.1 conversion of CDN causes a bandwidth amplification.



Bandwidth amplification factor: $49\text{B} / 11\text{B} = 4.45$

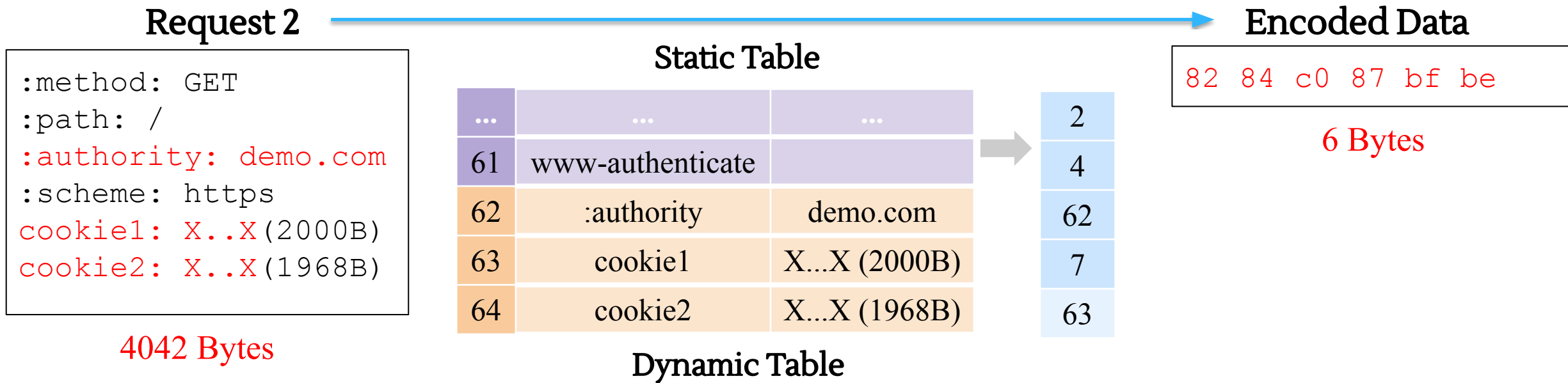
Attack-1.2: Using HPACK Dynamic Table

- ❖ Use an indexed dynamic table of previously seen headers to avoid repeatedly transferring headers in the table.
 - The firstly seen headers will be inserted into the dynamic table.



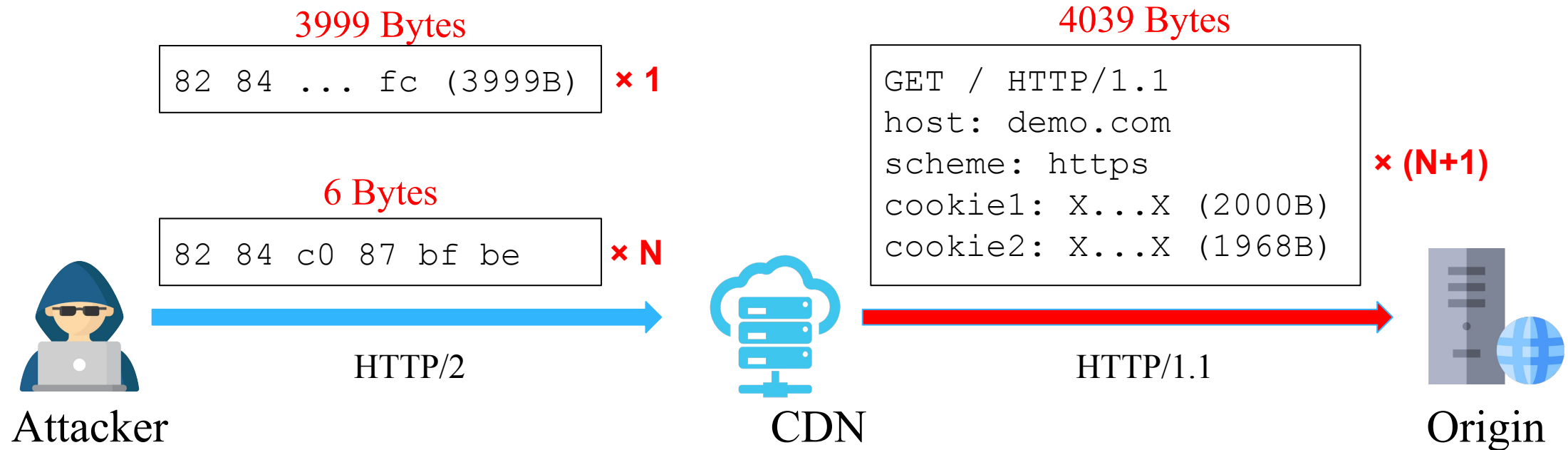
Attack-1.2: Using HPACK Dynamic Table

- ❖ Use an indexed dynamic table of previously seen headers to avoid repeatedly transferring headers in the table.
 - The subsequently repeated headers will be substituted as an index.



Attack-1.2: Using HPACK Dynamic Table

- ❖ The dynamic table enhances this kind of bandwidth amplification.

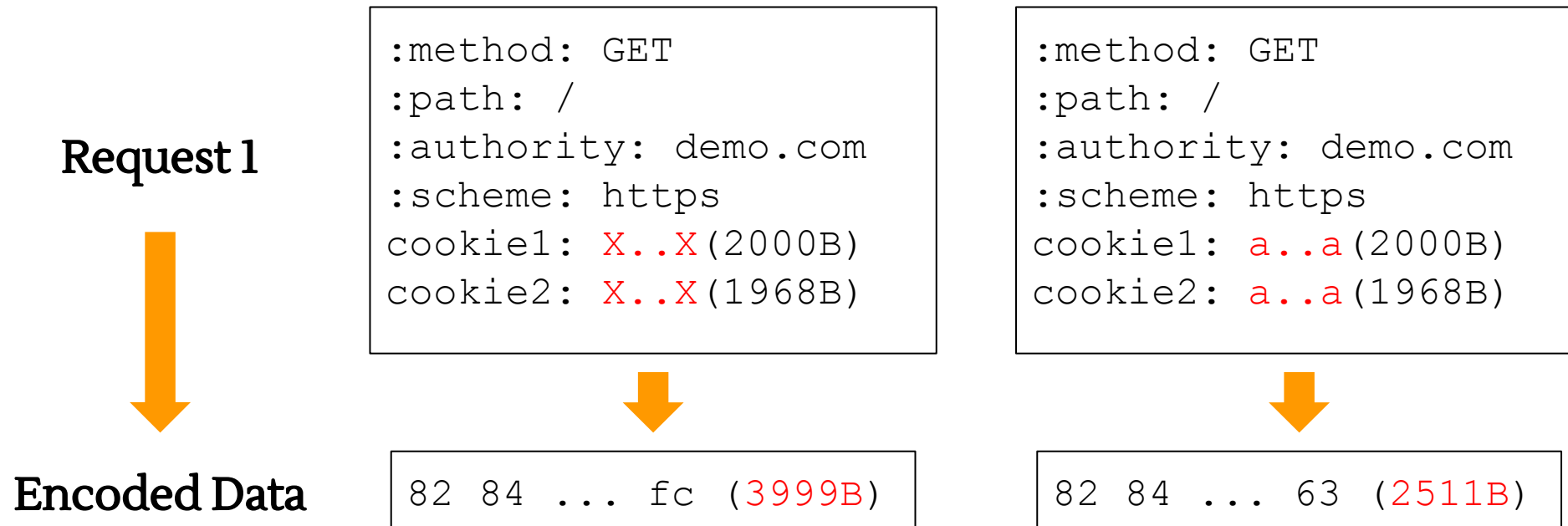


$$\text{Bandwidth amplification factor: } 4039B \times (N+1) / 3999B + 6B \times N = \frac{4039 + 4039N}{3999 + 6N}$$

For example, when N is 100, the factor is 88.70.

Attack-1.3: Improve with Huffman Encoding

- ❖ The shorter the Huffman encoding, the shorter the encoded data.
 - The Huffman encoding of 'X' is 8 bits in length.
 - Characters {0, 1, 2, a, c, e, i, o, s, t} have the shortest Huffman encoding (5 bits).



Attack-1.3: Improve with Huffman Encoding

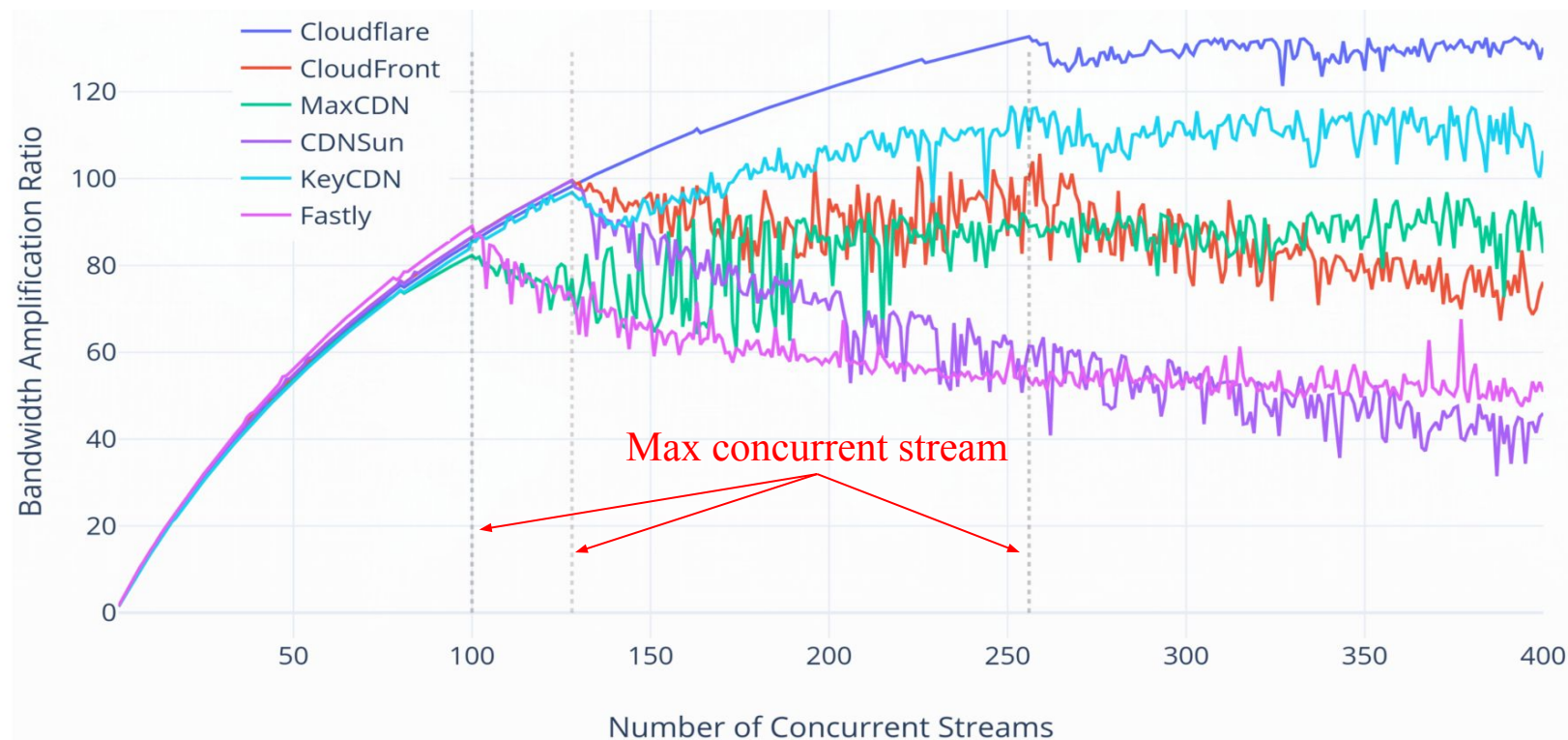
- ❖ The shorter the Huffman encoding, the larger the amplification factor.

	Huffman Encoding Length	Amplification Factor	
Character 'X'	8 bits	$\frac{4039 + 4039N}{3999 + 6N}$	88.70 when N is 100
Character 'a'	5 bits	$\frac{4039 + 4039N}{2511 + 6N}$	131.13 when N is 100

Note: N is the concurrent requests in the same HTTP/2 connection.

Amplification Evaluation

- ❖ Create multiple concurrent requests in one HTTP/2 connection.
 - The amplification factor grows with the number of concurrent requests.
 - The max factor is got at the position of the max concurrent streams.



Further improvement

- ❖ Our work achieved larger amplification factors than previous work.



	Max Streams	100		128			256
Our Attack	Evaluation Platform	MaxCDN	Fastly	CDNsun	CloudFront	KeyCDN	Cloudflare
	Amplification Factor	94.7	97.9	118.7	116.9	105.5	166.1
HTTP/2 Tsunami Attack	Evaluation Platform	HTTP/2 Proxies built with Nginx and Nhttp2					
	Amplification Factor	79.2		94.4			140.6

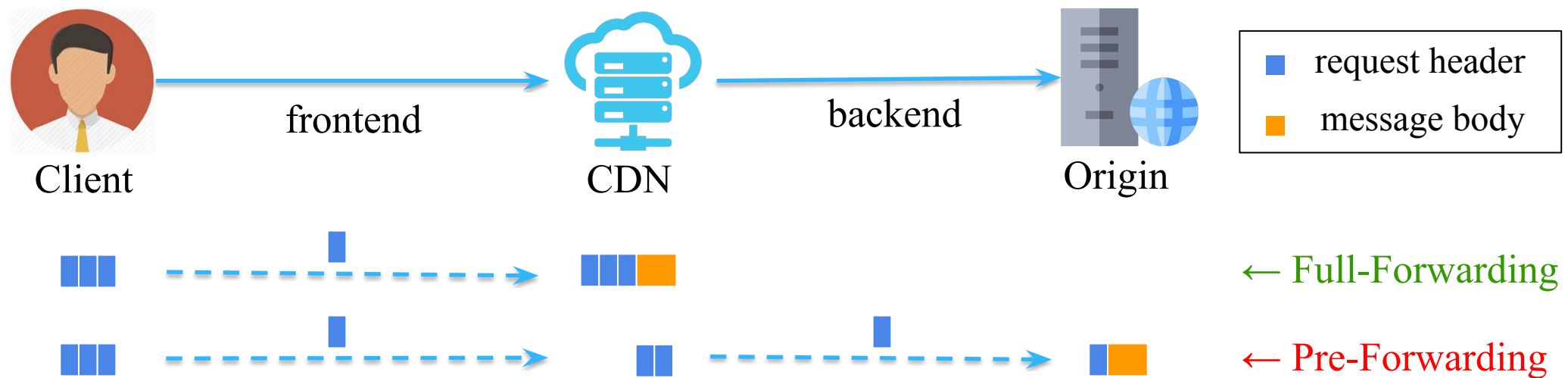
Attack-2

Pre-POST Slow HTTP Attack

CDN POST-Forwarding strategy

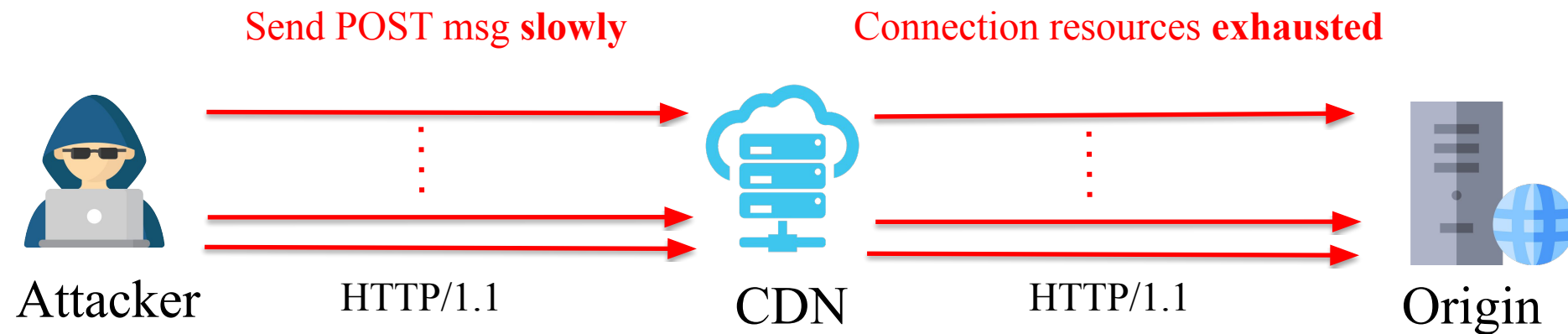
❖ Two POST-forwarding strategies

- Full-Forwarding: receive both header and the full message body, then forward 
- Pre-Forwarding: receive the header, then forward 



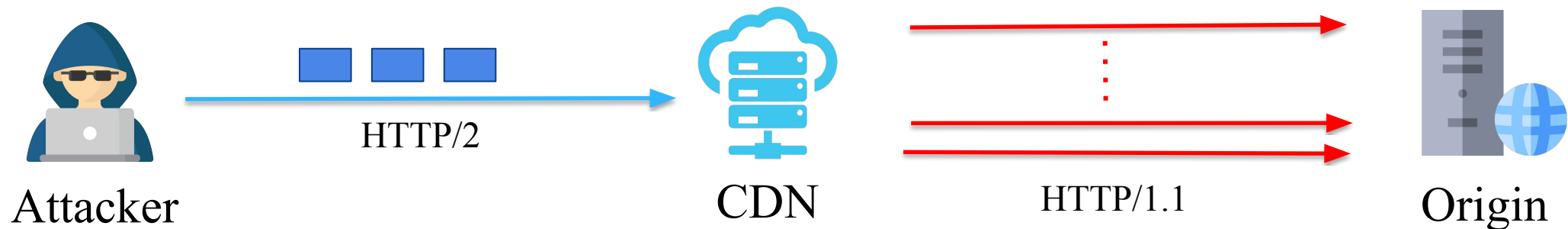
HTTP/1.1 Pre-POST Forwarding Attack

- ❖ Pre-Forwarding strategy can be abused to perform DDoS attack
 - Frontend connections: send POST messages slowly.
 - Backend connections: maintain for a long time.
 - However, the attacker has to consume TCP connections as much as the origin.



HTTP/2 Enhances the Attack

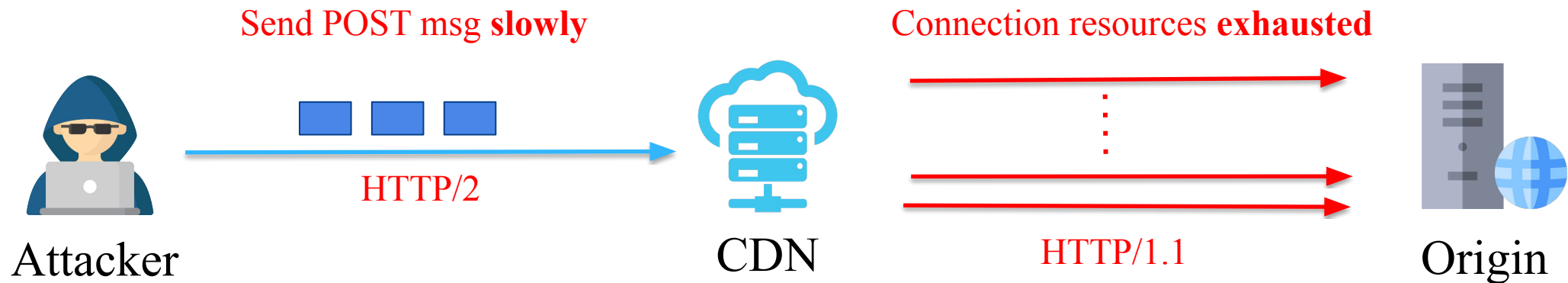
- ❖ CDN converts concurrent streams in one HTTP/2 connection to multiple HTTP/1.1 connections when forwarding.
 - The attacker consumes TCP connections much less than the origin.



	CloudFront	Cloudflare	CDNSun	Fastly	KeyCDN	MaxCDN
Max concurrent streams per HTTP/2 connection	128	256	128	100	128	100

Pre-POST Forwarding Attack

- ❖ Pre-Forwarding strategy can be abused to perform DDoS attack
- ❖ concurrent streams in one HTTP/2 connection → multiple HTTP/1.1 connections



	CloudFront	Cloudflare	CDNSun	Fastly	KeyCDN	MaxCDN
Max concurrent streams per HTTP/2 connection	128	256	128	100	128	100

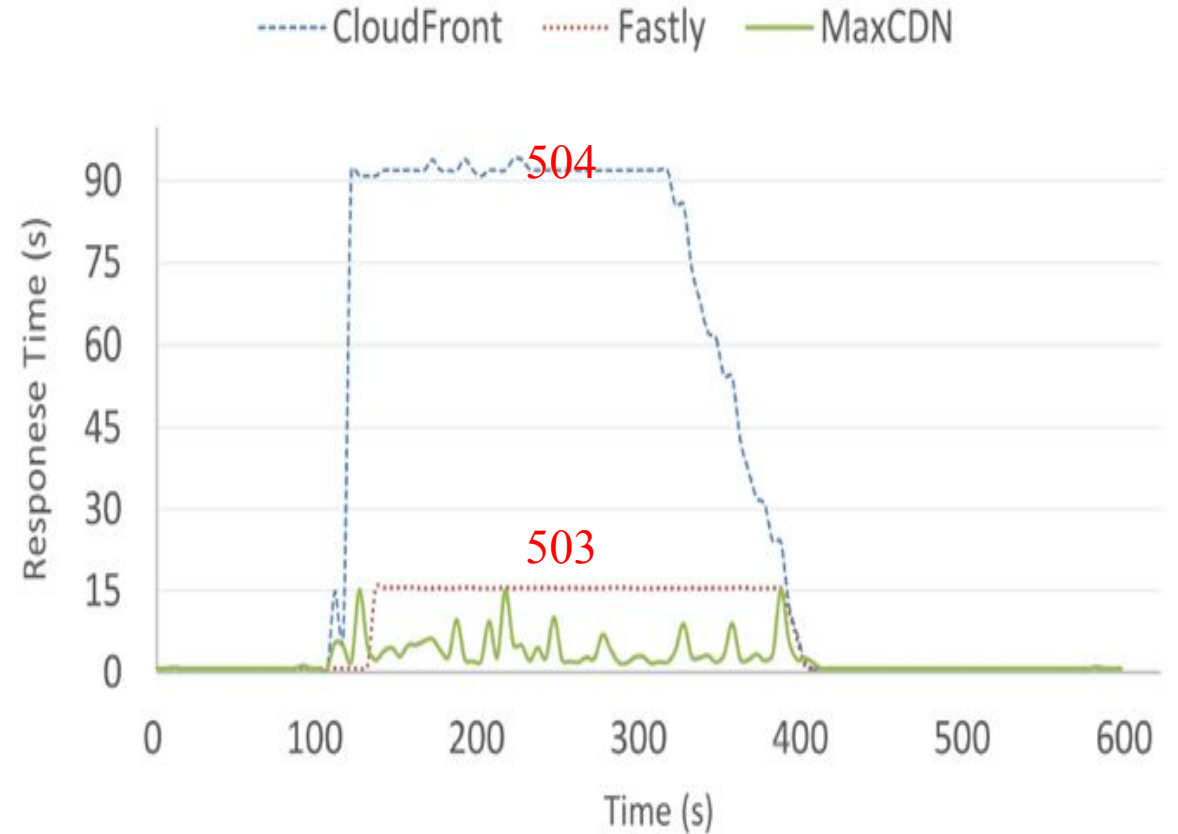
Real-world POST-forwarding Behaviors

- ❖ Some CDNs adopts Pre-Forwarding strategy to process POST request.
 - Step 1: send POST message slowly in 300 seconds.
 - Step 2: observe how long the backend connection maintains.
 - Conclusion: The attacker can control the survival time of backend connections.
 - Similar results were obtained using HTTP/1.1 or HTTP/2.

	CloudFront	Cloudflare	CDNSun	Fastly	KeyCDN	MaxCDN
Request Receiving Time in Origin	0.87s	300.29s	299.92s	0.55s	299.79s	0.74s
Connection Keep-alive Time in Origin	298.89s	0.12s	0.34s	299.32s	0.37s	15.01s
	Vulnerable	-	-	Vulnerable	-	Vulnerable

Proof of Concept

- ❖ Origin: max connections = 1000
- ❖ Attacker: send msg slowly for 300s
- ❖ Norml User: access website every 5s
- ❖ Result - the Origin is deny of service
 - CloudFront, HTTP 504 gateway timeout
 - Fastly, HTTP 503 service unavailable
 - MaxCDN, A QoS attack

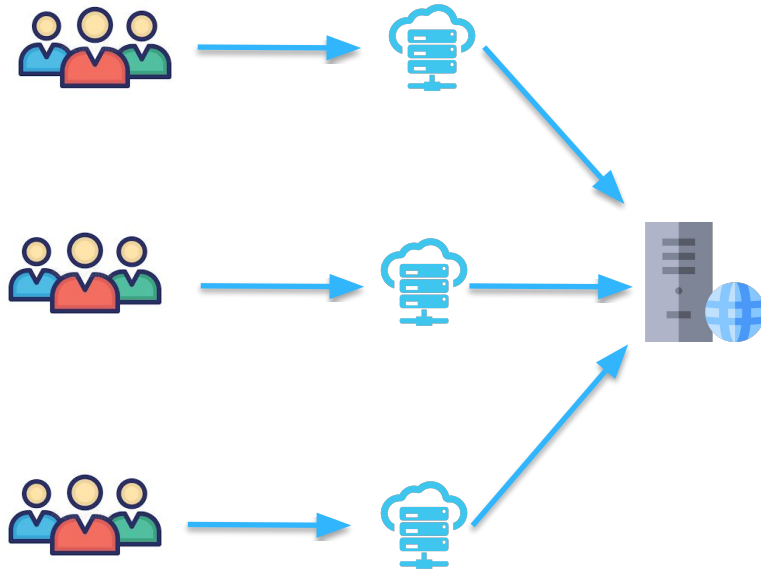


Attack-3

Egress IP Blocking Attack

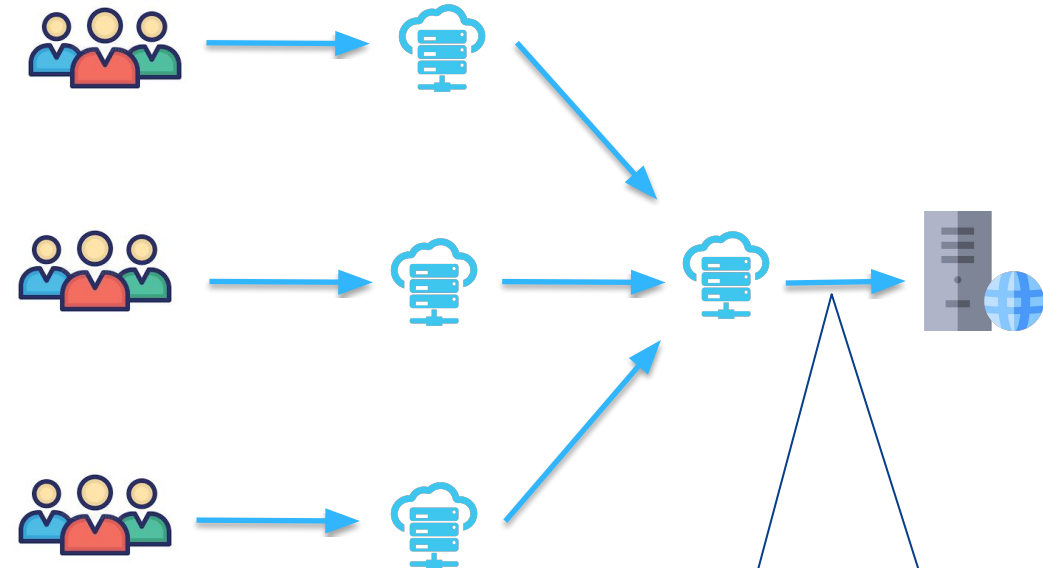
Origin Shield

Without Origin Shield



With Origin Shield

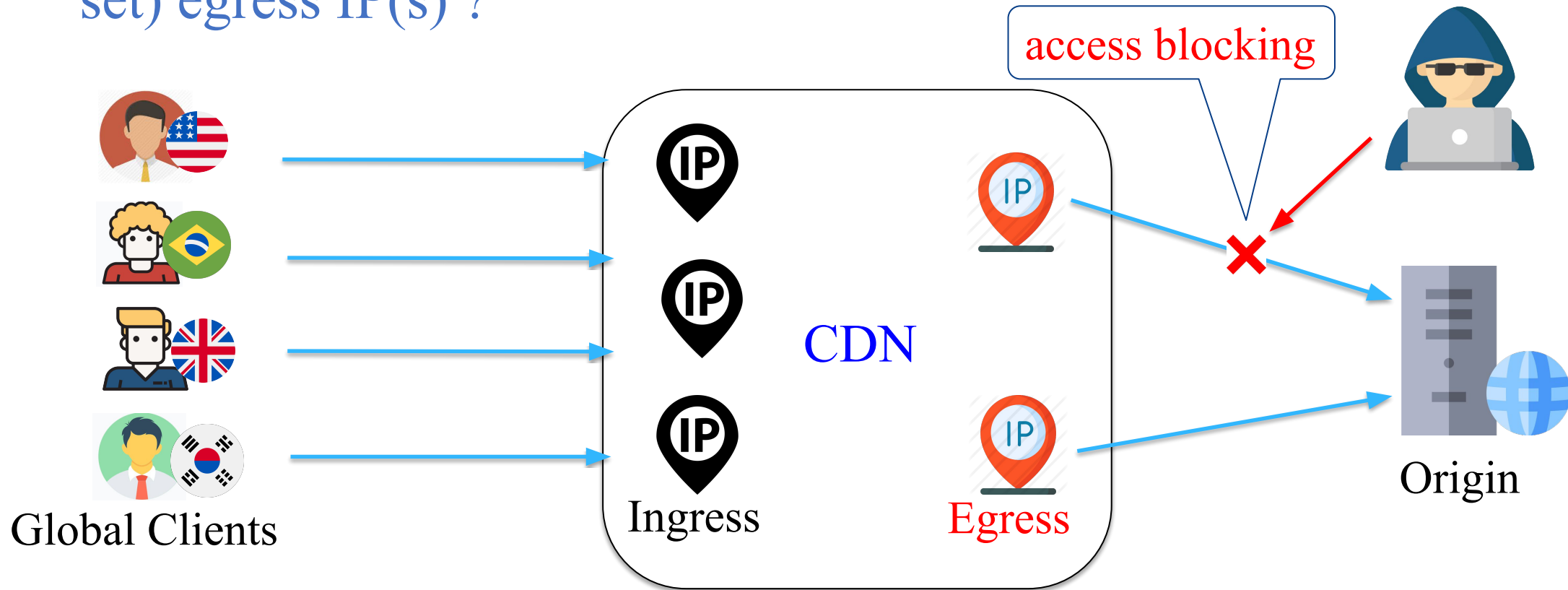
- offload the origin
- speed up cache-miss responses



backend connections
originated from less
egress IPs.

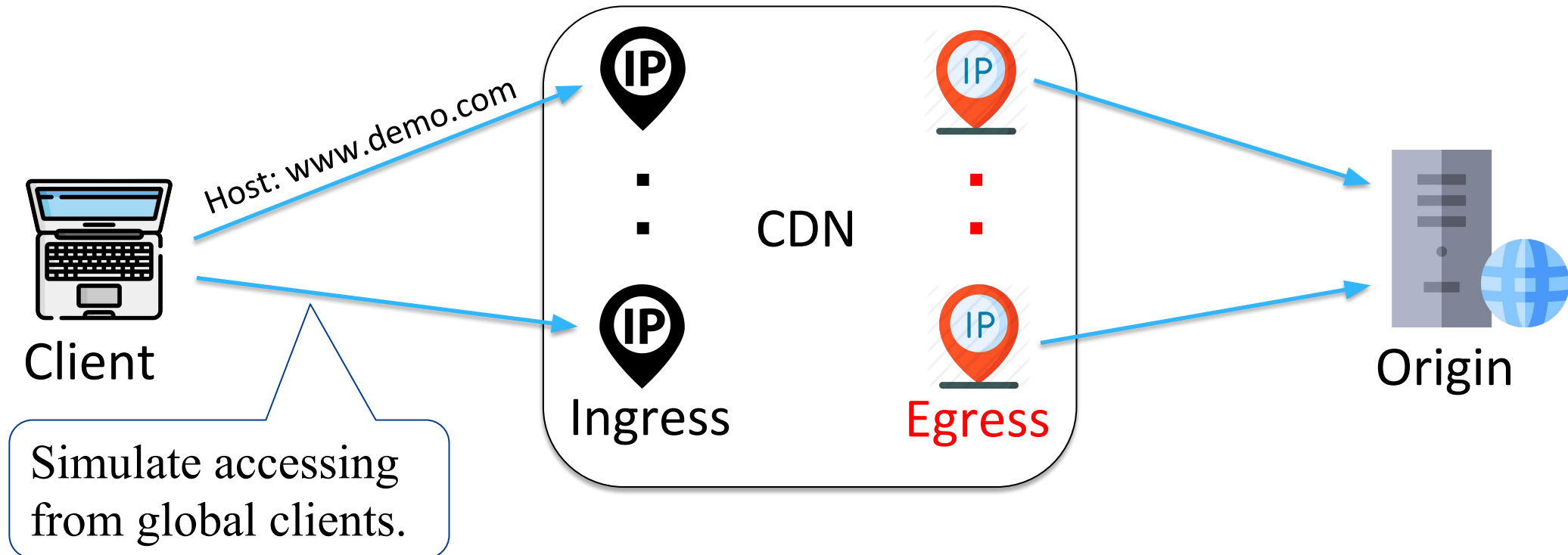
Threat Model

- ❖ Global clients will be affected when just blocking one (or a small set) egress IP(s) ?



Measurement of CDN Egress IP

- ❖ Simulate global clients
 - hourly send requests to all ingress IP addresses
- ❖ Monitor egress IP churning at our own origin



Characteristics of Egress IP distribution

❖ A small set of egress IPs

	Mapping	Ingress IPs	Egress IPs
CloudFront	DNS	128906	862
Cloudflare	Anycasting	490309	242
Fastly	DNS	64659	1136
MaxCDN	Anycasting	300	12
CDNSun	DNS	116	40
KeyCDN	DNS	95	39

❖ Churning of egress IPs (24 hours measurement)

- MaxCDN: 96.32% of the backend connections originated from the same egress IP. ❌
- Other CDNs churn egress IPs more fast, < 10% of the backend connections originated form the same egress IP. ✅

❑ Results are consistent with [Unveil the hidden presence, ICNP '19]

Real-world Case Study

CDN

- Churning of egress IP is slow

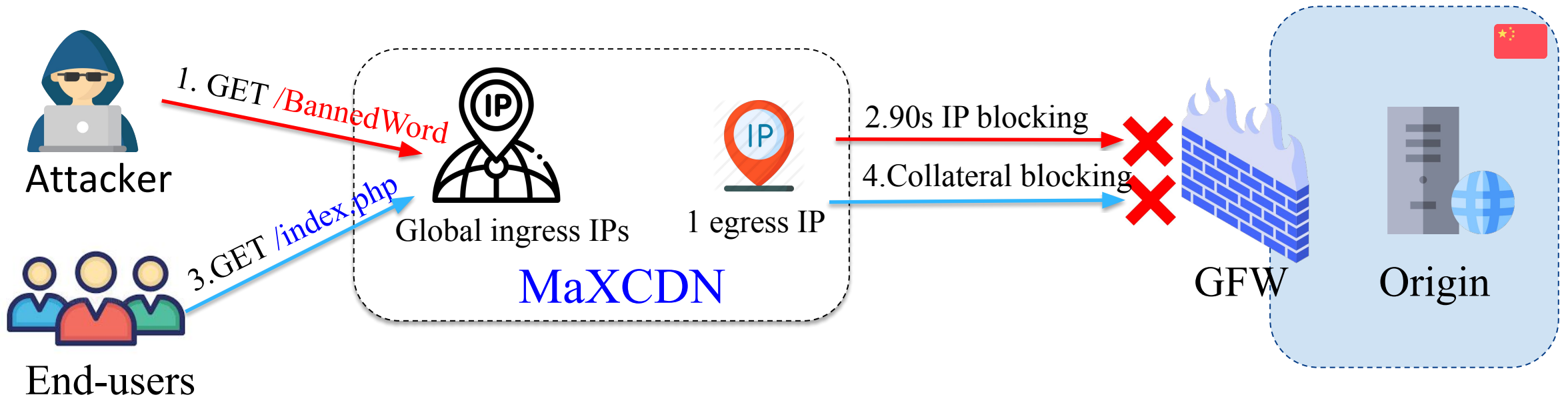
Backend connection

- GFW locates between CDN and origin
- GFW blocks censored IP pairs for 90s



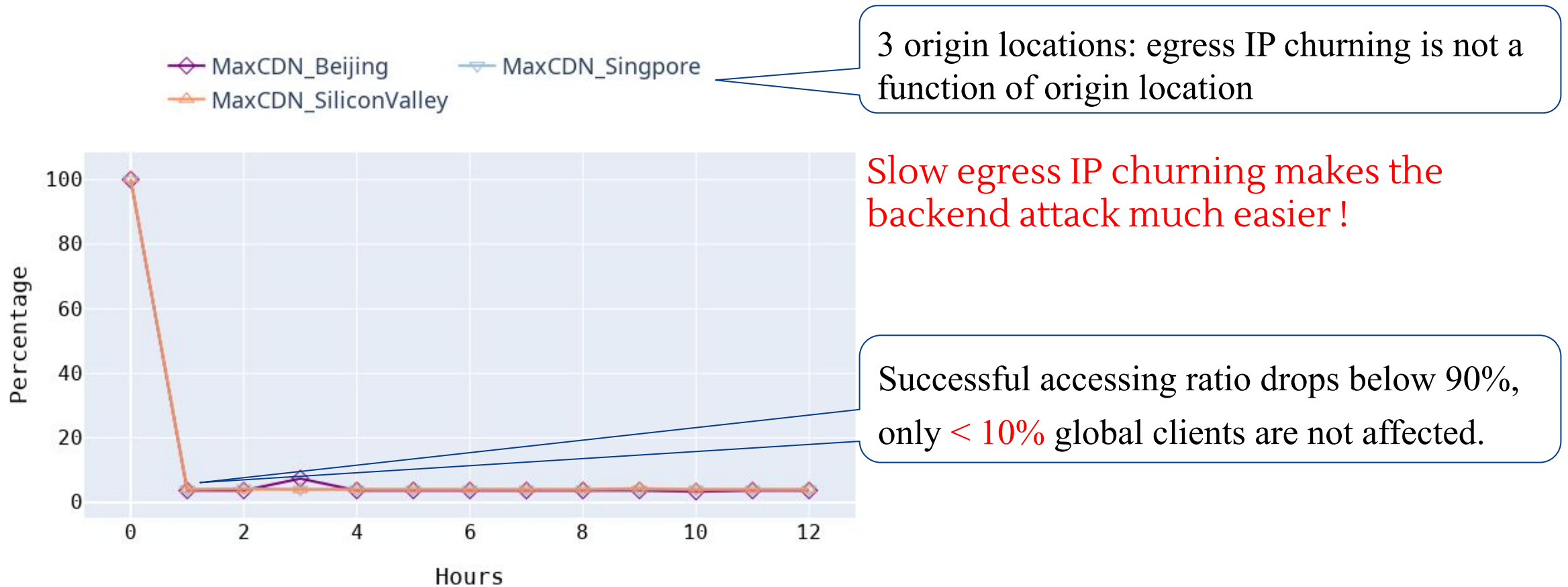
Collateral blocking

- Attacker sends requests to ingress IPs
- Global end-users are collaterally blocked



Egress IP Blocking Evaluation

- ❖ MaxCDN: block one egress IP for 12 hours



Summary

Mitigation

Threat	Recommendation
Attack-1	limit the backend network traffic.
Attack-2	enforce strict forwarding (store-then-forward).
Attack-3	apply unpredictable egress IP churning strategy.

Responsible Disclosure

- ❖ **Fastly**: Confirmed HTTP/2 and pre-POST threats, suggest to implement a request forwarding timeout, and offered us T-shirts.
- ❖ **Cloudflare**: reproduced HTTP/2 amplification with 126x, and rewarded us \$200 bonus.
- ❖ **CloudFront**: HTTP/2 issue is the product of HTTP/2 standard, suggest to use rate-based WAF rules to mitigate the attack.
- ❖ **MaxCDN**: HTTP/2 is already known. They think the egress IP blocking is out of scope as it involves with additional GFW infrastructure.
- ❖ **CDNSun** and **KeyCDN**: thanked for the messages and forwarded the issues to the developers, although no further response.

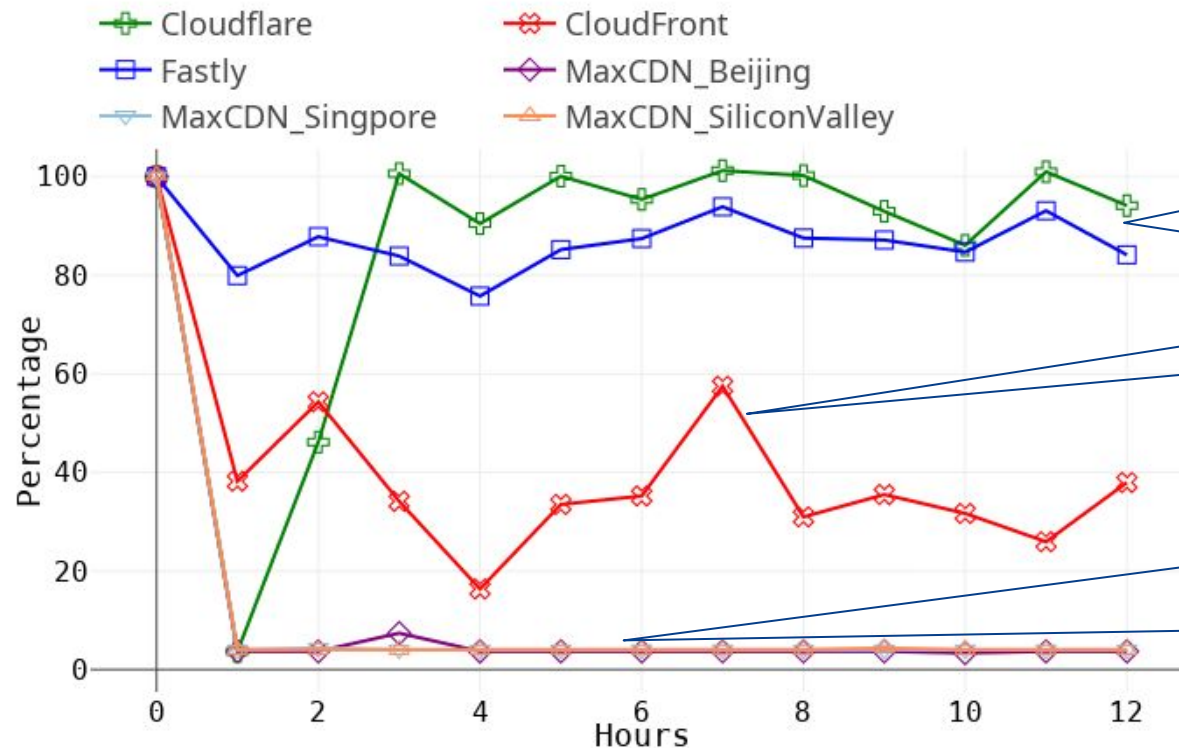
Summary

- ❖ CDN faces more security challenges in the increasing complicated Internet.
- ❖ Protocol or implementation weaknesses of CDN can be abused to break DDoS protection.
- ❖ Finding the balance between usability and security.

Thank you!

Egress IP Blocking Evaluation

- ❖ Block backend connection(s) for 12 hours
 - MaxCDN: block one egress IP
 - Other CDNs: block 16 egress IPs



Slow egress IP churning makes the backend attack much easier !

> 80% global clients are not affected

< 60% global clients are not affected

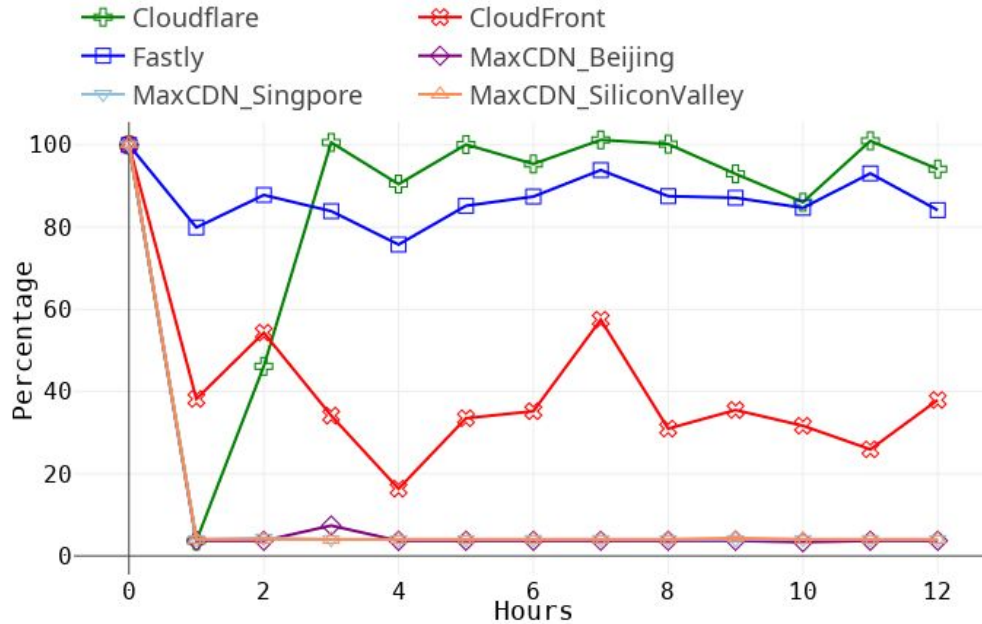
MaxCDN: Only < 10% global clients are not affected. (Origin located in Beijing/Singapore/Silicon Valley)

结论里说, Goals of CDN Vendors

- ❖ Under fierce business competition
- ❖ Strive to provide efficient & full-featured service

Negligence / sacrifice of security?

Experimental Blocking Evaluation



Experimental blocking

- From hour 0, send requests to global ingress IPs, simulating a global accessing
- From hour 1, block one egress IP of MaxCDN, block 16 egress IPs of other CDNs
- Successful global accessing ratio drops
 - MaxCDN, blew 10% within 12 hours
 - Other CDNs fluctuate because of faster egress IP churning rate

A low egress IP churning rate make the backend attacks more easier

- access blocking, e.g., on-path blocking or off-path “CrossFire” attack
- traffic eavesdropping
- ...

Q1:

*How to **globally measure** the hidden DNS interception?*

Q2:

*What are the **characteristics** of the hidden DNS interception?*

Collect vantage points

Diversify DNS requests

Identify egress IP

Amplification factors

- ❖ To achieve the maximum amplification factors
 - Streams
 - Use maximum streams (100, RFC-recommended value)
 - HPACK
 - Use a “:path” header field predefined in the static table, or a shorter one.
 - Use “cookie”, “user-agent” or other repeated fields.
 - Huffman encoding
 - Use characters in { 0, 1, 2, a, c, e, i, o, s, t } which have the shortest Huffman encoding defined in the RFC7541.

HTTP/2 amplification factors

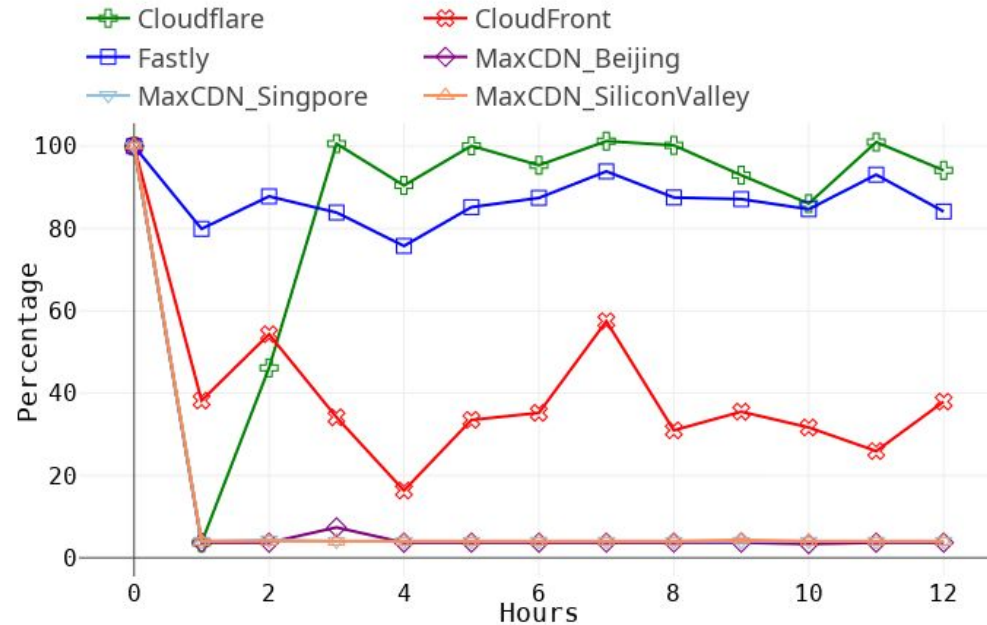
- ❖ The amplification factors are affected by the Huffman encoding and the :path header field.
- ❖ To achieve the maximum amplification factors:
 - Use characters in { o, 1, 2, a, c, e, i, o, s, t } which have the shortest Huffman encoding defined in the RFC7541.
 - Use a “:path” header field predefined in the static table, or a shorter one.

Applicable to all vendors we tested.

Experimental evaluation

Experimental blocking

- From hour 0, send requests to global ingress IPs, simulating a global accessing
- From hour 1, block one egress IP of MaxCDN, block 16 egress IPs of other CDNs
- Successful global accessing ratio drops
 - MaxCDN, blew 10% within 12 hours
 -



Backend blocking:

CDNs aim to access and cache web resources efficiently with few nodes

for 12 hours

Fewer egress IP resources

A much lower egress IP-churning rate

Degrade global availability

HTTP/2 Protocol

❖ RFC7540, released in 2015

HTTP1.1

A. *Binary framing message*



plain text
inconsistent interpretation

B. *Multiplexing streams*



header-of-line blocking
Concurrent multiplex streams

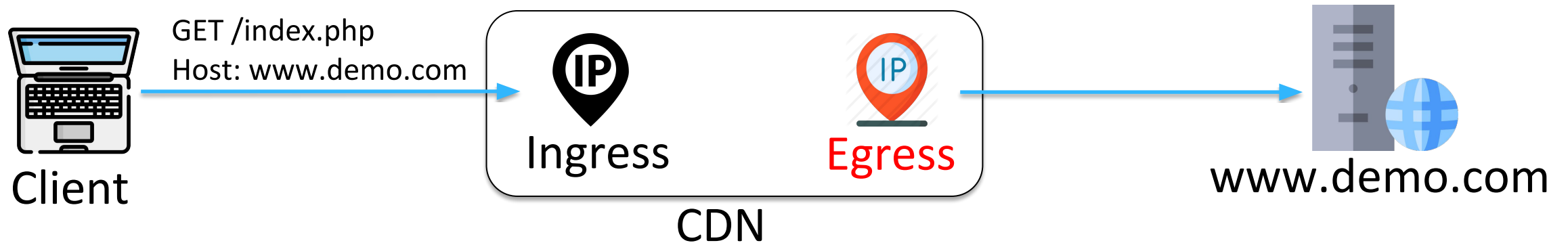
C. *HPACK header compression*



repeated redundant header
fields in each request and reply

Ingress IP & Egress IP

- ❖ A normal request is routed to a nearby CDN ingress IP.
- ❖ CDN connects the server with a egress IP

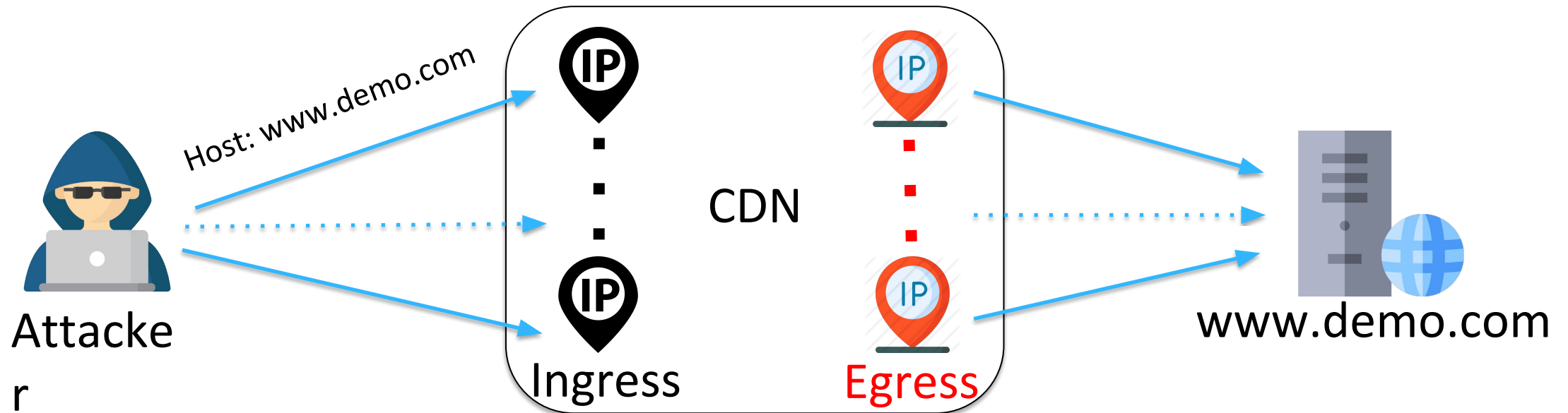


DoS to DDoS

- ❖ Global nodes can be accessed directly from one vantage point

HTTP/2 bandwidth amplification attack

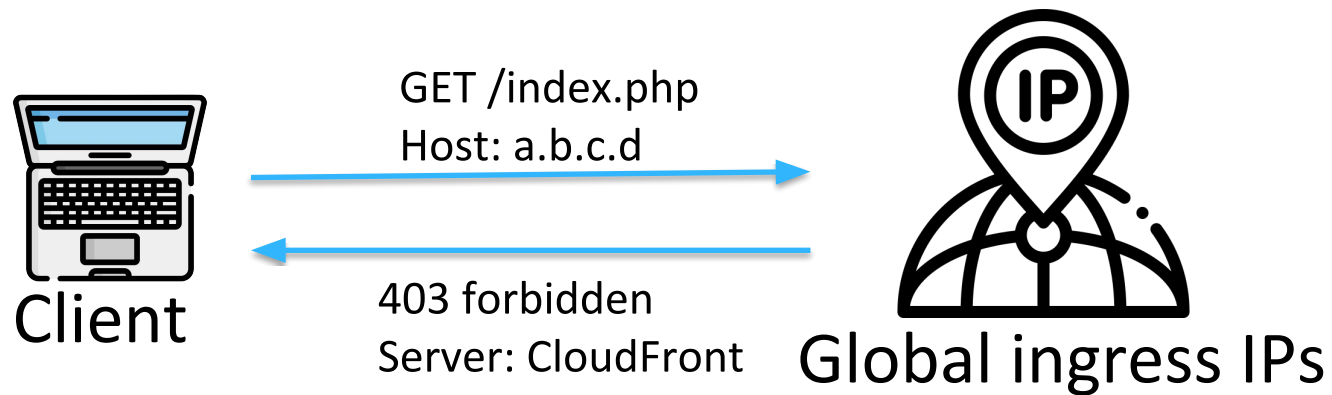
Pre-POST slow HTTP attack



Global CDN distribution has not been measured in large scale.

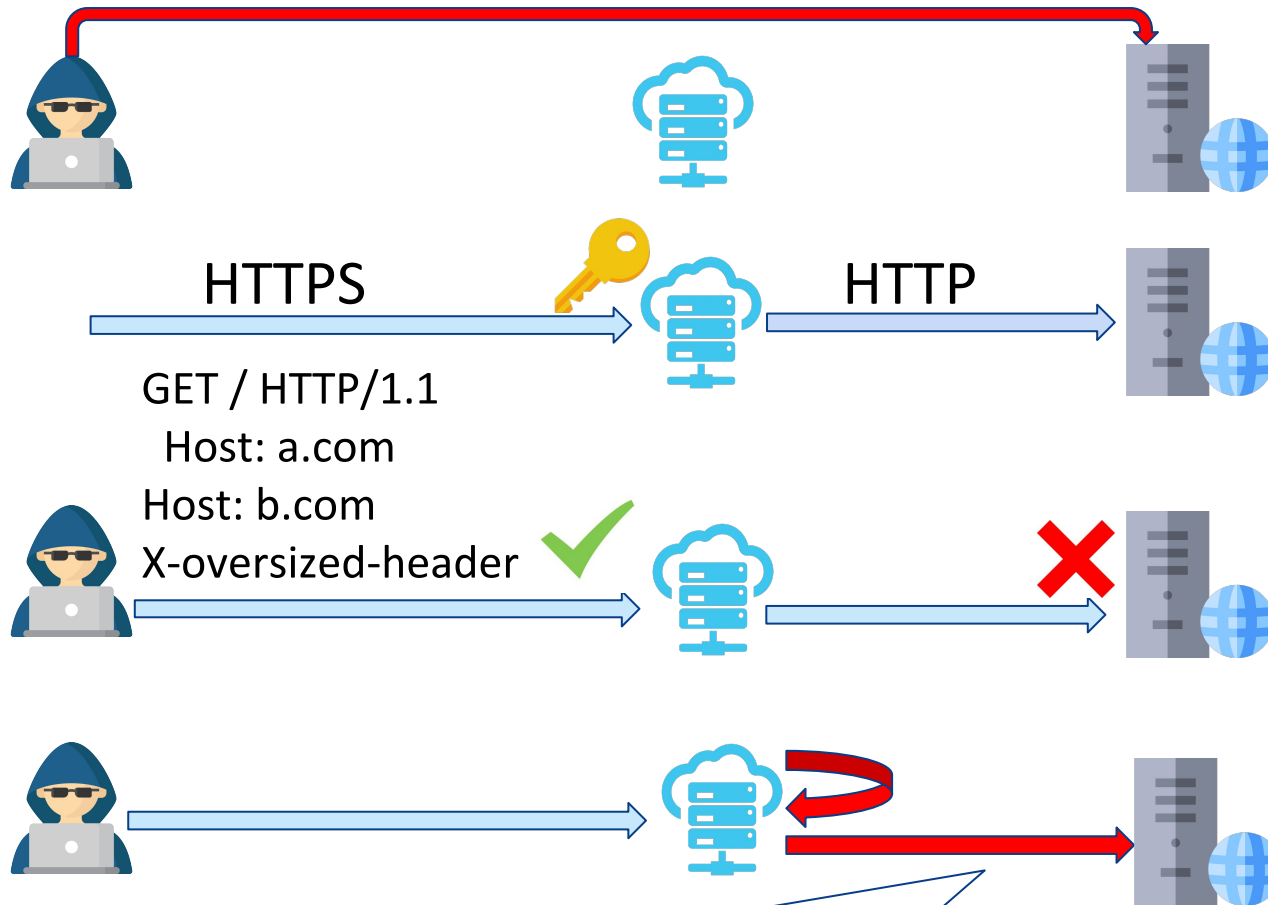
Global distribution of Ingress IP

- ❖ Collect ingress IP addresses
 - Internet-wide HTTP scanning (or censys.io)



	HTTP Status Code	HTTP Response(H: Header B: Body)
CloudFront	403	H: "Server: CloudFront"
Cloudflare	403	H: "Server: cloudflare"
Fastly	500	B: "Fastly error"

CDN Threat Model



Bypass CDN protection

[CloudPiercer, CCS '15]
[Residual Resolution, DSN '18]

Front-end security

[HTTPS meet CDN, IEEE S&P '14]
[TLS private key sharing, CCS '16]

Cache Poison

[Host of trouble, CCS '16]
[Cache fallen, CCS '19]
[Cached and Confused, USENIX security '20]

Attack Origin

[Protection or Threat, ESORICS '09]

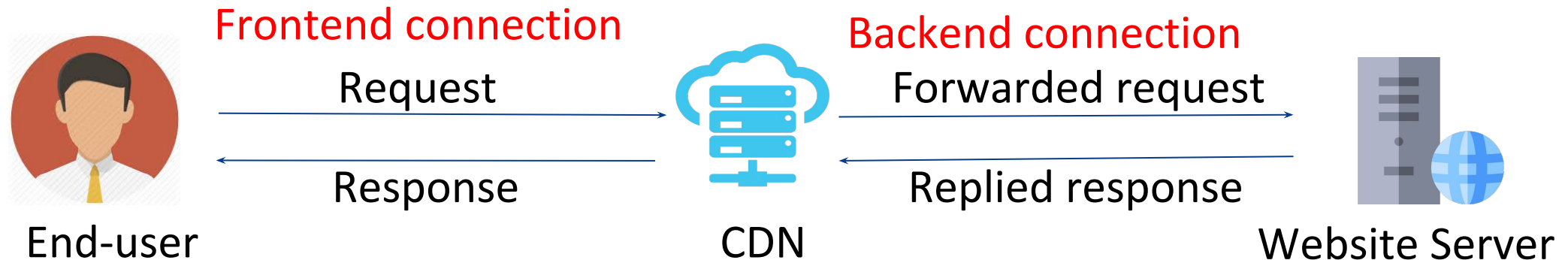
Attack CDN

[Forwarding loop attack, NDSS '16]

Send legal crafted requests to abuse the CDN to attack the origin ?

CDN Forwarding Process

Decoupled frontend and backend connections



Improve with Huffman Encoding

sym	code as bits aligned to MSB	code as hex aligned to LSB	len in bits
'a' (97)	00011	3	[5]
'b' (98)	100011	23	[6]
'c' (99)	00100	4	[5]
'd' (100)	100100	24	[6]
'e' (101)	00101	5	[5]
'f' (102)	100101	25	[6]
'g' (103)	100110	26	[6]
'h' (104)	100111	27	[6]

characters { 0, 1, 2, a, c, e, i, o, s, t } have the shortest Huffman encoding



Amplification ratio: 140 -> 166

2

4

1

Huffman("demo.com")

7

1000 0010

1000 0100

0100 0001 + 1xxx xxxx + 100100 00101 101001 00111 010111 00100 00111 101001

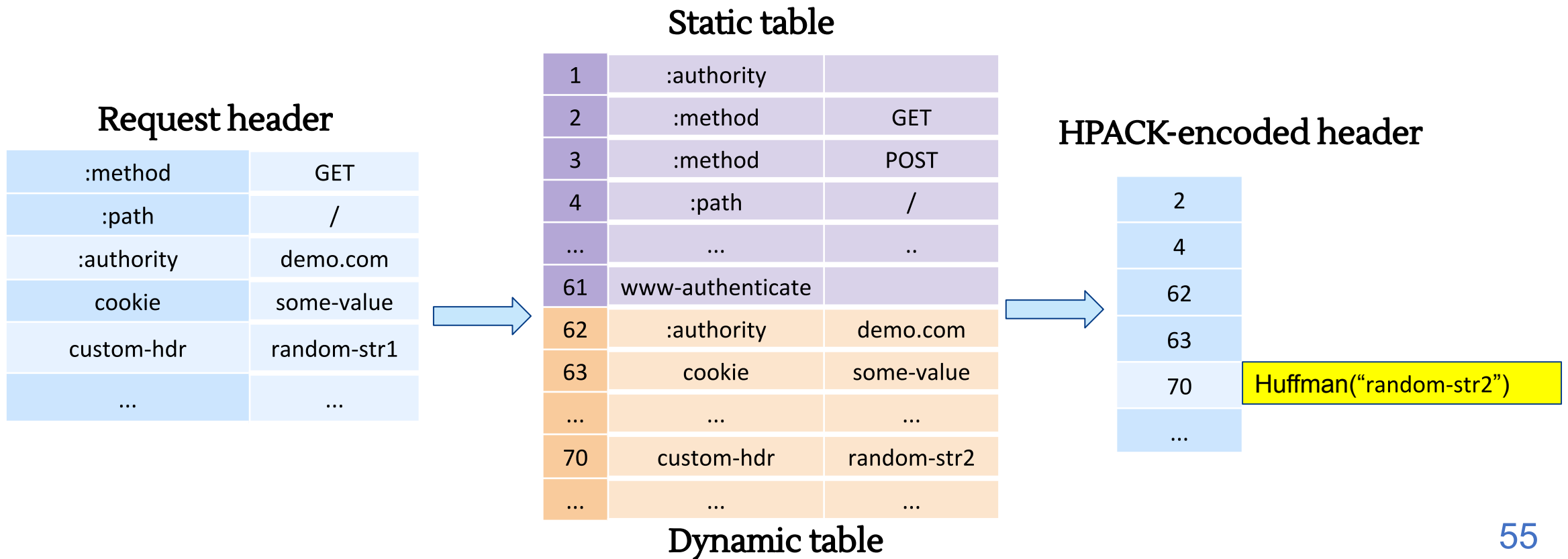
--> 1000 0110 + 1001 0000 1011 0100 1001 1101 0111 0010 0001 1110 1001 1111

1000 0111

82 84 41 86 90 b4 9d 72 1e 9f 87

HPACK: Header Compression for HTTP/2

HTTP/2 HPACK static table
半页表原理，半页举例压缩效果



Amplification factors

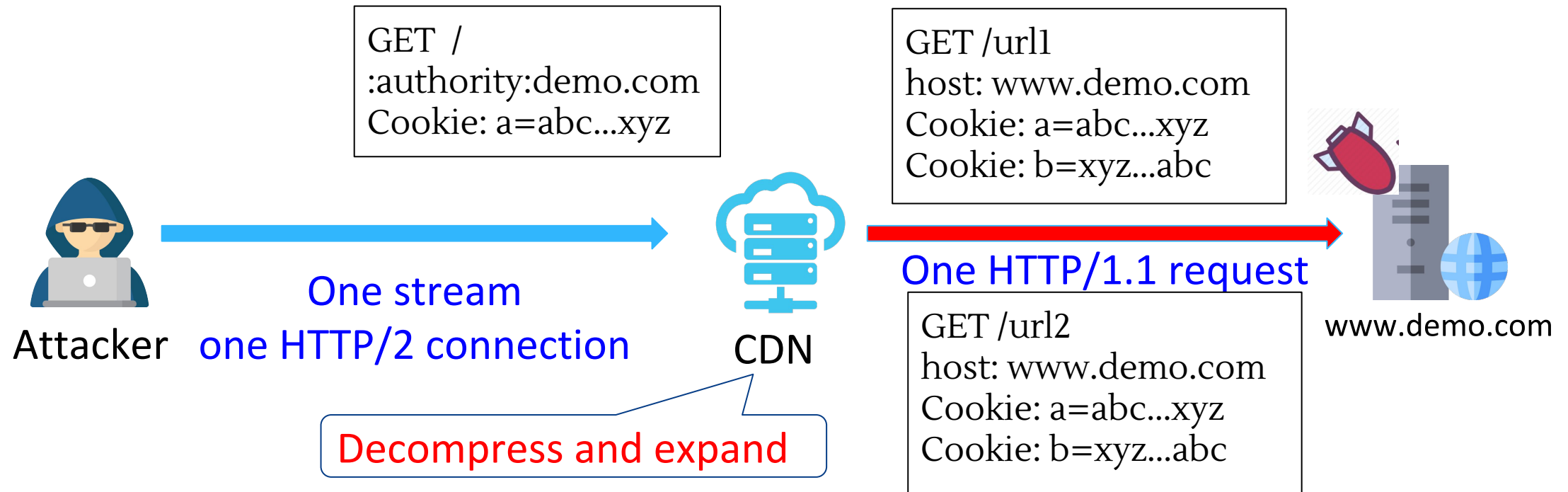
- ❖ To achieve the maximum amplification factors

Affecting features

- | | | |
|--------------------------------|---|---|
| A. <i>HPACK</i> | → | Use the repeated head fields with large-sized values, “cookies”, “user-agent” |
| B. <i>Multiplexing streams</i> | → | Use maximum streams |
| C. <i>Huffman encoding</i> | → | Use characters which have the shortest Huffman encoding |

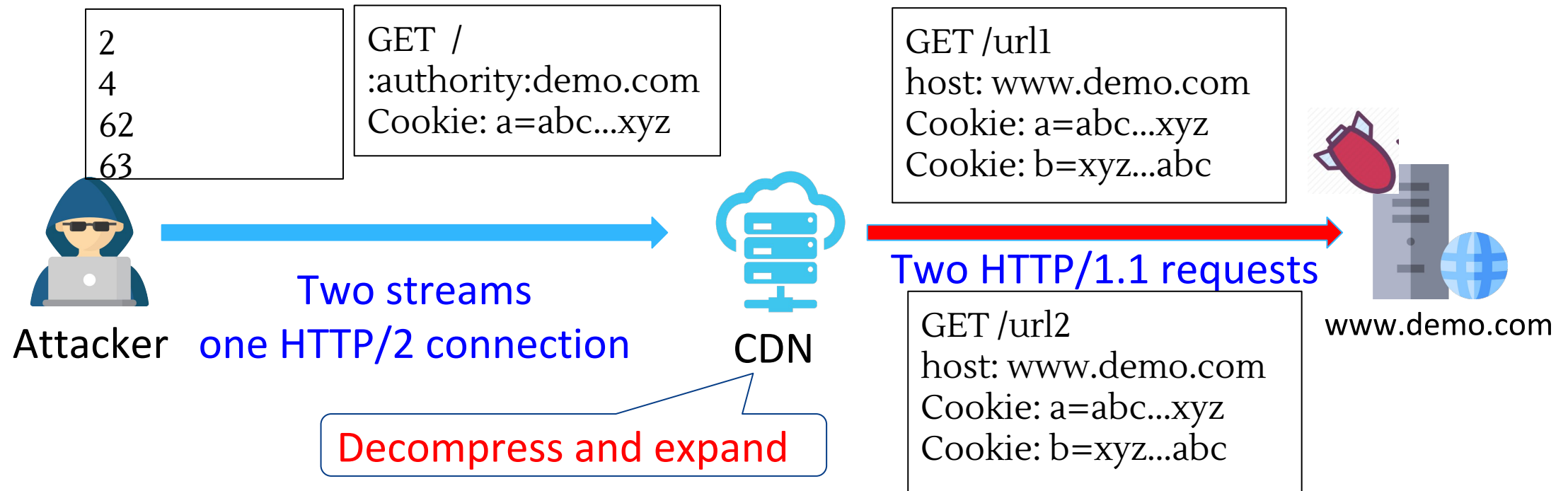
HTTP/2-1.1 Amplification Attack

one stream



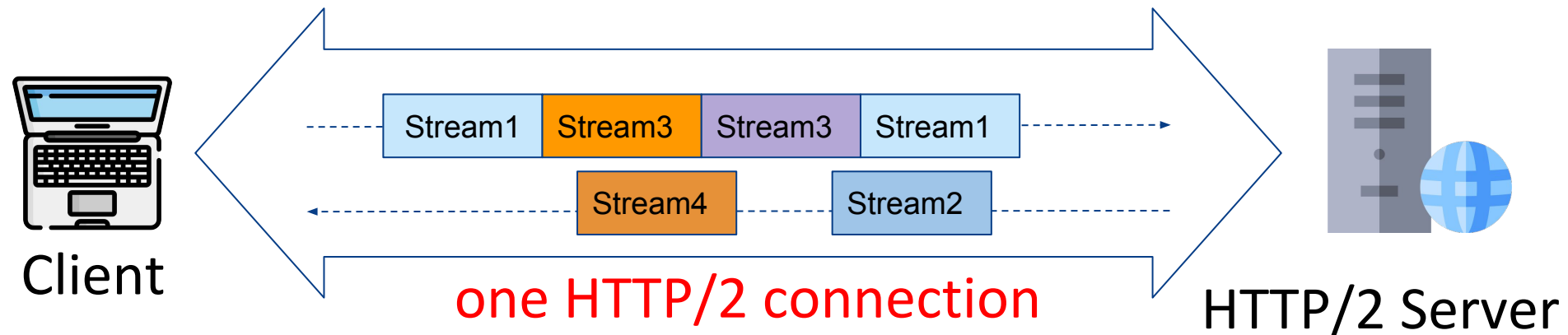
HTTP/2-1.1 Amplification Attack

two or more streams



Multiplexing Streams

- ❖ HTTP/1.1 performance inefficiency
 - Head-of-line blocking
- ❖ A TCP connection can send multiple HTTP requests and responses in parallel and staggered



Attack-1.2: Using HPACK Dynamic Table

Request 1

```

:method: GET
:path: /
:authority: demo.com
:scheme: https
cookie1: X..X(2000)
cookie2: X..X(1968)
    
```

Static Table

...
61	www-authenticate	

Dynamic Table

--	--	--

- 2
- 4
- 1
- 7

Huffman("demo.com")

Huffman("cookie1")	Huffman("X..X")
Huffman("cookie2")	Huffman("X..X")

Encoded Data

82 84 ... fc (3999)

Request 2

```

:method: GET
:path: /
:authority: demo.com
:scheme: https
cookie1: X..X(2000)
cookie2: X..X(1968)
    
```

Static Table

...
61	www-authenticate	

Dynamic Table

62	:authority	demo.com
63	cookie1	X..X (2000)
64	cookie2	X..X (1968)

- 2
- 4
- 62
- 7
- 63

Encoded Data

82 84 c0 87 bf be

Impact From “:path” Header Field

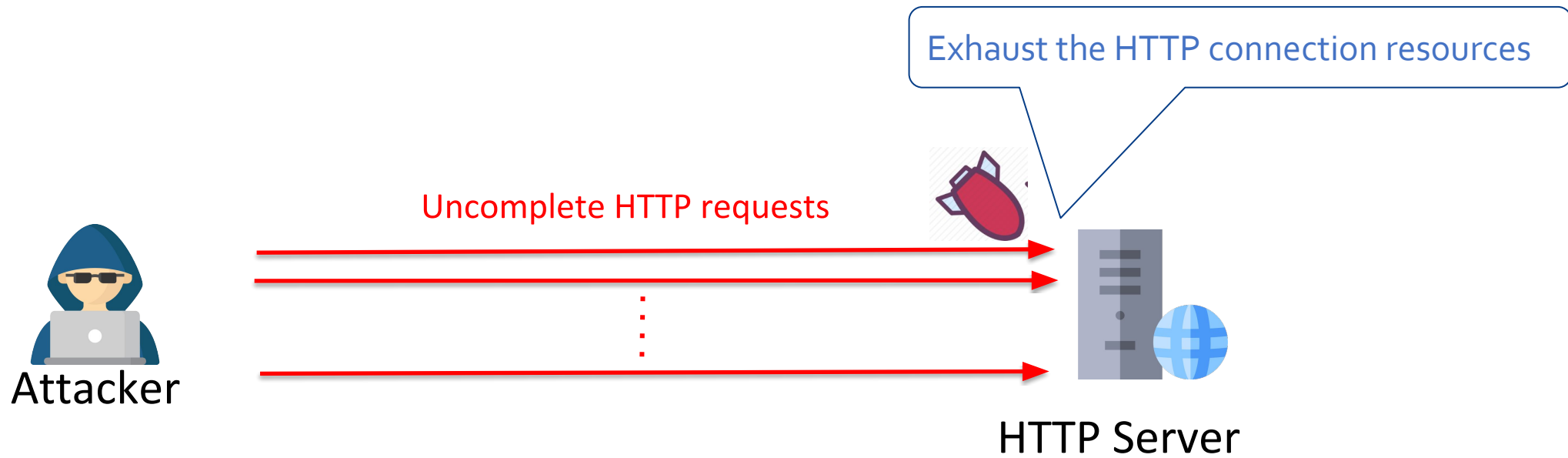
indexed as “4” in HTTP/2 table

:path: /
:authority:demo.com
other_field: ...

:path: /other_urls
:authority:demo.com
other_field: ...

	CloudFront	Cloudflare	CDNSun	Fastly	KeyCDN	MaxCDN
Max streams	128	256	128	100	128	100
Amplification ratio (:path /)	116.9	166.1	118.7	97.9	105.5	94.7
Amplification ratio (:path /8bytes_random)	99.6	132.6	99.5	89.0	96.8	82.3

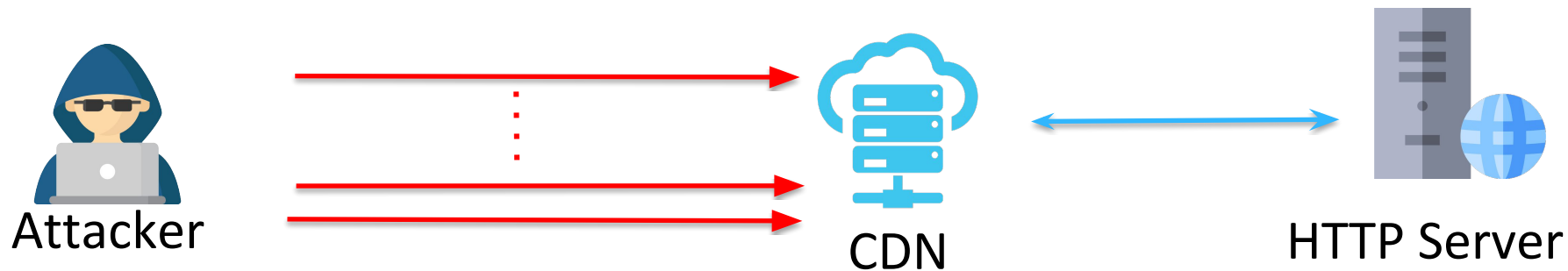
Slow HTTP Attack



HTTP is designed to keep connection open until the receiving of data is finished.

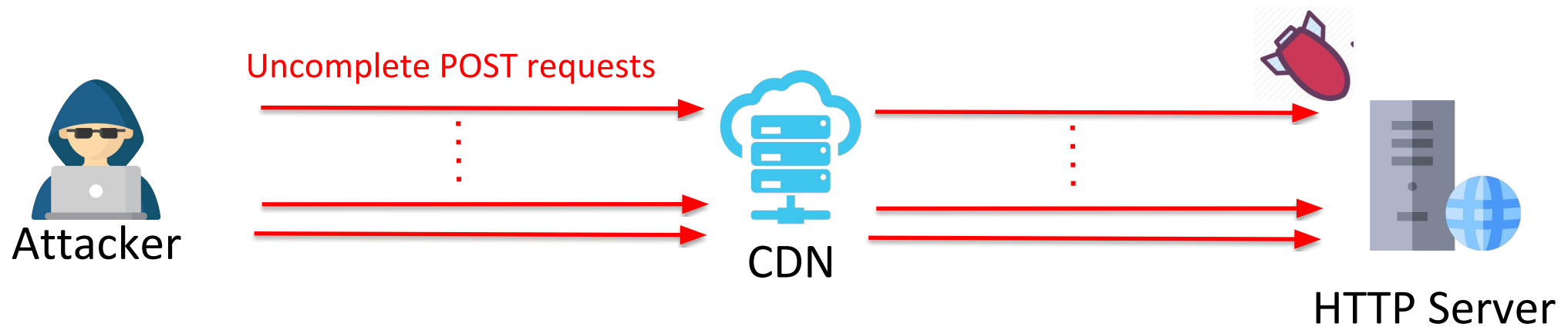
CDN mitigates slow HTTP attacks

- ❖ A CDN stops
 - slow header attack (receive the full header then forward)
 - slow read attack (no slow attribute in backend connection)
 - **slow POST attack ?**



Pre-POST Forwarding Attack

slow post 攻击图



HTTP/2-1.1 Amplification on CDN

❖ Our study

- Identify that HTTP/2-1.1 conversion of CDN will cause amplification attack.
- Improve the attack with the feature of Huffman encoding.
- Real-world measurement and evaluation.

