draft-cdn-loop-prevention-00

HTTPBIS WG

IETF 102

S. Ludin, M. Nottingham, **N. Sullivan**

Background

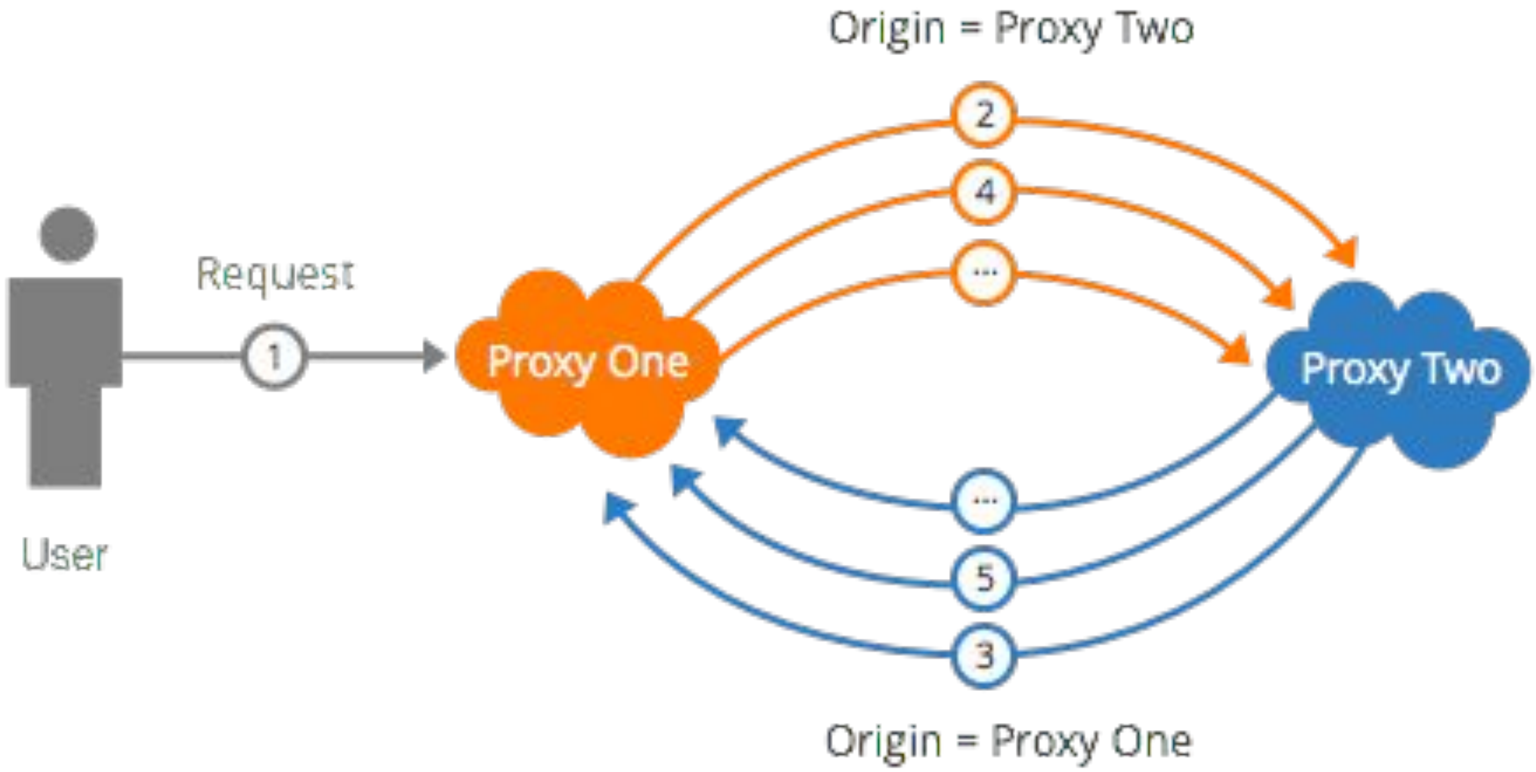
CDNs are often used as an HTTP reverse proxy for websites.

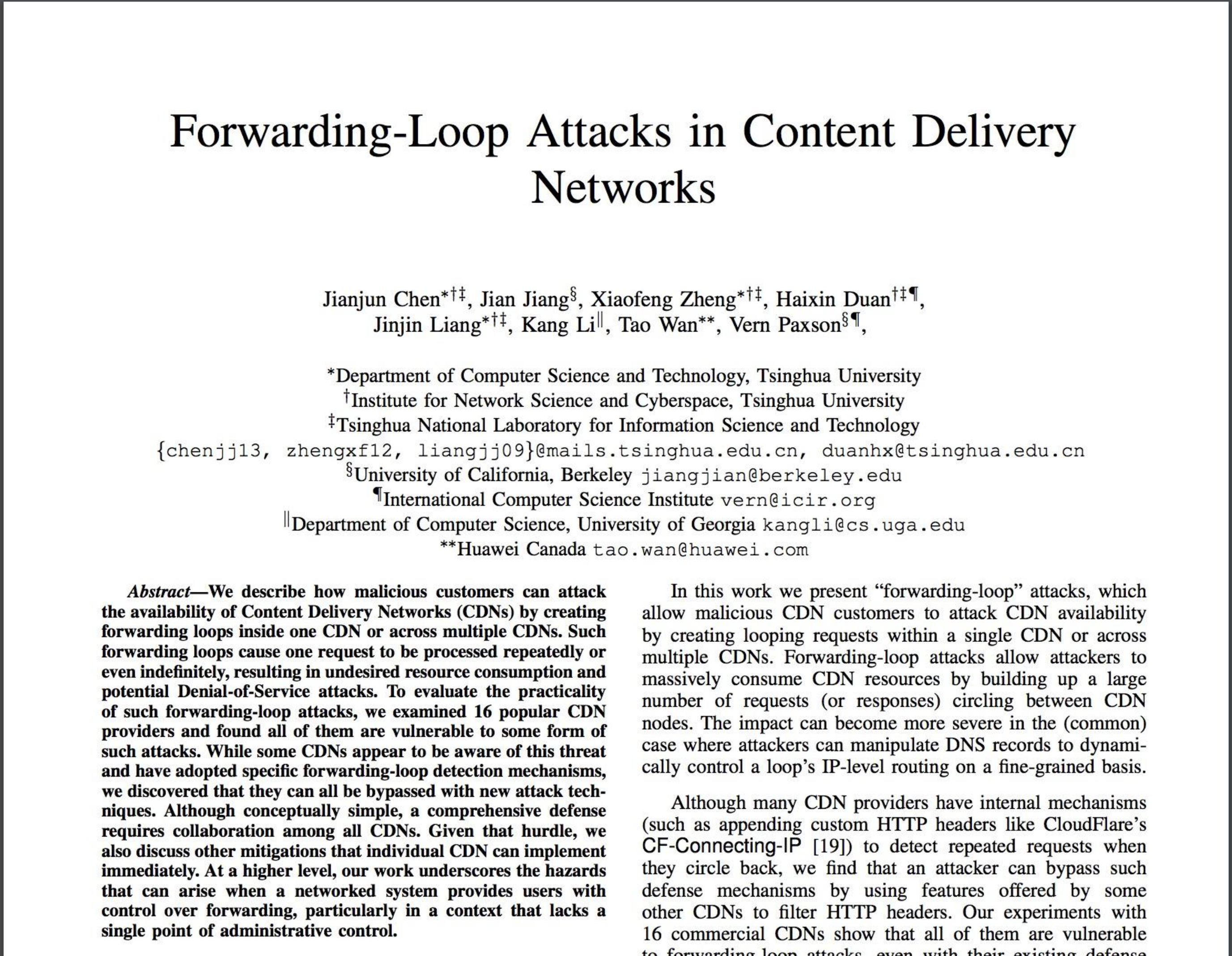
Headers are used to prevent loops.

- X-Forwarded-For, CF-Connecting-IP, Incap-Proxy-ID, RFC 7239 (Forwarded), Via

It is possible to point one reverse proxy to another.

CDN reverse proxies can be configured to modify/remove loop prevention headers.


Origin = Proxy Two
2
4
...
Request
1
Proxy One
Proxy Two
User
...
5
3
Origin = Proxy One



Forwarding-Loop Attacks in Content Delivery Networks

Jianjun Chen*[†‡], Jian Jiang[§], Xiaofeng Zheng*[†‡], Haixin Duan[†‡¶],
Jinjin Liang*[†‡], Kang Li[‖], Tao Wan**, Vern Paxson[§¶],

*Department of Computer Science and Technology, Tsinghua University
[†]Institute for Network Science and Cyberspace, Tsinghua University
[‡]Tsinghua National Laboratory for Information Science and Technology
{chenjj13, zhengxf12, liangjj09}@mails.tsinghua.edu.cn, duanhx@tsinghua.edu.cn
[§]University of California, Berkeley jiangjian@berkeley.edu
[¶]International Computer Science Institute vern@icir.org
[‖]Department of Computer Science, University of Georgia kangli@cs.uga.edu
**Huawei Canada tao.wan@huawei.com

***Abstract*—We describe how malicious customers can attack the availability of Content Delivery Networks (CDNs) by creating forwarding loops inside one CDN or across multiple CDNs. Such forwarding loops cause one request to be processed repeatedly or even indefinitely, resulting in undesired resource consumption and potential Denial-of-Service attacks. To evaluate the practicality of such forwarding-loop attacks, we examined 16 popular CDN providers and found all of them are vulnerable to some form of such attacks. While some CDNs appear to be aware of this threat and have adopted specific forwarding-loop detection mechanisms, we discovered that they can all be bypassed with new attack techniques. Although conceptually simple, a comprehensive defense requires collaboration among all CDNs. Given that hurdle, we also discuss other mitigations that individual CDN can implement immediately. At a higher level, our work underscores the hazards that can arise when a networked system provides users with control over forwarding, particularly in a context that lacks a single point of administrative control.**

In this work we present "forwarding-loop" attacks, which allow malicious CDN customers to attack CDN availability by creating looping requests within a single CDN or across multiple CDNs. Forwarding-loop attacks allow attackers to massively consume CDN resources by building up a large number of requests (or responses) circling between CDN nodes. The impact can become more severe in the (common) case where attackers can manipulate DNS records to dynamically control a loop's IP-level routing on a fine-grained basis.

Although many CDN providers have internal mechanisms (such as appending custom HTTP headers like CloudFlare's CF-Connecting-IP [19]) to detect repeated requests when they circle back, we find that an attacker can bypass such defense mechanisms by using features offered by some other CDNs to filter HTTP headers. Our experiments with 16 commercial CDNs show that all of them are vulnerable to forwarding-loop attacks, even with their existing defense

RFC 7230 Section 5.7.1. “Via”

For example, a request message could be sent from an HTTP/1.0 user agent to an internal proxy code-named "fred", which uses HTTP/1.1 to forward the request to a public proxy at p.example.net, which completes the request by forwarding it to the origin server at www.example.com. The request received by www.example.com would then have the following Via header field:

```
Via: 1.0 fred, 1.1 p.example.net
```

A sender SHOULD NOT combine multiple entries unless they are all under the same organizational control and the hosts have already been replaced by pseudonyms. A sender MUST NOT combine entries that have different received-protocol values.

Via is broken by default

IIS6: Set the variable "HcNoCompressionForProxies" to FALSE like so: in the IIS metabase properties.

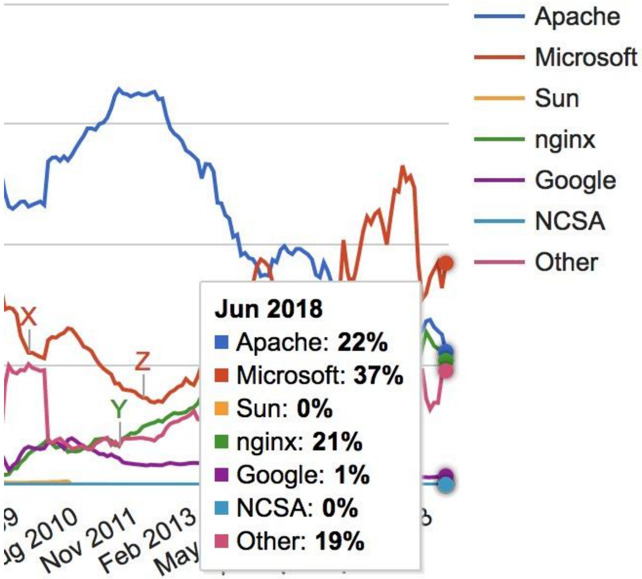
```
    HcNoCompressionForProxies="FALSE"
```

IIS7: Set the variables "noCompressionForHttp10” *and* ”noCompressionForProxies” to False in your server configuration.

```
    noCompressionForHttp10="FALSE"
    noCompressionForProxies="FALSE"
```

nginx: Add gzip_proxied to your configuration

Apache: Apache's mod_deflate (up to and including 2.4) will compress response bodies when the Via header is present, and should therefore require no changes.



Dedicated header: CDN-Loop

```
CDN-Loop = #cdn-id
cdn-id   = token *( OWS ";" OWS parameter )
```

For example:

```
CDN-Loop: FooCDN, barcdn; host="foo123.bar.cdn"
CDN-Loop: baz-cdn; abc="123"; def="456", anotherCDN
```

Requirements: CDN-Loop header

- Conforming Content Delivery Networks SHOULD add a value to this header field to all requests they generate or forward (creating the header if necessary).
- To be effective, intermediaries - including Content Delivery Networks MUST NOT remove this header field, or allow it to be removed (e.g., through configuration) and servers (including intermediaries) SHOULD NOT use it for other purposes.

Alternative proposals

RFC 7239: Forwarded HTTP Extension

4. Forwarded HTTP Header Field

[...]

If the request is passing through several proxies, each proxy can add a set of parameters; it can also remove previously added "Forwarded" header fields.

- Ambiguous?

RFC 7231 5.1.2. Max-Forwards

4. Forwarded HTTP Header Field

Each intermediary that receives a TRACE or OPTIONS request containing a Max-Forwards header field MUST check and update its value prior to forwarding the request. If the received value is zero (0), the intermediary MUST NOT forward the request; instead, the intermediary MUST respond as the final recipient.

[...]

A recipient MAY ignore a Max-Forwards header field received with any other request methods.

draft-cdn-loop-prevention-00

HTTPBIS WG

IETF 102

S. Ludin, M. Nottingham, **N. Sullivan**