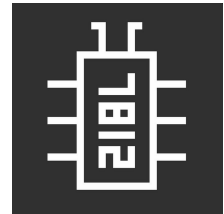


Weird proxies/2

Invicti 

About me

- Security researcher at Invicti
- Web security enthusiast / pentester
 - <https://github.com/GrrrDog>
 - <https://twitter.com/antyurin>
- Co-organizer of Defcon Russia 7812
 - <https://t.me/DCG7812>



Invicti 

Weird proxies/2

- ZeroNights 2018
 - “Reverse proxies & Inconsistency”
 - https://github.com/GrrrDog/weird_proxies
- Research Collisions
- Thanks to contributors

Reverse proxy

- Front-End
 - Reverse proxy/Load balancer/Cache proxy/...
- Back-end
 - Origin/Web-Server/...



Front-end

Request

- Route to back-end
- Route to endpoints
- Rewrite path/query
- Deny access
- Headers modification
- ...

Response

- Cache
- Headers modification
- Body modification
- ...

HTTP/1.1

GET /path/ HTTP/1.1

– Request-line

Host: target.com

Header: value

Request-line:

method SP request-target SP HTTP-version CRLF

Server

- Receive Request
- Parse
- Normalize (/path/../ -> /path/, urldecode)
- Apply rules (deny /admin, proxy to /endpoint2/)
- Recreate Request (urlencode, initial/norm. path)
- Send Request

Inconsistency between Front-end and Back-end

Host misrouting

Haproxy:

- Doesn't support Absolute URI
- Forwards as is

GET http://backend.com/q?name=X&type=Y HTTP/1.1

Host: target.com

Nginx:

- backend.com "rewrites" Host header

Host misrouting

GET **http://localhost/q?name=X&type=Y** HTTP/1.1
Host: target.com

Haproxy:

- target.com

Nginx:

- **localhost**

Nginx

- Accepts raw bytes (0x01-0x20) in path as-is

```
GET /path/<TAB>HTTP/1.1 HTTP/1.1
```

```
Path => /path/<TAB>HTTP/1.1
```

Nginx

- No trailing slash in proxy_pass
proxy_pass http://backend
- Forwards the initial path

After:

GET /path/<TAB>HTTP/1.1 HTTP/1.1

Gunicorn

- Reads until 1st whitespace with HTTP/1.1
- Accepts arbitrary string in HTTP version

GET /path/ HTTP/1.1 anything

Path misrouting

Nginx + Gunicorn

```
location /public/path {  
    proxy_pass http://backend_gunicorn;  
}
```

Path misrouting

GET /admin/<TAB>HTTP/1.1/../../../../public/path HTTP/1.1

Nginx:

- After normalization - /public/path
- Forwards - /admin/<TAB>HTTP/1.1/../../../../public/path

Gunicorn:

- After parsing - /admin/

Caddy

- urldecodes, but doesn't normalize the path
- Bypasses, misrouting - `//admin/` `./admin/` `/Admin/`
- Support fastcgi
 - php-fpm - as "back-end"
- Idea: path traversal in `SCRIPT_FILENAME` for php-fpm

Caddy

```
@phpFiles path *.php
  reverse_proxy @phpFiles 192.168.78.111:9000 {
    transport fastcgi {
      split .php
    }
  }
```

Caddy

- Caddy's fastcgi module internally normalizes path
 - <=2.4.?
- Path traversal vuln
 - `../../../../any/path/www/index.php HTTP/1.1`
- <https://github.com/caddyserver/caddy/pull/4207>
- no cve :(

URL/Header manipulation

Caddy:

```
route /prefix/* {  
    uri strip_prefix /prefix/  
    reverse_proxy 192.168.78.111:9999  
}
```

Before – GET /prefix/**http://localhost/admin** HTTP/1.1

After – GET **http://localhost/admin** HTTP/1.1

URL/Header manipulation

- Nginx \$uri - normalized URI (urldecoded)
- Works for any normalized value

```
location /uri {  
    proxy_pass      http://192.168.78.111:9999;  
    proxy_set_header X-uri $uri;  
}
```

URL/Header manipulation

- %0d%0a -> \r\n
- Request Splitting

GET

/uri/%0d%0a%0d%0aGET%20/admin%20HTTP/1.1%0d%0aHost:localhost%0d%0a%0d%0a HTTP/1.1

Deep rabbit hole

- Send url encoded value

```
location ~ /header/(.*)? {
```

```
    proxy_pass      http://192.168.78.11:9999/test/$1;
```

```
}
```

- Send url decoded value (\r\n)

```
location ~ /header/([^\s]*/[^\s]*)? {
```

```
    proxy_pass      http://192.168.78.11:9999/test/$1;
```

```
}
```

Deep rabbit hole

- Send url decoded values

```
location ~ /header/([^/]*/[^/]*)? {
```

```
    proxy_pass      http://192.168.78.11:9999/test/$1;
```

```
}
```

- O_o

Thanks to

<https://labs.detectify.com/2021/02/18/middleware-middleware-everywhere-and-lots-of-misconfigurations-to-fix/>

Complex systems

AWS, Fastly, CloudFlare, StackPath, etc (tested in 2019)

- many components (routing, caching, WAF, etc)
- inconsistency between internal components

Normalize (/path/../ -> /path/, urldecode)

Apply rules (deny /admin, proxy to /endpoint2/)

Recreate Request (urlencode, initial/norm. path)

Restriction bypass

- Restricted access to /admin
- Bypasses:
//admin/ /Admin/ /%61dmin/ /aaa../admin
- + Path misrouting

- AWS
- Fastly – patched(?) documentation
- CloudFlare – <https://developers.cloudflare.com/rules/normalization>

Nginx? Errors? Examples

AWS ELB

- // -> / , but /// -> ///
- Forwards spaces in path

AWS CloudFront

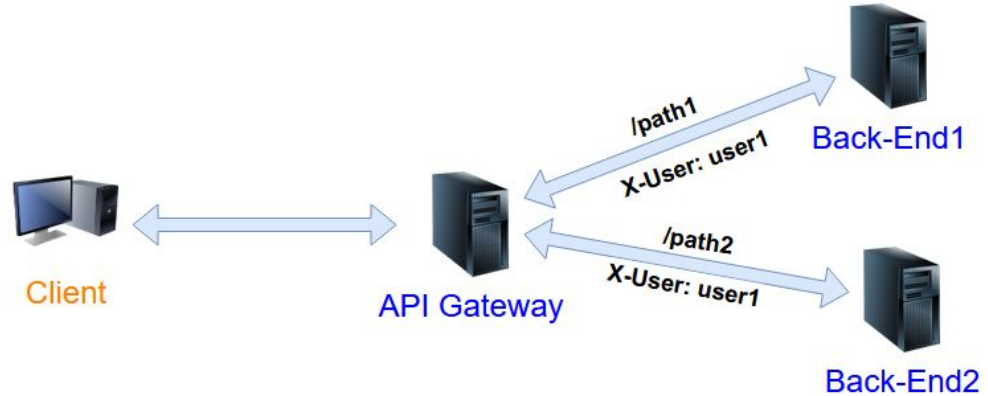
- Deletes spaces in path
- Forwards the initial path
 - If space in the path, forwards normalized path

StackPath

- Incorrectly normalizes /../ in some cases

API gateway

- Egress proxy
- Microservices
- Routing
- Authentication
 - Add header to request



API gateway

- Kong
 - Based on Nginx
- Didn't normalize path
 - /public/./admin -> /public/./admin

- CVE-2021-27306

<https://sewan.medium.com/cve-2021-27306-access-an-authenticated-route-on-kong-a-api-gateway-6ae3d81968a3>

API gateway

- Traefik doesn't normalize path
- Caddy doesn't normalize path

- Envoy depends on configuration
 - CVE-2019-9901 (<1.9.1)
 - Doesn't normalize path in older versions, by default(?)
 - Many options to setup normalization
 - Hard to configure properly

Header smuggling

API Gateway checks auth and adds header

- e.g x-user: admin

CGI* "-" is converted to "_"

- Traefik, Caddy, Envoy* proxy them
- Caddy proxies to fastcgi

Caddy: Underscore

To Caddy:

GET / HTTP/1.1

x_user: ADMIN

After Caddy* (fastcgi):

GET / HTTP/1.1

x-user: anonymous

x_user: ADMIN

Envoy: Double headers

- name: envoy.filters.http.ext_authz

typed_config:

"@type":

type.googleapis.com/envoy.extensions.filters.http.ext_authz.v3.ExtAuthz

http_service:

server_uri:

uri: ext_authz

cluster: ext_authz-http-service

timeout: 0.250s

authorization_response:

allowed_upstream_headers:

patterns:

- exact: x-user

Envoy: Double headers

To Envoy

GET / HTTP/1.1

x-user: asdasd

x-user: ADMIN

After Envoy

GET / HTTP/1.1

x-user: User_from_ext_auth

x-user: ADMIN

- Tested on 1.14.4

Special symbols

Test normalization of header names

Traefik(1.7.7)

- Accepts header with a trailing space
- Forwards without the trailing space
- net/http
- Before Go 1.13.1 and Go 1.12.10 (CVE-2019-16276)

Special symbols

To Traefik

GET / HTTP/1.1

x-user : ADMIN

After Traefik

GET / HTTP/1.1

x-user: ADMIN

x-user: anonymous

Web cache poisoning

Header smuggling with multiple headers

Nginx allows multiple Host headers

- Nginx uses 1st Host
- fastcgi_pass or uwsgi_pass gets 2nd Host
- App trusts Host header
- Cache proxy forwards multiple Host headers (smuggled), 2nd Host header is injection point

Web cache poisoning

Header smuggling and Range header

- Range header returns part of response body

Request:

- Range: bytes=33-

Response:

- Status - 206
- Content-Range: bytes 33-106/107

Range header

- 206 is allowed to cache, by standard
- Most cache proxies don't cache, by default
- Can be configured to cache

- Varnish doesn't proxy Range header*

Range header

```
sub vcl_fetch {  
    # ...  
    if (beresp.status >= 200 && beresp.status < 300) {  
        set beresp.cacheable = true;  
    }  
    # ...  
}
```

<https://docs.fastly.com/en/guides/http-status-codes-cached-by-default>

Range header

- Browser treats 206 as 200
- If 1 range, server returns same Content-Type
- Browser renders part of response

- Attack can control part of response
- Back-end must support Range

Range header

- Send request with smuggled Range header
- Cache proxy forwards it to Back-end
- Back-end returns a part of response
- Cache proxy caches the part
- Victim's browser renders only the part
- Attacker can escape context

HTTP/2

- Binary protocol
 - Length for fields
- Frames (Headers, Data)
- High-level compatibility with HTTP/1.1
- Pseudo-headers
 - :method, :scheme, :authority, :path
- HTTP/2 -> HTTP/1.1

HTTP/2

HTTP/1.1:

GET /path/ HTTP/1.1

Host: target.com

Header: value

HTTP/2:

:method:GET

:path:/path/

:authority:target.com

:scheme:https

header:value

Restriction bypass

Before

:method:GET
:path:<SP>/admin/
:authority:target.com
:scheme:https
Header: value

After

GET<SP><SP>/admin/ HTTP/1.1
Host: target.com
Header: value

<sp> – white space

<https://t.me/learningnets>

Collision

- :authority
- host header
- absolute uri

Host misrouting

Before Haproxy

:authority:target.com
host:**localhost**

After Haproxy

GET / HTTP/1.1
Host:**localhost**

Haproxy

- Prepend :scheme to path
 - If it's not http or https
- Allows any symbols in :scheme
 - except \x00 \x0a \x0d
- v2.4.0

Misrouting

Before Haproxy

:path:/any

:scheme:**http://localhost/admin?**

:authority:target.com

After Haproxy

GET **http://localhost/admin?://target.com/any** HTTP/1.1

Envoy

- Allows spec symbols and `\x20 \x09` in `:method`
- v1.18.3
- The same for Haproxy

Misrouting

Before Envoy

:method:**GET /admin?**

:path:/

After Envoy

GET /admin? / HTTP/1.1

Nginx:

/admin? /

<https://t.me/learningnets>

CPDoS and HTTP/2

- Cache Poisoning DoS
- When we can poison cache proxy with “broken response”
- Usually, when proxy caches 4xx, 5xx resps
- Meta symbols in header names
- Oversized headers

Conclusion

- New web-servers – old problems
- Lack of path normalization
- Configuration-dependent vulnerabilities
- New protocol – new opportunities

Next steps

- Reading list of related articles/presentations
- Results
- Docker images

https://github.com/GrrrDog/weird_proxies

Thank you

Questions