Security & Performance: Breaking the Conundrum …Again!

@soniaburney @sabrina_burney
Why Again??
STAR WARS: 'LESSONS LEARNED'

DEATH STAR’S SINGLE POINT OF FAILURE

#velocityconf
Scale
Security

DDoS
Financial Fraud
Data Harvest
SEO Poisoning
Spamming

#velocityconf
3<sup>rd</sup> Party Popularity

Average number of embedded page objects from Alexa’s Top 100

1<sup>st</sup> party resources: 48
3<sup>rd</sup> party resources: 62
Technology Trends

HTTP/2
For Faster and Safer Internet

Service Workers
Service Worker Life Cycle

**INSTALLING**
This stage marks the beginning of registration. It’s intended to allow setup of worker-specific resources such as offline caches.

Use `event.waitUntil()` passing a promise to extend the installing stage until the promise is resolved.

Use `self.skipWaiting()` anytime before activation to skip installed stage and directly jump to activating stage without waiting for currently controlled clients to close.

**INSTALLED**
The service worker has finished its setup and it’s waiting for clients using other service workers to be closed.

**ACTIVATING**
There are no clients controlled by other workers. This stage is intended to allow the worker to finish the setup or clean other worker’s related resources like removing old caches.

Use `event.waitUntil()` passing a promise to extend the activating stage until the promise is resolved.

Use `self.clients.claim()` in the activate handler to start controlling all open clients without reloading them.

**ACTIVATED**
The service worker can now handle functional events.

**Events**

- **install**
- **activate**
- **message**
- **fetch**
- **sync**
- **push**

**Functional events**

1. After registering a service worker to control a scope, each time a client in that scope requests a resource from anyplace in the network...
   - Page under scope
   - The Internet

2. The service worker for that scope enters the game and dispatches the request.
   - Page under scope
   - Service Worker
   - The Internet

3. Through `event.respondWith()` and passing a response object, the service worker can answer the client’s request with a custom response made out of different sources: the network, storage or its own code.
   - Page under scope
   - Service Worker
Service Worker Gotchas!

- HTTPs
- Not supported on all browsers
- Browser can terminate Service Worker at any time
- No DOM Access
- No sync xhr/local storage supported
- No Global State
- No Credentials/Non CORS fails By Default
- Fetch events limited to scope and no fetch events for iframes, service workers, requests triggered within service workers
Service Workers: Analytics Monitoring
Third Party Analytics Monitoring

Adobe Analytics

mPulse

WEBPAGETEST

dynatrace

New Relic

NGINX

#velocityconf
What do third party analytics tools have to do with Service Workers?
Offline Analytics
Tracking Metrics with Service Workers

- navigator.connect to track metrics both online and offline

```javascript
self.addEventListener('activate', function(event) {
  event.waitUntil(
    navigator.services.connect('https://thirdparty.com/services/analytics', {name: 'analytics'}));
});

self.addEventListener('fetch', function(event) {
  navigator.services.match({name: 'analytics'}).then(
    port.postMessage('log fetch'));
});
```

...Where does performance and security come in?

- Less client side JavaScript logic
  - After Service Worker is installed, no need for browser to re-parse analytics reporting code
  - Reduce risk of script injection (no DOM access)
- Asynchronous metric reporting (non blocking)
- Avoid SPOF
- User Agent can terminate the Service Worker at ANY time
Why don’t popular analytics tools leverage Service Workers?
<iframe> & Content-Security-Policy
What Are Sites Using Now?

From Alexa’s Top 100 domains

65% of sites use `<iframe>`

2% of sites use `Content-Security-Policy` header
Why? ..... 3rd Party Vulnerabilities

- Single-Point-of-Failure
- Compromised content
Vendor Related Breaches:
- Walmart Canada looks into possible credit card data breach
- CVS probes card breach at online photo unit
- Expedia, Travelocity, Hotels.com warn customers of phishing scam

Ad Network Related Breaches:
- Hackers hit Google AdWords and Adsense networks
- DoubleClick ad fraud campaign lures victims with promise of adult videos
- 1 million machines hacked for AdSense revenue in Click Fraud attacks
How can `<iframe>` or Content-Security-Policy help?
**<script> vs <iframe>**

```javascript
var myIframe=document.createElement("IFRAME");
myIframe.src="about:blank";
myIframe.addEventListener('load',
    function (e) {
        var myElement=document.createElement("SCRIPT");
        myElement.type="text/javascript";
        myElement.src="http://3rdparty.com/object.js";
        myIframe.contentDocument.body.appendChild(myElement);
    }, false);

document.body.appendChild(myIframe);
```

- **Security**
  - Limited access to site content

- **Performance**
  - No Single-Point-of-Failure
<script> with Content-Security-Policy


- Prevent Single-Point-Of-Failure with unknown 3rd parties
Performance Comparison: Single-Point-of-Failure

Original

CSP

iFrame

#velocityconf
How Can We Enhance Security?
<iframe>: sandbox

Original HTML

```html
<script src="http://3rdparty.com/object.js"></script>
```

Modified HTML

```html
<iframe src="http://3rdparty.com/object.html" sandbox></iframe>
```

```html
<iframe src="http://3rdparty.com/object.html" sandbox="allow-scripts"></iframe>
```

allow-forms allow-modals allow-orientation-lock allow-pointer-lock allow-popups allow-popups-to-escape-sandbox allow-same-origin allow-scripts allow-top-navigation
Content-Security-Policy: sandbox

Content-Security-Policy:

default-src 'self' ;

img-src 'self' https://*.google.com ;

script-src 'self' http://www.google-analytics.com 'unsafe-inline' http://*gstatic.com ;

style-src 'self' https://fonts.googleapis.com ;

font-src 'self' https://themes.googleusercontent.com ;

frame-src 'self' http://3rdparty.com

sandbox ‘allow-scripts’;

allow-forms allow-modals allow-orientation-lock allow-pointer-lock allow-popups allow-popups-to-escape-sandbox allow-same-origin allow-scripts allow-top-navigation
Inline Code Considerations

Original HTML: Sample Analytics Code

```html
<script type="text/javascript">
    var _comscore = _comscore || [];
    _comscore.push({ c1: "2", c2: "3005660" });
    (function () {
        var s = document.createElement("script"), el = document.getElementsByTagName("script")[0]; s.async = true;
        s.src = (document.location.protocol == "https://sb.scorecardresearch.com/beacon.js";
        el.push.parentNode.insertBefore(s, el );
    })();
</script>
```
Modified HTML: Sample Analytics Code

```html
<iframe srcdoc='&lt;script&gt; var _comscore = _comscore || []; _comscore.push({ c1: "2", c2: "3005660" }); (function () {
    var s = document.createElement("script"), el = document.getElementsByTagName("script")[0]; s.async = true;
    el.push.parentNode.insertBefore(s, el);
})();&lt;/script&gt;
sandbox= "allow-scripts"
&lt;/iframe&gt;
```
User Session Considerations

- Referer: https://firstparty.com/sabrina.burney.3?ut=3856&pt=1209&name=sabrinaburney
- Information Leakage
<iframe>: referrerpolicy

Original HTML

```html
<script src="http://3rdparty.com/object.js"></script>
```

Modified HTML

```html
<iframe src="http://3rdparty.com/object.js" referrerpolicy="no-referrer"></iframe>
```

no-referrer no-referrer-when-downgrade origin origin-when-cross-origin unsafe-url
Content-Security-Policy: referrer

Content-Security-Policy:

default-src 'self' ;
img-src 'self' https://*/.google.com ;
script-src 'self' http://www.google-analytics.com 'unsafe-inline'
http://*gstatic.com ;
style-src 'self' https://fonts.googleapis.com ;
font-src 'self' https://themes.googleusercontent.com ;
frame-src 'self' http://3rdparty.com ;
referrer origin ;

no-referrer no-referrer-when-downgrade origin origin-when-cross-origin unsafe-url
<iframe> & CSP: Secure and Optimal for 3rd Party Content
Web <link>ing
Resource Hints & Preload

```html
<link rel="dns-prefetch" href="//example.com">

<link rel="preconnect" href="//example.com">

<link rel="prerender" href="/product-page.html">

<link rel="prefetch" href="/common.js">

<link rel="preload" href="/homestyle.css">
```
Where Does Security Fit In?
Preload & Prefetch: The AS attribute

AS media script style image worker embed object document font

<Link rel="prefetch" href="/dir/common.js" as="script"/>

Link: <dir/common.js>; rel=prefetch; as=script

<Link rel="preload" href="/dir/styles.css" as="style"/>

Link: <dir/styles.css>; rel=preload; as=style
Preload, Prefetch & Content-Security-Policy

<meta http-equiv="Content-Security-Policy" content="default-src 'self'; style-src 'self' https://fonts.googleapis.com; script-src 'self' http://*.google-analytics.com 'unsafe-inline'">

One Security Policy to protect them all? NO

Not all resources are created equal!
Service Workers: 3rd Party Content Control
Client Reputation

- Typically, Client Reputation Strategies are **Backend**
- Leverage Service Workers to control third party content based on specified criteria to improve performance and security

Expand to an Adaptive Reputation Strategy…
Service Worker Reputation Implementation

Pseudocode:

```javascript
self.onfetch = function(event) {
  event.respondWith(
    // retrieve list of acceptable third party domains ahead of time and other
    // information from remote txt or database

    if request URL has domain that matches ones in list ?
    if request domain matches other required criteria ?
    adaptive - timing/payload/counter metrics ?
      fetch the event.request
    else block
      // whitelist or blacklist or both, to grow based on incoming requests
  );
};
```
Service Worker Reputation Implementation

Example:

```javascript
self.addEventListener('activate', function(event) {
    if (self.clients && clients.claim) {
        clients.claim();
    }

    var policyRequest = new Request('thirdparty_urls.txt');

    fetch(policyRequest).then(function(response) {
        return response.text().then(function(text) {
            result=text.toString();
        });
    });
});
```
Service Worker Reputation Implementation

Example:

```javascript
self.addEventListener('fetch', function(event) {

  var urlstring = event.request.url;
  var hostname = (urlstring.match(/\//[\^\]\]*/i).toString()).replace(/\//g, "").toString();
  var regexHostname = new RegExp(hostname);

  if(regexHostname.test(result)) {
    getCounter(event, rspFcn);
    var rspFcn = function(event){
      if (flag > 0){
        //If counter exceeded, retrieve from cache or serve 408
        caches.open('sabrina_cache').then(function(cache) {
          var cachedResponse = cache.match(event.request.url).then(function(response) {
            if(response) {console.log("Found response in cache"); return response;}
            else{
              console.log("Did not find response in cache");
              return (new Response('', {status: 408,statusText: 'Request timed out.'}));
            }
          }).catch(function() {
            return (new Response('', {status: 408,statusText: 'Request timed out due to error.'}));
          });
        event.respondWith(cachedResponse);
      }
    }
    return;
  }
});
```
Service Worker Reputation Implementation

Example:

```javascript
else{
    Promise.race([timeout(500), fetch(event.request.url, {mode: 'no-cors'})]).then(function(value){
        if(value=="timeout"){
            console.log("Timeout threshold reached, update counter");
            updateCounter(event.request.url); //use promises here
        } else console.log("Timeout threshold not reached, retrieve request as needed without updating counter");
    //If counter not exceeded (normal request), then add to cache
    caches.open('sabrina_cache').then(function(cache) {
        console.log("Adding to cache");
        cache.add(event.request.url);
    }).catch(function(error) {
        console.error("Error" + error);
        throw error;
    });
    });
}
else {
    event.respondWith(fetch(event.request));
}
});
```
Why Obfuscate 3rd Party Content?

“Attackers will look at what content is dynamically placed into a site, and look to compromise one of those providers. If the target site blindly trusts the content sent from a provider, the attacker knows the site can be compromised by malicious content sent by the provider.” ~ www.stateoftheinternet.com

“Most companies do not have a process for assessing security third-party partner capabilities before they do business with them.” ~ blog.securityscorecard.com
URL Obfuscation: Proxy Critical 3rd Party Assets

Let's take a Third Party URL

```html
<img src="http://image.thirdparty.com/images/header_banner.jpg"/>
```

1. First Party Origin Servers
2. Reverse Proxy
3. Third Party Network
URL Obfuscation: Benefits

What if we apply URL rewrites to multiple 3rd party resources in a page?

- **Security**
  - Hide content from malicious end users
  - Enforce additional layer of privacy

- **Performance**
  - Control over SPOF
  - New caching rules
  - Faster delivery and rendering
URL Obfuscation: DNS lookup and Connection

- Connection time delays per individual hostnames
- Connection time reduced with rewritten URLs
- Connection time further reduced with rewritten URLs and HTTP/2


- Risks
  - Exposing content sources
  - Large header size
URL Obfuscation: Content-Security-Policy

Replace:

```
```

With:

```
Content-Security-Policy: script-src 'self' *.obf1.firstparty.com 'unsafe-inline'
*.obf2.firstparty.com ; report-uri /cspreport
```

- **Benefits**
  - Hide content sources
  - Smaller header size
Proxy and obfuscate critical 3rd party assets
Other Service Worker Applications

- Input Validation

- Geo Content Control with the Geofencing API

- Load Balancing
  - Switching off between multiple server URLs

- Dependency Injector
  - Distinguishing production versus staging or production versus a fallback
A Performance Solution Can Be A Security Solution!

- Bridge the gap
- First Party Attackers & Third Party Attackers
- Explore Service Worker Applications

Questions?
@soniaburney @sabrina_burney