Mozilla Firefox 0-day

Browser Side-Channel Attack to Leak Installed Applications

AVTOKYO2024

Satoki Tsuji

Ricerca Security, Inc.

WebSec, AISec, Pentesting

𝕏 : @satoki00

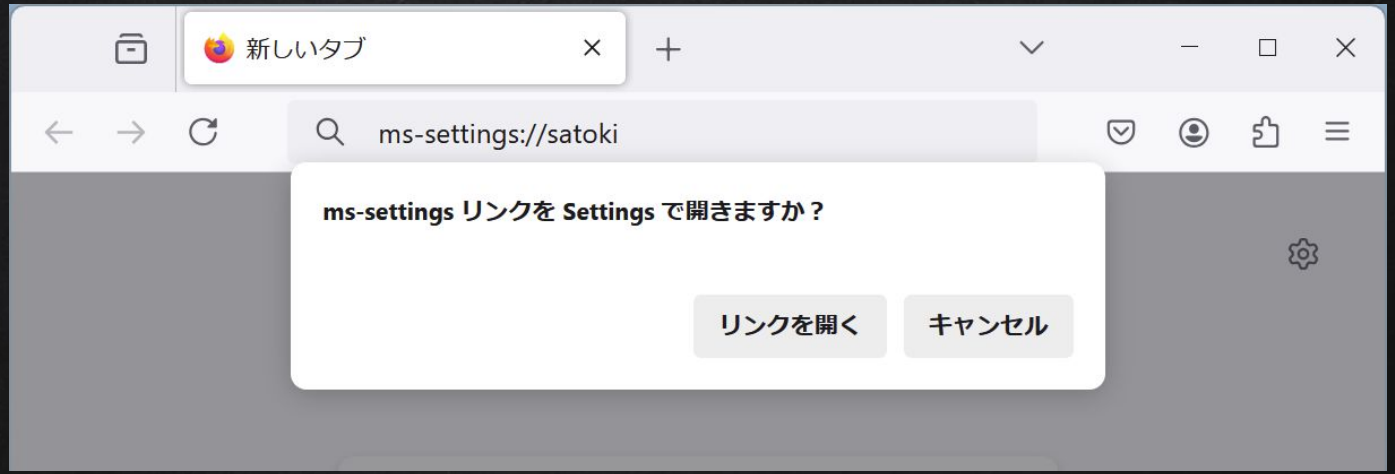
AVTOKYO2020

Techniques for restoring room images from virtual background images and their automation

AVTOKYO2023

Techniques for Prompt Injection and Filter Bypass in AI

Today's Target



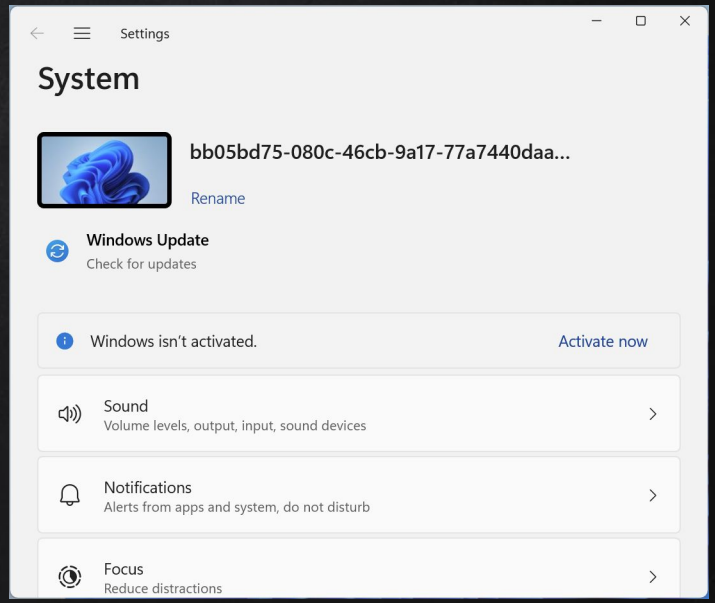
URL Protocol Handler

URL Protocol Handler

Mechanism for invoking applications via scheme.

ms-settings://satoki

Custom Scheme



Registered Application

URL Protocol Handler

Any application can register the schemes.
i.e., Steam, Slack, Zoom, Skype, Spotify, and Twitch

satoki00.reg

C: > satoki00.reg

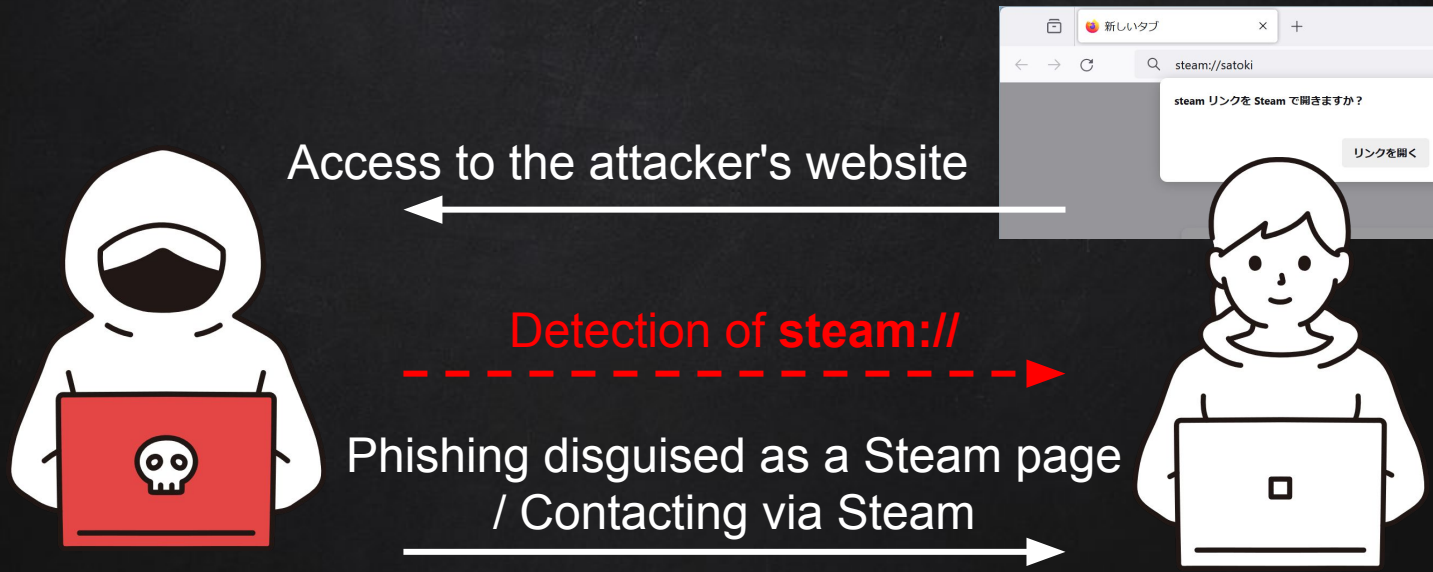
```
1   Windows Registry Editor Version 5.00
2
3   [HKEY_CLASSES_ROOT\satoki00]
4   @="URL:satoki00 Protocol"
5   "URL Protocol"=""
6
7   [HKEY_CLASSES_ROOT\satoki00\shell]
8
9   [HKEY_CLASSES_ROOT\satoki00\shell\open]
10
11  [HKEY_CLASSES_ROOT\satoki00\shell\open\command]
12  @="\"C:\\app.exe\" \"%1\""
```

URL Protocol Handler

Profit?

URL Protocol Handler Leaks

If an attacker can tell **steam://** existence remotely:



URL Protocol Handler Leaks

If the existence of protocol handlers can be enumerated:

Leak protocol handlers

- Hobby-use PC
- Young person
- **No Avast products!**

instagram:// → Exist
skype:// → Not Exist
slack:// → Not Exist
spotify:// → Exist
steam:// → Exist
twitch:// → Exist
twitter:// → Exist
whatsapp:// → Exist
avastpam:// → Not Exist

URL Protocol Handler Leaks

If the handlers can be detected across multiple websites:

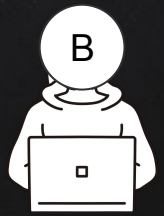
Web Site 1

Web Site 2

The same person!
→ **User tracking**

User A

instagram:// → Exist
skype:// → Not Exist
slack:// → Not Exist
spotify:// → Exist
steam:// → Not Exist
twitch:// → Exist
twitter:// → Exist
whatsapp:// → Exist
zoommtg:// → Not Exist



User B

instagram:// → Exist
skype:// → Not Exist
slack:// → Not Exist
spotify:// → Exist
steam:// → Not Exist
twitch:// → Exist
twitter:// → Exist
whatsapp:// → Exist
zoommtg:// → Not Exist

Browser Side-Channel Attacks

How to capture behavior outside of the website?

What's important in this attack:

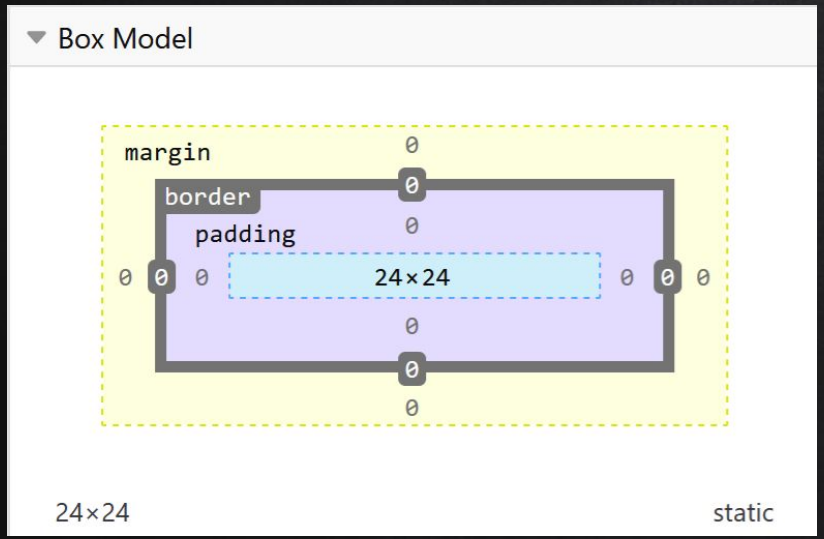
- Enumeration
- Stealth
- Speed

Onblur triggered by popups → It works, but not stealth

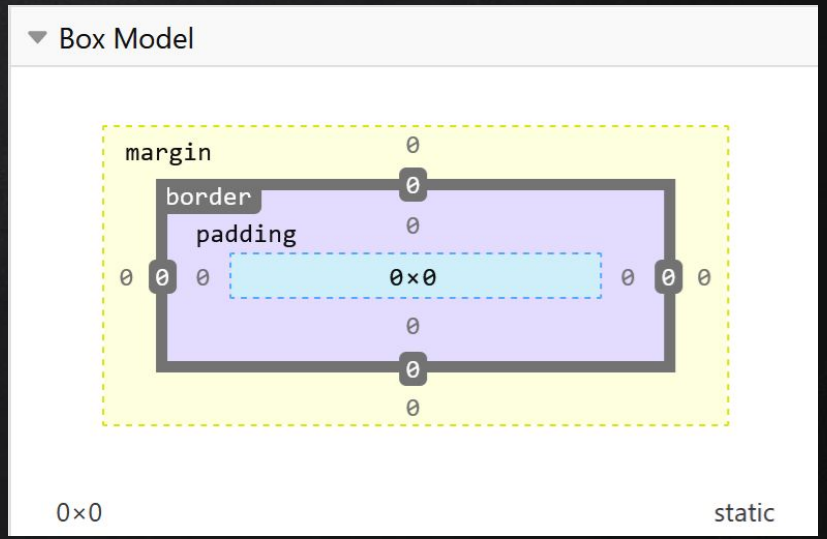
Browser Side-Channel Attacks

CVE-2020-15680 (Reported by Rotem Kerner)

<img src="ms-settings://satoki">



<img src="satoki://satoki">

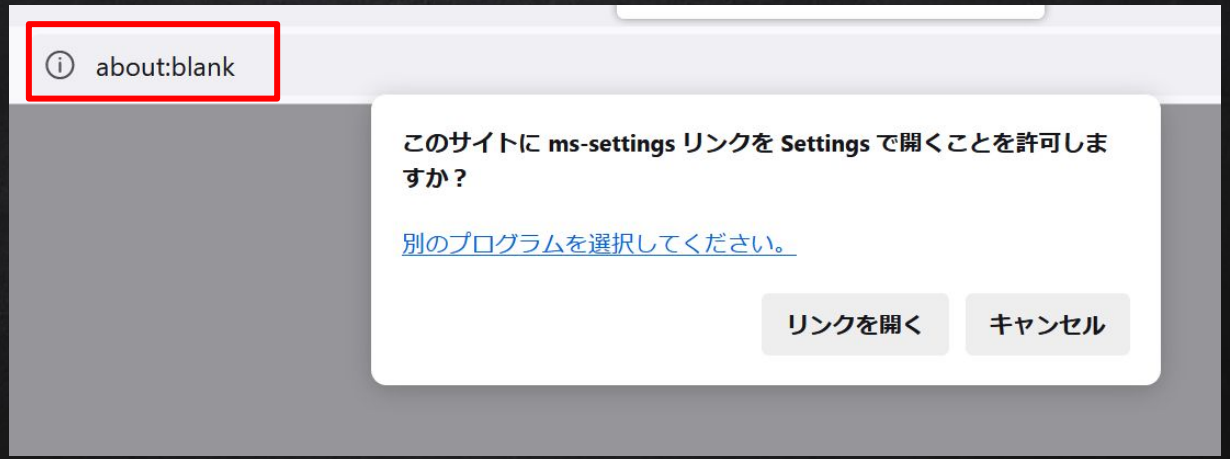


Browser Side-Channel Attacks

CVE-2024-9398
(0-day 1)

Browser Side-Channel Attacks

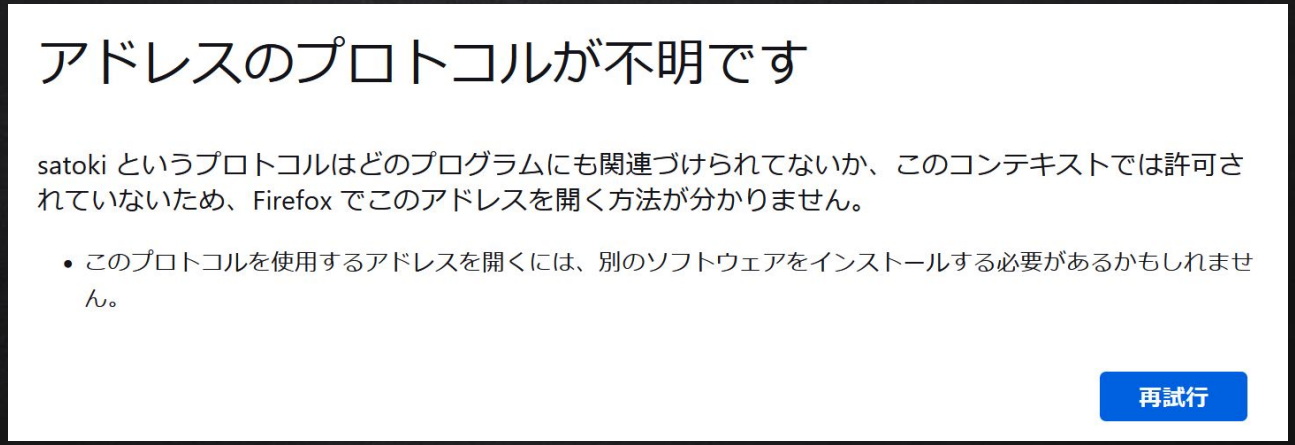
CVE-2024-9398 (0-day 1)




```
open01 = window.open("ms-settings://satoki");
```

Browser Side-Channel Attacks

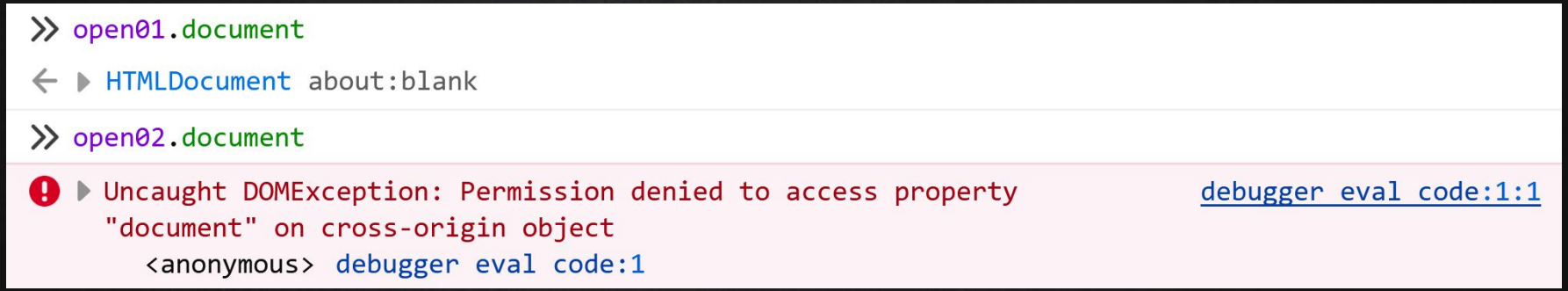
CVE-2024-9398 (0-day 1)




```
open02 = window.open("satoki://satoki");
```

Browser Side-Channel Attacks

CVE-2024-9398 (0-day 1)



open01 → ms-settings://satoki → about:blank → Accessible

open02 → satoki://satoki → Cross-Origin Error

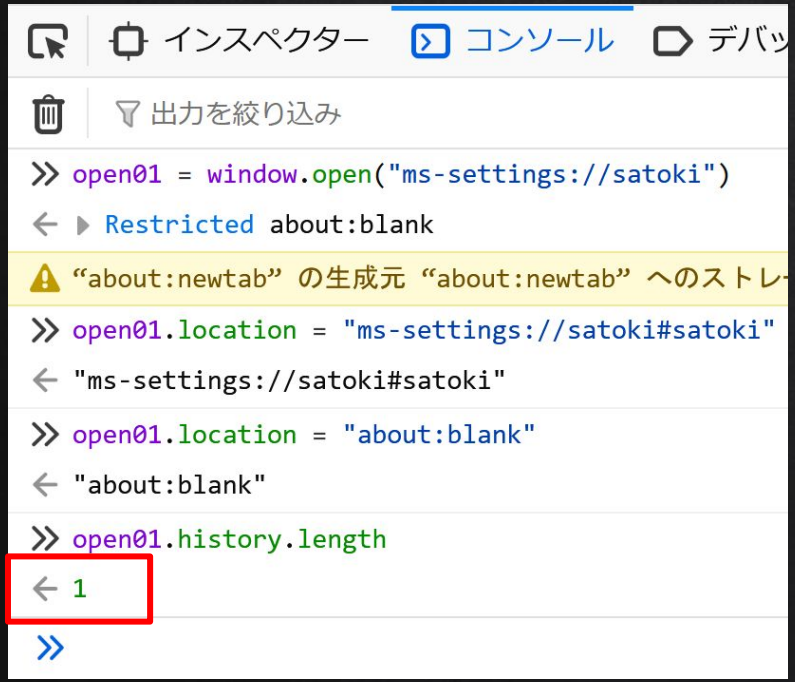
Browser Side-Channel Attacks

CVE-2024-9398 (0-day 1)

Error-Based Oracle

Browser Side-Channel Attacks

CVE-2024-9398 (0-day 1)



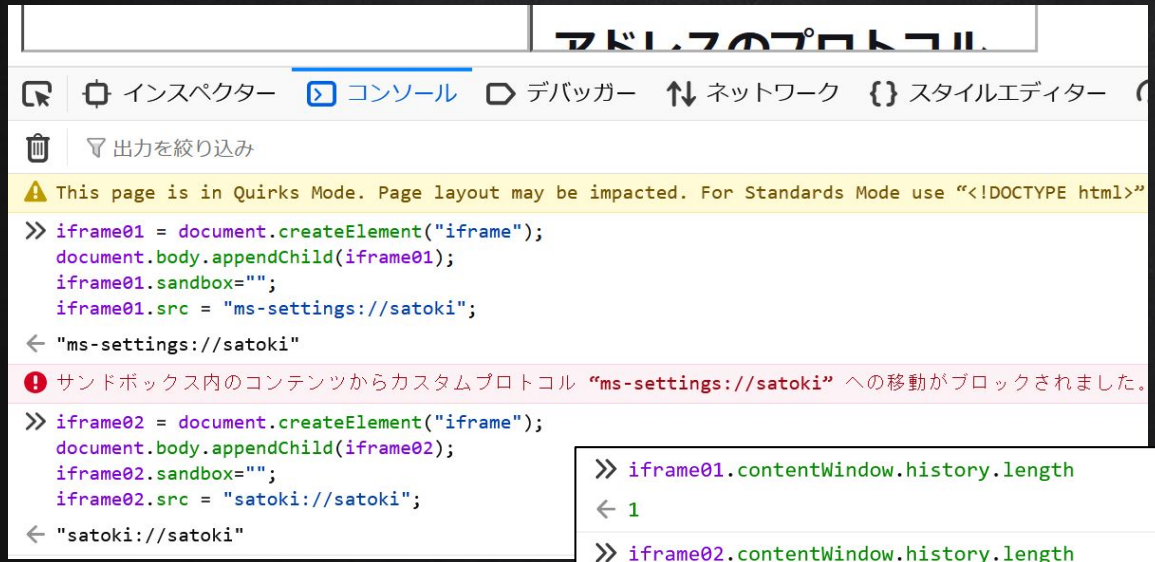
Browser Side-Channel Attacks

CVE-2024-9398 (0-day 1)

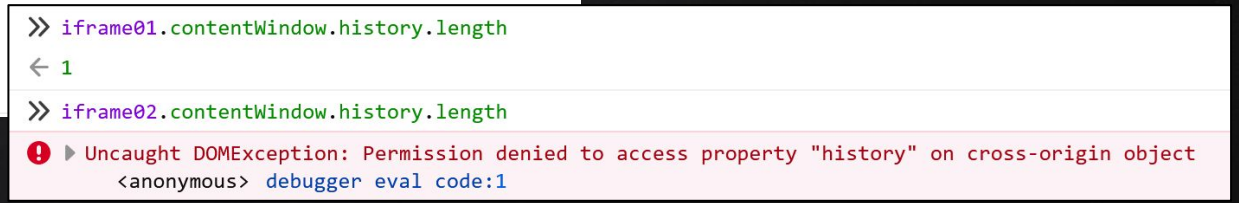
History-Based Oracle

Browser Side-Channel Attacks

😵 No window open permission, popups aren't stealthy!



iframe.sandbox = "";



Browser Side-Channel Attacks

CVE-2024-5690
(0-day 2)

Browser Side-Channel Attacks

CVE-2024-5690 (0-day 2)

```
<img src="ms-settings://satoki" onerror="alert('ms-settings')">
<img src="satoki://satoki" onerror="alert('satoki')">
```

The image size is the same,
but what about the time until the event handler fires?

Browser Side-Channel Attacks

CVE-2024-5690 (0-day 2)

```
» measureLoadTime("ms-settings://satoki", 10000).then(time => console.log(time));
← ▶ Promise { <state>: "pending" }
  13952
» measureLoadTime("satoki://satoki", 10000).then(time => console.log(time));
← ▶ Promise { <state>: "pending" }
  6019
» measureLoadTime("satoki://satoki", 10000).then(time => console.log(time));
← ▶ Promise { <state>: "pending" }
  5864
» measureLoadTime("ms-settings://satoki", 10000).then(time => console.log(time));
← ▶ Promise { <state>: "pending" }
  13013
```

ms-settings://satoki 6000ms, satoki://satoki 13000ms

Browser Side-Channel Attacks

CVE-2024-5690 (0-day 2)

Time-Based Oracle

Browser Side-Channel Attacks

CVE-2024-5690
DUPLICATE
(0-day 3)

Browser Side-Channel Attacks

CVE-2024-5690:DUPLICATE (0-day 3)

```
<html>
    <head>
    <meta http-equiv="Content-Security-Policy" content="img-src 'self';">
    </head>
    <body>
    <img src="ms-settings://satoki">
    <img src="satoki://satoki">
    </body>
</html>
```

Browser Side-Channel Attacks

CVE-2024-5690:DUPLICATE (0-day 3)

ms-settings://satoki → Blocked by CSP

satoki://satoki → Image fails to load

Browser Side-Channel Attacks

CVE-2024-5690:DUPLICATE (0-day 3)

How to catch CSP violations?

report-uri

Browser Side-Channel Attacks

CVE-2024-5690:DUPLICATE (0-day 3)

```
{
  "csp-report": {
        "blocked-uri": "ms-settings",
        "column-number": 1,
        "disposition": "enforce",
        "document-uri": "http://localhost:5555/",
        "effective-directive": "img-src",
        "original-policy": "img-src 'self'; report-uri http://localhost:5555/omg",
        "referrer": "",
        "status-code": 200,
        "violated-directive": "img-src"
  }
}
```

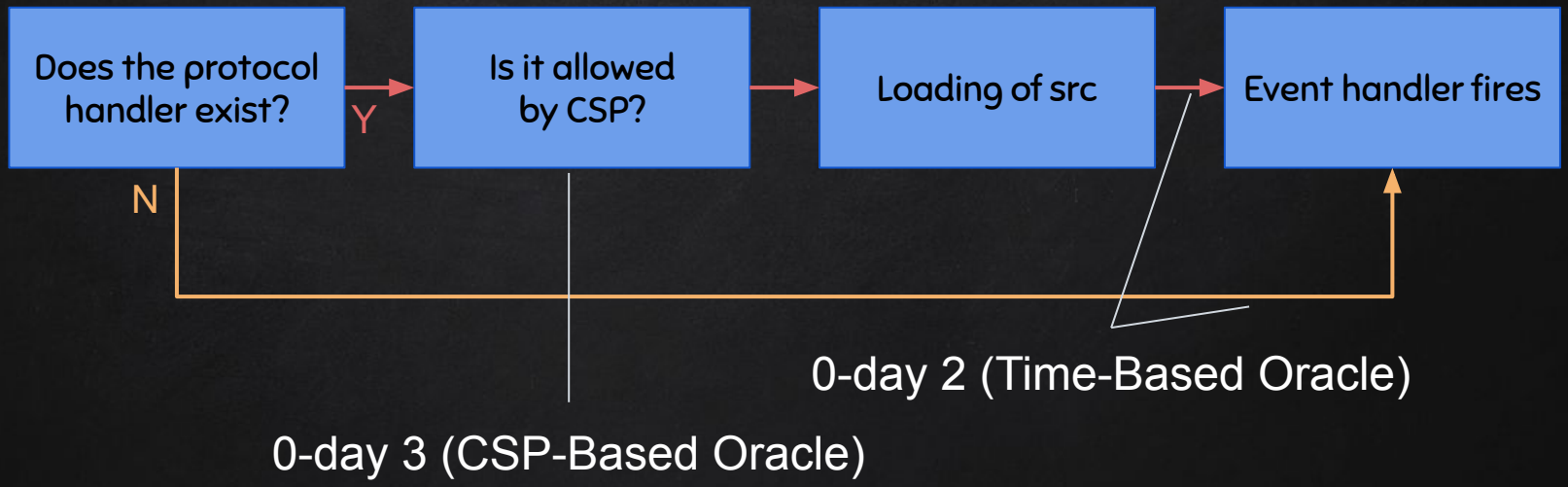
Browser Side-Channel Attacks

CVE-2024-5690:DUPLICATE (0-day 3)

CSP-Based Oracle

Cause of Side-Channel Attacks

Caused by the difference in the browser processing flow.



Find Your Side-Channel Attacks

There may be various other techniques out there!

Welcome to browser side-channel world!

※ It is not effective for tracking **Tor** users because the URL protocol handler is not enabled.

Special Thanks

・Rotem Kerner
Basic idea of the attack
https://www.fortinet.com/blog/threat-research/leaking-browser-url-protocol-handlers
・st98 (@st98_)
Inspiration for the time-based oracle
・ptr-yudai (@ptrYudai)
Inspiration for the stealth error-based oracle

The End

The End