

CHROME ALONe

Transforming a Browser into a C2 Platform



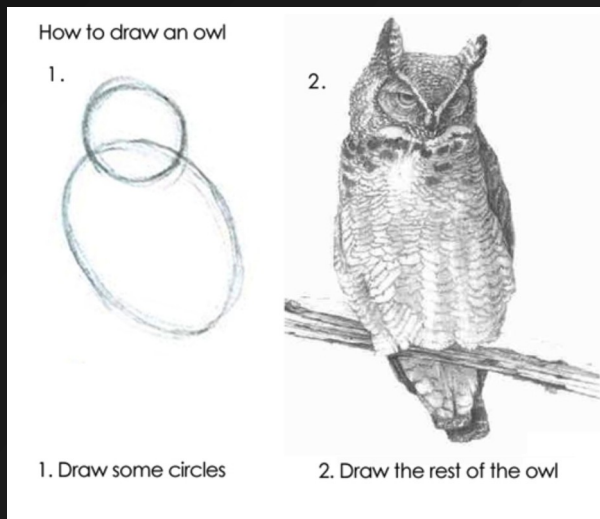
whoami

- **Mike Weber**
 - Building offensive Chrome extensions since 2018
- **Principal Security Engineer at Praetorian Security**
 - Praetorian Labs Researcher
 - Offensive Tool Developer
 - Occasional 0-day hunter
- **Who I am not**
 - ~~Director of Attack & Penetration at Protiviti~~
 - ~~Senior Cyber Security Consultant at EY~~
 - ~~Senior Advisor at Steelhead Advisors LLC~~
 - ~~Principal Scientist at BAE Systems~~
 - ~~Software Engineer at Lockheed Martin~~



Code + Slides Online

github.com/praetorian-inc/ChromeAlone



Why target the browser?

- It's whitelisted by EDR
- It's how the user accesses most internal resources
- It's everywhere
- Chromium has so many built-in features, it's practically an OS

Browser Extensions

- What you used to use to install adblockers
- Convenience features for power users



Ad Blocker: Stands AdBlocker
Extension



Adblock for Youtube™
Extension



AdBlock — block ads across the web
Extension



AdBlocker Ultimate
Extension



Adblock Plus - free ad blocker
Extension



AdGuard AdBlocker
Extension



Pie Adblock - A Powerful Free Ad Blocker
Extension

Known Extension Abuse

- Malware authors have bought or published apps in the Chrome Store to make a quick buck
 - Adware / Click Fraud
 - Crypto Mining
 - Credential + Cookie Theft
- LummaC2 sideloaded extensions as a post-exploitation attack
- Supply Chain Attacks
 - Cyberhaven was used to attack other extensions and infect them

The Hacker News

Malicious Browser Extensions Infect Over 700 Users Across Latin America Since Early 2025

A phishing campaign infects 722 users with malicious browser extensions in Latin America to steal bank login data.

1 week ago



GBHackers News

Over 40 Malicious Chrome Extensions Impersonate Popular Brands to Steal Sensitive Data

Follow Us on Google News. Cybersecurity firm LayerX has uncovered over 40 malicious Chrome browser extensions, many of which are still available...

3 weeks ago



New York Post

Google Chrome users warned to delete 16 popular extensions due to 'malicious' threat risk

Google Chrome users warned to delete 16 popular extensions due to 'malicious' threat risk.

Feb 28, 2025



BleepingComputer

Malicious Browser Extensions are the Next Frontier for Identity Attacks

A recent campaign targeting browser extensions illustrates that they are the next frontier in identity attacks. Learn more about these...

Jan 7, 2025



Malicious Extensions for Pentesting

- CursedChrome
 - Cookiejacking + HTTP Traffic Proxying
- RedExt
 - Information Extraction (History, Cookies, Page DOM, Screenshots, etc.)
- Sliver
 - Hijacks / modifies any existing extensions on disk
- SquareX
 - Syncjacking / hijacking existing extensions in store with OAuth phishing

Extension Abuse Prevention

- Manifest v3 (RIP adblockers)
 - Reduced capabilities of extensions regarding request interception
- Chrome Store Review Process
 - Every permission needs explicit justification / manual review if you ask for wide permissions
- Chrome Apps Deprecated
 - Lots of powerful capabilities like raw socket access removed from the browser...or not

Isolated Web Apps (IWAs)

Why do Isolated Web Apps Exist? (In Google's words):

So you want to make a new Web API

Follow the TAG design principles! (<https://www.w3.org/TR/design-principles/>)

1.2. "It should be safe to visit a web page"

If it's not:

- Change your API so it's safe
- Change the Web Platform to make it safe (see Cross-Origin Isolation)

1.4. "Ask users for meaningful consent"

If you can't:

- Figure out how to
- Maybe enterprise only

Isolated Web Apps (IWAs)

For when you want something in Chrome that does something
DANGEROUS

- Direct Sockets (raw TCP/UDP send/recv)
- Unrestricted WebUSB
- GetAllScreensMedia

Isolated Web App Protections

How do they work?

Bundling & Signing

All app resources are bundled in a Signed Web Bundle.

The bundle is signed by a developer-owned key.

Strict CSP

Guarantees* that all executable content comes from the bundle.

* apps could ship a JS interpreter to bypass CSP eval restrictions

Isolation

Mandatory Cross-Origin Isolation.

Storage Isolation (StoragePartitions instead of double-keying).

Isolated Web App Protections

Isolated Web Apps - Quotes from the Devs

"...[This] lands us in a place where the API is very abusable, it creates this sort of attractive nuisance of capability..."

"You can design a broken application...it does require that the overall application architecture...trying to be a secure application"

"This is only as secure as chrome apps ever were"

"Who's going to use this besides malware authors?"

Case Study

Let's talk about a "Nightmare" Red Teaming scenario



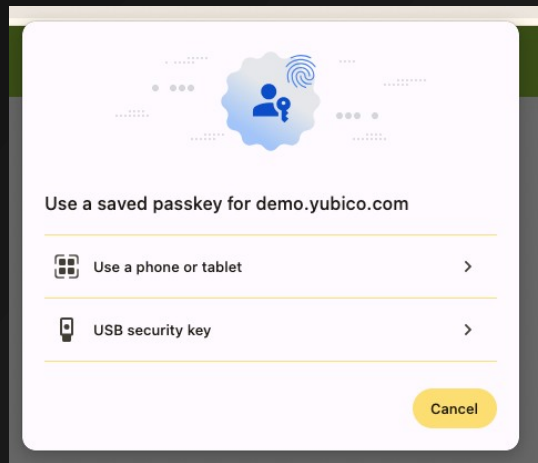
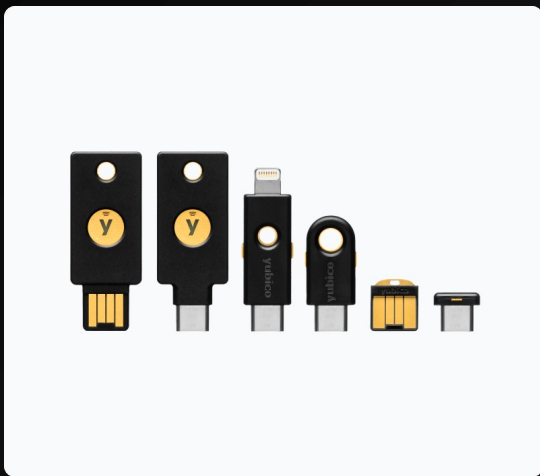
HashiCorp
Vault

**Endpoint
Detection
and Response
(EDR)**



Case Study (Assumptions)

- The user actively logs into SSO once a day using Chrome
- The user NEVER logs directly into Vault
- Our goal is a machine that exposes RDP to specific credentials ONLY in Vault
- Vault is accessible internally via SSO with MFA on a physical Security Key

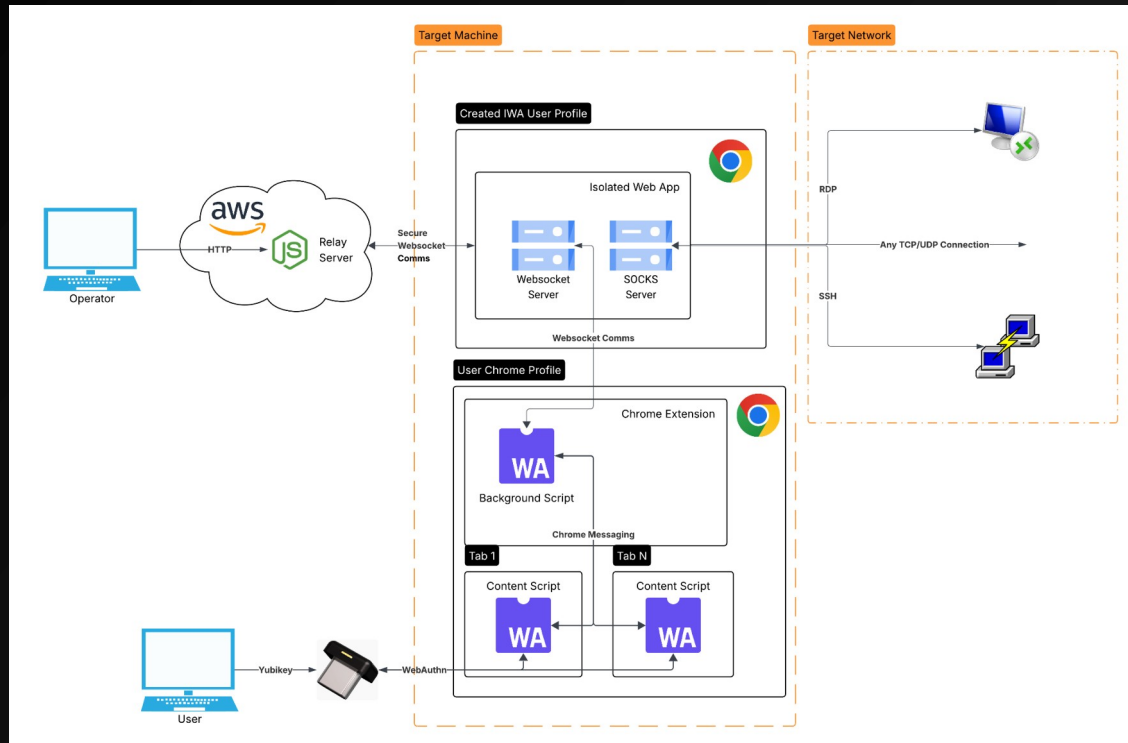


Case Study (Assumptions, Cont.)

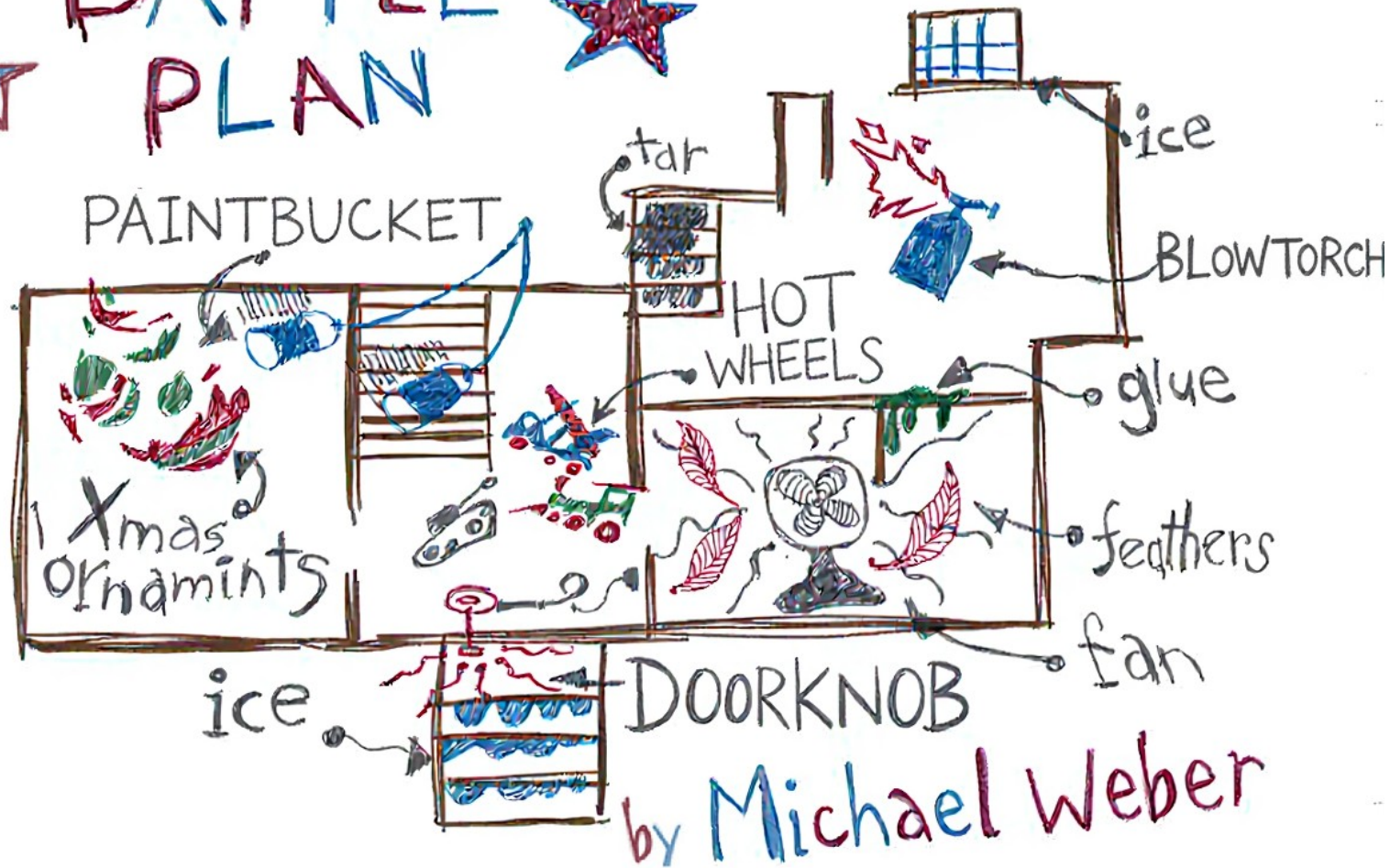
- We trick the user to run one Powershell script on a Windows host
 - This is a Post-Exploitation tool, Initial Access is out of scope
- Minus the Powershell script, EDR is fine-tuned and the SOC is good
- Chrome is whitelisted by EDR for network traffic



Architecture



BATTLE PLAN



by Michael Weber

The Plan

- **Sideload our Extension + Isolated Web App**
- Capture Okta SSO Credentials at Login + Session-ride
- Tunnel Traffic through User via SOCKS
- Trick User into Authenticating to Vault
- RDP into Target via SOCKS
- WIN!

Sideloading Chrome Extensions

- Stored in JSON file on disk
 - Could be Preferences or Secure Preferences
- Individual Extensions request permissions
- Very common to ask for dangerous permissions
 - This one is Adobe Acrobat

```
,
"efaidnbmnnnibpcajpcgiclfefindmkaj": {
  "account_extension_type": 0,
  "ack_external": true,
  "active_bit": false,
  "active_permissions": {
    "api": [
      "alarms",
      "contextMenus",
      "cookies",
      "downloads",
      "nativeMessaging",
      "storage",
      "tabs",
      "webNavigation",
      "webRequest",
      "scripting",
      "offscreen",
      "sidePanel"
    ],
    "explicit_host": [
      "<all_urls>"
    ],
    "manifest_permissions": [],
    "scriptable_host": [
      "<all_urls>",
      "file:///.*",
      "http://.*",
      "https://.*",

```

Sideloaded Chrome Extensions

- Stored in JSON in Preferences/Secure Preferences file on disk
- Secured against modification by hash calculations
- Hash calculations replicated by Trotus/Elex adware in 2016
 - Discussed by tigzy from Adlice
- Public PoCs from Nicholas Murray + Gordon Long for Windows + MacOS

```
{
  "extensions": {
    "settings": {
      "aebldkhhhdcdjpfifhbbdiojplfjncoa": "F0A55962F2B38DA9B61C3B6CC036F2E89A96B10C060E7B0750B36948C089250F",
      "ahfgeienlihckogmohjhadlkjgocpleb": "9F200BB8946C813D9BACEF7976036CF3343441A0244CFBF697617F77E9035ED4",
      "bcpobdlnflhohlfiiecfmhjcnfihfdcl": "2154D6A34154AC1D5920F4808464B4560CDDE9EB292888C95B88BA699A645FA5",
      "efaidnbmninnibpcajpcglclefindmkaj": "78680A663BDB11400B0CDEDBE882EFDFF545489BD7E8FF5C4A83854E3ADD014A",
      "fignfifoniblkonapihmkfakmlgkbkcf": "8D8146F26D8FA887E483EC7477CC81959F356381552BE2CB21F0B5A1F5F27C40",
      "ghbmnnjooekpmoecnninlnbndlohkhi": "B1DAEB53C437DEDD46DEA6A095D8151FAE8845C70E262FBE5B6C912EA5CDE425",
      "mhjfbmdgcfjbbpaeojofohoeofgiehjai": "B9F8B4CBAEA9B323F89907FE73AEC43E6B3B7D6B698F58ACC213DF0F90B7BC15",
      "neajdpdkcdipfabeoofebfddakdcjhd": "038200A2718A2DE48709DF08122857DE2B55DD5D2990AB53C8383C8B3346C085",
      "nkeimhogjdpnpccoofpliimaahmaaome": "84EAB86DAA00BFF2786BF2589C4AAB77DF0E43E8AA1B0A7017CBCC39737FB2C7",
      "nmhkhkegccagdldgiimedpiccmgmieda": "9F3D76F3DAC40B2F0F2E0441B6FBE65F01C9BCDD693075AE809F25BEBFBFF1A"
    }
  },
  "ui": {
    "developer_mode": "227EEFE7D3ADEC0BF2D96CB232558D146196112E1976DBA1D2B485E25EE7650C"
  }
},
```

Sideloaded Chrome Extensions

Permissions

- Read and change your browsing history on all your signed-in devices
- Block content on any page
- Read data you copy and paste
- Communicate with cooperating native applications

Site access

Allow this extension to read and change all your data on websites you visit: ? On all sites

Site settings 🔗

Pin to toolbar 🔌

Allow in Incognito 🔵

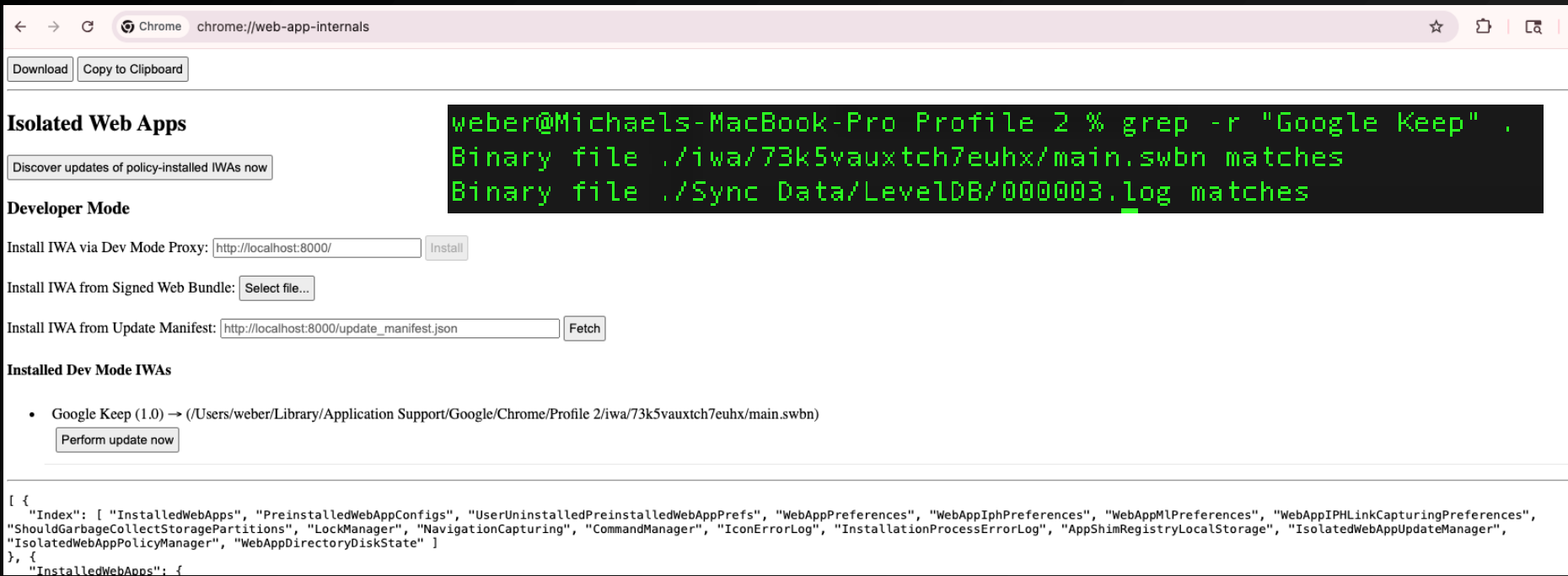
Warning: Google Chrome cannot prevent extensions from recording your browsing history. To disable this extension in Incognito mode, unselect this option.

Allow access to file URLs 🔵

Collect errors 🔵

Installing Isolated Web Apps

- Unlike extensions, IWAs don't touch any JSON files



chrome://web-app-internals

Download Copy to Clipboard

Isolated Web Apps

Discover updates of policy-installed IWAs now

Developer Mode

Install IWA via Dev Mode Proxy:

Install IWA from Signed Web Bundle:

Install IWA from Update Manifest:

Installed Dev Mode IWAs

- Google Keep (1.0) → (/Users/weber/Library/Application Support/Google/Chrome/Profile 2/iwa/73k5vauxtch7euhx/main.swbn)

```
weber@Michaels-MacBook-Pro Profile 2 % grep -r "Google Keep" .  
Binary file ./iwa/73k5vauxtch7euhx/main.swbn matches  
Binary file ./Sync Data/LevelDB/000003.log matches
```

```
[  
  "Index": [  
    "InstalledWebApps", "PreinstalledWebAppConfigs", "UserUninstalledPreinstalledWebAppPrefs", "WebAppPreferences", "WebAppIphPreferences", "WebAppMlPreferences", "WebAppIPHLLinkCapturingPreferences",  
    "ShouldGarbageCollectStoragePartitions", "LockManager", "NavigationCapturing", "CommandManager", "IconErrorLog", "InstallationProcessErrorLog", "AppShimRegistryLocalStorage", "IsolatedWebAppUpdateManager",  
    "IsolatedWebAppPolicyManager", "WebAppDirectoryDiskState"  
  ],  
  {  
    "InstalledWebApps": {
```


Isolated Web Apps on Disk

- That's not JSON at all...

```
0000000: 110e fa6e 2700 0101 0000 0000 0000 0001 ...n'..... 00000b50: 9004 0098 0400 a004 00a8 0400 b804 00d4 .....
0000010: 0000 0001 165f 6d74 735f 7363 6865 6d61 ....._mts_schema 00000b60: c5b7 a8ac 0901 0a00 0000 0000 0000 0100 .....
0000020: 5f64 6573 6372 6970 746f 7202 0801 acc7 _descriptor.... 00000b70: 0000 012c 7765 625f 6170 7073 2d64 742d ...,web_apps-d
0000030: 460e 0c00 0102 0000 0000 0000 0000 0000 F..... 00000b80: 6a70 6e65 626f 6f65 6f65 6463 6b6e 6f63 jpnebooeoedckn
0000040: 0063 fe72 4c2e 0001 0200 0000 0000 0000 .c.rL..... 00000b90: 6d6c 6368 6e6e 616d 6f67 706e 636e 6e6b mlchnnamogpncn
0000050: 0100 0000 011d 7765 625f 6170 7073 2d64 .....web_apps-d 00000ba0: f012 0a87 060a 4869 736f 6c61 7465 642d .....Hisolate
0000060: 742d 4441 5441 4241 5345 5f4d 4554 4144 t-DATABASE_METAD 00000bb0: 6170 703a 2f2f 6e34 6d6b 6534 6e73 7833 app://n4mke4ns
0000070: 4154 4102 0803 acc5 b8a4 8301 0103 0000 ATA..... 00000bc0: 376a 6978 6d62 356c 3637 6134 7774 6d67 7jixmb5l67a4wt
0000080: 0000 0000 0001 0000 0001 2c77 6562 5f61 .....web_a 00000bd0: 6a65 7975 6d69 7179 6469 376f 7a6a 6933 jeyumiqydi7ozj
0000090: 7070 732d 6474 2d66 6d67 6a6a 6d6d 6d6c pps-dt-fmgjjmmml 00000be0: 346c 6b73 6a61 6f36 6571 6161 6963 2f12 4lksjao6eqaaic
00000a0: 666e 6b62 7070 6e63 6162 666b 6464 626a fnkbppncabfkddb 00000bf0: 0b47 6f6f 676c 6520 4b65 6570 1803 2a48 .Google Keep..
00000b0: 696d 6366 6e63 6dc7 020a 780a 3268 7474 imcfncm...x.2htt 00000c00: 6973 6f6c 6174 6564 2d61 7070 3a2f 2f6e isolated-app:/
00000c0: 7073 3a2f 2f6d 6169 6c2e 676f 6f67 6c65 ps://mail.google 00000c10: 346d 6b65 346e 7378 3337 6a69 786d 6235 4mke4nsx37jixm
00000d0: 2e63 6f6d 2f6d 6169 6c2f 3f75 7370 3d69 .com/mail/?usp=i 00000c20: 6c36 3761 3477 746d 676a 6579 756d 6971 l67a4wtmgjeyum
00000e0: 6e73 7461 6c6c 6564 5f77 6562 6170 7012 nstalled_webapp. 00000c30: 7964 6937 6f7a 6a69 3334 6c6b 736a 616f ydi7ozji34lksj
00000f0: 0547 6d61 696c 1801 2a1d 6874 7470 733a .Gmail..*.https: 00000c40: 3665 7161 6169 632f 3265 0830 125f 6973 6eqaaic/2e.0._
0000100: 2f2f 6d61 696c 2e67 6f6f 676c 652e 636f //mail.google.co 00000c50: 6f6c 6174 6564 2d61 7070 3a2f 2f6e 346d olated-app://n
0000110: 6d2f 6d61 696c 2f4a 1a6d 6169 6c2f 3f75 m/mail/J.mail/?u 00000c60: 6b65 346e 7378 3337 6a69 786d 6235 6c36 ke4nsx37jixmb5
0000120: 7370 3d69 6e73 7461 6c6c 6564 5f77 6562 sp=installed_web 00000c70: 3761 3477 746d 676a 6579 756d 6971 7964 7a4wtmgjeyumiq
0000130: 6170 7012 0547 6d61 696c 2200 2801 321d app..Gmail".(.2. 00000c80: 6937 6f7a 6a69 3334 6c6b 736a 616f 3665 i7ozji34lksjao
0000140: 6874 7470 733a 2f2f 6d61 696c 2e67 6f6f https://mail.goo 00000c90: 7161 6169 632f 696d 6167 6573 2f69 636f qaaic/images/i
0000150: 676c 652e 636f 6d2f 6d61 696c 2f3a 1c08 gle.com/mail/... 00000ca0: 6e73 2d76 6563 746f 722e 7376 6718 0132 ns-vector.svg.
0000160: 0010 0018 0020 0028 0130 0038 0048 0050 .....(.0.8.H.P 00000cb0: 6508 3012 5f69 736f 6c61 7465 642d 6170 e.0._isolated-
0000170: 0058 0060 0068 0070 0078 0040 0248 0058 .X.`.h.p.x.@.H.X 00000cc0: 703a 2f2f 6e34 6d6b 6534 6e73 7833 376a p://n4mke4nsx3
0000180: 2058 3058 4058 6058 8001 58c0 0158 8002 X0X@X`X..X..X.. 00000cd0: 6978 6d62 356c 3637 6134 7774 6d67 6a65 ixmb5l67a4wtmg
```


LevelDB

- An open source Google DB Format
 - With mediocre open source tooling...

LevelDB Viewer

File

	Key	Value
1	bytearray(b'_mts_schema_descriptor')	bytearray(b'\x08\x01')
2	bytearray(b'web_apps-dt-...')	bytearray(b'\x08\x01')
3	bytearray(b'web_apps-dt-...')	bytearray(b'\n\n https://drive.google.com/?lfhs=2\x12\x0cGoogle Drive\x18\x01*\x19https://drive.google.com/\x07?...')
4	bytearray(b'web_apps-dt-...')	bytearray(b'\n\n%https://www.youtube.com/?feature=ytca\x12\x07YouTube\x18\x01*\x18https://www.youtube.com/\x1r?...')
5	bytearray(b'web_apps-dt-...')	bytearray(b'\n\n\x91\x01\n:https://docs.google.com/spreadsheets/?usp=installed_webapp\x12\x06Sheets\x18\x01*%https://...')
6	bytearray(b'web_apps-dt-...')	bytearray(b'\nx\n2https://mail.google.com/mail/?usp=installed_webapp\x12\x05Gmail\x18\x01*\x1dhttps://...')
7	bytearray(b'web_apps-dt-...')	bytearray(b'\n\n\x91\x01\n:https://docs.google.com/presentation/?usp=installed_webapp\x12\x06Slides\x18\x01*%https://...')
8	bytearray(b'web_apps-dt-...')	bytearray(b'\nT\n\x1dhttps://mail.google.com/chat/\x12\x0bGoogle Chat\x18\x01*\x1dhttps://mail.google.com/chat/...')
9	bytearray(b'web_apps-dt-...')	bytearray(b'\n\n\x83\x01\n6https://docs.google.com/document/?usp=installed_webapp\x12\x04Docs\x18\x01*!https://...')
10	bytearray(b'web_apps-dt-...')	bytearray(b'\n\n\x87\x06\nHisolated-app://d6sii27z6gonfdc7kcudjbxhfpdvgcqdy2toutlissffdi7cfzqaic/\x12\x0bGoogle ...')

LevelDB

- Actually fairly straightforward layout
 - But that bit about CRC-32 isn't QUITE right...

Offset	Length	Meaning
0	4	CRC-32 of block data
4	2	Int16 data length for this block
6	1	Block type (see below): 1: Full 2: First 3: Middle 4: Last

CRC-32C

- It's this: <https://github.com/google/leveldb/blob/main/util/crc32c.h>
- Save yourself the time and just use their implementation.
 - Don't forget to mask

```
// Return the crc32c of data[0,n-1]
inline uint32_t Value(const char* data, size_t n) { return Extend(0, data, n); }

static const uint32_t kMaskDelta = 0xa282ead8ul;

// Return a masked representation of crc.
//
// Motivation: it is problematic to compute the CRC of a string that
// contains embedded CRCs.  Therefore we recommend that CRCs stored
// somewhere (e.g., in files) should be masked before being stored.
inline uint32_t Mask(uint32_t crc) {
    // Rotate right by 15 bits and add a constant.
    return ((crc >> 15) | (crc << 17)) + kMaskDelta;
}
```

Protobuf

The value of each key is a serialized Protobuf object

Found at https://chromium.googlesource.com/chromium/src/+main/chrome/browser/web_applications/proto/web_app.proto#198

```
194 // Full WebApp object data. See detailed comments in
195 // chrome/browser/web_applications/web_app.h. Note on database identities:
196 // app_id is a hash of launch_url. app_id is the client tag for sync system.
197 // app_id is the storage key in DataStore.
198 message WebApp {
199     reserved "handle_links", "is_in_sync_install", "is_placeholder",
200         "file_handler_os_integration_state",
201         "run_on_os_login_os_integration_state", "url_handlers", "update_token";
202     // Synced data. It is replicated across , all devices with WEB_APPS.
203     //
204     // [sync_data.name] and [sync_data.theme_color] are read by a device to
205     // generate a placeholder icon. Any device may write new values to synced
206     // [name] and [theme_color]. A random last update wins.
207     optional sync_pb.WebAppSpecifics sync_data = 1; // Required.
```

```
223 // Local data. May vary across devices. Not to be synced.
224 //
225 optional string name = 2; // Required.
226 optional uint32 theme_color = 3;
227 optional string description = 4;
228 optional DisplayMode display_mode = 5;
229 optional string scope = 6;
230 optional Sources sources = 7; // Required.
231 // This used to be 'bool is_locally_installed'.
232 required proto.InstallState install_state = 8; // Required.
```

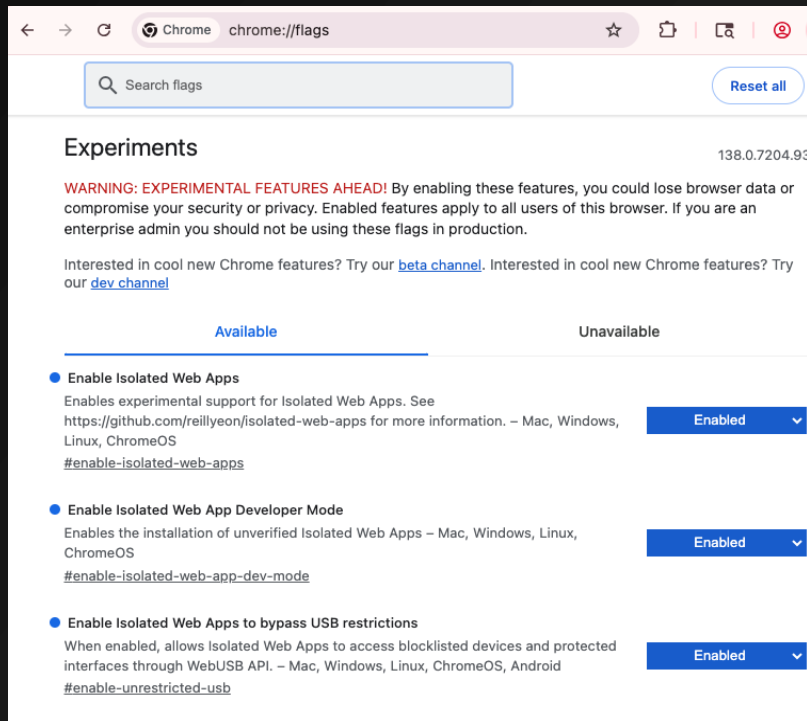
```
66 // A set to track simultaneous installs and uninstalls from multiple install
67 // sources. This should stay in sync with |WebAppManagement| in
68 // chrome/browser/web_applications/web_app_constants.h
69 // and the WebAppManagement enum below.
70 message Sources {
71     optional bool system = 1;
72     optional bool policy = 2;
73     optional bool web_app_store = 3;
74     optional bool sync = 4;
75     optional bool default = 5;
76     optional bool sub_app = 6;
77     optional bool kiosk = 7;
78     reserved 8;
79     optional bool oem = 9;
80     optional bool one_drive_integration = 10;
81     optional bool aps_default = 11;
82     optional bool iwa_shimless_rma = 12;
83     optional bool iwa_policy = 13;
84     optional bool iwa_user_installed = 14;
85     optional bool user_installed = 15;
86 }
```

Sideloading Isolated Web Apps

- Serialize your app into a Protobuf object
- Calculate the CRC-32C for the object
- Add it into the SyncData LevelDB structure
- Make sure these flags are turned on
 - Thankfully these ARE in JSON in

// Local State

```
"browser": {  
  "enabled_labs_experiments": [  
    "enable-isolated-web-app-dev-mode@1",  
    "enable-isolated-web-apps@1",  
    "enable-unrestricted-usb@1"  
  ],  
}
```



chrome://flags

Search flags

Reset all

Experiments

138.0.7204.93

WARNING: EXPERIMENTAL FEATURES AHEAD! By enabling these features, you could lose browser data or compromise your security or privacy. Enabled features apply to all users of this browser. If you are an enterprise admin you should not be using these flags in production.

Interested in cool new Chrome features? Try our [beta channel](#). Interested in cool new Chrome features? Try our [dev channel](#)

Available	Unavailable
Enable Isolated Web Apps Enables experimental support for Isolated Web Apps. See https://github.com/reillyeon/isolated-web-apps for more information. – Mac, Windows, Linux, ChromeOS #enable-isolated-web-apps	
Enable Isolated Web App Developer Mode Enables the installation of unverified Isolated Web Apps – Mac, Windows, Linux, ChromeOS #enable-isolated-web-app-dev-mode	
Enable Isolated Web Apps to bypass USB restrictions When enabled, allows Isolated Web Apps to access blocklisted devices and protected interfaces through WebUSB API. – Mac, Windows, Linux, ChromeOS, Android #enable-unrestricted-usb	

Sideloading Isolated Web Apps

- Other Details like:
 - How to calculate the Isolated Web App ID
 - How to run the app from the command line
 - How to run the app without showing it to the user

How to draw an owl

1.



1. Draw some circles

2.



2. Draw the rest of the owl

DOORKNOB

- Powershell scripts to sideload Chrome Extensions + IWAs
 - Provide an extension folder or a signed web app bundle and it generates the installer for you.
- Loads with minimal user interaction
 - Chrome needs to restart to load the extension, but we can maintain window/tab state



The Plan

- ~~Sideload our Extension + Isolated Web App~~
- **Capture Okta SSO Credentials at Login + Session-ride**
- Tunnel Traffic through User via SOCKS
- Trick User into Authenticating to Vault
- RDP into Target via SOCKS
- WIN!

Malicious Chrome Extensions

- Since we're sideloading, we ask for ALL the scary permissions
 - background keeps Chrome running even if the user closes it
 - clipboardRead is for capturing copied credentials
 - cookies is for stealing user sessions
 - declarativeNetRequest is for disabling CSP + X-Frame-Options
 - history gives context for what the user visits regularly and when
 - <all_urls> gives us these permissions for EVERYTHING
- Not how we capture most credentials though
 - Thanks Manifest v3

```
{
  "manifest_version": 3,
  "name": "ChromeAlone",
  "version": "1.0",
  "description": "",
  "permissions": [
    "activeTab",
    "background",
    "clipboardRead",
    "cookies",
    "declarativeNetRequest",
    "history",
    "nativeMessaging",
    "scripting",
    "tabs"
  ],
  "host_permissions": [
    "<all_urls>"
  ],
}
```

Malicious Chrome Extensions

- Content scripts do the heavy lifting
 - Can inject into every page
 - Intercept EVERY form submission
- We need to run in both “Worlds”
 - MAIN
 - ISOLATED
- Won't the user notice?
 - Remember that bit about `wasm-unsafe-eval`...

```
"content_scripts": [  
  {  
    "matches": ["<all_urls>"],  
    "js": ["wasm/wasm_exec.js", "content.js"],  
    "run_at": "document_end"  
  },  
  {  
    "matches": ["<all_urls>"],  
    "js": ["maintoisolated.js"],  
    "all_frames": true,  
    "run_at": "document_start"  
  },  
  {  
    "matches": ["<all_urls>"],  
    "js": ["inject.js"],  
    "run_at": "document_start",  
    "all_frames": true,  
    "world": "MAIN"  
  }  
],  
"content_security_policy": {  
  "extension_pages": "script-src 'self' 'wasm-unsafe-eval';"  
},
```

WASM

- If you thought javascript sucked to analyze...
- Oh yeah, it's also the entire Golang runtime compiled into WASM

[illegible]

```
main.wasm x
0x000174b    block $label29
0x000174d    loop $label30
0x000174f    block $label28
0x0001751    block $label27
0x0001753    block $label26
0x0001755    block $label25
0x0001757    block $label24
0x0001759    block $label23
0x000175b    block $label22
0x000175d    block $label21
0x000175f    block $label20
0x0001761    block $label19
0x0001763    block $label18
0x0001765    block $label17
0x0001767    block $label16
0x0001769    block $label15
0x000176b    block $label14
0x000176d    block $label13
0x000176f    block $label12
0x0001771    block $label11
0x0001773    block $label10
0x0001775    block $label9
0x0001777    block $label8
```

WASM

- There's no good tooling for it - Ghidra chokes on Hello World.
- `wasm-unsafe-eval` is an all or nothing permission
 - Cannot load WASM without this CSP permission
- You can call back out into javascript from your WASM to essentially have `unsafe-eval`
 - Or you can use it to load more WASM you dynamically decrypt and/or receive over the wire

WASM vs. Javascript

```
//go:build js && wasm
// +build js,wasm

package main

import (
    _ "embed"
    "syscall/js"
)

//go:embed embedded.wasm
var wasmModule []byte

func main() {
    // Convert decompressed data to Uint8Array
    wasmArray := js.Global().Get("Uint8Array").New(len(wasmModule))
    js.CopyBytesToJS(wasmArray, wasmModule)

    // Create a new Go object
    goObj := js.Global().Get("Go").New()

    // Call WebAssembly.instantiate with the Go importObject
    promise := js.Global().Get("WebAssembly").Call("instantiate", wasmArray, goObj.Get("importObject"))
    promise.Call("then", js.FuncOf(func(this js.Value, args []js.Value) interface{} {
        instance := args[0].Get("instance")
        goObj.Call("run", instance)
        return nil
    })))
}
```

```
const go = new Go();
const wasmModule = await WebAssembly.instantiateStreaming(
    fetch(chrome.runtime.getURL('wasm/') + 'main.wasm'),
    go.importObject
);
go.run(wasmModule.instance);
```


WASM

- The Chrome Store review process has no way to deal with this

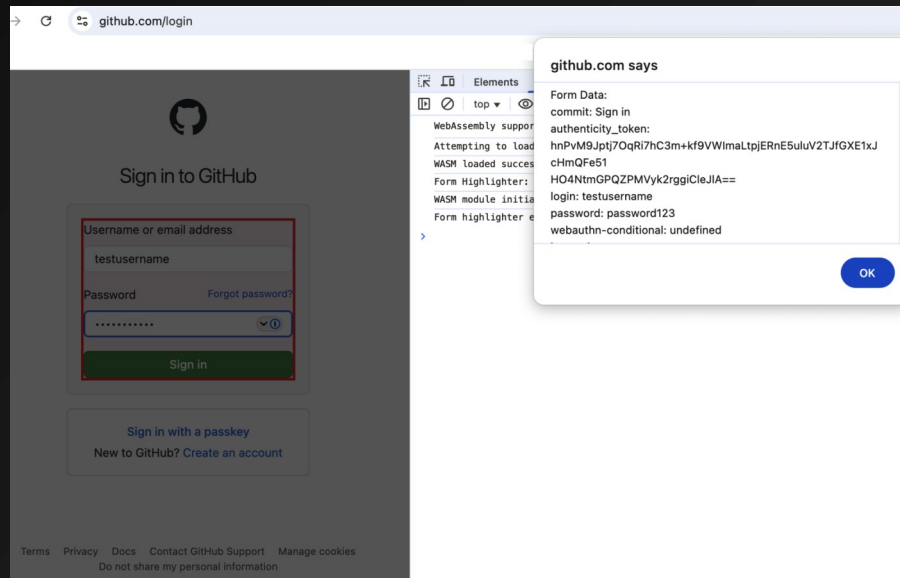
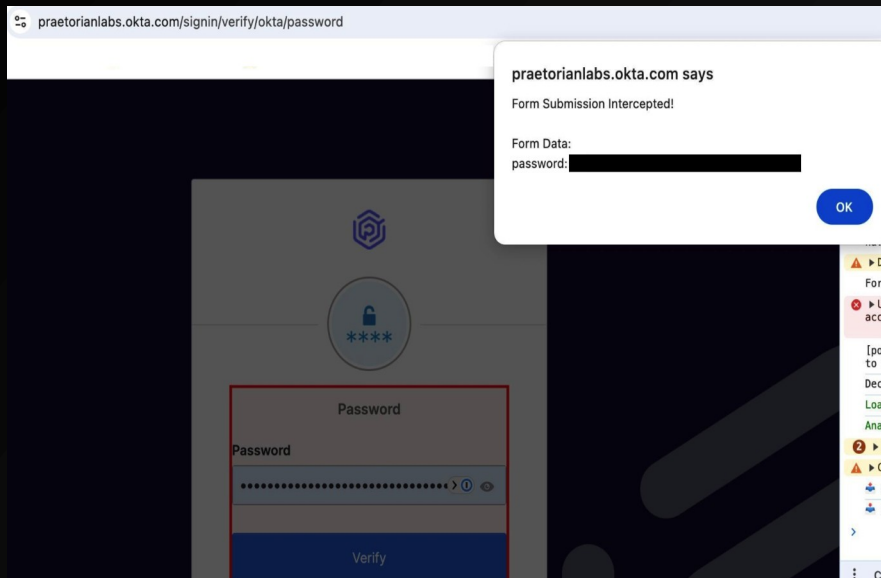


Capturing Credentials

- Javascript is super flexible
- We can just add extra listeners to anything
 - `targetForm.addEventListener("submit", credStealerFunction);`
- If you want to fully replace functions, you can do that too
 - `navigator.credentials.get = ourWebAuthnInterceptorFunction`

Capturing Credentials

- Every form submission gets relayed back to our command & control
 - No red highlighting or alerts in the actual deployed extension



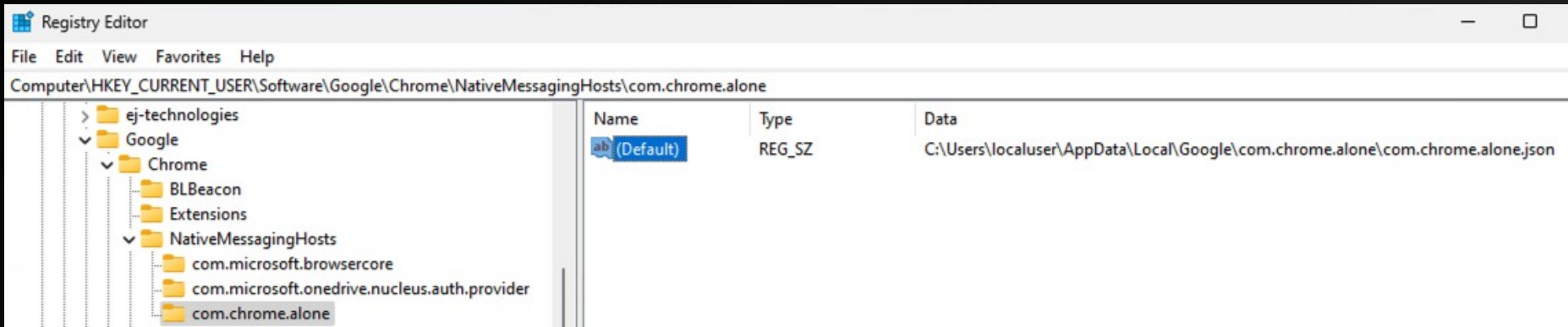
Shelling Out

- This talk is about only using Chrome
 - But sometimes we need help
- NativeMessaging to the Rescue!
- A mechanism to run other programs



Native Messaging

- Requires writing registry keys and files to disk
 - Since we've already sideloaded we get this for free
 - DOORKNOB has this built in if you want to use it



Native Messaging

- The configuration JSON is fairly static
 - Allowed_origins, name, and path are all that matter
- All binaries are required to process stdio for passing messages

```
{  
  "name": "com.my_company.my_application",  
  "description": "My Application",  
  "path": "C:\\\\Program Files\\My Application\\chrome_native_messaging_host.exe",  
  "type": "stdio",  
  "allowed_origins": ["chrome-extension://knldjmfmpopnolahpmmgbagdohdnhkik/"]  
}
```

Native Messaging

- Not exactly subtle

chrome.exe	14600	63.18 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	15856	6.43 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	9004	19.05 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	9620	18.36 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	8512	8.9 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	5776	25.46 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	2140	69.08 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	16060	60.5 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	8028	93 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	14152	26.13 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	7716	86.73 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	8692	7.14 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	3988	15.35 MB	GOADF9725...\localuser	Google Chrome
chrome.exe	9976	12.74 MB	GOADF9725...\localuser	Google Chrome
cmd.exe	15036	1.86 MB	GOADF9725...\localuser	Windows Command Processor
conhost.exe	8	5.67 MB	GOADF9725...\localuser	Console Window Host
NativeAppHost.exe	7132	4.53 MB	GOADF9725...\localuser	NativeAppHost

Native Messaging

- Not exactly subtle

- `C:\Windows\system32\cmd.exe /d /s /c "c:\path\to\native_host.exe" chrome-extension://ihdbjhckmijbmdbfnebmioahfogge/ --parent-window=0" < \\. \pipe\chrome.nativeMessaging.in.b4b2575ecc3200cd > \\. \pipe\chrome.nativeMessaging.out.b4b2575ecc3200cd`

- ALL NativeMessaging looks like this, so EDR can't ban chrome.exe spawning cmd.exe

- Binary needs to be written to process Chrome specific messages...

- It DOES load the binary though, so you CAN get code execution through DLL sideloading

The Plan

- ~~Sideload our Extension + Isolated Web App~~
- ~~Capture Okta SSO Credentials at Login + Session-ride~~
- **Tunnel Traffic through User via SOCKS**
- Trick User into Authenticating to Vault
- RDP into Target via SOCKS
- WIN!

Tunneling Traffic

- We could just use CursedChrome...
- But we have all these DANGEROUS IWA powers



Direct Sockets

- The primary IWA feature that we will abuse

Use cases

The initial motivating use case is to support creating a web app that talks to servers and devices that have their own protocols incompatible with what's available on the web. The web app should be able to talk to a legacy system, without requiring users to change or replace that system.

- [Secure Shell](#)
- [Remote Desktop Protocol](#)
- [printer protocols](#)
- Mail
- [IRC](#)
- [IOT](#) smart devices
- [Distributed Hash Tables for P2P systems](#)
- [Resilient collaboration using IPFS](#)
- [Virtual Desktop Infrastructure \(VDI\)](#)

Direct Sockets

- Slightly weird usage restrictions hidden in the specification
 - Not sure how this number got picked...isn't $\frac{1}{2}$ of 65536 32768?

6. Perform the following steps in parallel.

1. If the requested *localPort* is less than 32678, queue a global task on the relevant global object of this using the TCPServerSocket task source to run the following steps:
 1. Reject the [[openedPromise]] with a "NotAllowedError" DOMException.
 2. Reject the [[closedPromise]] with a "NotAllowedError" DOMException.

NOTE

While this condition could technically be checked at construction time, it's deliberately addressed here to allow for potentially loosening it with additional user-facing permissions.

Direct Sockets

THE I WANTED 10 YEARS AGO



THE I WANT TODAY



Direct Sockets

- Isolated Web Apps are supposed to remain isolated from the browser
 - But you can host a TCPSocketServer which the browser can connect to
- Implement a Websocket Server using Direct Sockets for internal comms
 - Now we have the power of chrome extensions + IWAs working together
- Then implement a SOCKS server using Direct Sockets

How to draw an owl

1.



1. Draw some circles

2.



2. Draw the rest of the owl

BLOWTORCH

- Provides a full SOCKS5 proxy to the attacker from the victim's machine
 - All comms originate from chrome.exe which is typically whitelisted
 - Combine with cookie stealing from HOTWHEELS to session ride with network position
- Yes, RDP works

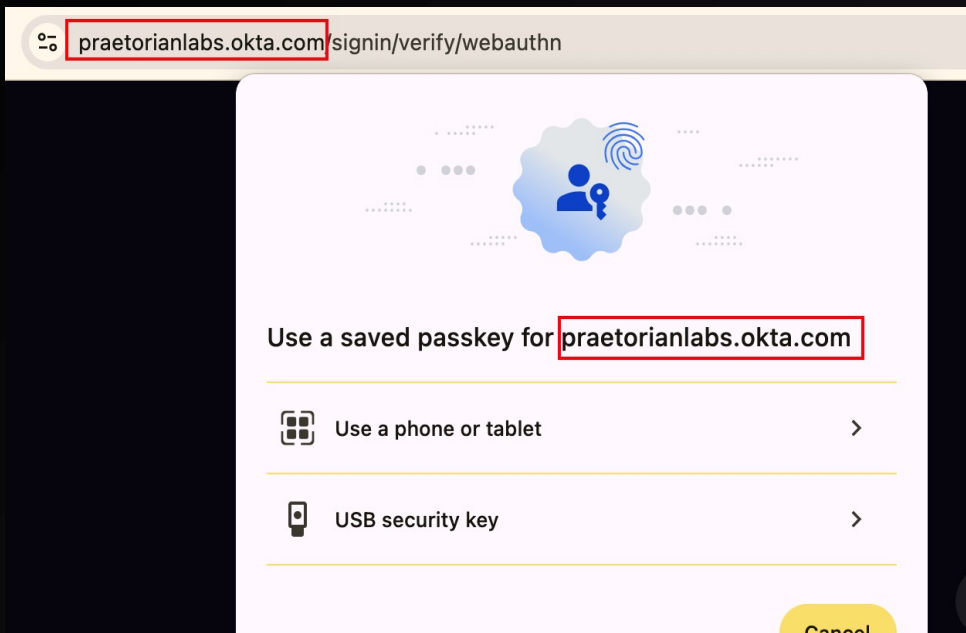


The Plan

- ~~Sideload our Extension + Isolated Web App~~
- ~~Capture Okta SSO Credentials at Login + Session-ride~~
- ~~Tunnel Traffic through User via SOCKS~~
- **Trick User into Authenticating to Vault**
- RDP into Target via SOCKS
- WIN!

WebAuthn

- Passwordless authentication using keys stored on external devices
 - Often used as an additional authentication factor
 - Phishing Resistant



Unrestricted WebUSB Permissions

- Isolated Web Applications can access restricted devices
 - Brings back an old attack vector from 2018

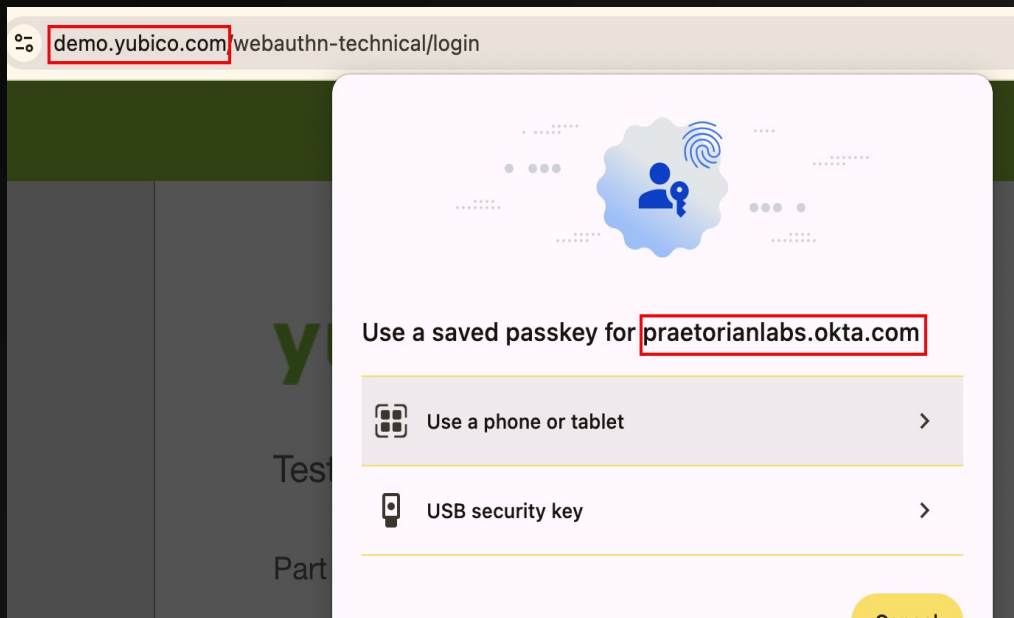


Unrestricted WebUSB Permissions

- Couldn't get WebUSB to interact with YubiKeys, Keyboard, or Mouse
 - Could be partial / buggy implementation
 - Could be a skill issue
- Isolated Web Apps can't do this...maybe extensions can?

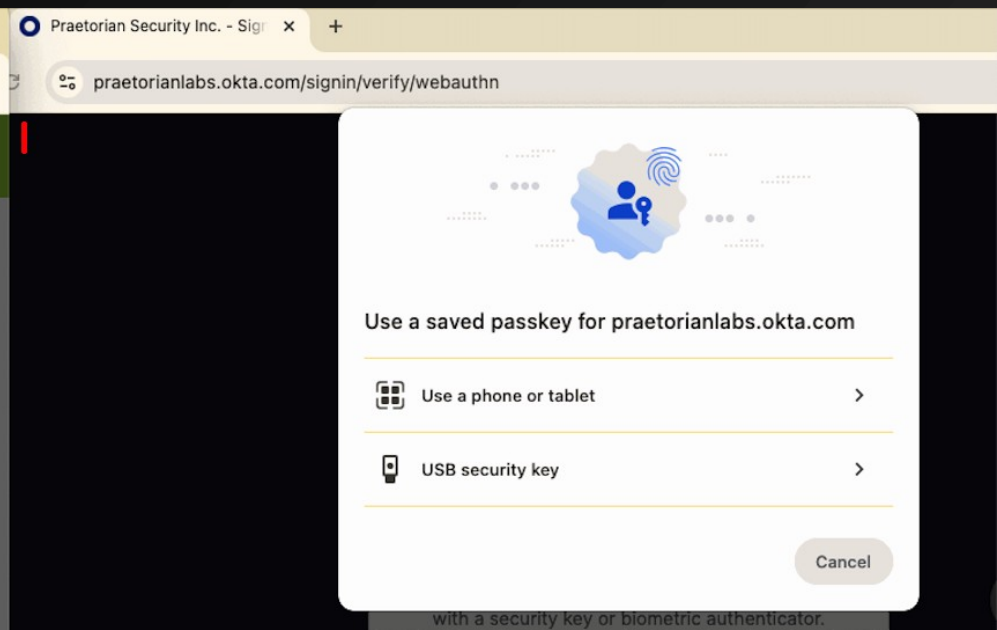
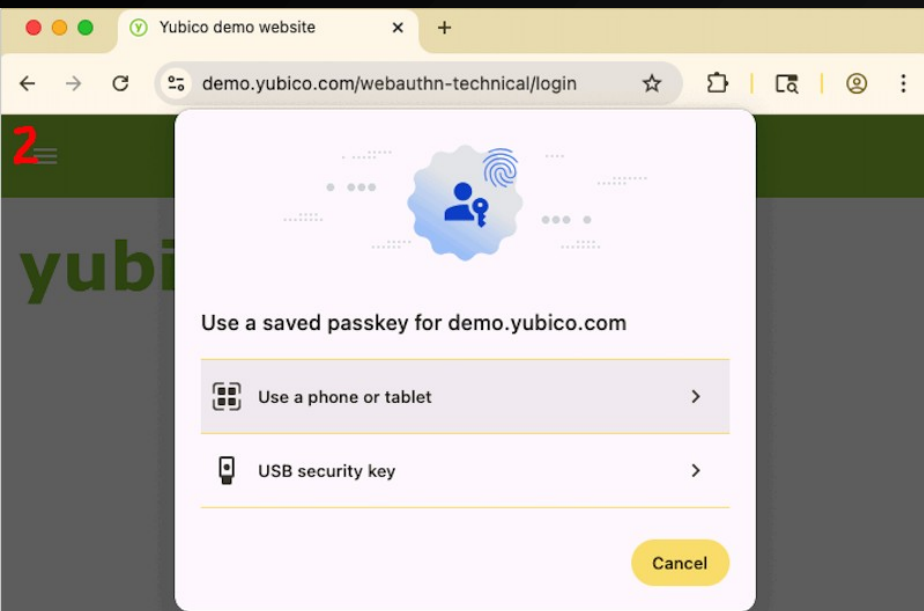
WebAuthn

- Can we break this with Chrome Extensions?
 - First open an iFrame for the target site on page
 - Then have our injected content script trigger the desired WebAuthn request



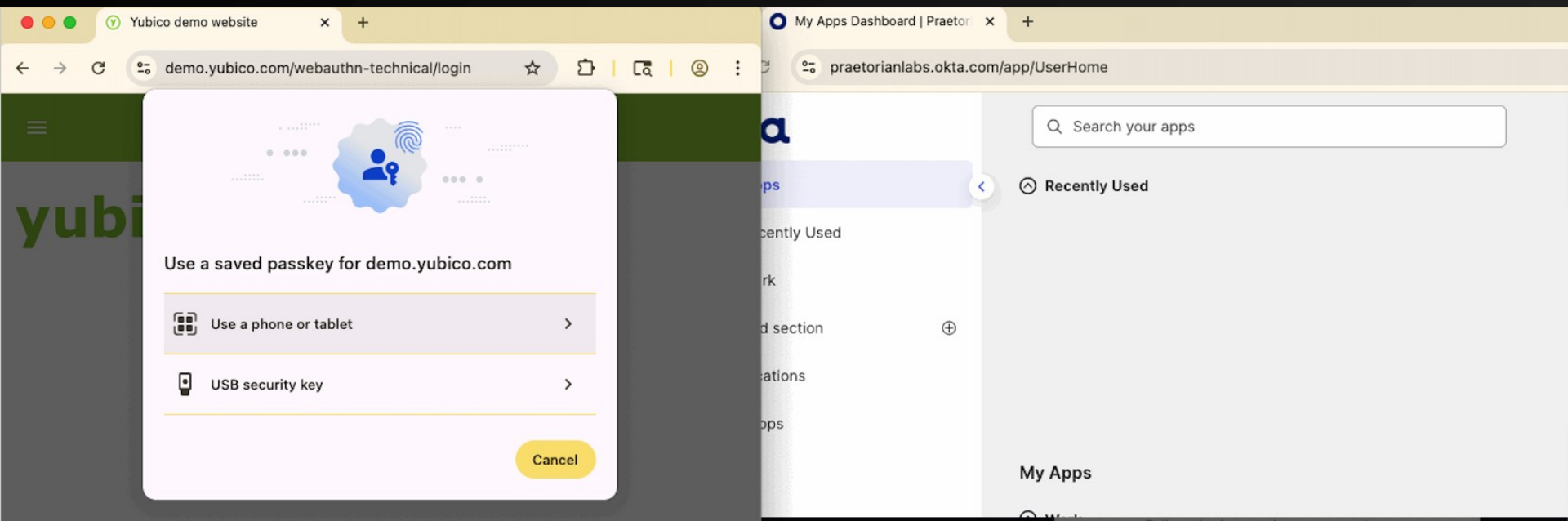
WebAuthn

- What happens if you trigger a WebAuthn request in one tab then change tabs and do it again?
 - When you press your Yubikey, which tab's challenge is resolved first?



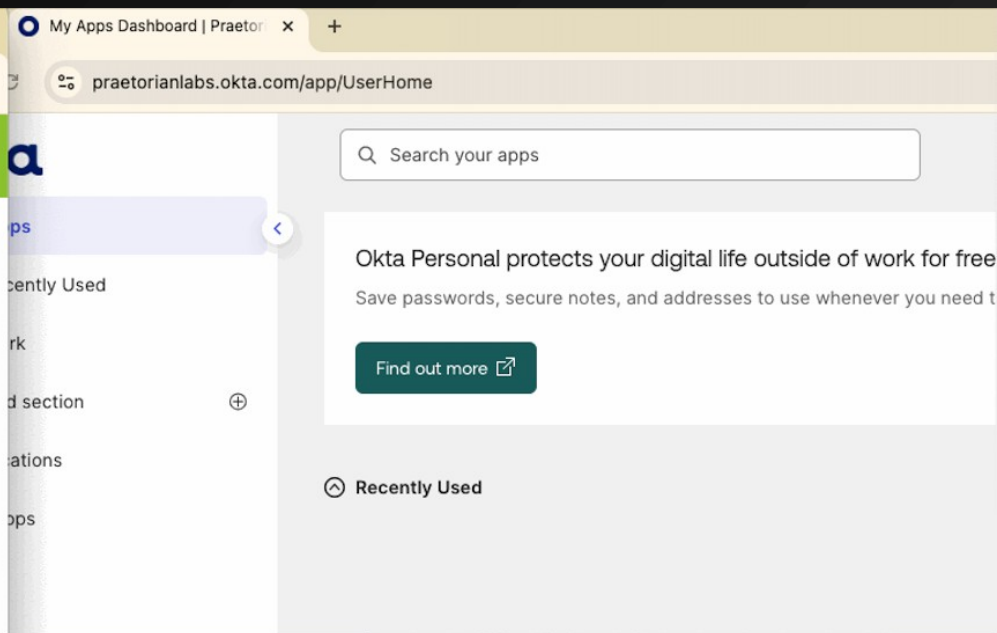
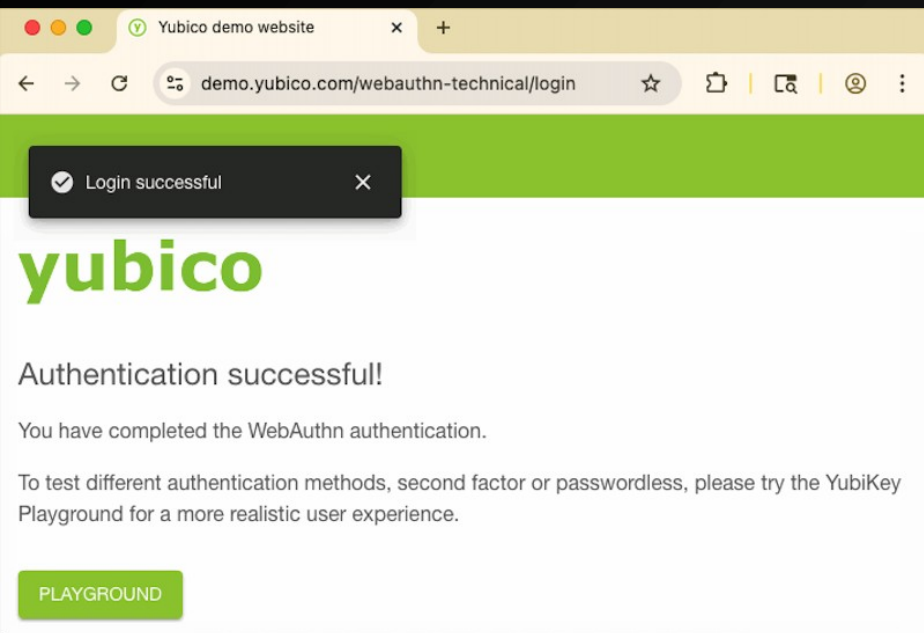
WebAuthn

- What happens if you trigger a WebAuthn request in one tab then change tabs and do it again?
 - The first request takes precedence over the active window/tab



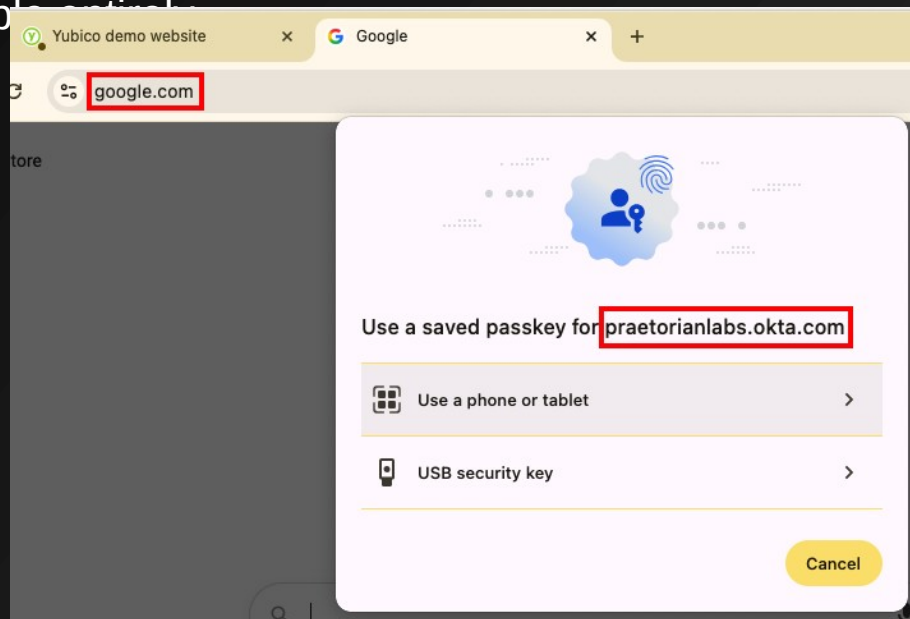
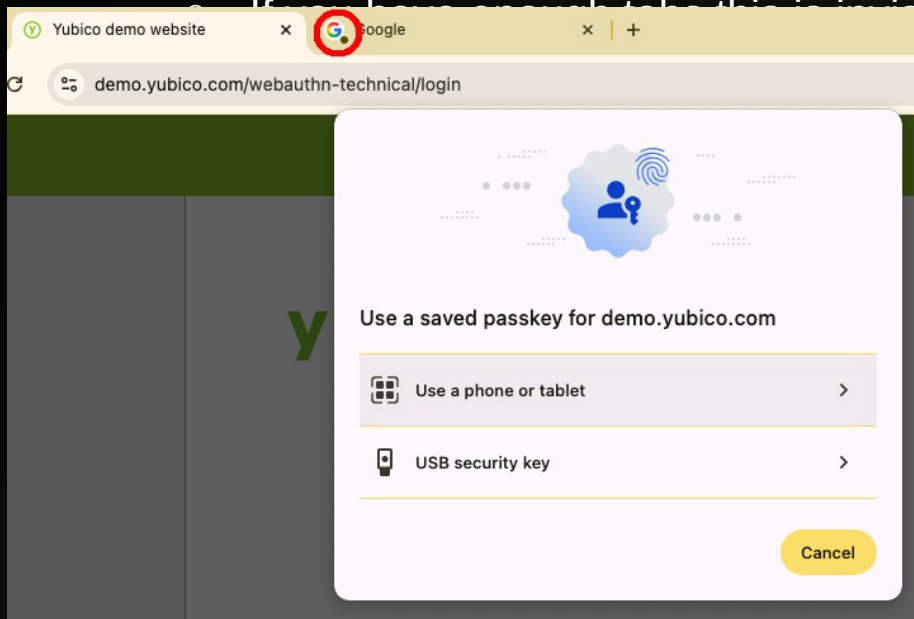
WebAuthn

- What happens if you trigger a WebAuthn request in one tab then change tabs and do it again?
 - But it will resolve if we press the Yubikey again



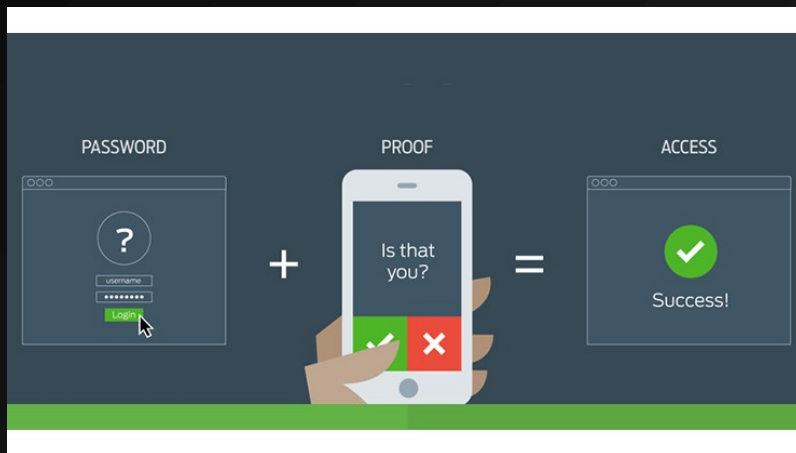
WebAuthn

- Now hook `navigator.credentials.get()`
 - Every time you try to auth on a tab, we secretly open an iFrame in a background tab
 - Looks like the first Yubikey touch didn't register



WebAuthn

- Automate capture of a WebAuthn challenge for our target every N minutes
 - Push it to the client, when they eventually use WebAuthn they'll get hit
- Turn WebAuthn targets into Push Notification Spam
 - If you're feeling aggressive, just constantly pop up WebAuthn requests until they press it



PAINTBUCKET

- Secretly coerce your target into answering WebAuthn challenges
 - Just provide a captured challenge to the target and wait



The Plan

- ~~Sideload our Extension + Isolated Web App~~
- ~~Capture Okta SSO Credentials at Login + Session-ride~~
- ~~Tunnel Traffic through User via SOCKS~~
- ~~Trick User into Authenticating to Vault~~
- **RDP into Target via SOCKS**
- **WIN!**

DEMO

Limitations

- Edge and Chrome have different implementation details
 - No Unrestricted WebUSB in Edge, for example
 - But it DOES load IWAs...it got stuck in an infinite loop until recently
- Isolated Web Applications are constantly changing
 - They might restrict installations to apps signed by Google by default instead of anyone
 - Some of the permissions don't have working demos
- DOORKNOB is Windows + Chrome only...for now

Defensive Recommendations

- Enterprise Browser Policy stops (some of) this
 - ExtensionInstallBlocklist = *
 - <https://chromeenterprise.google/policies/#ExtensionInstallBlocklist>
 - ExtensionInstallAllowList = <what your org needs>
 - <https://chromeenterprise.google/policies/#ExtensionInstallAllowlist>
 - There is nothing to explicitly disable Isolated Web Apps right now
- Watch for any file modifications of Chrome data not by Chrome
 - Secure Preferences, Preferences, Local State, or LevelDB files
- Watch for NativeMessaging registry entry modifications
 - HKCU\HKLM\SOFTWARE\Google\Chrome\NativeMessagingHosts\com.my_company.my_application

Thank You!

Code is online at github.com/praetorian-inc/ChromeAlone

- If you have questions or success stories let me know via [@bouncyhat](https://twitter.com/bouncyhat)



References

- Offensive Browser Extension Development - <https://www.irongeek.com/i.php?page=videos/derbycon8/track-4-02-offensive-browser-extension-development-michael-weber>
- Browser Extension Supply Chain Attacks - <https://www.darktrace.com/blog/cyberhaven-supply-chain-attack-exploiting-browser-extensions>
- LummaC2 Sideloaded - <https://www.esentire.com/blog/lummac2-malware-and-malicious-chrome-extension-delivered-via-dll-sideloaded>
- BlinkOn 19 - <https://www.youtube.com/watch?v=Q3b5NB-7HQQ>
- CursedChrome - <https://github.com/mandatoryprogrammer/CursedChrome>
- RedExt - <https://github.com/Darkrain2009/RedExt>
- Browser Syncjacking - <https://labs.sqrx.com/browser-syncjacking-cc602ea0cbd0>
- Isolated Web Apps Explainer - <https://github.com/WICG/isolated-web-apps>
- LevelDB Writeup - <https://www.cclsolutionsgroup.com/post/hang-on-thats-not-sqlite-chrome-electron-and-leveldb>
- DirectSockets Explainer - <https://wicg.github.io/direct-sockets/>
- WebUSB Yubikey Phishing - <https://www.wired.com/story/chrome-yubikey-phishing-webusb/>
- Silent Chrome - https://github.com/asaurusrex/Silent_Chrome