CYCRAFT

Original Sin of SSO: macOS PRT Cookie Theft & Entra ID Persistence via Device Forgery

Shang-De ‘John’ Jiang

Kazma Ye

Echo Lee

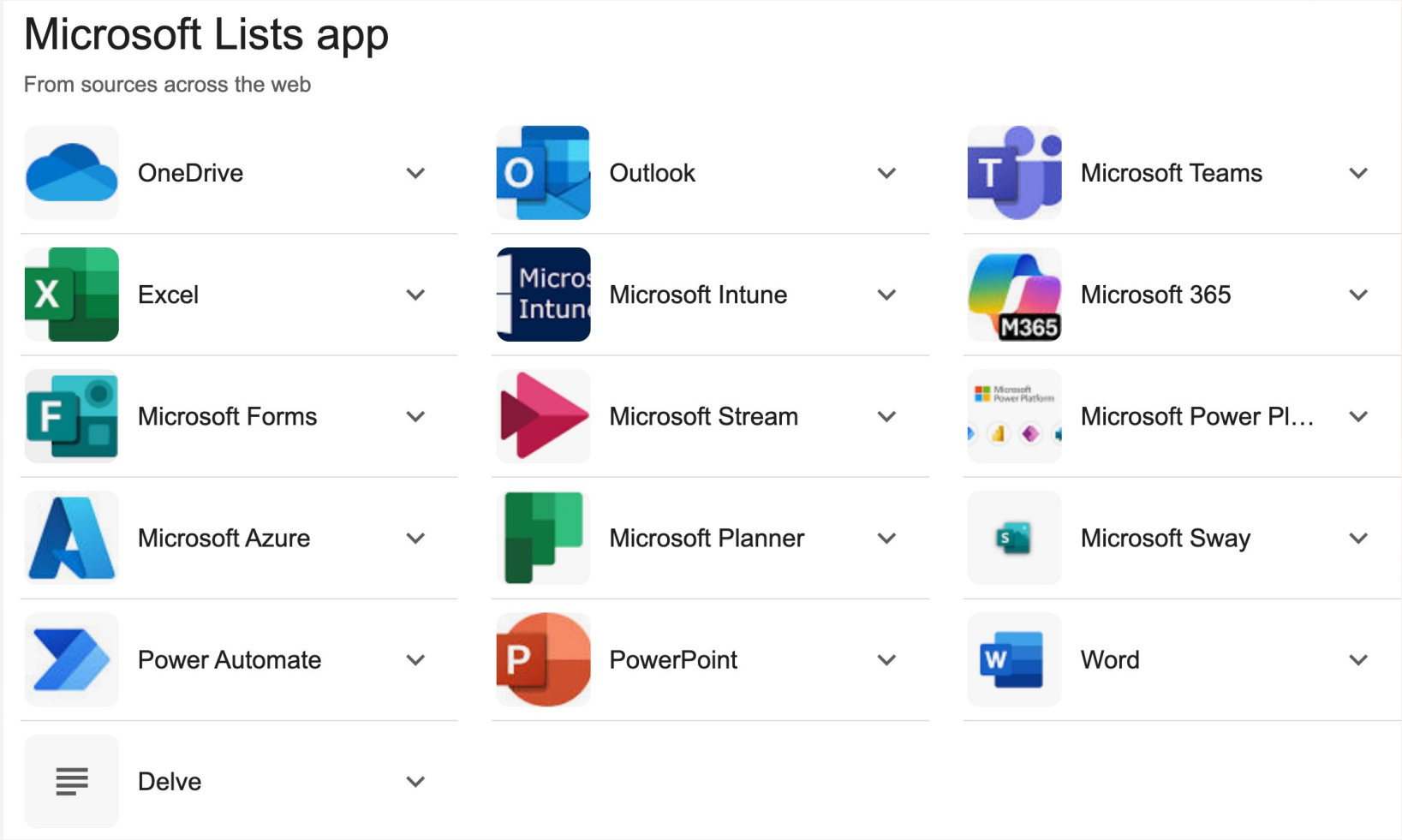
$ whoami



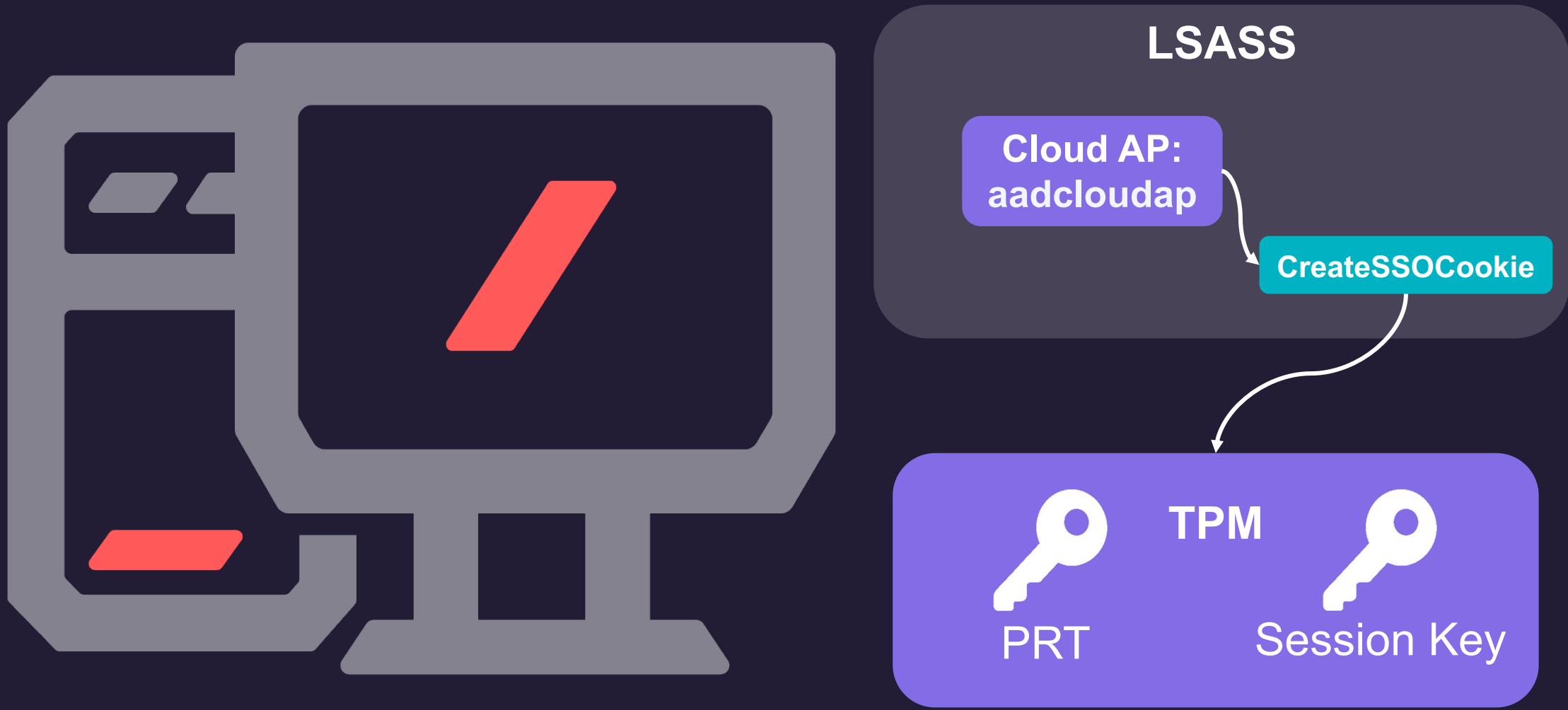
- Shang-De ‘John’ Jiang (@SecurityThunder)
- Deputy Director of Research at CYCRAFT
- UCCU Hacker Co-Founder
- Blog: HackerPeanutJohn
- Speaker at the following technical conferences: BlackHat USA, CodeBlue, HITCON , HITB, TROOPERS, Sans Blue Team Summit …



A lots of service! You need SSO

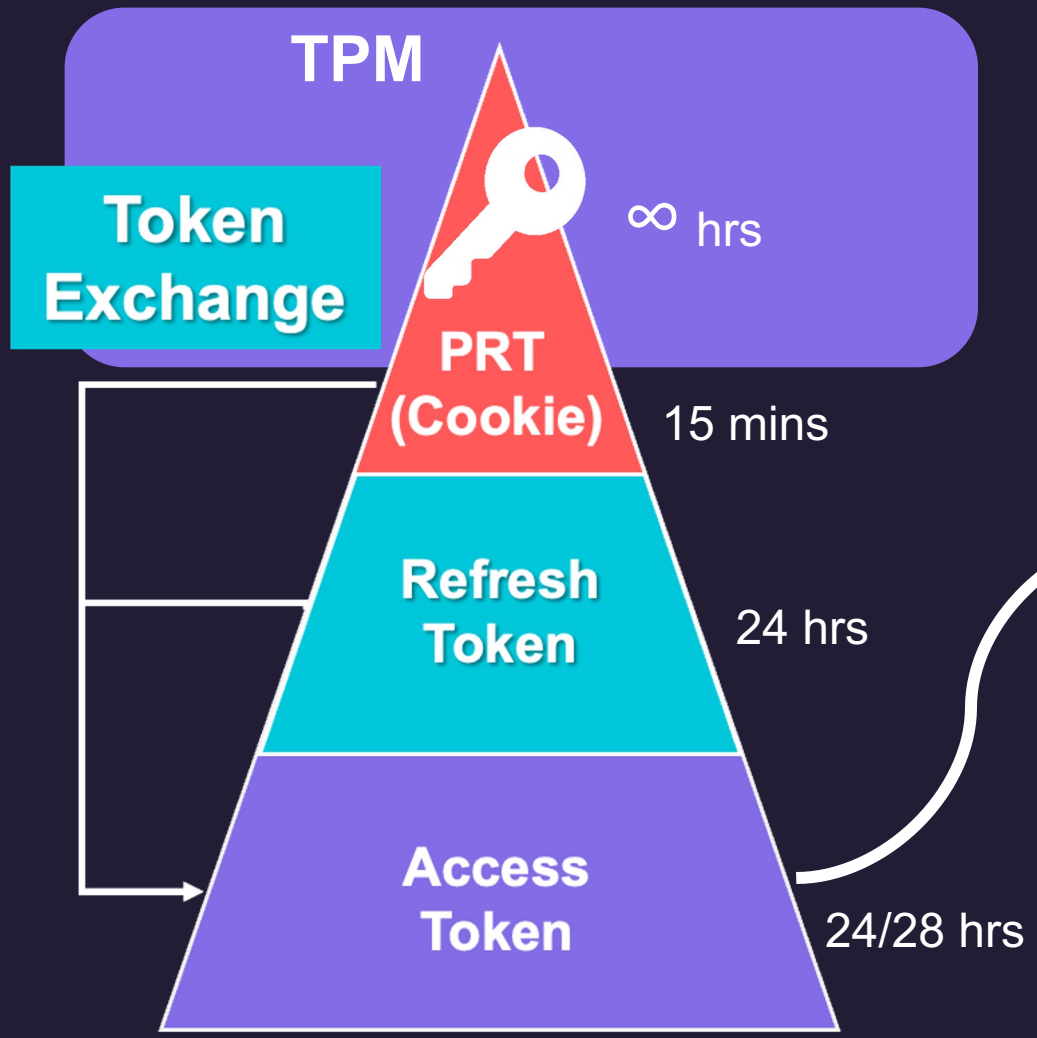


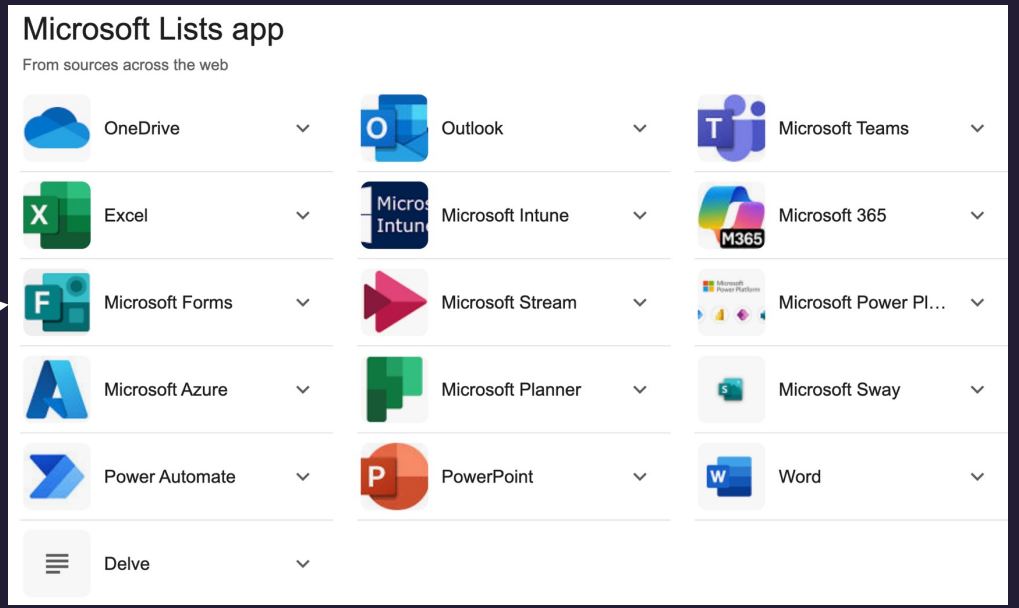
SSO token generate in Windows OS



CYCRAFT

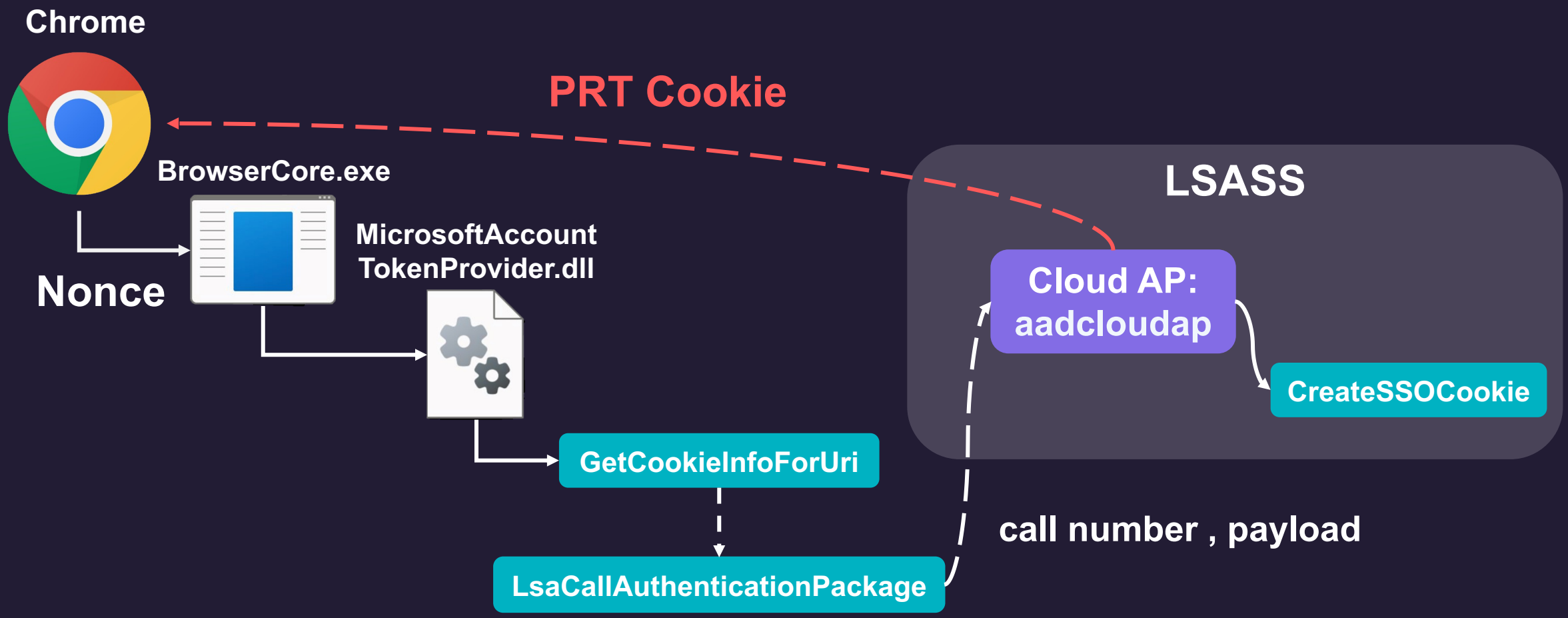
PRT Can Exchange Everything We Wanted





CYCRAFT

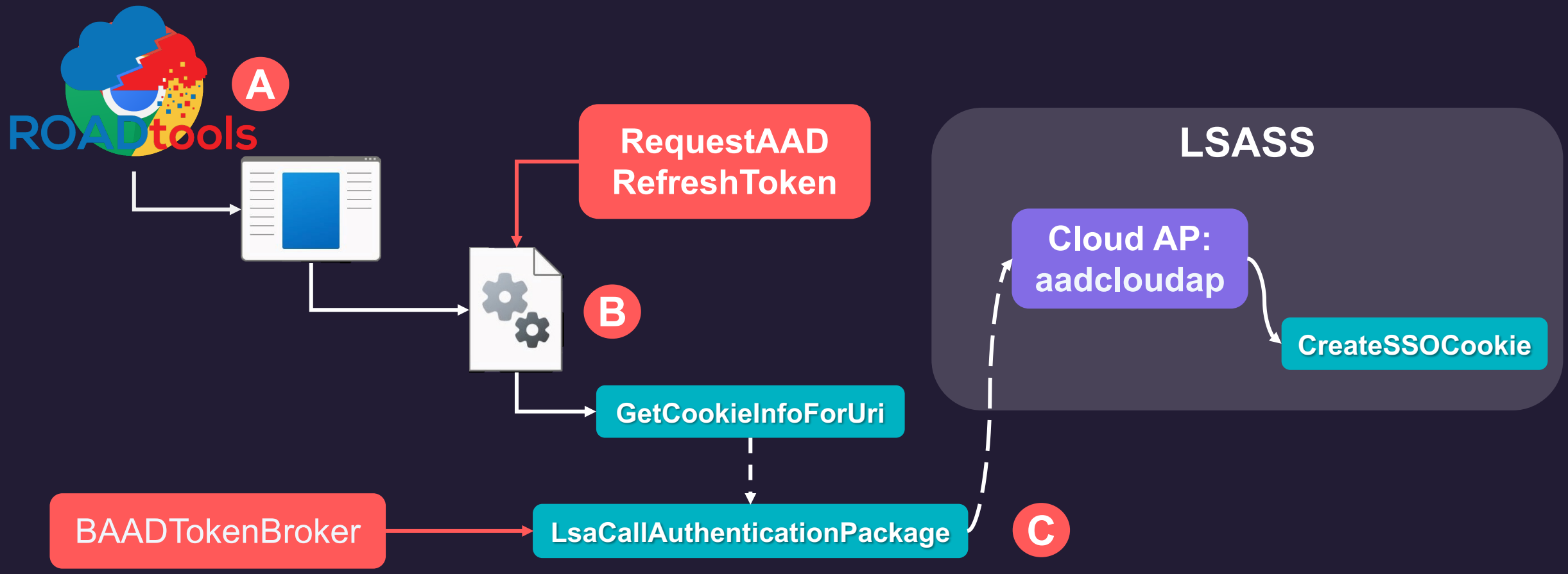
Browser SSO on Windows



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

CYCRAFT

Abuse Browser SSO on Windows

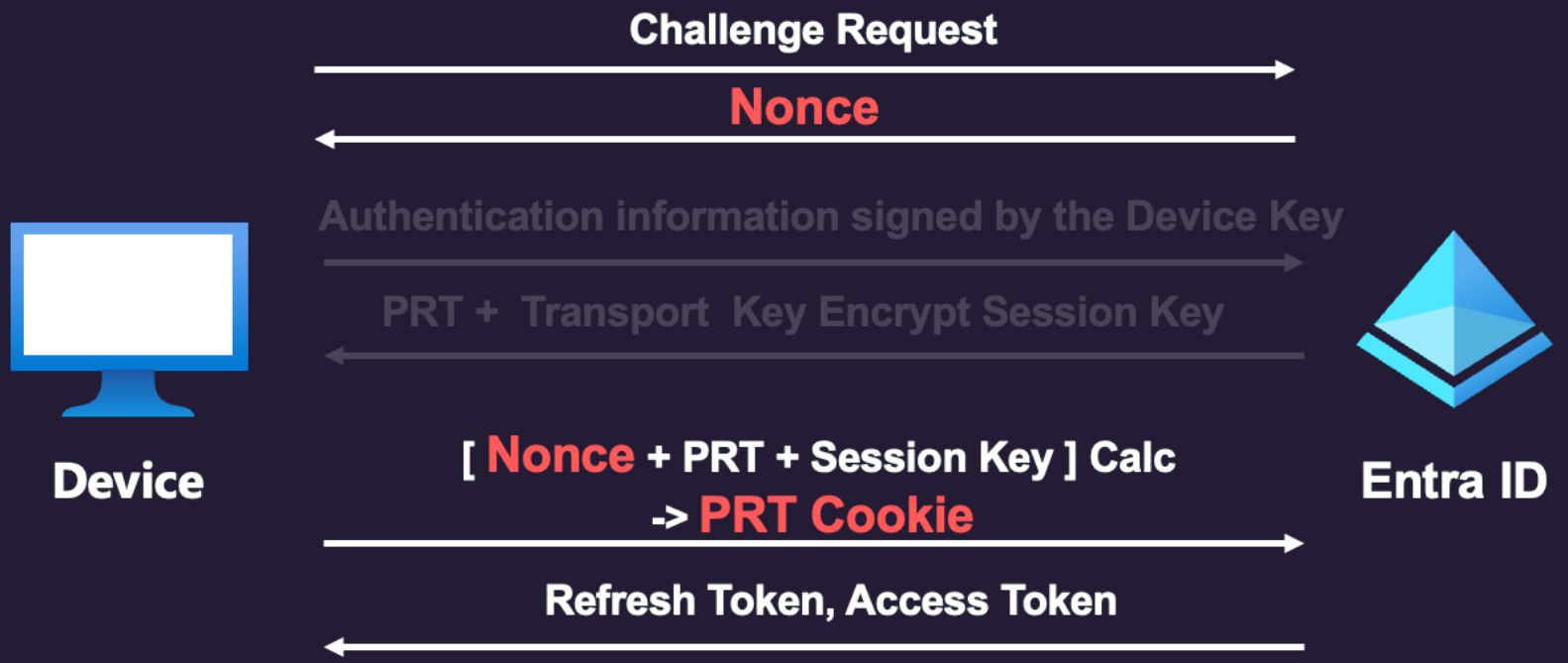


Ref: https://i.blackhat.com/Asia-24/Presentations/Asia-24-Chudo-Bypassing-Entra-ID-Conditional-Access-Like-APT.pdf

CYCRAFT

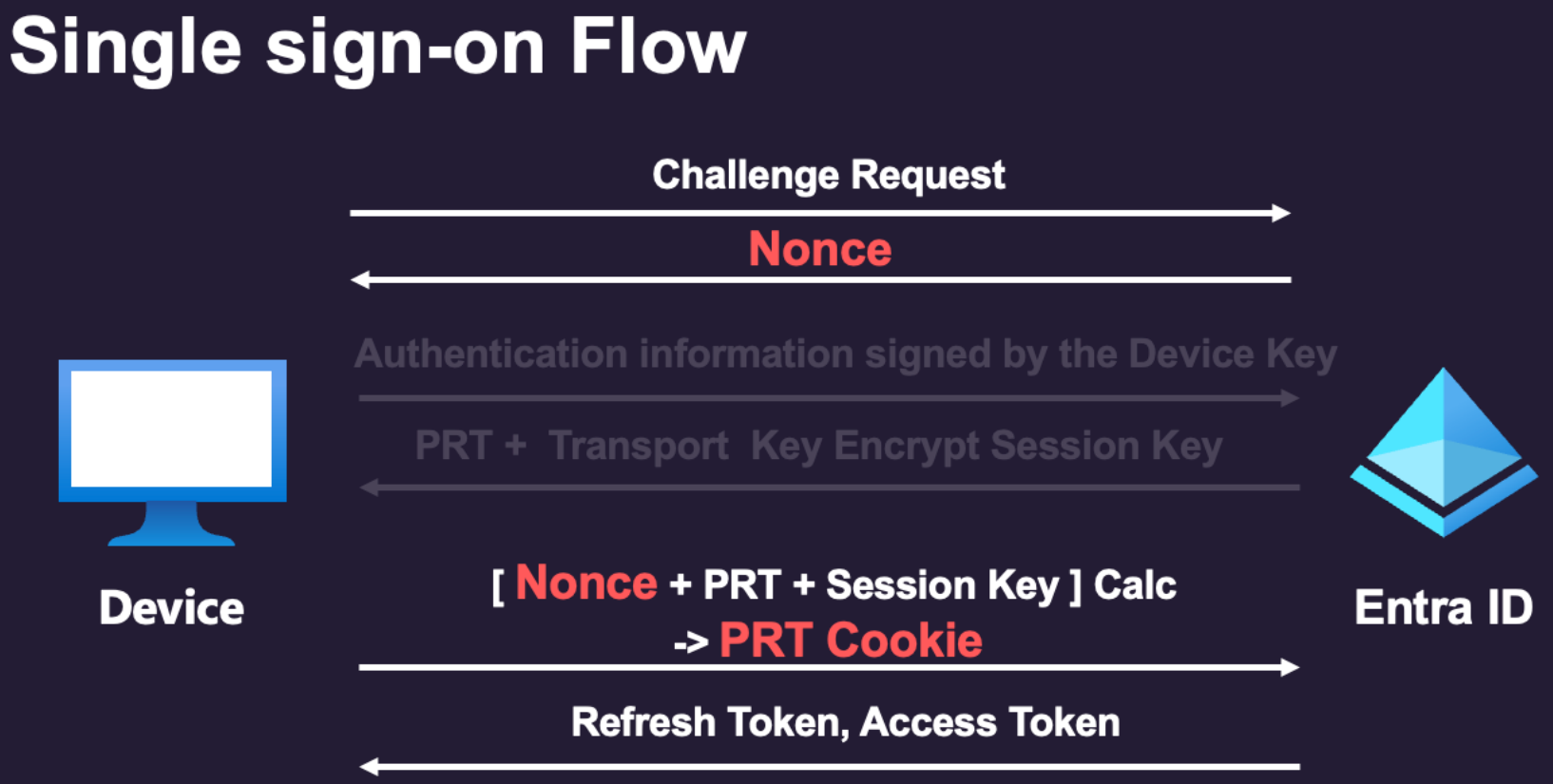
The PRT Cookie includes user identity + linked device information

Single sign-on Flow



CYCRAFT

PRT Cookie From Device Can Include MFA Claim + Device Claim



CYCRAFT

Today's Objectives:
From PRT Cookie to Persistence

- Steal PRT Cookie from macOS
- Establish Long-Term Persistence & Compliance Security Requirement

  Convert temporary access into a permanent foothold by forging a new device in Entra ID. Use the token's embedded MFA and Device claims to defeat most Conditional Access policies
- Goal: Access Critical Resources

  Impersonate the user to gain unrestricted access to data in Azure and Microsoft 365

PRT Cookie Theft
on macOS

$ whoami

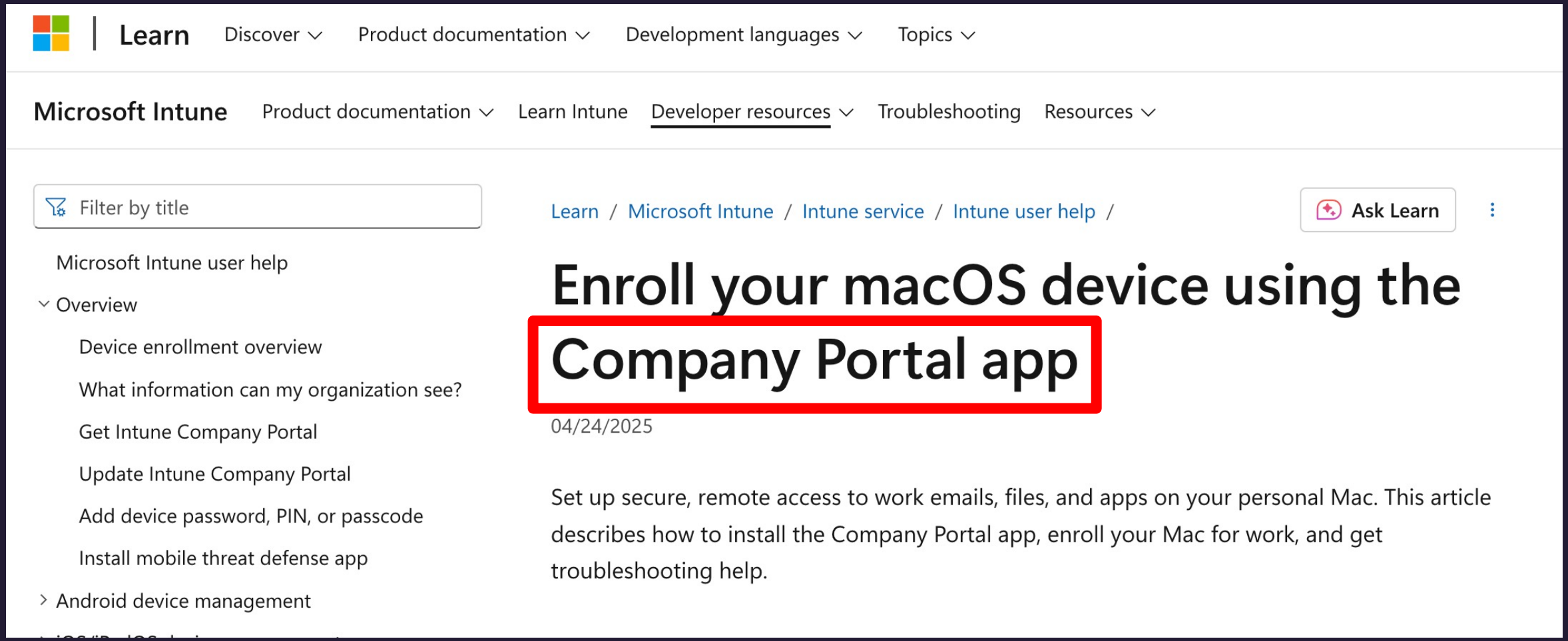


- 🥷 Kazma Ye
- University Student in Taiwan 🇹🇼
- Leader @ CTF Team B33F50UP
- Cybersecurity Researcher @ CYCRAFT
- Founder of Taiwan Security Club & NCKUCTF
- 1st Place AIS3 EOF | 3rd Place WorldSkills Cybersecurity
- 10th Place HITCON CTF (1st in Taiwan)
- Talks @ TROOPERS25, DEF CON 33, RomHack

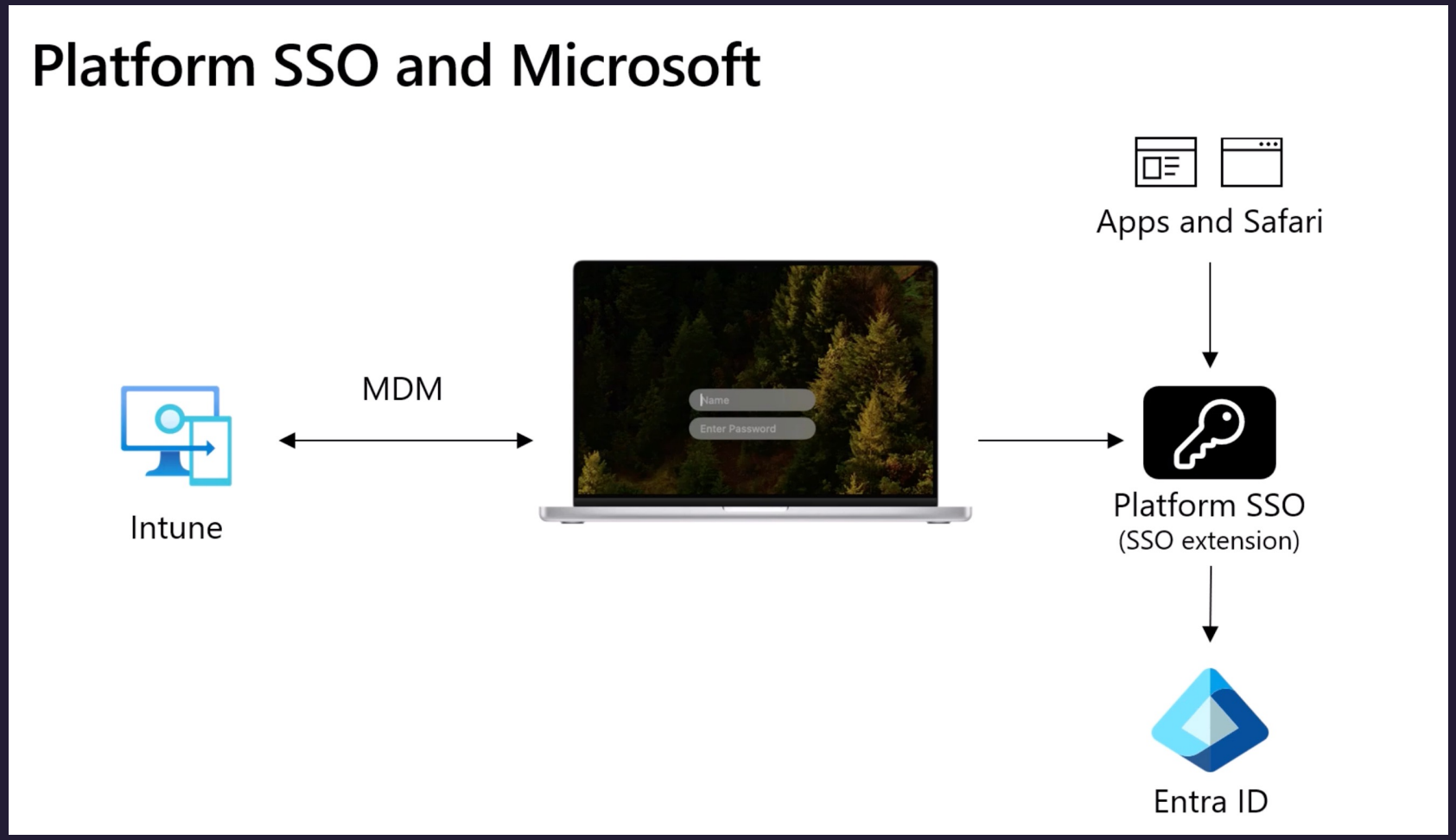
Why Steal macOS PRT Cookies?

- 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?

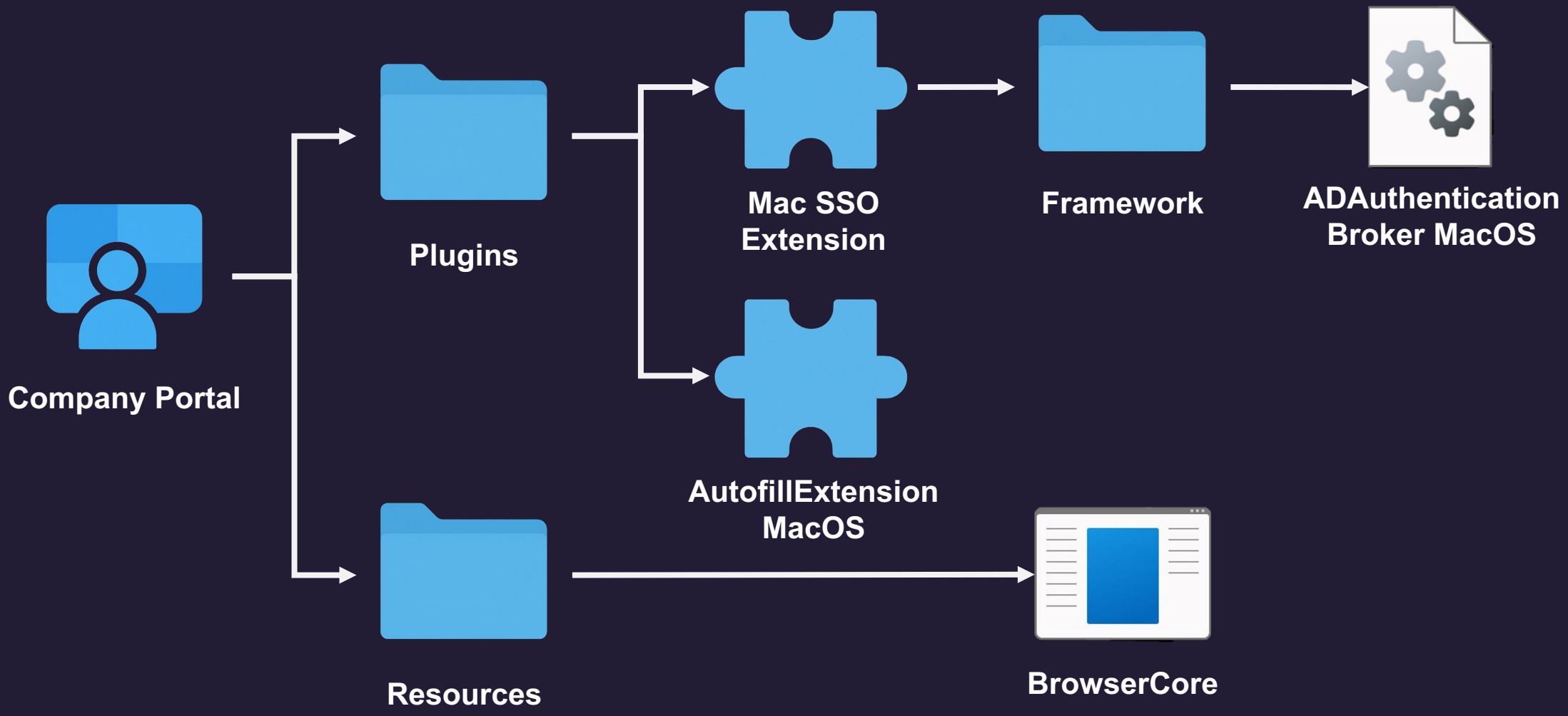


Company Portal on macOS

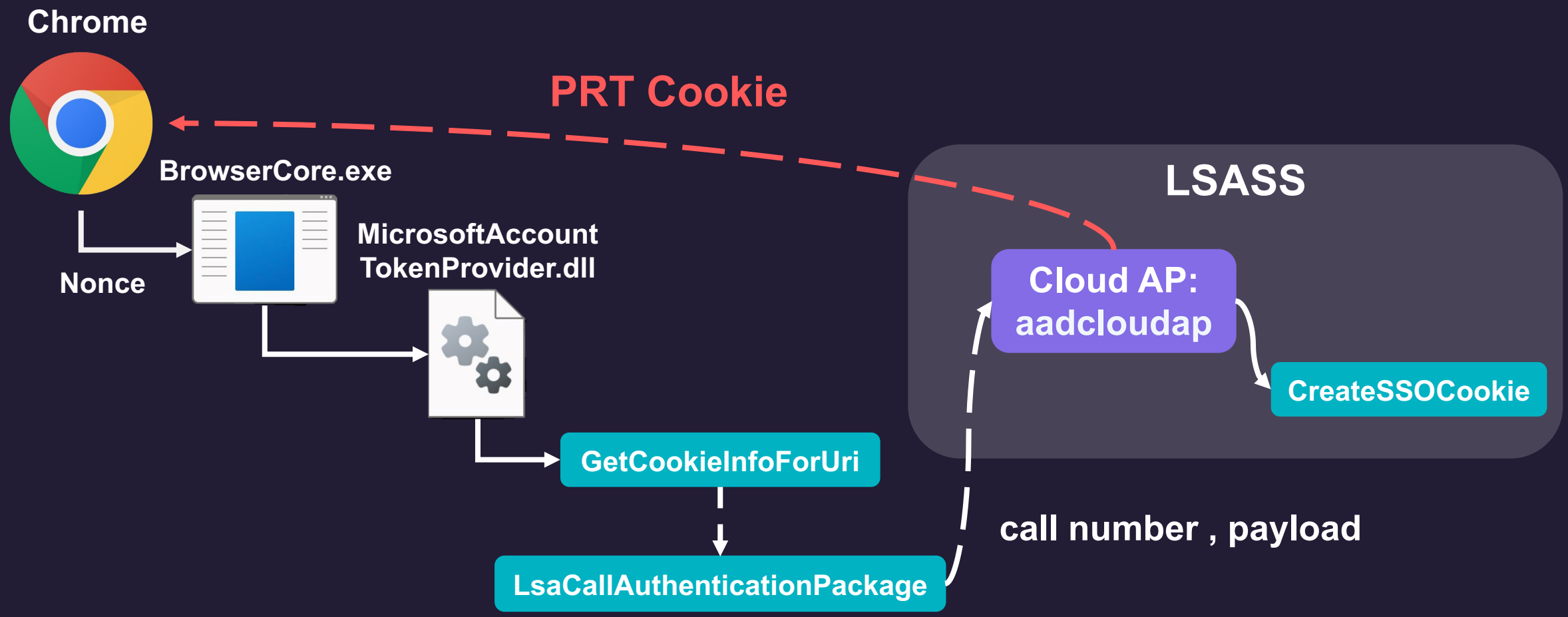


Ref: https://youtu.be/awckSIpCPMg?si=18uS-Ot0jNSeMpUs

Main Structure of Company Portal

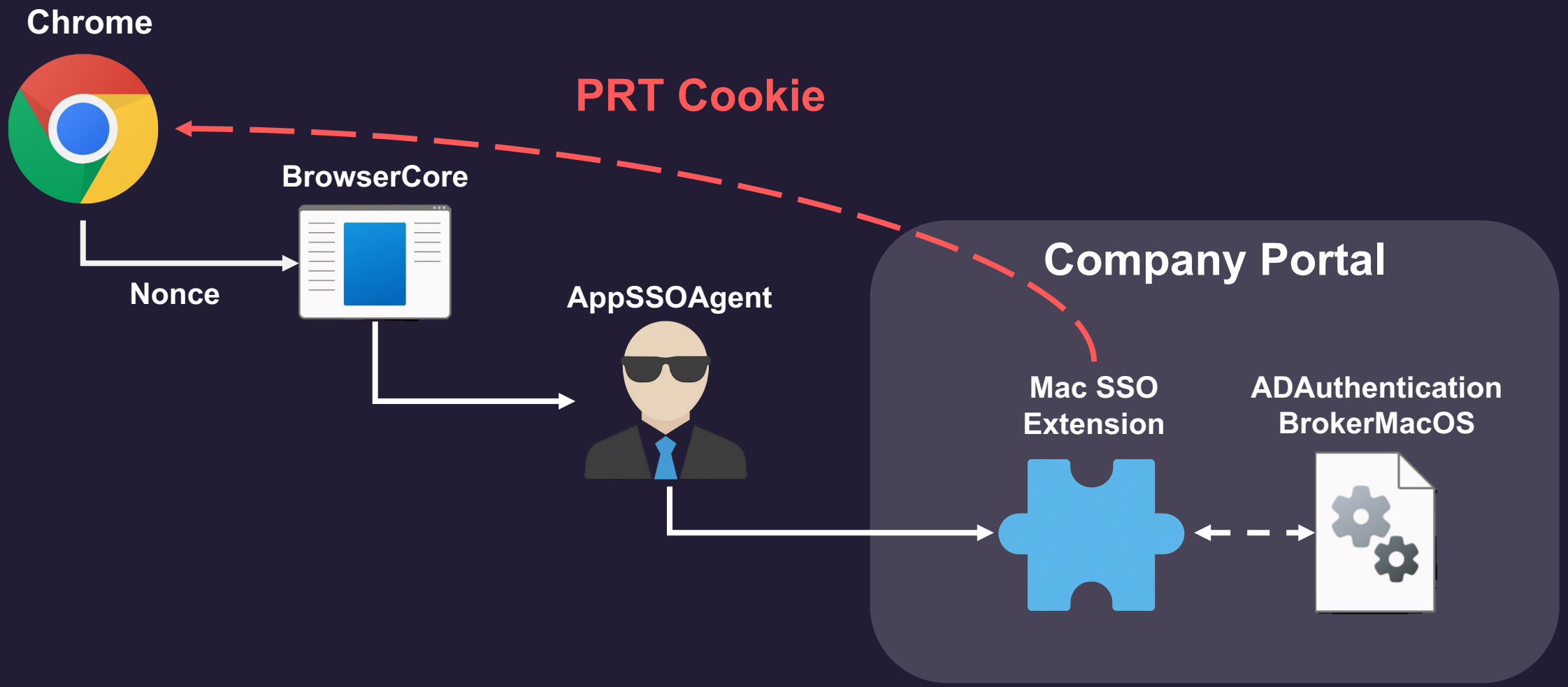


Browser SSO on Windows



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

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

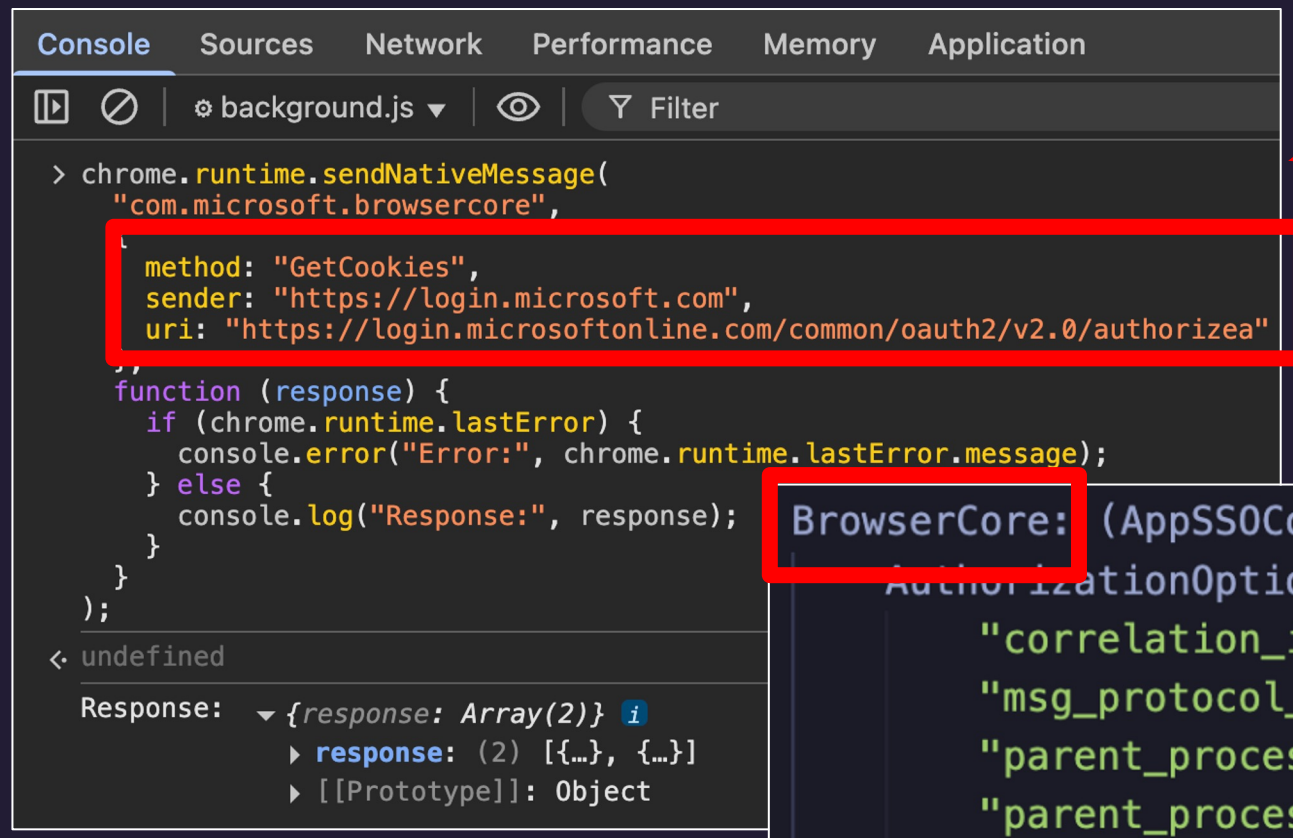


Headless Browser-Based Native Messaging Abuse



BrowserCore on macOS

Microsoft SSO Chrome extension





CYCRAFT

Headless Browser Method Condition

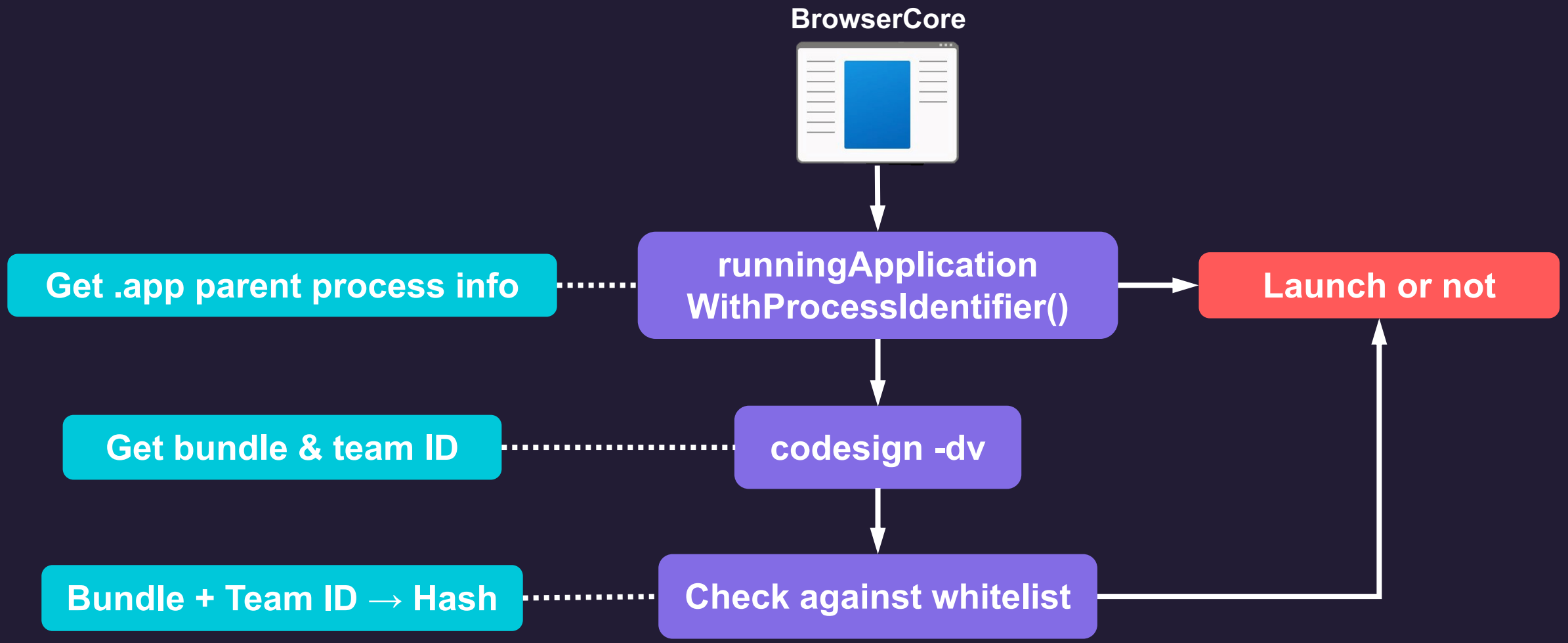
- Victim must be logged into desktop session
- Headless browser ≠ no GUI dependencies
- Only works on official Chrome, Edge

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



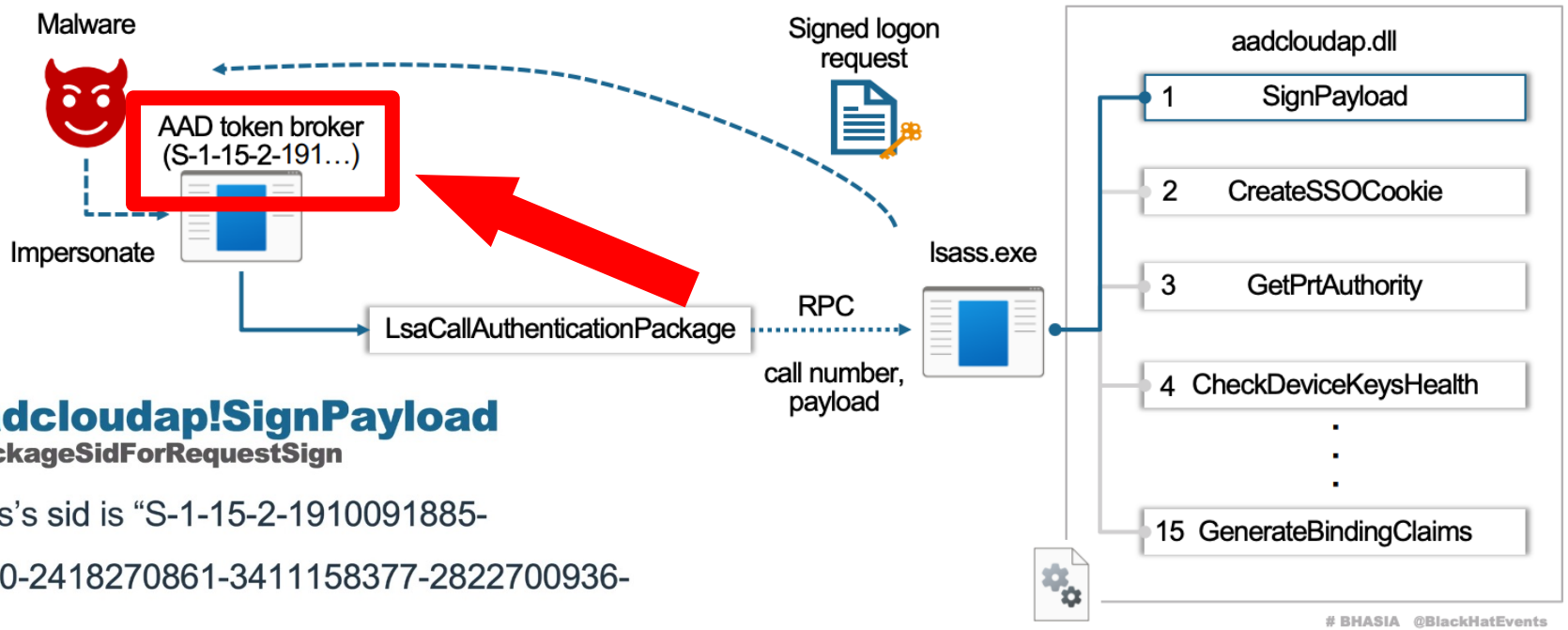
CYCRAFT

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

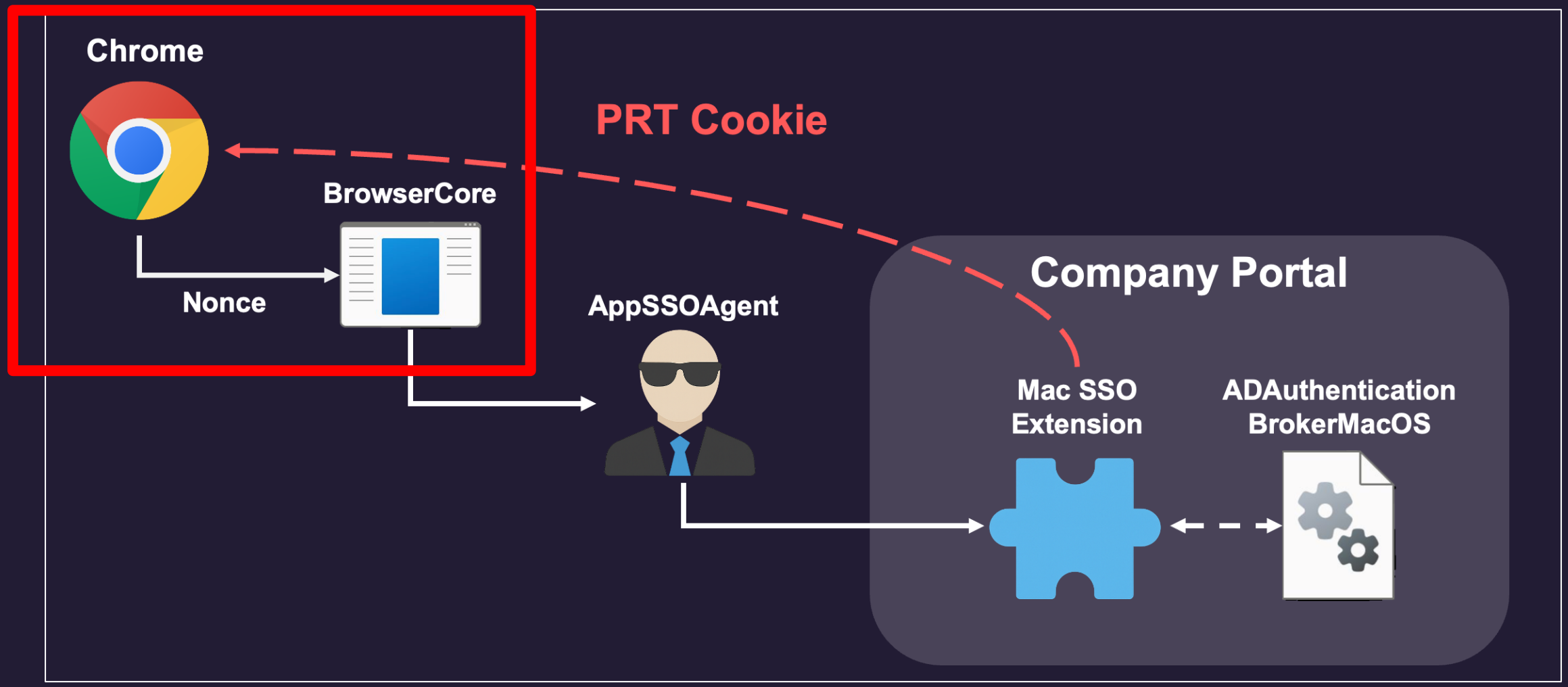
CheckPackageSidForRequestSign

- 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

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

CYCRAFT

Two Different Callers in SSO Flow

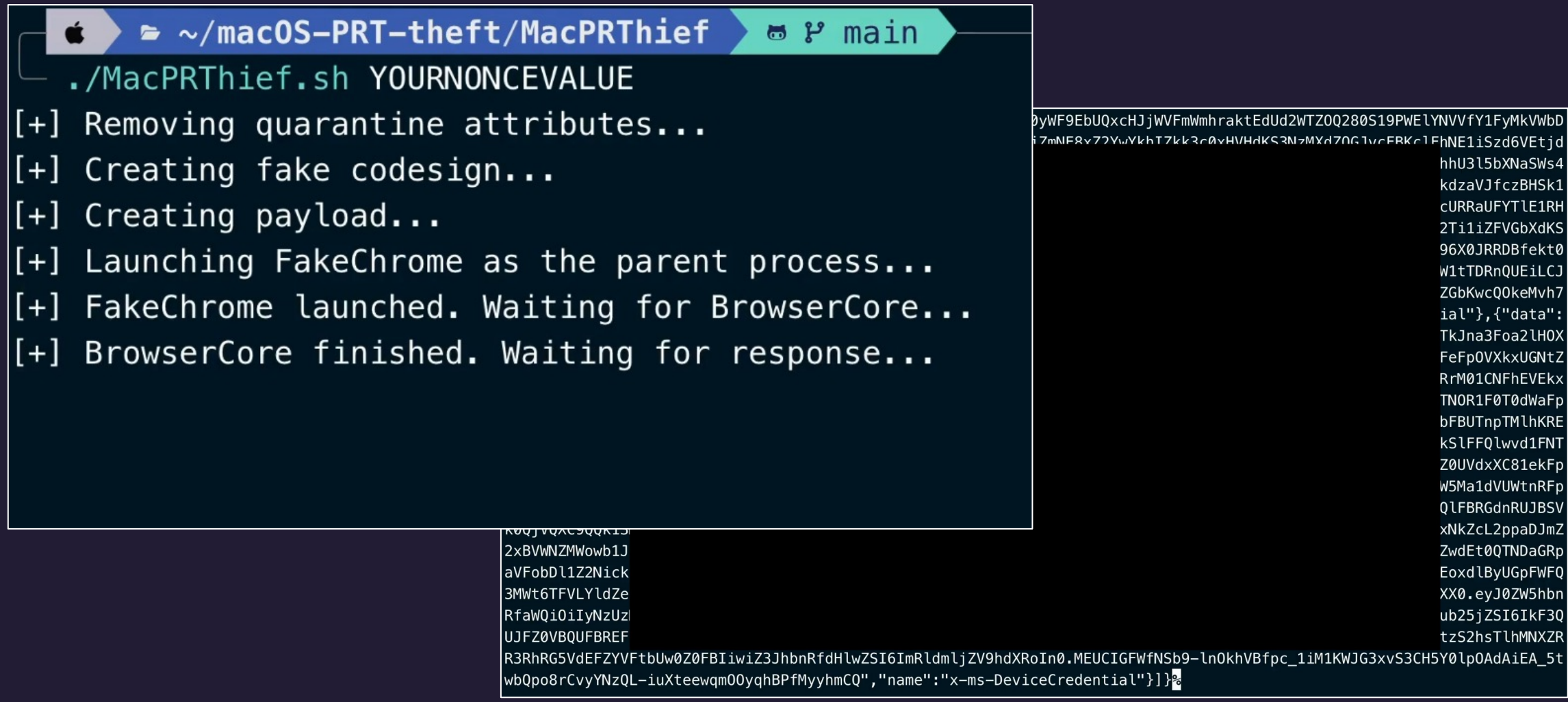


CYCRAFT

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 💥



CYCRAFT


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...
3TkJna3Foa2lHOXcwQkFRc0ZBREI0TVhZd0VRW
)9ESmtZbUZqWVRRdE0yVTRNUzAwTm1OaExUbGp
XRTNOR1F0T0dWaFpEUXhObUZpWm1VM01Ga3dFd
zg0T3FWc2J6NWRxUktPQjBEQ0J6VEFNQmdOVkh
JWTZ0UVdxXC81ekFpQmdzcWhraUc5eFFCQllJY
```


CYCRAFT

Log Diff

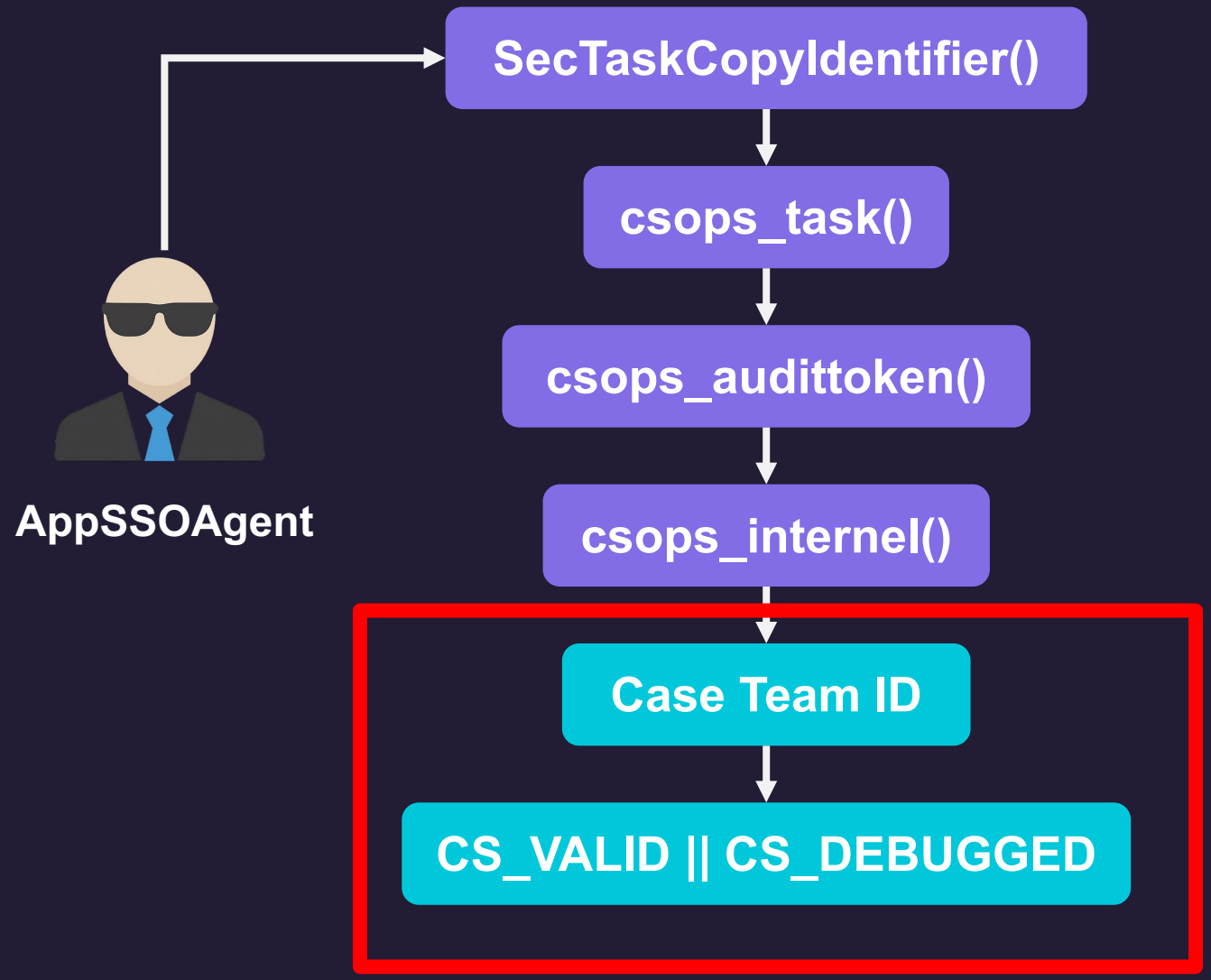
❌ Fail:

```
 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:

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

How AppSSOAgent Validates its Parent



User Space
darling-security/SecTask.c

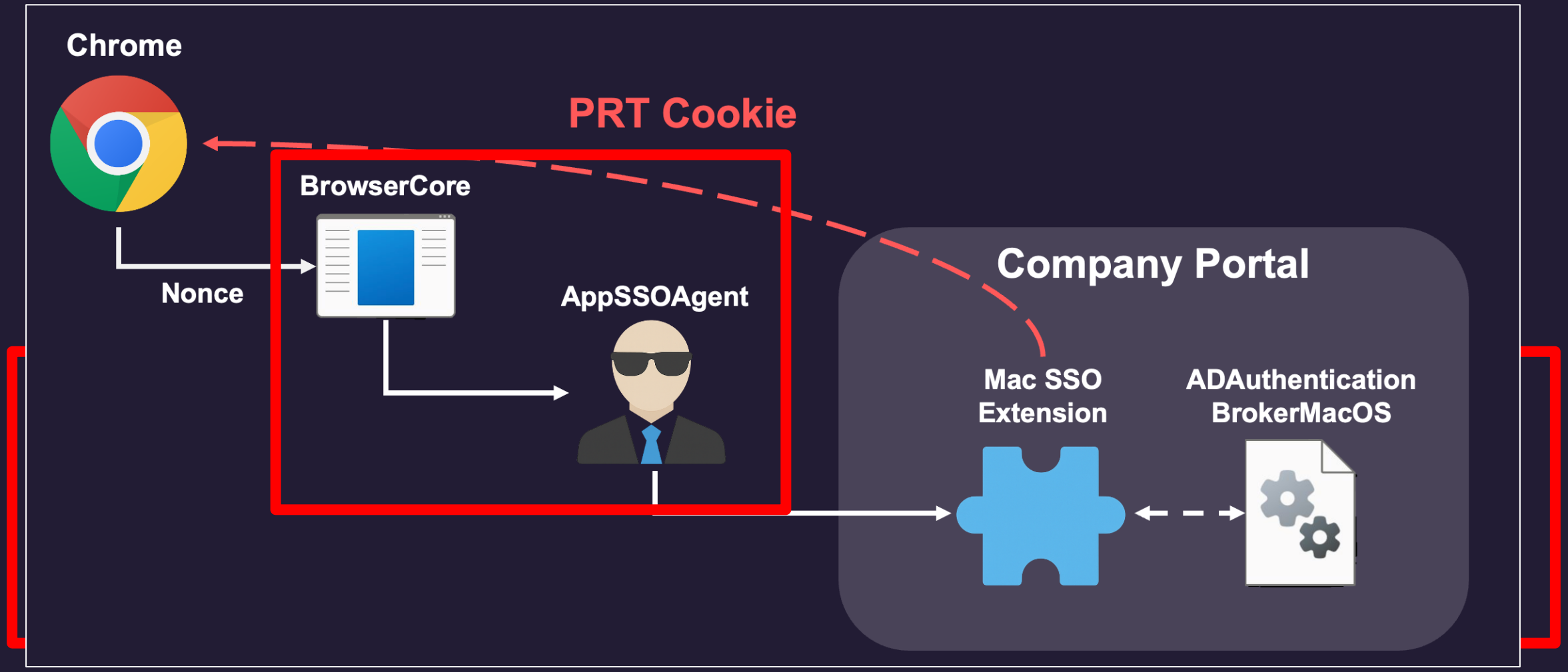
Kernel Space
darwin-xnu/kern_proc.c

Actually...☹️



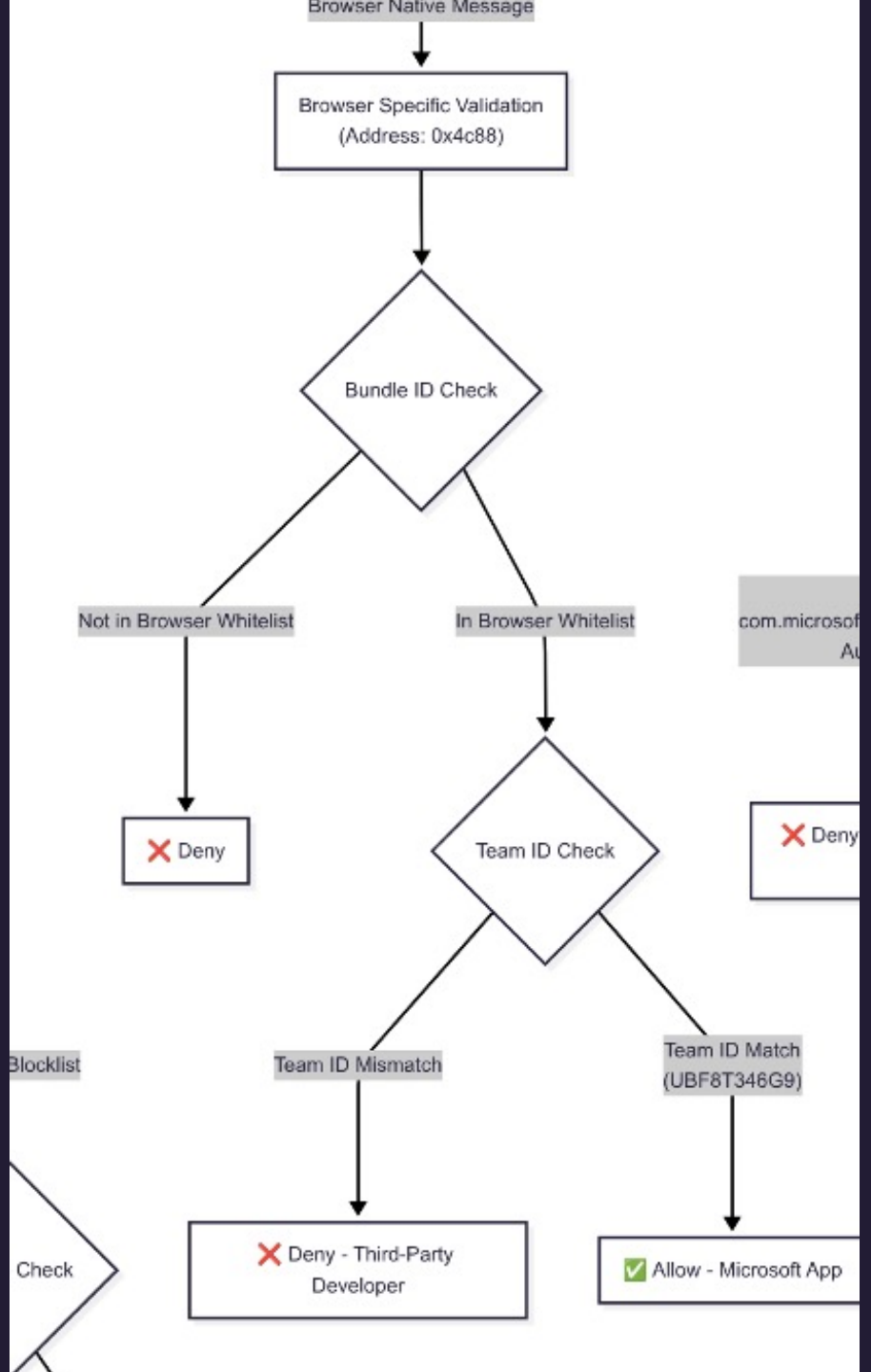

```
└─ csrutil status
System Integrity Protection status: disabled.
```

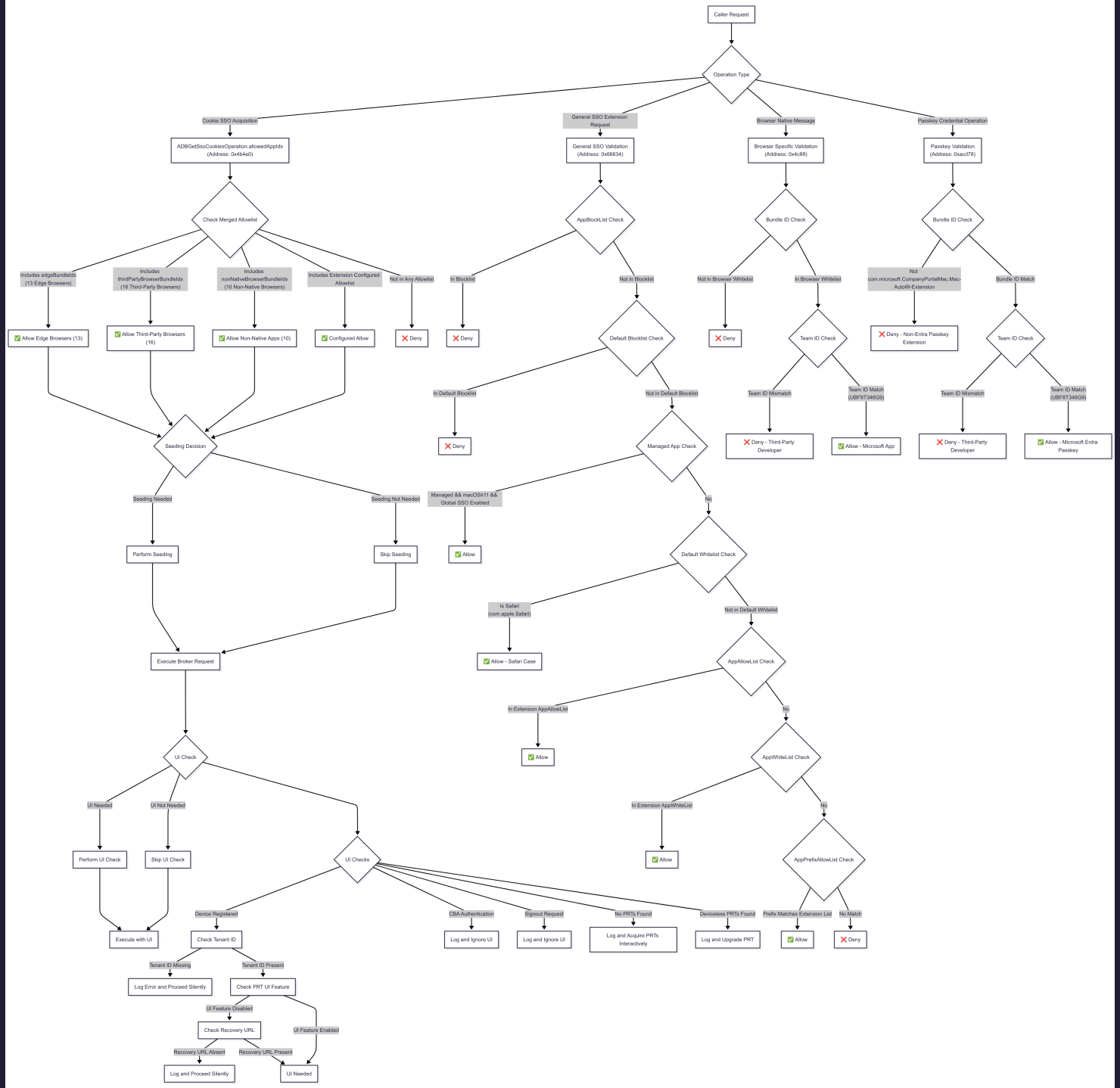
Two Different Callers in SSO Flow



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


Browser Native Message
Browser Specific Validation
(Address: 0x4c88)
Bundle ID Check
Not in Browser Whitelist
In Browser Whitelist
com.microsof
Au
Deny
Team ID Check
Deny
Blocklist
Team ID Mismatch
Team ID Match
(UBF8T346G9)
Check
Deny - Third-Party
Developer
Allow - Microsoft App


Caller Request
Operation Type
Cookie SSO Acquisition
General SSO Extension Request
Browser Native Message
Passkey Credential Operation
ADBGetSsoCookiesOperation.allowedAppIds (Address: 0x4b4a0)
General SSO Validation (Address: 0x66634)
Browser Specific Validation (Address: 0x4c88)
Passkey Validation (Address: 0xacd78)
Check Merged Allowlist
AppBlockList Check
Bundle ID Check
Bundle ID Check
Includes edgeBundleIds (13 Edge Browsers)
Includes thirdPartyBrowserBundleIds (16 Third-Party Browsers)
Includes nonNativeBrowserBundleIds (10 Non-Native Browsers)
Includes Extension Configured Allowlist
Not in Any Allowlist
In Blocklist
Not in Blocklist
Not in Browser Whitelist
In Browser Whitelist
Not com.microsoft.CompanyPortalMac.Mac-Autofill-Extension
Bundle ID Match
✅ Allow Edge Browsers (13)
✅ Allow Third-Party Browsers (16)
✅ Allow Non-Native Apps (10)
✅ Configured Allow
❌ Deny
❌ Deny
Default Blocklist Check
❌ Deny
Team ID Check
❌ Deny - Non-Entra Passkey Extension
Team ID Check
In Default Blocklist
Not in Default Blocklist
Team ID Mismatch
Team ID Match (UBF8T346G9)
Team ID Mismatch
Team ID Match (UBF8T346G9)
Seeding Decision
❌ Deny
Managed App Check
❌ Deny - Third-Party Developer
✅ Allow - Microsoft App
❌ Deny - Third-Party Developer
✅ Allow - Microsoft Entra Passkey
Seeding Needed
Seeding Not Needed
Managed && macOS≥11 && Global SSO Enabled
No
Perform Seeding
Skip Seeding
✅ Allow
Default Whitelist Check
Is Safari (com.apple.Safari)
Not in Default Whitelist
Execute Broker Request
✅ Allow - Safari Case
AppAllowList Check
In Extension AppAllowList
No
UI Check
✅ Allow
AppWhiteList Check
UI Needed
UI Not Needed
In Extension AppWhiteList
No
Perform UI Check
Skip UI Check
UI Checks
✅ Allow
AppPrefixAllowList Check
Device Registered
CBA Authentication
Signout Request
No PRTs Found
Deviceless PRTs Found
Prefix Matches Extension List
No Match
Execute with UI
Check Tenant ID
Log and Ignore UI
Log and Ignore UI
Log and Acquire PRTs Interactively
Log and Upgrade PRT
✅ Allow
❌ Deny
Tenant ID Missing
Tenant ID Present
Log Error and Proceed Silently
Check PRT UI Feature
UI Feature Disabled
Check Recovery URL
UI Feature Enabled
Recovery URL Absent
Recovery URL Present
Log and Proceed Silently
UI Needed

Direct SSO Invocation via Apple’s API

Four SSO Methods



Browser Native Message → Check Bundle & Team ID



Passkey Credential Operation → Check Bundle & Team ID



General SSO Extension Request → Check Only Bundle ID



Cookie SSO Acquisition → Check Only Bundle ID

CYCRAFT

Cookie SSO Acquisition

edgeBundleIds (13)

thirdPartyBrowserBundleIds (16)

nonNativeBrowserBundleIds (10)

- **Bundle IDs:**
  - `com.microsoft.edgemac` (Edge Stable)
  - `com.microsoft.edgemac.Beta` (Edge Beta)
  - `com.microsoft.edgemac.Dev` (Edge Dev)
  - `com.microsoft.edgemac.Canary` (Edge Canary)
  - `com.google.Chrome` (Google Chrome)
  - `org.mozilla.firefox` (Mozilla Firefox)
  - `com.apple.Safari` (Safari)
  - `com.opera.Opera` (Opera)
  - `com.vivaldi.Vivaldi` (Vivaldi)
  - `com.brave.Browser` (Brave)
  - `com.yandex.browser` (Yandex)
  - `com.microsoft.edgemac.Enterprise` (Edge Enterprise)
  - `com.microsoft.edgemac.Insider` (Edge Insider)
  - `com.microsoft.edgemac.Nightly` (Edge Nightly)
  - `com.microsoft.edgemac.Preview` (Edge Preview)
  - `com.microsoft.edgemac.Test` (Edge Test)
  - `com.microsoft.edgemac.Experimental` (Edge Experimental)
  - `com.microsoft.edgemac.Internal` (Edge Internal)
  - `com.microsoft.edgemac.Staging` (Edge Staging)
  - `com.microsoft.edgemac.Production` (Edge Production)
  - `com.microsoft.edgemac.Release` (Edge Release)
  - `com.microsoft.edgemac.Alpha` (Edge Alpha)
  - `com.microsoft.edgemac.BetaChannel` (Edge Beta Channel)
  - `com.microsoft.edgemac.DevChannel` (Edge Dev Channel)
  - `com.microsoft.edgemac.CanaryChannel` (Edge Canary Channel)
  - `com.microsoft.edgemac.StableChannel` (Edge Stable Channel)
  - `com.microsoft.edgemac.NightlyChannel` (Edge Nightly Channel)
  - `com.microsoft.edgemac.PreviewChannel` (Edge Preview Channel)
  - `com.microsoft.edgemac.TestChannel` (Edge Test Channel)
  - `com.microsoft.edgemac.ExperimentalChannel` (Edge Experimental Channel)
  - `com.microsoft.edgemac.InternalChannel` (Edge Internal Channel)
  - `com.microsoft.edgemac.StagingChannel` (Edge Staging Channel)
  - `com.microsoft.edgemac.ProductionChannel` (Edge Production Channel)
  - `com.microsoft.edgemac.ReleaseChannel` (Edge Release Channel)
  - `com.microsoft.edgemac.AlphaChannel` (Edge Alpha Channel)

Here Comes our Second POC



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

PRT Cookie survives only 15 minutes…

How can we achieve persistence?

$ 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

Ref : https://dirkjanm.io/

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

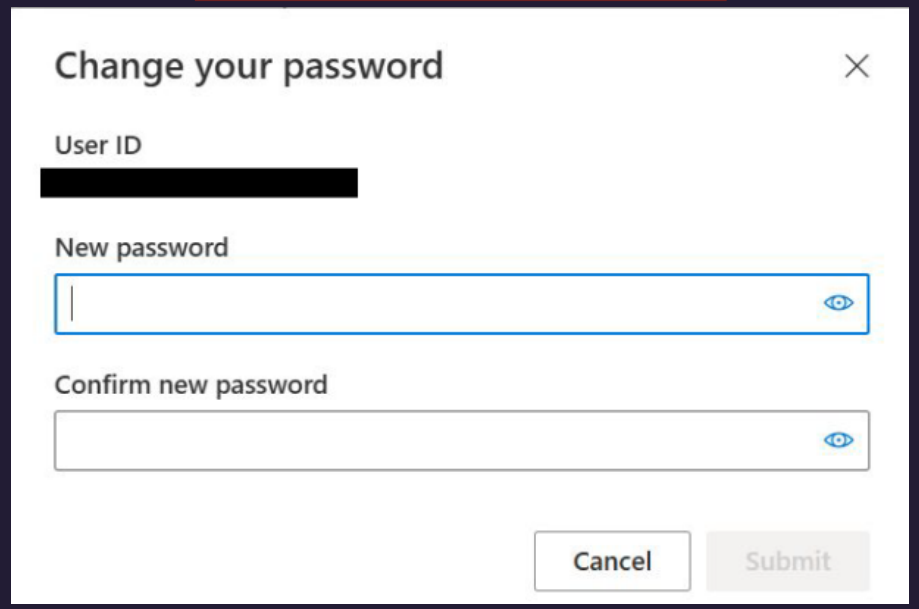
Get a Token without deviceId

- **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

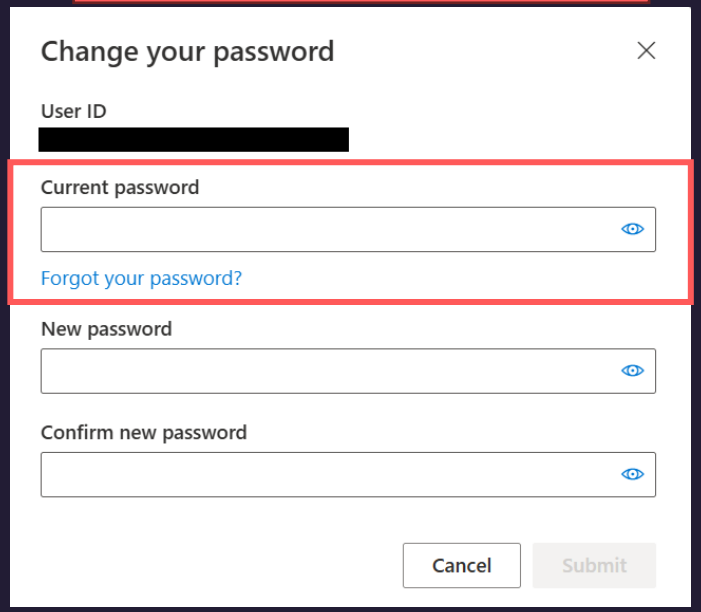
- **My Signins:** https://mysignins.microsoft.com/ front end logic
- login with Access Token
  (pwd + mfa Claim)

/api/password/reset



- login with Access Token
  (only mfa Claim)

/api/password/change



CYCRAFT

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      responese = requests.post(f'https://api.mysignins.microsoft.com/api/password/reset',
26                                headers = header,
27                                data    = json.dumps(body))
28      return responese
```

Get a Token without deviceId

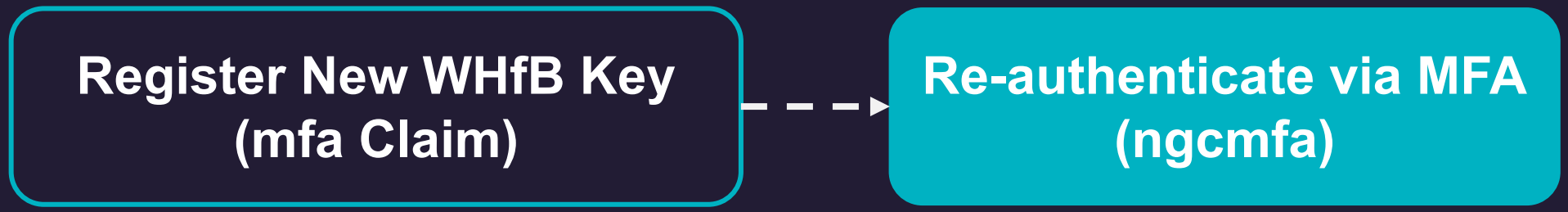
- **Phishing**
  - **Upside:** Does not require compromising the device
  - **Downside:** unreliable
- **(Kerberos) Seamless Single Sign On**
  - **Upside:** Keeps the noise level low
  - **Downside:** Less than 25% of tenants have this setting enabled
- **Reset Password**
  - **Upside:** used across 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:**
    - Interactive authentication 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

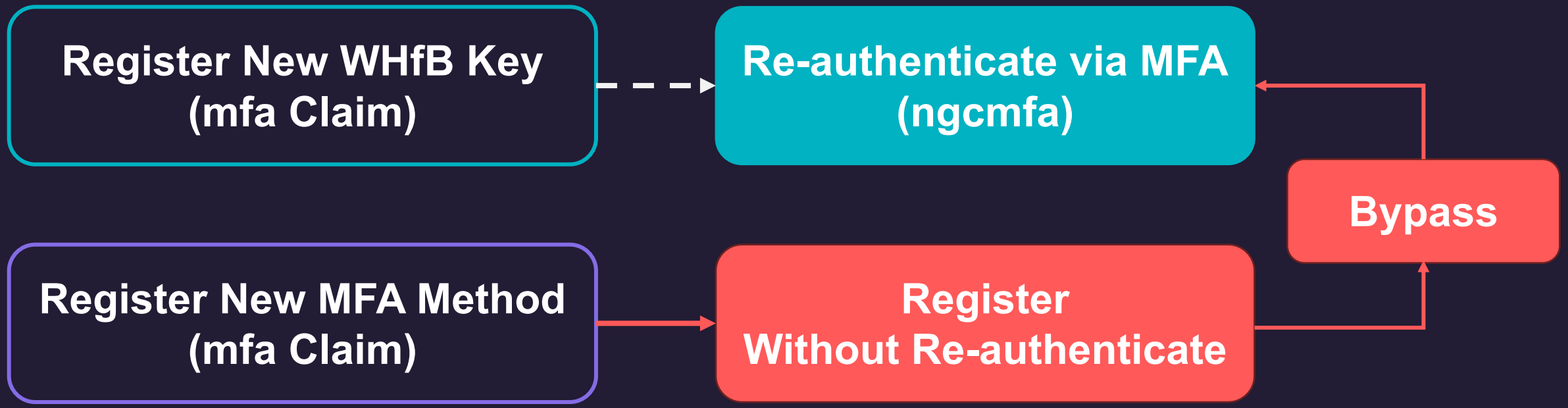
- WHfB Key enrollment token should contain the ngcmfa claim
  - Indicates recent (~10 mins) MFA was performed



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

CYCRAFT

Demo: Attack Chain Overview

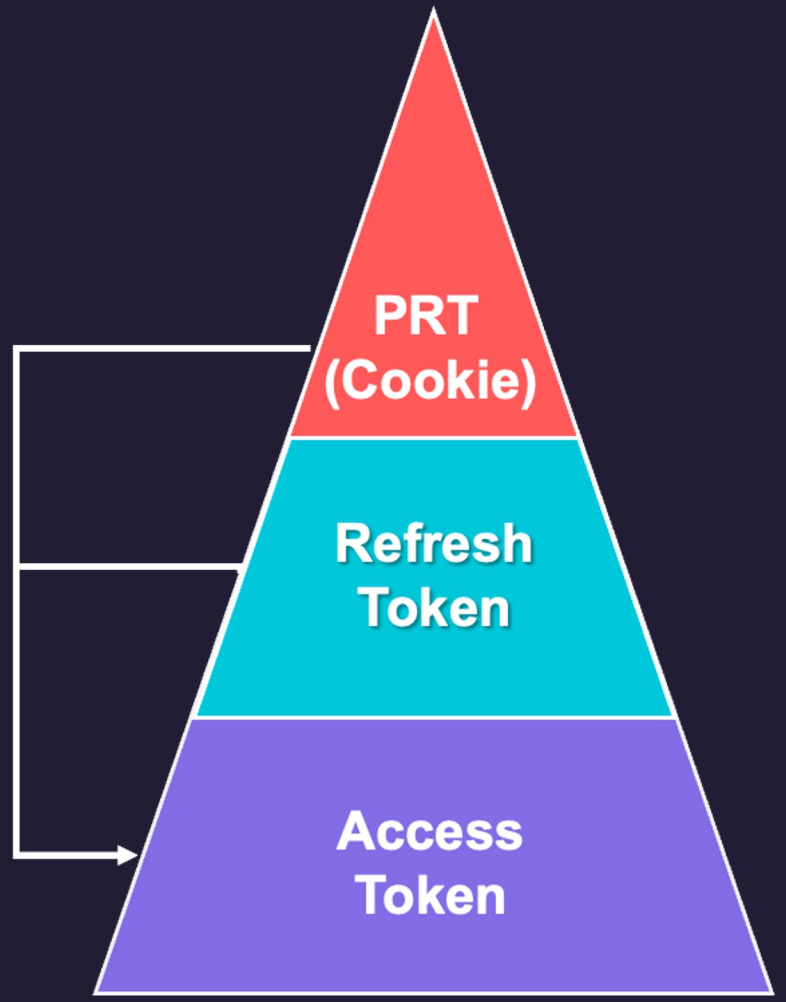
- **Request PRT cookie through direct SSO invocation**
- **Abusing device join scenario to achieve persistence**
  - **Reset Password**
  - Get a Token without deviceId
  - Register a new (fake) device
  - (Skip) Make new device pass Device Compliance
  - **Add MFA Method**
  - Register new WHfB key

DEMO

CYCRAFT


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
Home > MSFT | Overview > Conditional Access | Overview > Policies >
adelev@AdeleVs-Virtual-Machine macprt %
Victim
Save

User Identity(PRT) + Strong MFA + 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





Owned WHfB

- Require MFA for “Register or join devices”
- 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

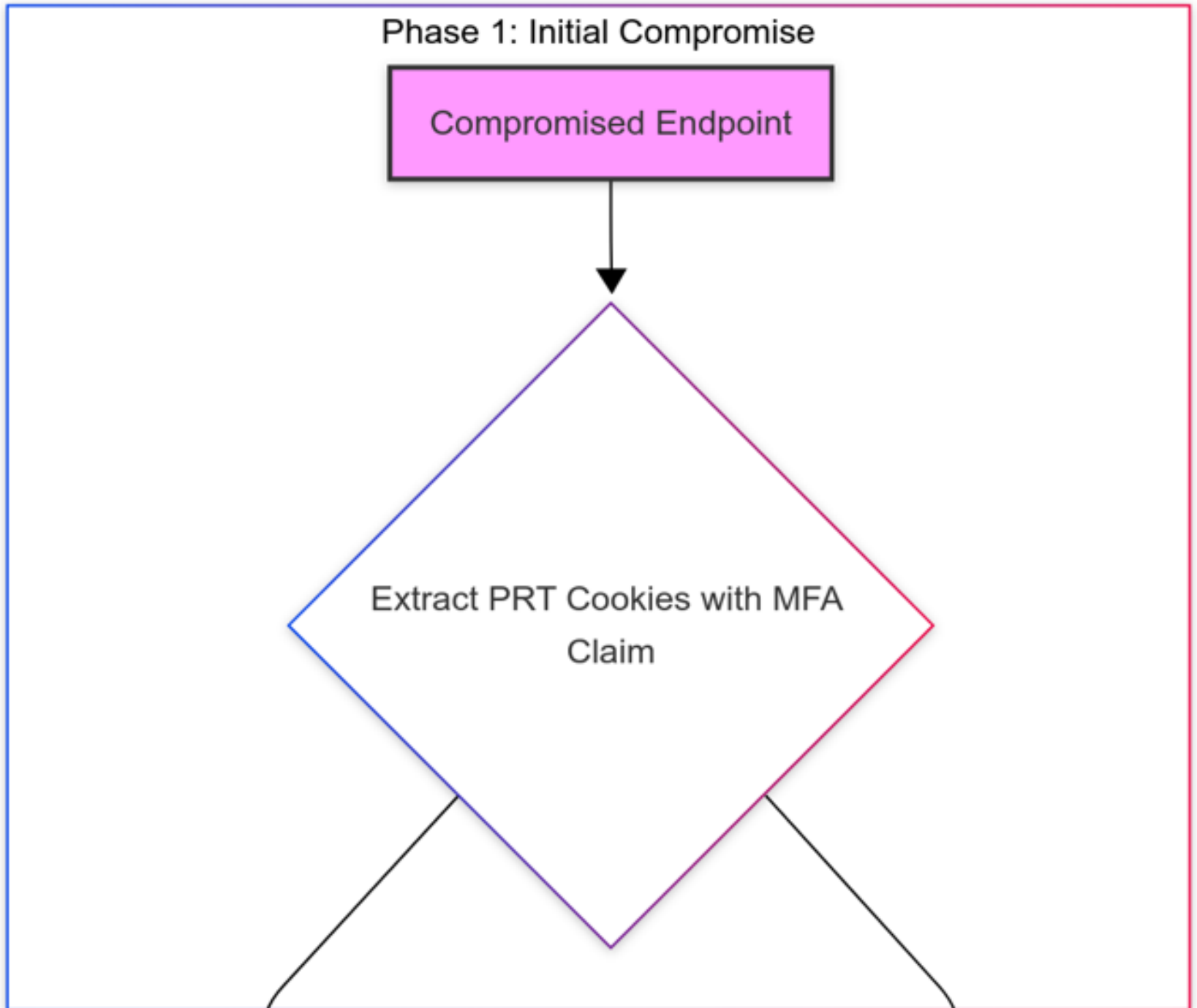


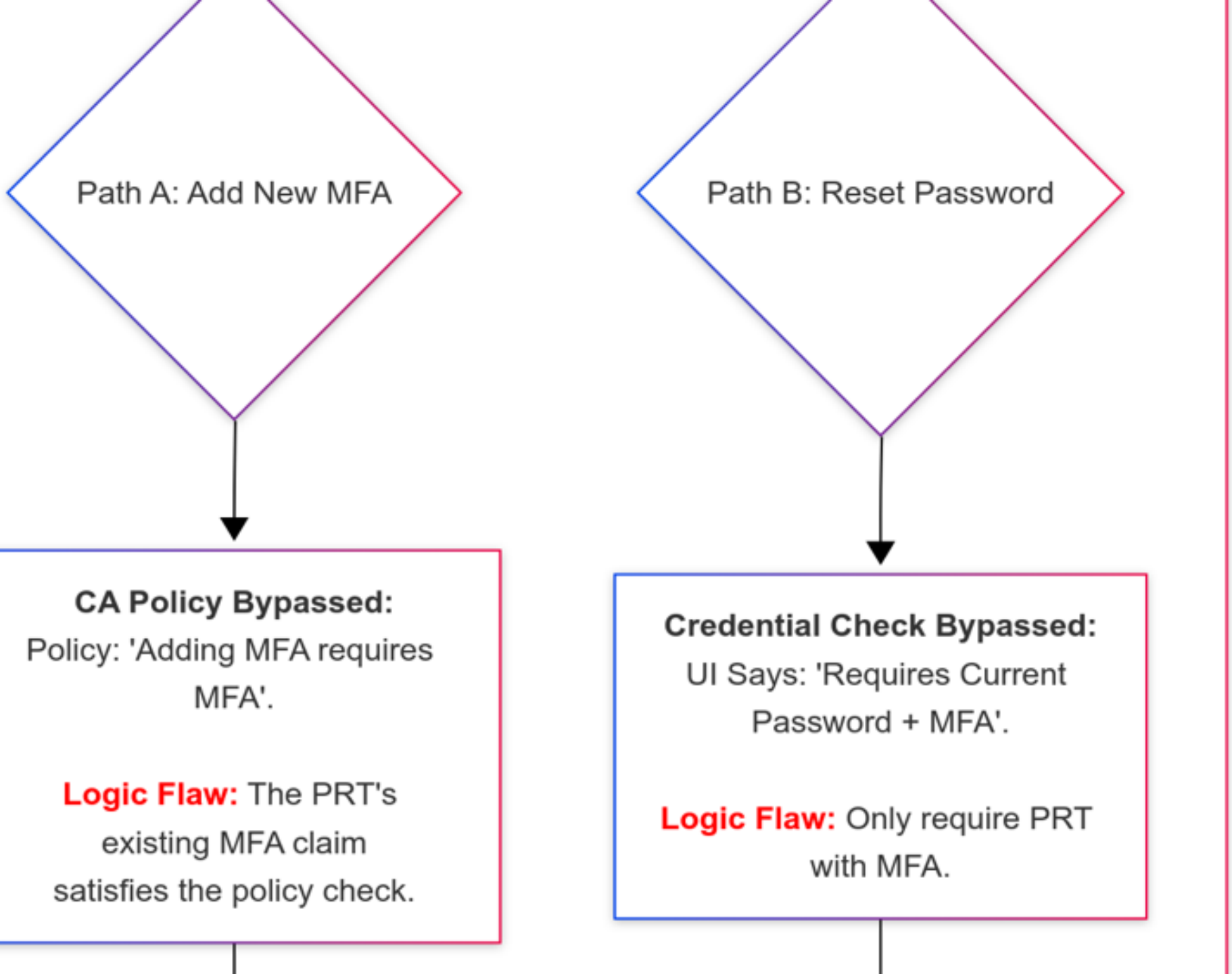
CYCRAFT



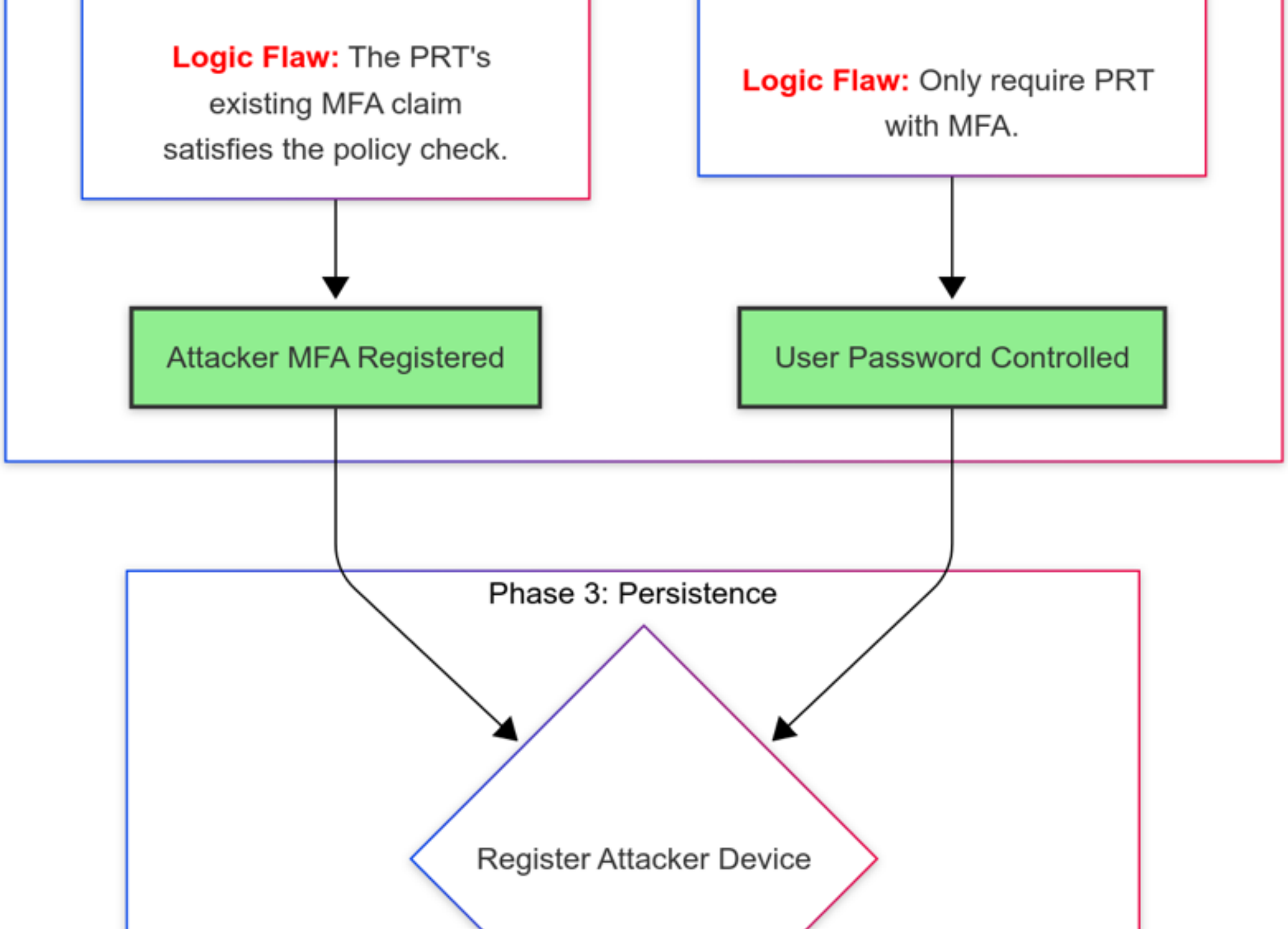
Even with these CA policies. This is not enough.




Phase 1: Initial Compromise
Compromised Endpoint
Extract PRT Cookies with MFA Claim


Path A: Add New MFA
Path B: Reset Password
CA Policy Bypassed:
Policy: 'Adding MFA requires MFA'.
Logic Flaw: The PRT's existing MFA claim satisfies the policy check.
Credential Check Bypassed:
UI Says: 'Requires Current Password + MFA'.
Logic Flaw: Only require PRT with MFA.

CYCRAFT


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



requires MFA'.

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

New Device with PRT & WHfB



End Goal

PERSISTENCE ACHIEVED

MSRC response

- Acknowledged & Patched: VULN-151266
  - This vulnerability involves the "macOS PRT Cookie" technique. Now this is fixed.
- Acknowledged - No Fix Planned: VULN-148636
  - Upon investigation, we have determined that this is not a vulnerability. This is because we discussed the case details again and determined that **this issue requires having access to someone's device to get their PRT cookie in the first place.** At that point the user is already compromised.

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.
- No fixed for today persistence methods. Pray for your endpoint never been compromised and keep monitoring.
- Following might be help you against from persistence technique (Not Quite):
  - Create Conditional Access use weak criteria such as IP/Location/UserAgent as requirement to restrict Add MFA or Join Device actions.
  - Monitor the new joined device and user add new MFA

Thank You

> Tool release: