



Silent Infiltration

Chromium Preferences Attacks





Riadh Bouchahoua

@rb-x

- 2 years in IT security
- Pentest
- Works at Synacktiv, an Offensive security company

Agenda



- Introduction
- Chromium extension structure
- Extension loading mechanism
- Administrative restrictions
- Prevention & Detection

Introduction

Introduction

Why ?

Why Browser extension ?

- LSASS.exe is heavily monitored protected
- OS became bootloader to your browser
- Browser extensions have access to some very interesting APIs

Introduction

Why Chromium-based browsers?

- ~80% market share
- Non-store extensions remain persistent
- Development less restrictive than Safari



Introduction

What are Chromium extension

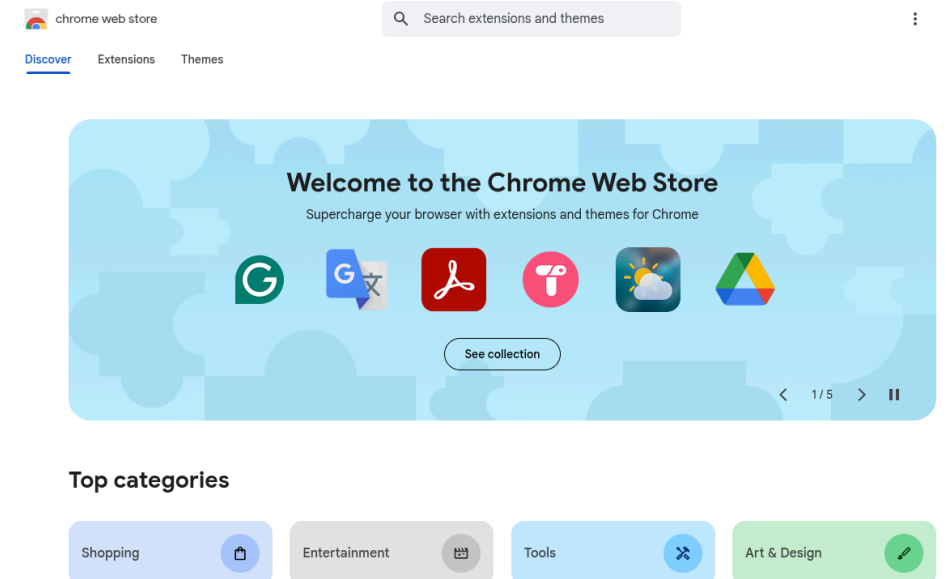
- Extend and customize the browser
- Deep interaction with webpages
- Access to sensitive browser APIs



Introduction

Installation methods

- **Chrome Web Store / Edge Add-ons Store**
- Command-line arguments at startup (loads unpacked extensions)
- Developer mode



Introduction

Installation methods

- Chrome Web Store / Edge Add-ons Store
- **Command Line arguments at startup**

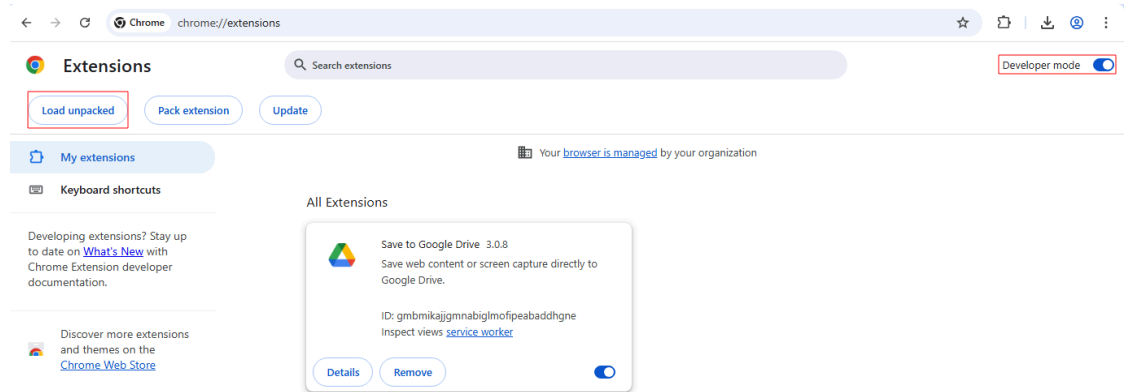
```
PS1> chrome.exe --disable-features=DisableLoadExtensionCommandLineSwitch --load=C:\unpacked_ext\
```

- Developer mode

Introduction

Installation methods

- Official Chrome Web Store or Edge add-ons Store
- Command Line arguments at startup
- **Developer mode**
 - Load unpacked extension from folder
 - Install self-signed CRX



Introduction

Extension format

- **Packed extensions**

- Chrome Web Store: signed **CRX** (ZIP archive + metadata + Web Store signature)
- Self-signed CRX: possible, but only installable in **Developer Mode** (not trusted by default)

- **Unpacked extensions**

- Source files in a local folder
- Loaded via **Developer Mode** (persists across sessions)

Chromium extension structure

Chromium extension structure

Extension anatomy

-  **unpacked-extension/**
 -  **manifest.json**
 - Defines an extension's **info, permissions**, etc...
 -  content-scripts
 -  service worker script

```
$ cat unpacked-extension/manifest.json
{
  "manifest_version": 3,
  "name": "Simple extension",
  "version": "1.0",
  "description": "A minimalistic MV3 Chrome extension example.",
  "permissions": [
    "cookies",
    "tabs",
    "storage",
    "scripting"
  ],
  "host_permissions": ["<all_urls>"],
  "background": {
    "service_worker": "background.js"
  },
  "content_script": [
    {
      "matches": ["<all_urls>"],
      "js": ["content-script.js"]
    }
  ]
}
```

Chromium extension structure

Extension anatomy

-  **unpacked-extension/**
 -  manifest.json
 -  **content-script**
 - (Read / Modify) the DOM of the page
 -  service worker script

```
$ cat unpacked-extension/content-script.js

console.log("Hello from content-script !");

const dumpStorage = (storage) =>
  [...Array(storage.length).keys()]
    .map(i => {
      const key = storage.key(i);
      return { key, value: storage.getItem(key) };
    });

console.log("localStorage contents:", dumpStorage(localStorage));
console.log("sessionStorage contents:", dumpStorage(sessionStorage));
```

Chromium extension structure

Extension anatomy

-  **unpacked-extension/**
 -  manifest.json
 -  content-scripts
 -  **service worker script**
 - Handle events and have access to the browser API

```
$ cat unpacked-extension/background.js  
  
console.log("Hello from service worker !")  
// Get all cookies  
chrome.cookies.getAll({}, (cookies) => console.log('All cookies:', cookies))  
// List Tabs  
chrome.tabs.query({}, (tabs) => console.log('Open tabs:', tabs))  
// List history, screenshot capture and many more ...
```

Chromium extension structure

service worker script in action

```
manifest.json  JS content-script.js  JS background.js x
```

```
backdooring_chromium > demo > extension > JS background.js > ...
1 console.log("Hello from service worker !")
2
3 chrome.cookies.getAll({}, (cookies) => {
4   console.log('All cookies:', cookies);
5 });
6
```

Extensions

Load unpacked Pack extension Update

Your browser is managed by your organization

All Extensions

Simple extension 1.0
A minimalistic MV3 Chrome extension example.
ID: ndlpefnfkohemhdpkfgbfbchpneif
Inspect views: [service worker](#)

Chromium PDF Viewer 1
ID: mhjfbmdgcfjbbpaeojofohoefghehjai

DevTools

Console Application Sources Network Performance Memory

background... Filter

Hello from service worker !

All cookies:

```
Array(16)
  0: {domain: '.google.com', expirationDate: 1772123543.863798, hostOnly: false, httpOnly: true, name: 'AEC', ...}
  1: {domain: '.google.com', expirationDate: 1790758241.35724, hostOnly: false, httpOnly: true, name: '__Secure-ENID', ...}
  2: {domain: '.www.linkedin.com', hostOnly: false, httpOnly: false, name: 'JSESSIONID', path: '/', ...}
  3: {domain: '.linkedin.com', hostOnly: false, httpOnly: false, name: 'lang', path: '/', ...}
  4: {domain: '.linkedin.com', expirationDate: 1788107560.014274, hostOnly: false, httpOnly: false, name: 'bcookie', ...}
  5: {domain: '.www.linkedin.com', expirationDate: 1788107560.014304, hostOnly: false, httpOnly: true, name: 'bscookie', ...}
  6: {domain: '.linkedin.com', expirationDate: 1772123560.01433, hostOnly: false, httpOnly: false, name: 'li_gc', ...}
  7: {domain: '.linkedin.com', expirationDate: 1756657960.014353, hostOnly: false, httpOnly: false, name: 'lidc', ...}
  8: {domain: '.linkedin.com', expirationDate: 1756573360.014378, hostOnly: false, httpOnly: true, name: '__cf_bm', ...}
  9: {domain: '.www.linkedin.com', expirationDate: 1788107560, hostOnly: true, httpOnly: false, name: 'li_alerts', ...}
  10: {domain: 'accounts.google.com', expirationDate: 1791131561.433705, hostOnly: true, httpOnly: true, name: '__Host-GAPS', ...}
  11: {domain: 'accounts.google.com', expirationDate: 1759163562, hostOnly: true, httpOnly: false, name: 'OTZ', ...}
  12: {domain: 'azure.microsoft.com', expirationDate: 1788107607.955191, hostOnly: true, httpOnly: false, name: 'MicrosoftApplicati...', ...}
  13: {domain: 'azure.microsoft.com', expirationDate: 1756573407, hostOnly: true, httpOnly: false, name: 'ai_session', ...}
  14: {domain: '.microsoft.com', expirationDate: 1788107619.284136, hostOnly: false, httpOnly: false, name: 'MC1', ...}
  15: {domain: '.microsoft.com', expirationDate: 1756573419.284224, hostOnly: false, httpOnly: false, name: 'MS0', ...}
  length: 16
  [[Prototype]]: Array(0)
```


Chromium extension structure

Content-script in action

The image displays a Chromium extension's internal structure and its execution. On the left, a code editor shows the `content-script.js` file with the following code:

```
1 console.log("Hello from content-script !");
2
3 const dumpStorage = (storage) =>
4   [...Array(storage.length).keys()]
5     .map(i => {
6       const key = storage.key(i);
7       return { key, value: storage.getItem(key) };
8     });
9
10 console.log("localStorage contents:", dumpStorage
    (localStorage));
11 console.log("sessionStorage contents:", dumpStorage
    (sessionStorage));
12
```

On the right, a browser window shows the Google homepage. The browser's developer tools are open to the Console tab, displaying the output of the content script:

```
localStorage contents: content-script.js:10
  ▶ [{...}]

sessionStorage contents: content-script.js:11
  (16) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...},
  {...}, {...}, {...}, {...}, {...}, {...}, {...}]
```

Chromium extension structure

Offensive capabilities

Capability	Content Script	Service Worker
 Access ALL cookies	 No	 Yes
 Steal sensitive data (DOM/storage)	 Yes	 Limited
 Inject malicious code in pages	 Yes	 No
 Take screenshots	 No	 Yes
 Intercept/modify network requests	 No	 Yes

Chromium extension structure

Offensive capabilities

- Communication between **service worker** and **content-script**

```
// content-script.js
chrome.runtime.sendMessage({
  localStorageData: localStorage,
  sessionStorageData: sessionStorage
});
```

```
// background.js
[...]

chrome.runtime.onMessage.addListener((message, _sender, _sendResponse) => {
  let { localStorageData, sessionStorageData } = message;
  handle(localStorageData);
  handle(sessionStorageData);
});
```

Chromium extension structure

Offensive capabilities > Filesystem access



Post-exploitation

Offensive capabilities > Filesystem access

- **Webpages** can't access the local file system because of the Same-Origin Policy (SOP)
- **The browser** can browse and access files though **file://** scheme
 - as can extension through service worker

Post-exploitation

Offensive capabilities > Filesystem access

Service Worker can leverage **tabs** and **host_permissions** to:

- Open a background window of 1 px on **file://<location>**
- Parse and extract tab content
- Exfiltrate data as required

Post-exploitation

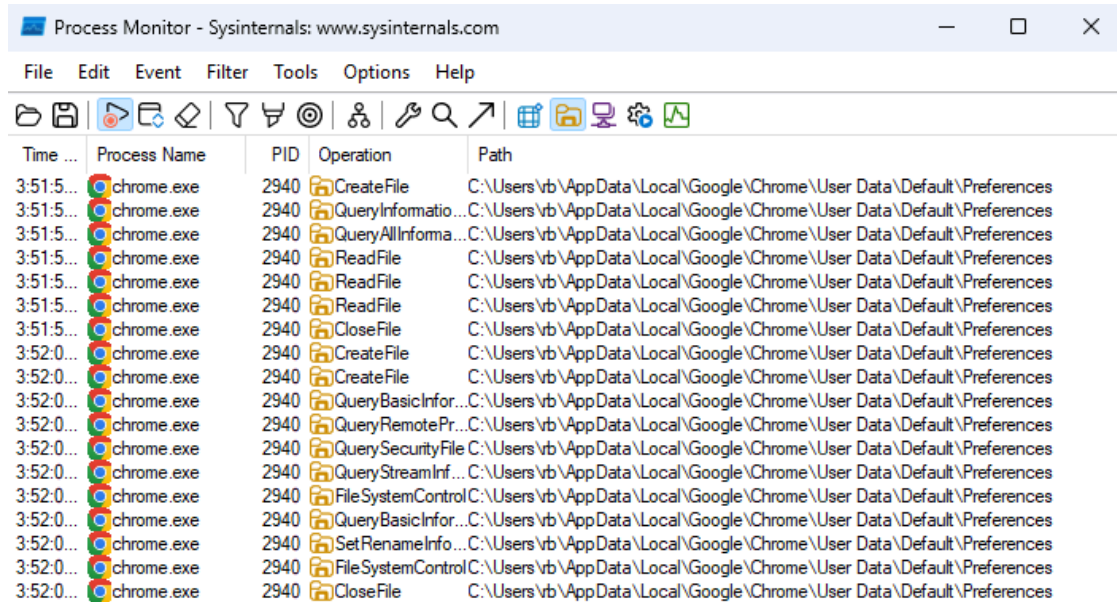
Offensive capabilities > Filesystem access

- MIME type that can be displayed (and exfiltrated):
 - **text/* (plain, html, xml, csv...)**
 - application/xml
 - application/javascript
 - application/json
 - application/ld+json

Extension loading mechanism

Extension loading mechanism

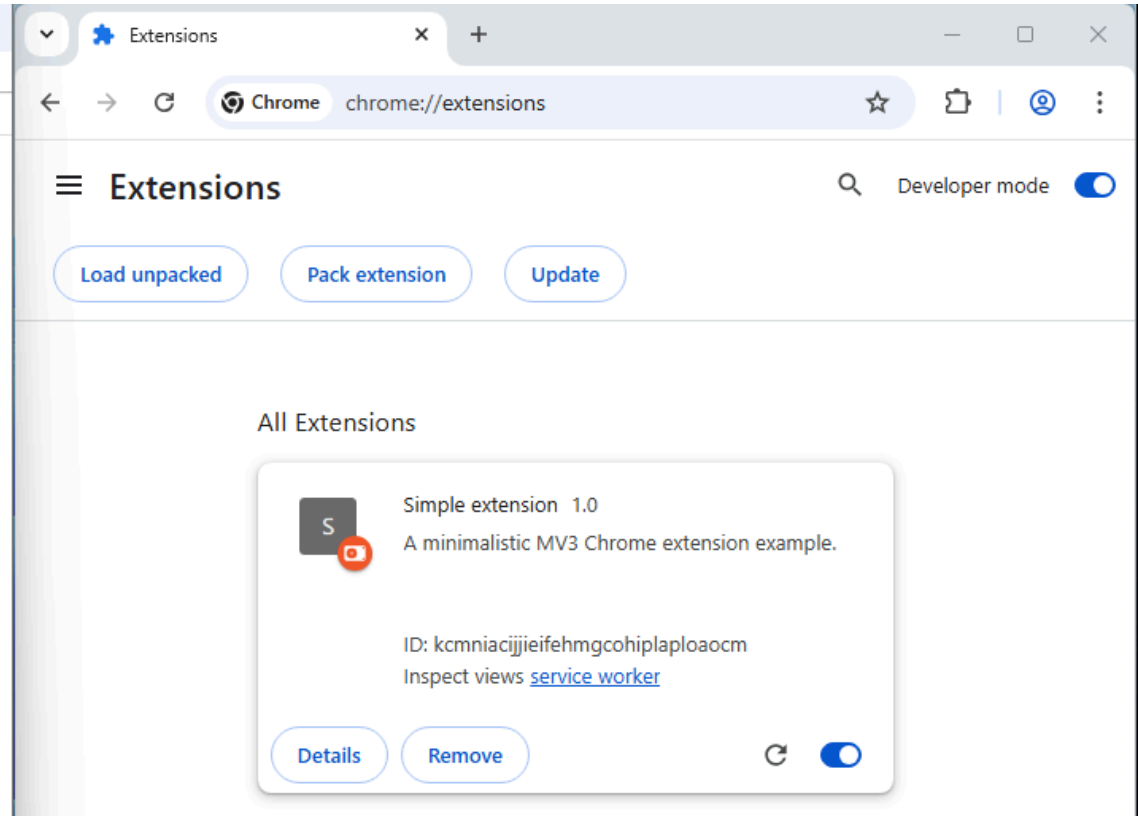
Preferences file



Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path
3:51:5...	chrome.exe	2940	CreateFile	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:51:5...	chrome.exe	2940	QueryInformatio...	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:51:5...	chrome.exe	2940	QueryAllInforma...	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:51:5...	chrome.exe	2940	ReadFile	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:51:5...	chrome.exe	2940	ReadFile	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:51:5...	chrome.exe	2940	ReadFile	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:51:5...	chrome.exe	2940	CloseFile	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:52:0...	chrome.exe	2940	CreateFile	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:52:0...	chrome.exe	2940	QueryBasicInfor...	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:52:0...	chrome.exe	2940	QueryRemotePr...	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:52:0...	chrome.exe	2940	QuerySecurityFile	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:52:0...	chrome.exe	2940	QueryStreamInf...	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:52:0...	chrome.exe	2940	FileSystemControl	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:52:0...	chrome.exe	2940	QueryBasicInfor...	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:52:0...	chrome.exe	2940	SetRenameInfo...	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:52:0...	chrome.exe	2940	FileSystemControl	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences
3:52:0...	chrome.exe	2940	CloseFile	C:\Users\vb\AppData\Local\Google\Chrome\User Data\Default\Preferences



Extension loading mechanism

Preferences file

All Extensions

 Simple extension 1.0
A minimalistic MV3 Chrome extension example.

ID: kcmniacijjiefhmgcohiplaploaocm

Inspect views [service worker \(Inactive\)](#)

Details Remove  

```
Windows PowerShell
PS C:\Users\rb\AppData\Local\Google> Get-ChildItem -Path . -File -Recurse -ErrorAction SilentlyContinue |
>> Select-String -Pattern "kcmniacijjiefhmgcohiplaploaocm" -SimpleMatch -ErrorAction SilentlyContinue |
>> Select-Object -ExpandProperty Path |
>> ForEach-Object { Split-Path $_ -Leaf } |
>> Sort-Object -Unique
000003.log
Preferences
QuotaManager
Tabs_13401233015145867
PS C:\Users\rb\AppData\Local\Google>
```

Extension loading mechanism

Preferences files

```
{
  "extensions":
  {
    "settings":
    {
      "<extension_hash>":{
        "name" : "Extension name",
        [...rest of manifest.json]
      },
      {...}
    },
    "ui": {
      "developer_mode": false
    }
  }
  [...]
  "protection":{
    "macs":{
      "extensions":{
        "settings":{
          "<extension_hash>" : "<MAC>"
        }
      },
      "extensions": {
        "ui": {
          "developer_mode": "<MAC>"
        }
      }
    }
  }
}
```

Extension loading mechanism

Goals

- Obtain a static Chromium extension ID
- Modify the Preferences file accordingly

Extension loading mechanism

Extension id calculation

- Computed by hashing the absolute path of the extension folder.
 - `chromium/src/components/crx_file/id_util.cc:GenerateIdForPath`
- **Generated by hashing the public key of an RSA X.509 certificate into a 32-character string using letters a through p.**
 - `chromium/src/components/crx_file/id_util.cc:GenerateId`

```
$ python3 gen_crx_id.py
```

```
pubkey: MIIBIjANBgkqhkiG9w0[...]  
crx_id_from_pubkey: aifbejmahijpgmclcdffcjbkochehfoi
```

```
$ cat manifest.json
{
  "key" : "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQE...",
  "manifest_version": 3,
  "name": "Simple extension",
  "version": "1.0",
  "[...]"
}
```

All Extensions

 Simple extension 1.0
A minimalistic MV3 Chrome extension example.

ID: `aifbejmahijpgmclcdffcjbkochefoi`
Inspect views [service worker](#)

Details Remove  

Extension loading mechanism

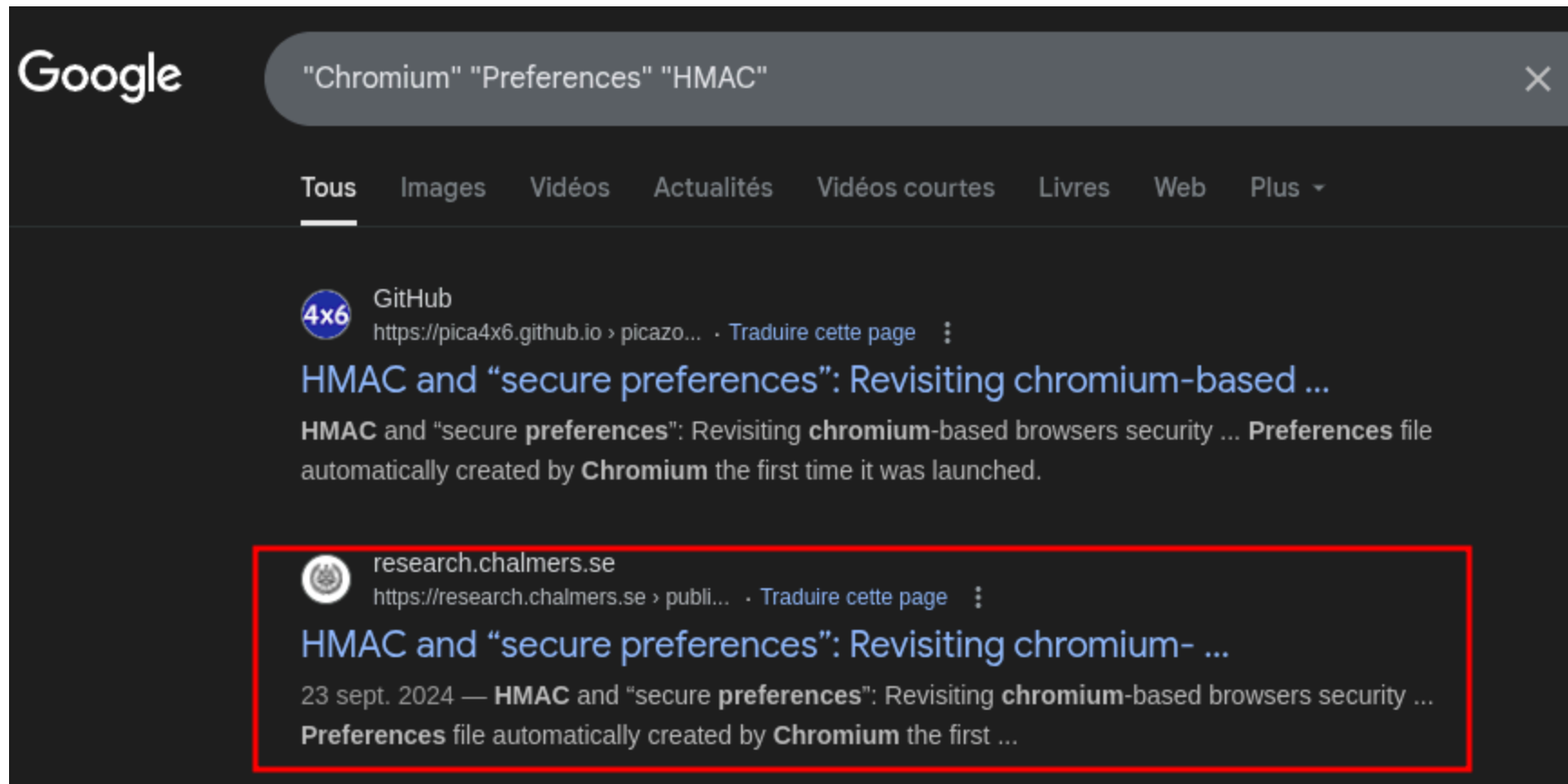
Goals

- **Obtain a static Chromium extension ID** 
- **Modify the Preferences file accordingly**
 - Register our extension 
 - HMAC SHA 256 signature of the extension
 - HMAC SHA 256 signature of developer_mode (v134+)

Extension loading mechanism

HMAC key

- (2020) HMAC and Secure Preferences: Revisiting Chromium-based Browsers Security, **Pablo Picazo-Sanchez**



Extension loading mechanism

Preferences file > HMAC key recovery

- That Chrome browser uses a **hardcoded** key
 - C:\Program Files\Google\Chrome\Application**<version>**\resources.pak
- Chromium browser this key is **null**

Extension loading mechanism

Preferences file > HMAC key recovery

```
<!-- browser_resources.grd => ressources.pak !-->
<if expr="_google_chrome">
  <include name="IDR_PREF_HASH_SEED_BIN" file="resources\settings\internal\pref_hash_seed.bin" type="BINDATA" />
  <include name="IDR_ADDITIONAL_MODULE_IDS" file="${additional_modules_list_file}" use_base_dir="false" type="BINDATA" />
</if>
```

```
// chrome_pref_service_factory.cc:277
std::unique_ptr<ProfilePrefStoreManager> CreateProfilePrefStoreManager(
    const base::FilePath& profile_path) {
    std::string seed;
    CHECK(ui::ResourceBundle::HasSharedInstance());
    #if BUILDFLAG(GOOGLE_CHROME_BRANDING)
        seed = std::string(ui::ResourceBundle::GetSharedInstance().GetRawDataResource(IDR_PREF_HASH_SEED_BIN));
    #endif
    return std::make_unique<ProfilePrefStoreManager>(profile_path, seed);
}
```

Extension loading mechanism

Preferences file > HMAC key recovery

```
# https://github.com/Pica4x6/SecurePreferencesFile/blob/main/Seed.py#L60
# Pablo Picazo-Sanchez, Gerardo Schneider and Andrei Sabelfeld: "HMAC and 'Secure Preferences' (Chrome 85)
[...]
```

```
'expected_seed': b'\xe7H\xf36\xd8^\xa[...]\xa8'
```

- Same HMAC-SHA256 key found on Chrome v130 !

	resources	resources.pak															
Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00006730	C8	70	B1	FA	FE	D0	48	70	09	FF	56	A8	A7	49	84	DD	Èp±úþÐHp.ÿV`SI,,Ý
00006740	F9	77	5B	AA	49	1E	52	A4	E2	46	86	23	2A	F2	0D	3B	ùw[°I.R×âF†#*ò.;
00006750	6D	D4	B1	1D	7B	B3	A3	8A	B2	8A	ED	99	9A	4D	64	07	mÔ±.(°£Š°Ši™šMd.
00006760	46	6A	44	76	94	3F	4D	6A	04	76	E2	D7	DD	AA	99	62	FjDv`°?Mj.vâ×Ý°°°b
00006770	67	86	B2	C2	D0	2E	F2	13	C9	80	E3	8A	5A	16	27	B8	gt°âÐ.ò.É€âŠZ.' ,
00006780	98	57	7C	6E	8E	3D	64	F3	54	4B	D7	30	5B	16	B1	A2	°W nŽ=dóTK×O[.±°
00006790	04	CF	29	AF	A4	02	CF	F7	08	6D	3A	B7	13	94	5F	60	.İ)°×.İ÷.m:°°°°`
000067A0	C3	F3	42	D3	C6	8B	CA	63	6A	8C	97	F8	CB	AD	47	A7	šóBÓE<ÉcjE-øE.GS
000067B0	F0	65	FE	B8	39	62	E2	15	85	34	F1	2A	53	8B	8A	07	šep,9bâ....4ñ°S<Š.
000067C0	DF	D7	94	E7	5D	1F	AF	33	12	FB	26	DE	50	88	8F	37	š×°°ç].°3.ú&P^°.7
000067D0	15	12	E1	2D	85	34	F1	36	EB	2C	4E	1A	02	EF	F0	97	..â-...4ñ6ē,N..ið-
000067E0	E3	D1	04	EF	B2	19	48	3A	6F	D0	EF	31	45	BA	20	B7	šÑ.i°°.H:oÐi1E°°
000067F0	F1	3E	43	99	9B	67	F8	80	A3	C0	F3	B0	42	57	9E	85	ñ>C™>gø€£âó°BWž...
00006800	4D	6C	A5	A2	08	5B	EB	B2	13	39	91	6D	BA	64	13	03	Ml¥°. [ë°.9°m°d..
00006810	33	3F	5C	B6	CD	8C	FF	03	47	69	0E	1C	2F	1F	00	00	3?\\gÍËÿ.Gi.../...
00006820	E7	48	F3	36	D8	5E	A5	F9	DC	DF	25	D8	F3	47	A6	5B	çHó6ø^¥uÜš°øóG! [
00006830	4C	DF	66	76	00	F0	2D	F6	72	4A	2A	F1	8A	21	2D	26	Lšfv.š-örJ°ñŠ!-š
00006840	B7	88	A2	50	86	91	0C	F3	A9	03	13	69	68	71	F3	DC	°°°P†°.ó@...ihqóÜ
00006850	05	82	37	30	C9	1D	F8	BA	5C	4F	D9	C8	84	B5	05	A8	.,70É.ø°\OÙÈ„µ..

Extension loading mechanism

Preferences file > HMAC key recovery

- chromium/chromium/src/refs/heads/main/./tools/grit/pak_util.py

```
$ python3 pak_util.py extract resources.pak -o resources_v139/  
$ python3 pak_util.py extract resources.pak -o resources_v139_dirty/  
  
$ diff -bry resources_v139 resources_v139_dirty  
Binary files resources_v139/146 and resources_v139_dirty/146 differ  
$ xxd -p resources_v139/146  
e748f336d85ea5f9dcdf25d8f347a65b4cdf667600f02df67[...]5a8
```

Extension loading mechanism

Preferences file > Enabling Developer mode ui

- `developer_mode` MAC key
 - Same HMAC key as the extension hash

```
// Preferences
{
  "extensions":
  {
    "settings":
    {
      "<extension_hash>":{
        "name" : "Extension name",
        [...rest of manifest.json]
      },
      {...}
    },
    "ui": {
      "developer_mode": true
    }
  }
  [...]
  "protection":{
    "macs":{
      "extensions":{
        "settings":{
          "<extension_hash>" : "<MAC>"
        }
      },
      "extensions": {
        "ui": {
          "developer_mode": "<MAC>"
        }
      },
    },
  }
}
```

Extension loading mechanism

Preferences file > Pseudocode

SHA256 MAC of the JSON section corresponding to the extension using the seed.

```
prefs = load_pref_json()

# Update extension settings array and dev mode
prefs["extensions"]["settings"][ext_id] = ext_settings
prefs["extensions"]["ui"]["developer_mode"] = True

# Calculate HMACs (device_key == sid in windows)
prefs["protection"]["macs"][ext_id] = HMAC(seed, sid + ext_id + ext_settings)
prefs["protection"]["ui"]["developer_mode"] = HMAC(seed, sid + "developer_mode" + "true")

save_json(prefs)
```

Extension loading mechanism

Goal

- Generate a static extension ID
- Place the unpacked extension on disk or on a readable network share (UNC)
- Edit Preferences entries under extensions.settings for the extension
- Recompute Secure Preferences HMACs for all modified keys so the changes are accepted
- Restart or kill Chrome to reload preferences

Extension loading mechanism

Automatizing all the steps + Post-exploitation

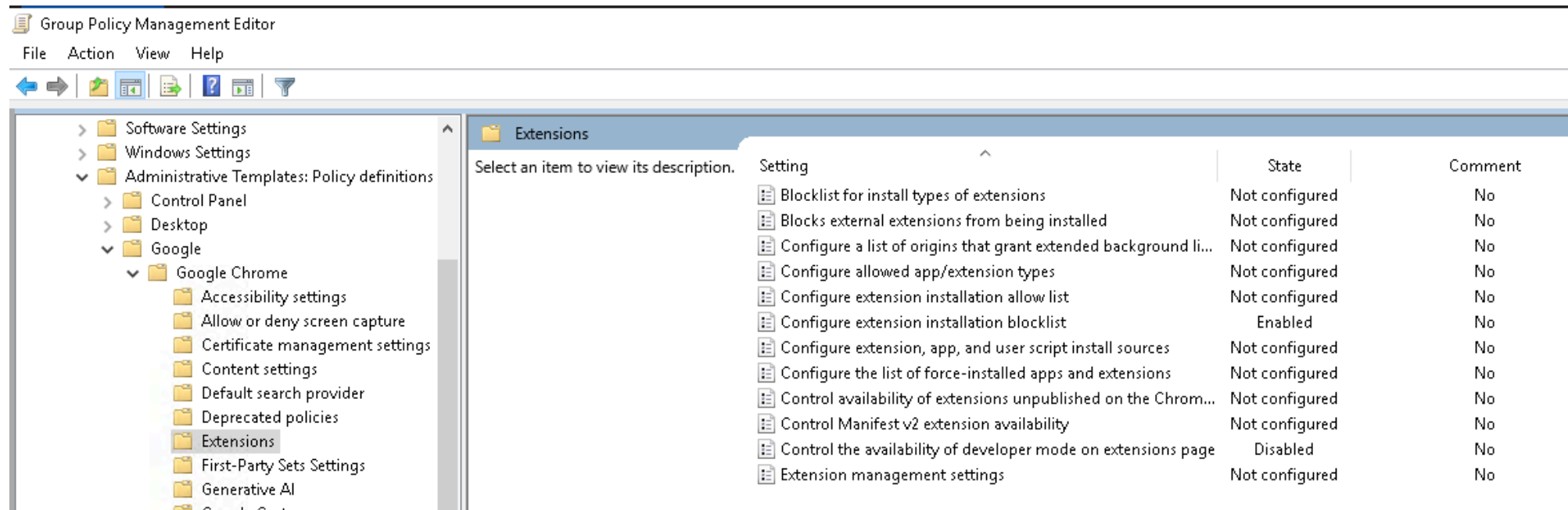


Administrative restrictions

Administrative restrictions

Group policy object

- Group Policy Object (GPO) on Windows
 - Require the installation of Chrome ADM/ADMX templates



Administrative restrictions

Group policy object

The image shows two side-by-side browser windows. The left window displays the 'Policies' page in Chrome, with the URL 'chrome://policy'. It shows a list of policies. The 'ExtensionInstallBlocklist' policy is highlighted with a red box. The right window shows the Chrome Web Store page for 'Adobe Acrobat: PDF edit, convert, sign tools'. A red box highlights a message: 'Your admin has blocked this item (ID: efaidnbmnnnibpcajpcgiclfndmkaj)'. Below this, the extension details are visible, including the name 'Adobe Acrobat: PDF edit, convert, sign tools' and a rating of 4.4 stars.

Policy Name	Value	Platform	Mach...	Mand...	Status	Action
CloudManagementEnr...	8cf5b468-3a87-4deb-bc6b-d...	Platfor...	Machi...	Mand...	OK	Show...
ExtensionInstallBlocklist	["*"]	Cloud	Machi...	Mand...	OK	Show...
GenAiDefaultSettings	1	Cloud	Machi...	Mand...	OK	Show...
MandatoryExtensionsF...	[]	Cloud	Machi...	Mand...	Unrele...	Show...
AIModelSettings					Not set.	Show...

Administrative restrictions

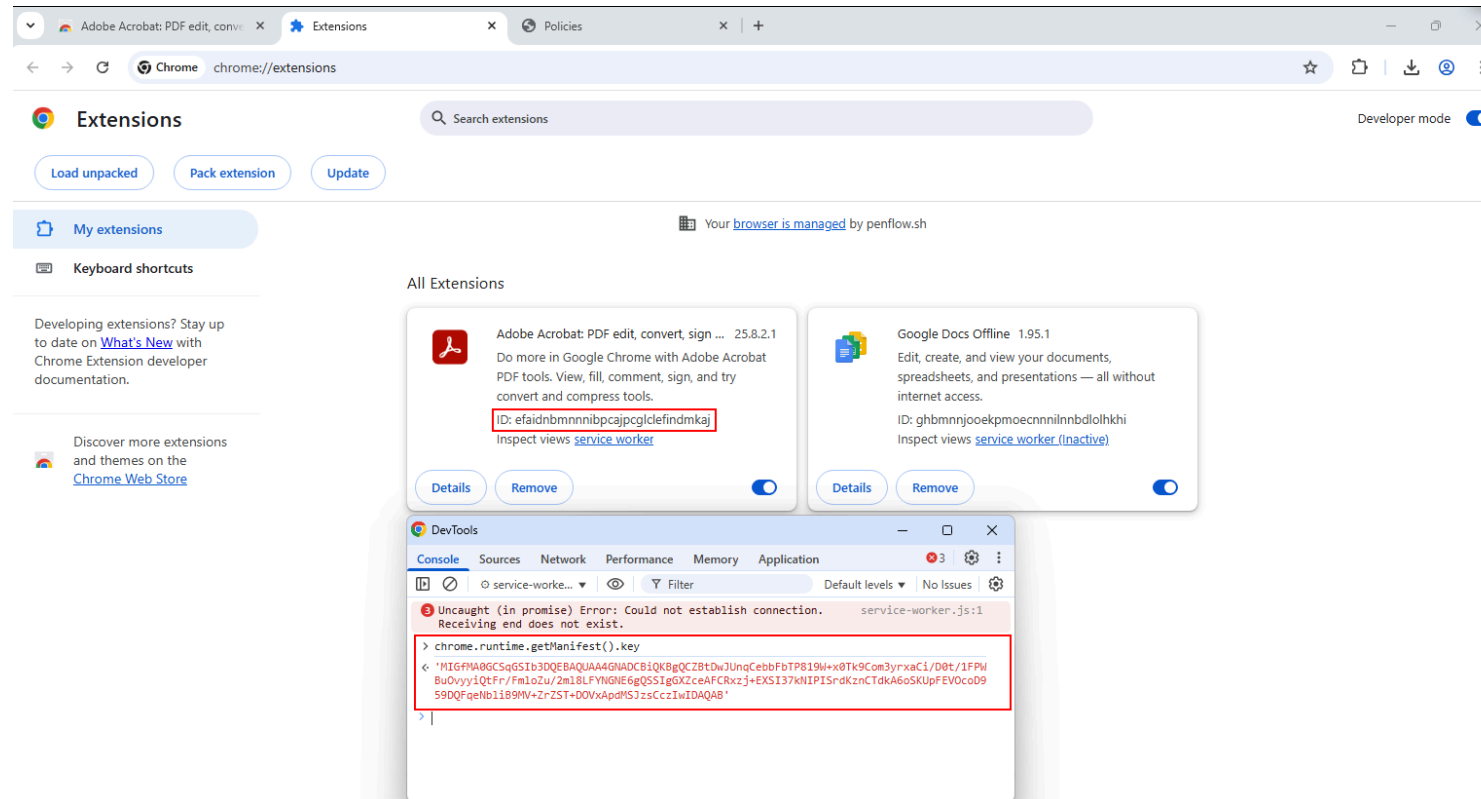
Group policy object

- Configure extension installation Allow List GPO
 - **HKEY_CURRENT_USER\SOFTWARE\Policies\Google\Chrome\ExtensionAllowlist**

Administrative restrictions

Extension id spoofing bypass

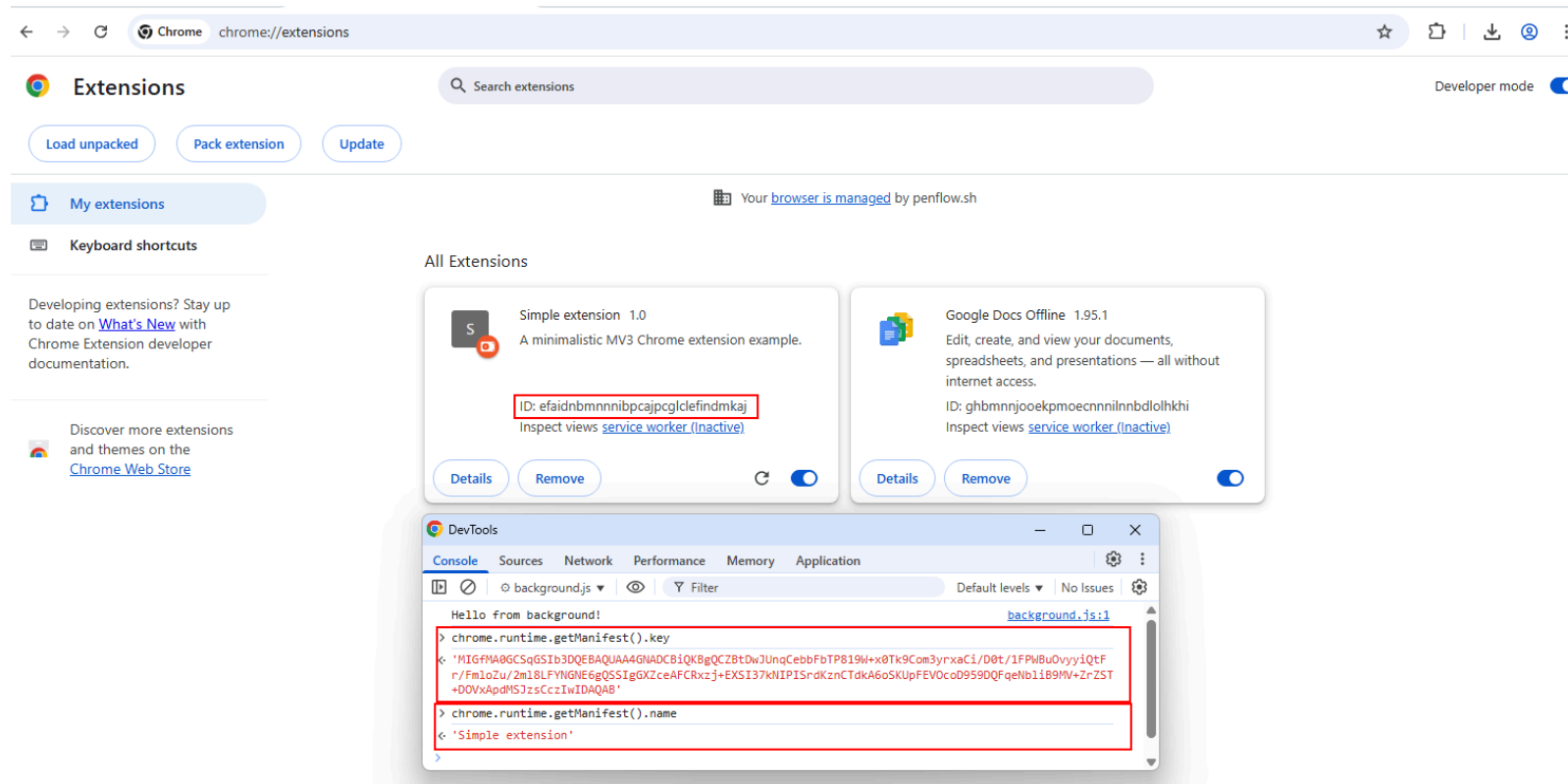
- Extension ID can be precaculated by defining a **public key**
- Since it's present in the manifest, what if we spoof a legitiamete extension id ?



Administrative restrictions

Extension stomping bypass

- If the extension is already installed
- Then our unpacked extension will stomp it and make the other one invisible from the UI !



Prevention & Detection

Prevention & Detection

HKLM Policy definition

- **HKLM** > Cloud > HKCU (Default via GPO)

ExtensionInstallBlocklist	["from HKLM"]	Platform	Machine	Mandatory	Error, Warnin...	Show less
Value	["from HKLM"]					1
Error	Value doesn't match format.					
Warning	This policy is working as intended but a conflicting value is set elsewhere and is overridden by this policy.					
Value (conflict)	["from HKCU"]	Platform	Current user	Mandatory		3
Value (conflict)	["*"]	Cloud	Machine	Mandatory		2

Prevention & Detection

User (Detection/Prevention)

- Define manual entries **HKLM ExtensionInstallAllowList** and **ExtensionInstallBlockList**
 - HKLM > Cloud > HKCU (Default via GPO)
- Common registry hives and paths to monitor for Chrome policies:
 - **HKEY_CURRENT_USER\SOFTWARE\Policies\Google\Chrome/**
 - **HKEY_CURRENT_USER\SOFTWARE\Policies\Google\Chrome/**
- Monitor Preferences file

Prevention & Detection

Admin user (Detection)

- With admin privileges, all security measure can be disabled (HKLM/HKCU)...
- If admin is compromised it's almost too late at this point to prevent the software

Prevention & Detection

Tooling

- POC and C2 has been recently released few weeks ago !
 - Chrome alone C2 && Loader (Michael Weber / Praetorian)
 - REDext C2 (DarkRain009)
 - SharpSilentChrome Loader (ChoiSG)

- Chrome for developers documentation
- Google Chrome и Secure Preferences (2015) - Kaimi.io
- HMAC and “Secure Preferences” (2020) - Pablo Picazo-Sanchez

Thanks!

Questions?

The logo for SYNACKTIV features a stylized icon on the left, composed of a 3x3 grid of squares with a red dot in the center square. To the right of the icon, the word "SYNACKTIV" is written in a bold, sans-serif font. "SYN" is in white, and "ACKTIV" is in red.

SYNACKTIV

