SYNACKTIV

Silent Infiltration

Chromium Preferences Attacks

OrangeCon 2025

whoami

SYNACKTIV

Riadh Bouchahoua

@rb-x

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

Agenda

SYNACKTIV

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

Introduction

Introduction

Why ?

SYNACKTIV

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?

SYNACKTIV

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

Introduction

What are Chromium extension

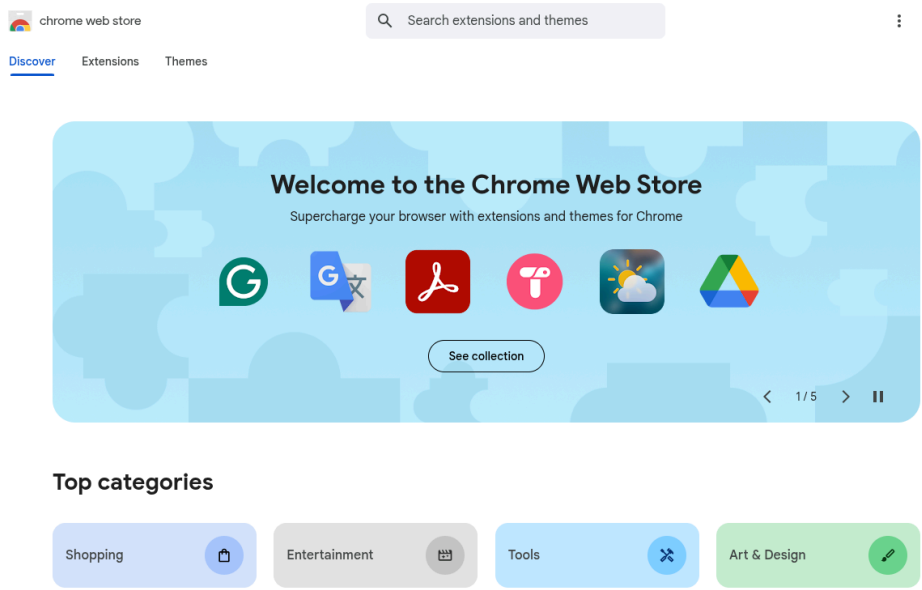
SYNACKTIV

- 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

SYNACKTIV

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

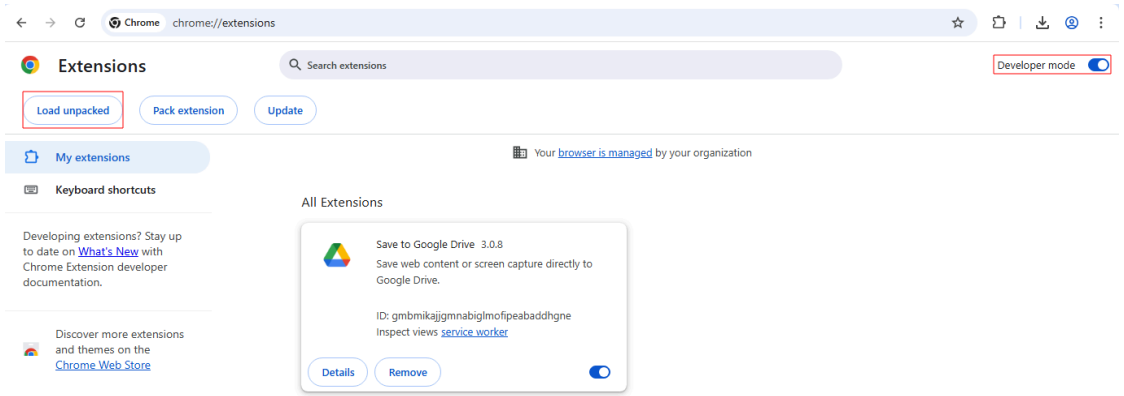
```
PS1> chrome.exe --disable-features=DisableLoadExtensionCommandLineSwitch --load=C:\unpacked_ext\
```

- 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

SYNACKTIV

- **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

SYNACKTIV

- 📁 **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

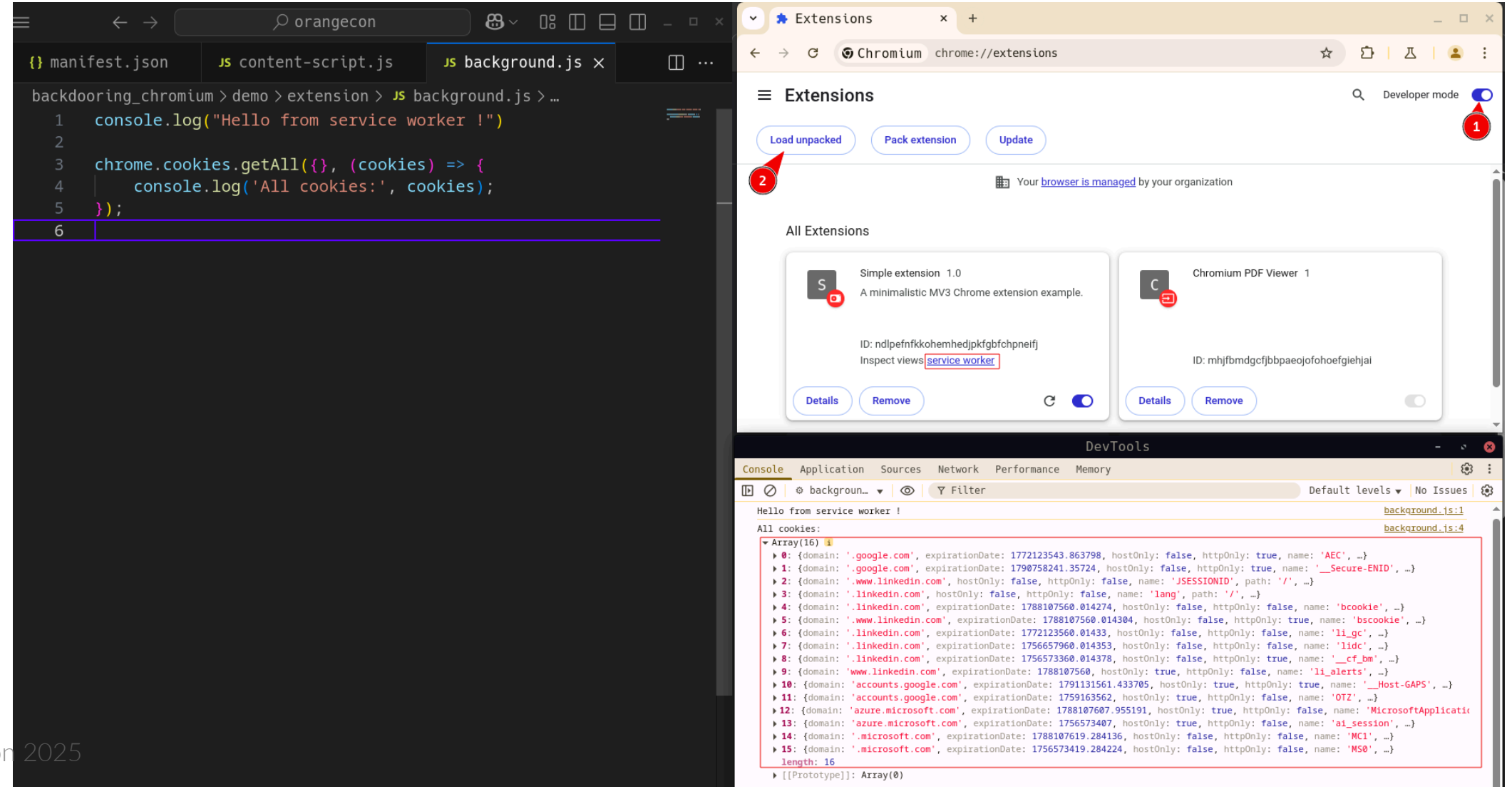
- 📁 **unpacked-extension/**
  - 📄 manifest.json
  - 📄 content-scripts
  - 📄 **service worker script**
    - Handle events and have access to the browser API

```
$ 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

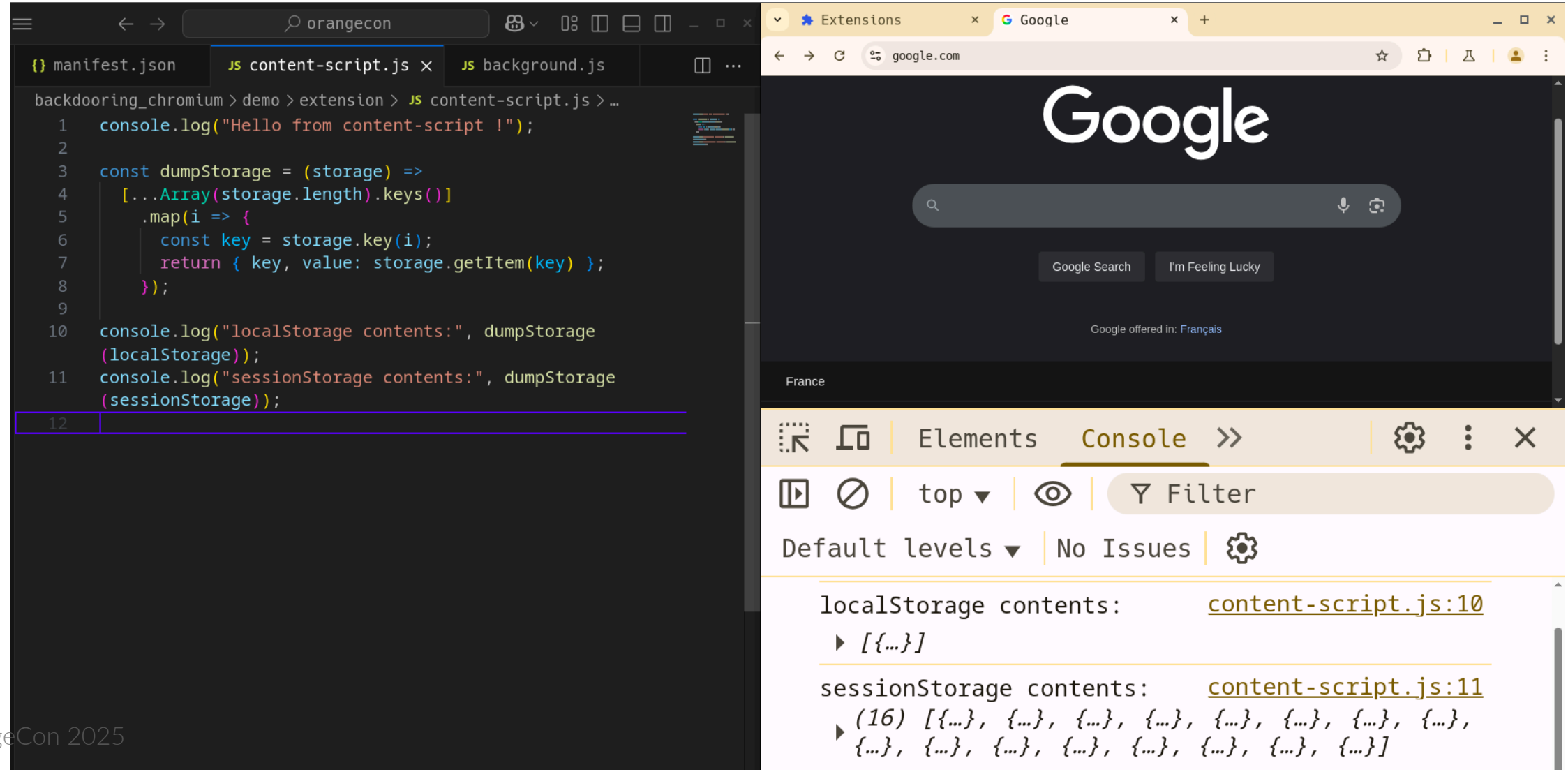
service worker script in action



Chromium extension structure

SYNACKTIV

Content-script in action



Chromium extension structure

Offensive capabilities

SYNACKTIV

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

SYNACKTIV

- 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

SYNACKTIV

- **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

SYNACKTIV

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

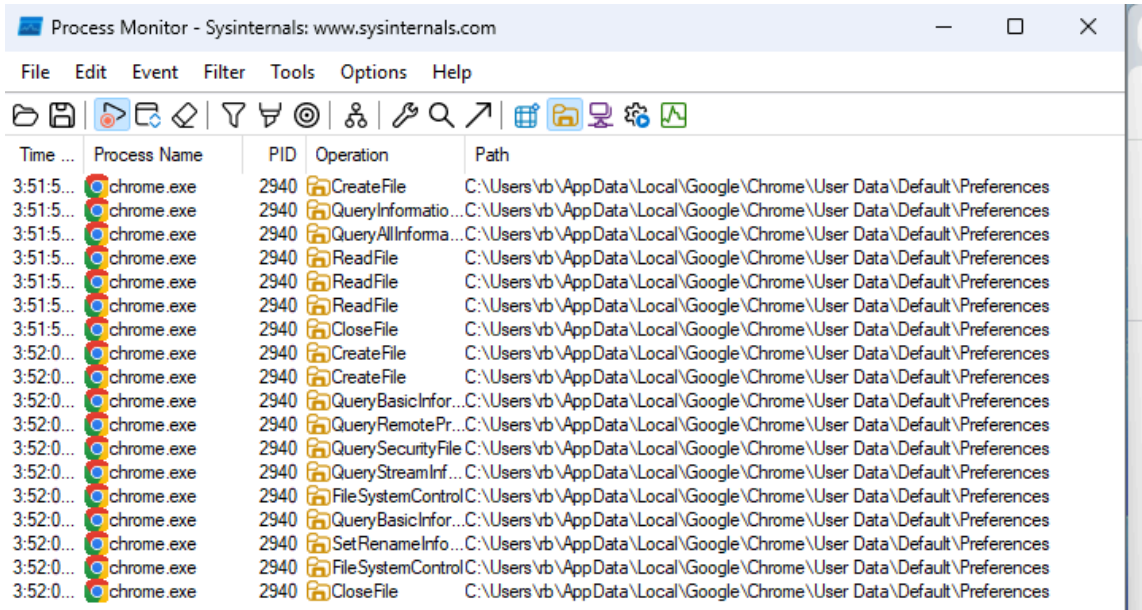
SYNACKTIV

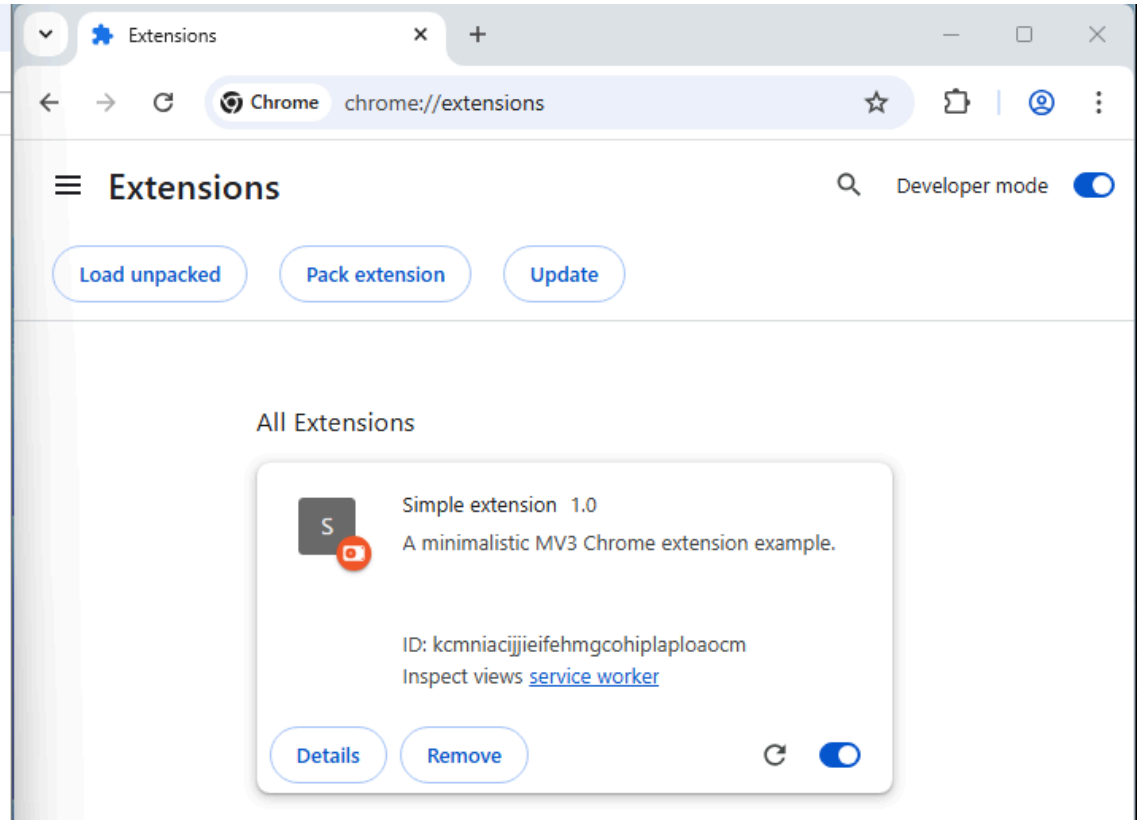
- MIME type that can be displayed (and exfiltrated):
  - **text/* (plain, html, xml, csv…)**
  - application/xml
  - application/javascript
  - application/json
  - application/ld+json

Extension loading mechanism

Extension loading mechanism

Preferences file

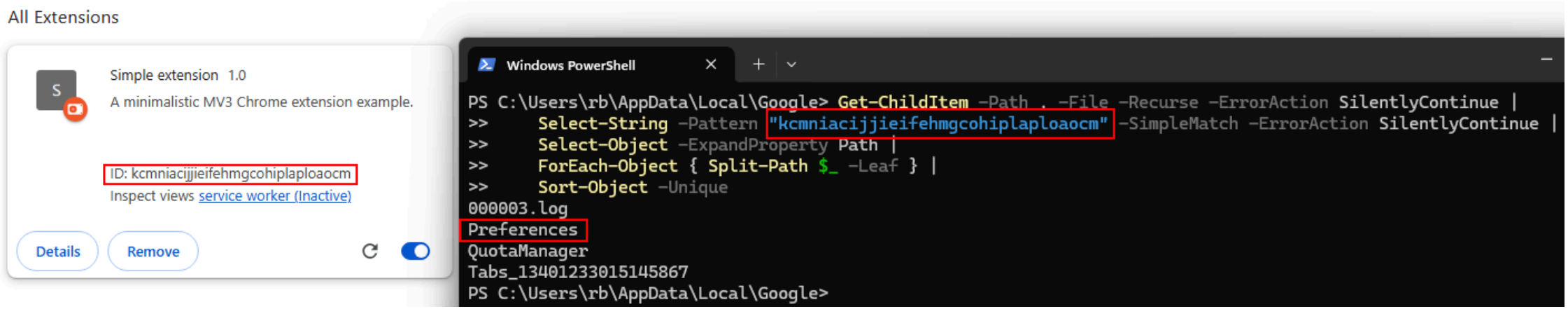




Extension loading mechanism

Preferences file

SYNACKTIV



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

SYNACKTIV

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

Extension loading mechanism

Extension id calculation

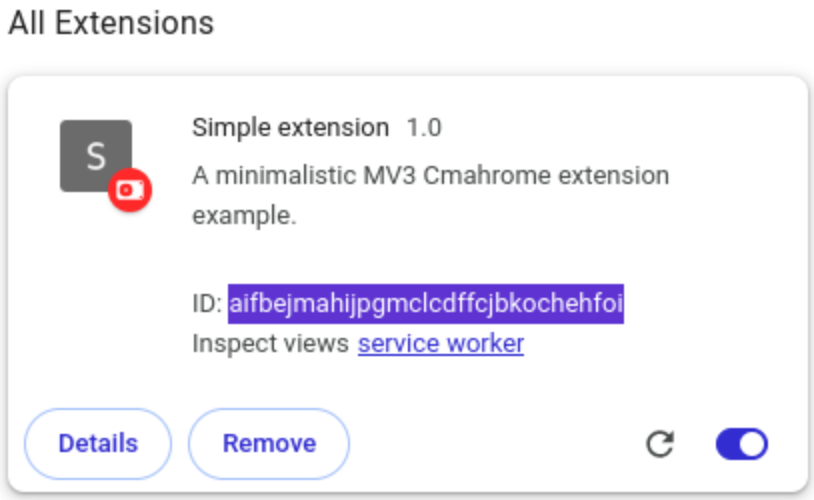
SYNACKTIV

- 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",
    "[...]"
}
```



Extension loading mechanism

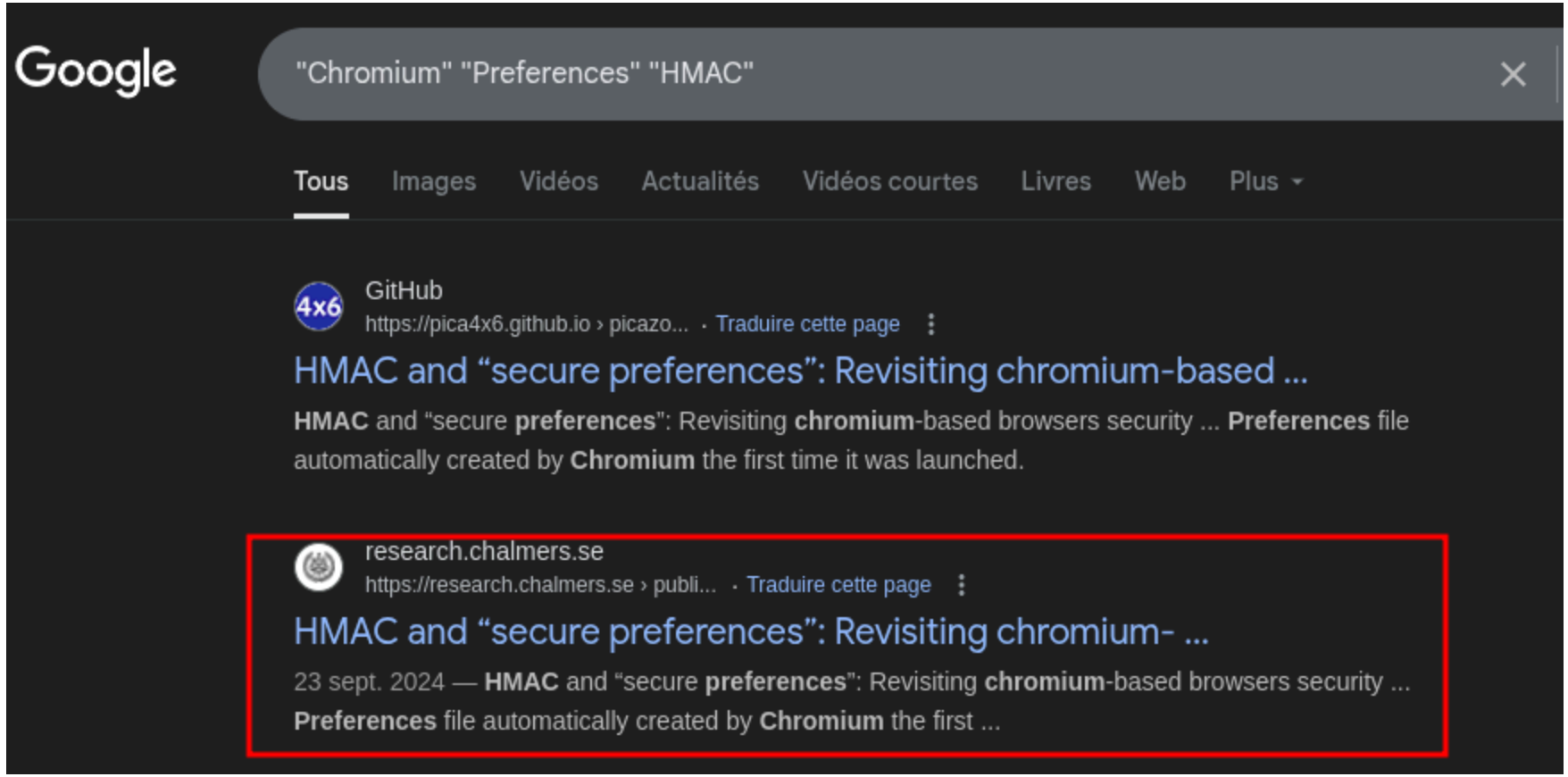
Goals

- **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 developper_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

SYNACKTIV

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

Extension loading mechanism

SYNACKTIV

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>
```

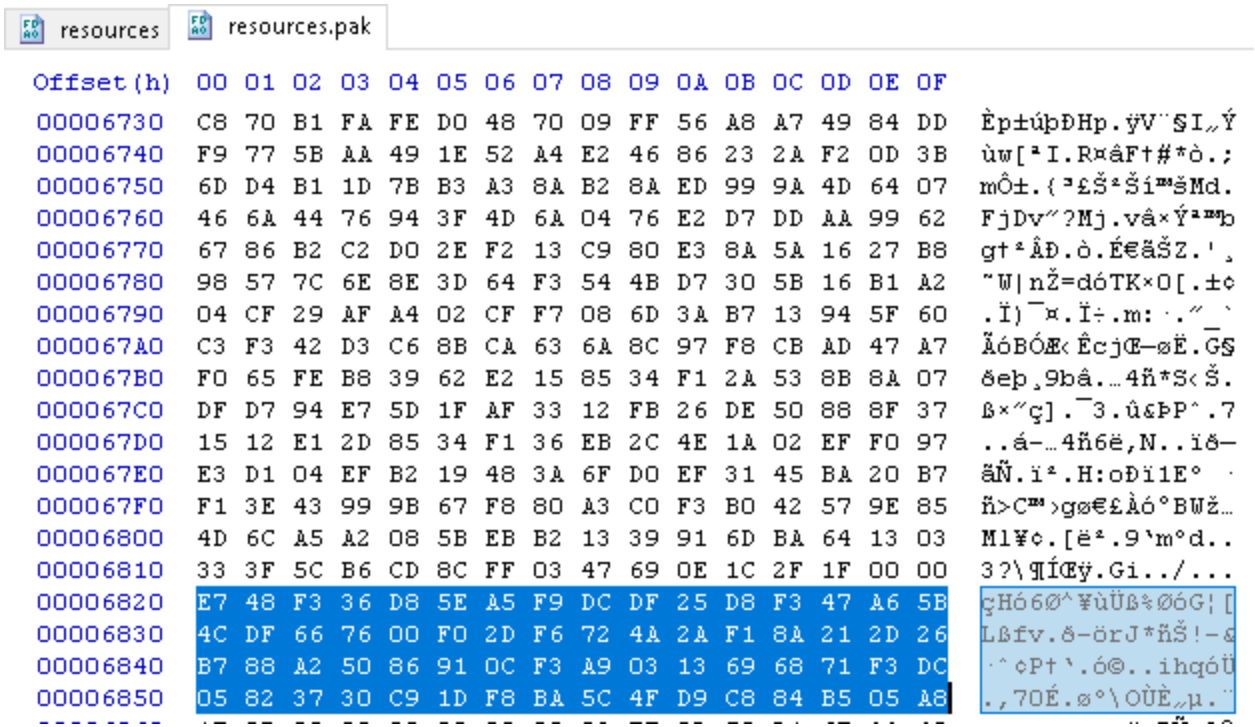
```
// chrome_pref_service_factory.cc:277
std::unique_ptr<ProfilePrefStoreManager> CreateProfilePrefStoreManager(
    const base::FilePath& profile_path) {
  std::string seed;
  CHECK(ui::ResourceBundle::HasSharedInstance());
#if BUILDFLAG(GOOGLE_CHROME_BRANDING)
  seed = std::string(ui::ResourceBundle::GetSharedInstance().GetRawDataResource(IDR_PREF_HASH_SEED_BIN));
#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[...]a8
```

- **Same HMAC-SHA256 key found on Chrome v130 !**



Extension loading mechanism

Preferences file > HMAC key recovery

SYNACKTIV

- **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

SYNACKTIV

Preferences file > Enabling Developper mode ui

- `developper_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

SYNACKTIV

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

SYNACKTIV

- 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

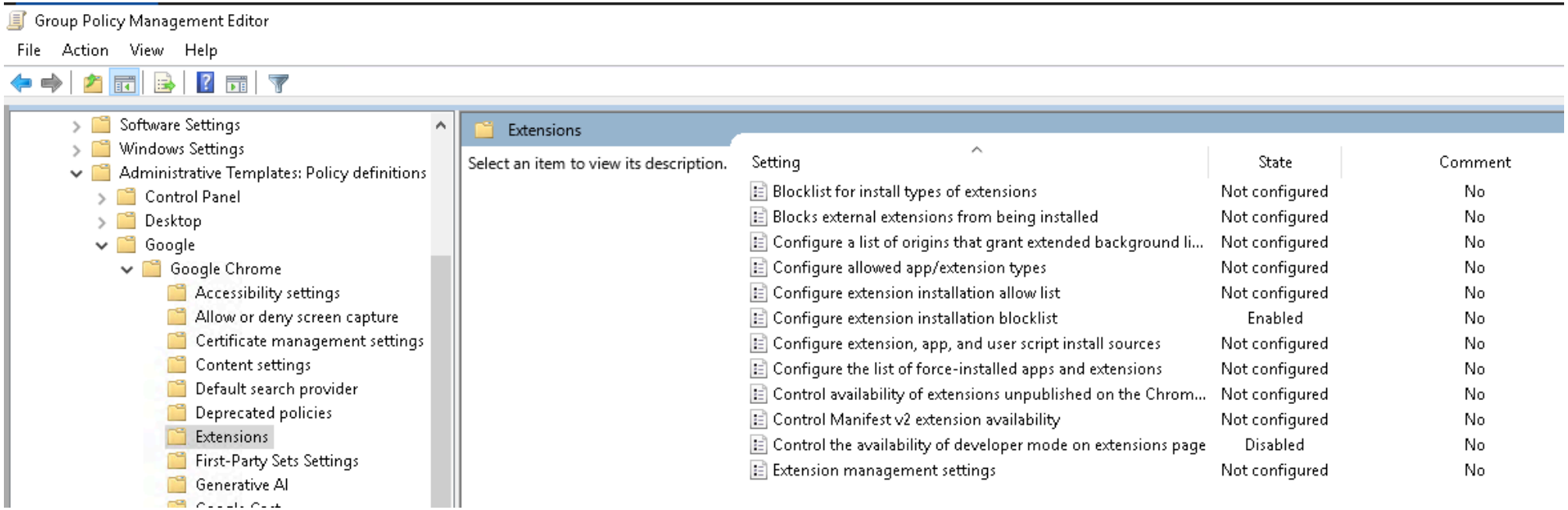
Administrative restrictions

Administrative restrictions

Group policy object

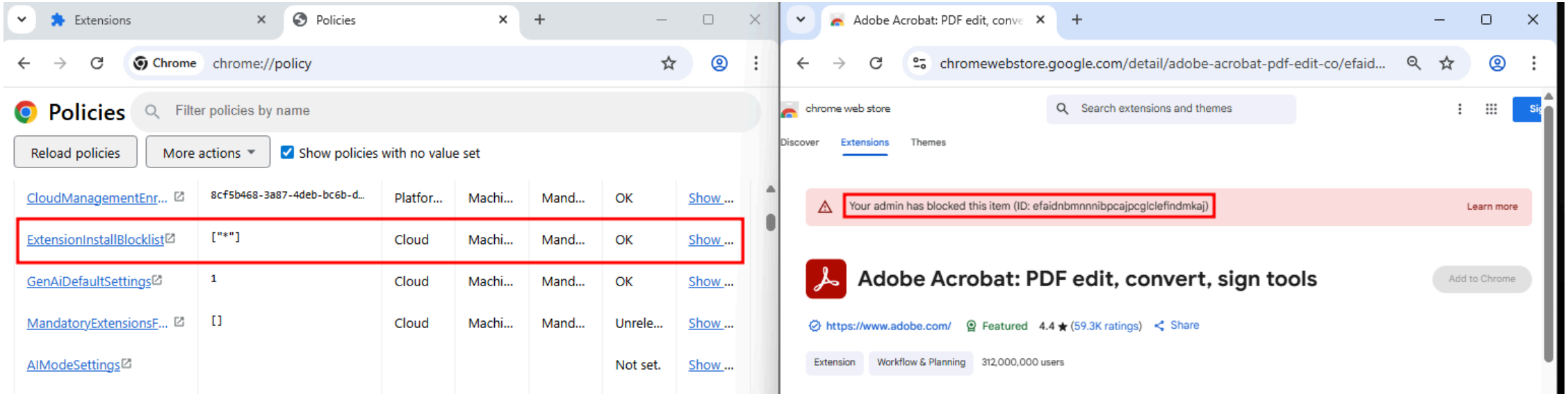
SYNACKTIV

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



Administrative restrictions

Group policy object



Administrative restrictions

Group policy object

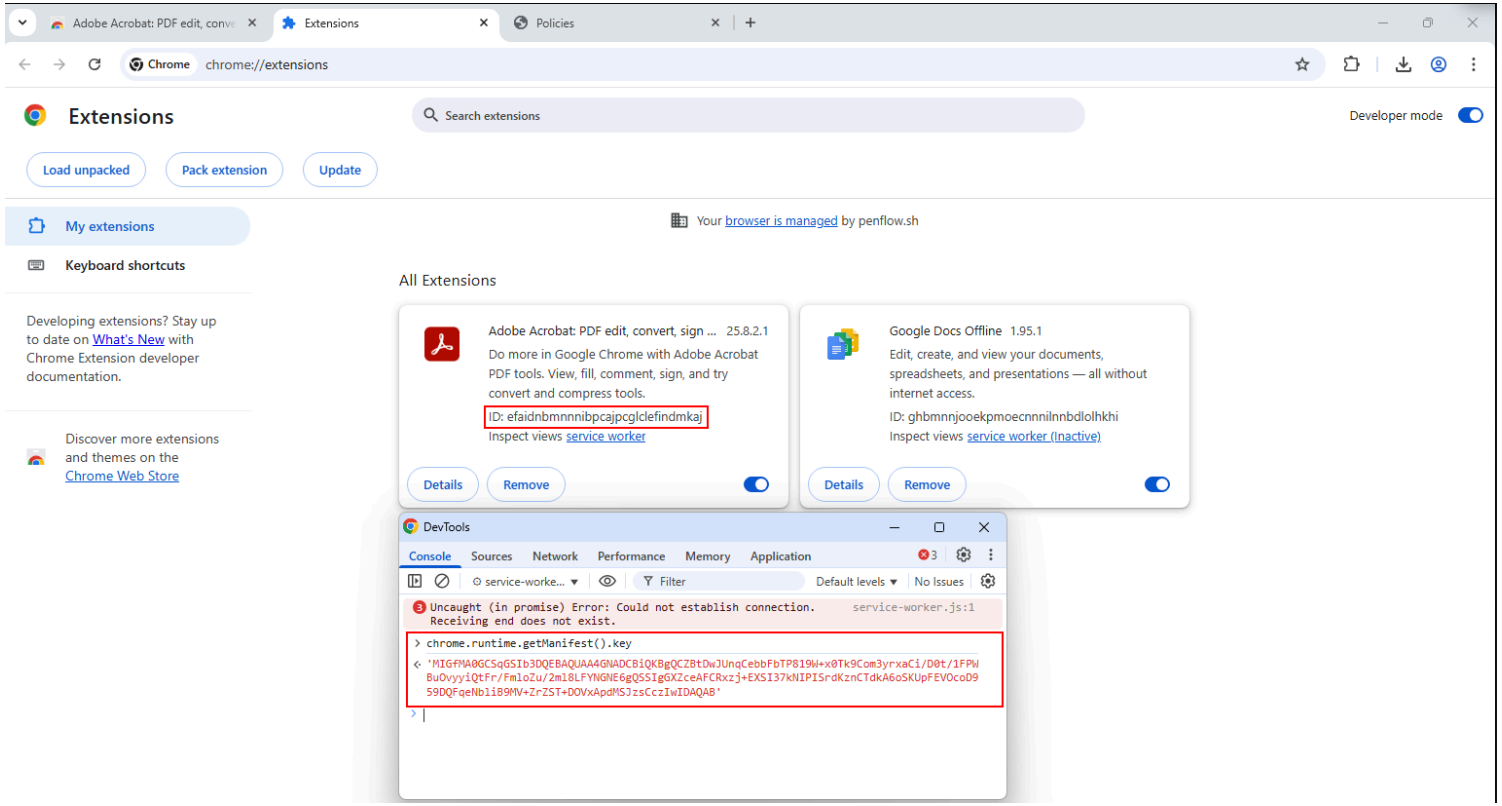
SYNACKTIV

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

Administrative restrictions

Extension id spoofing bypass

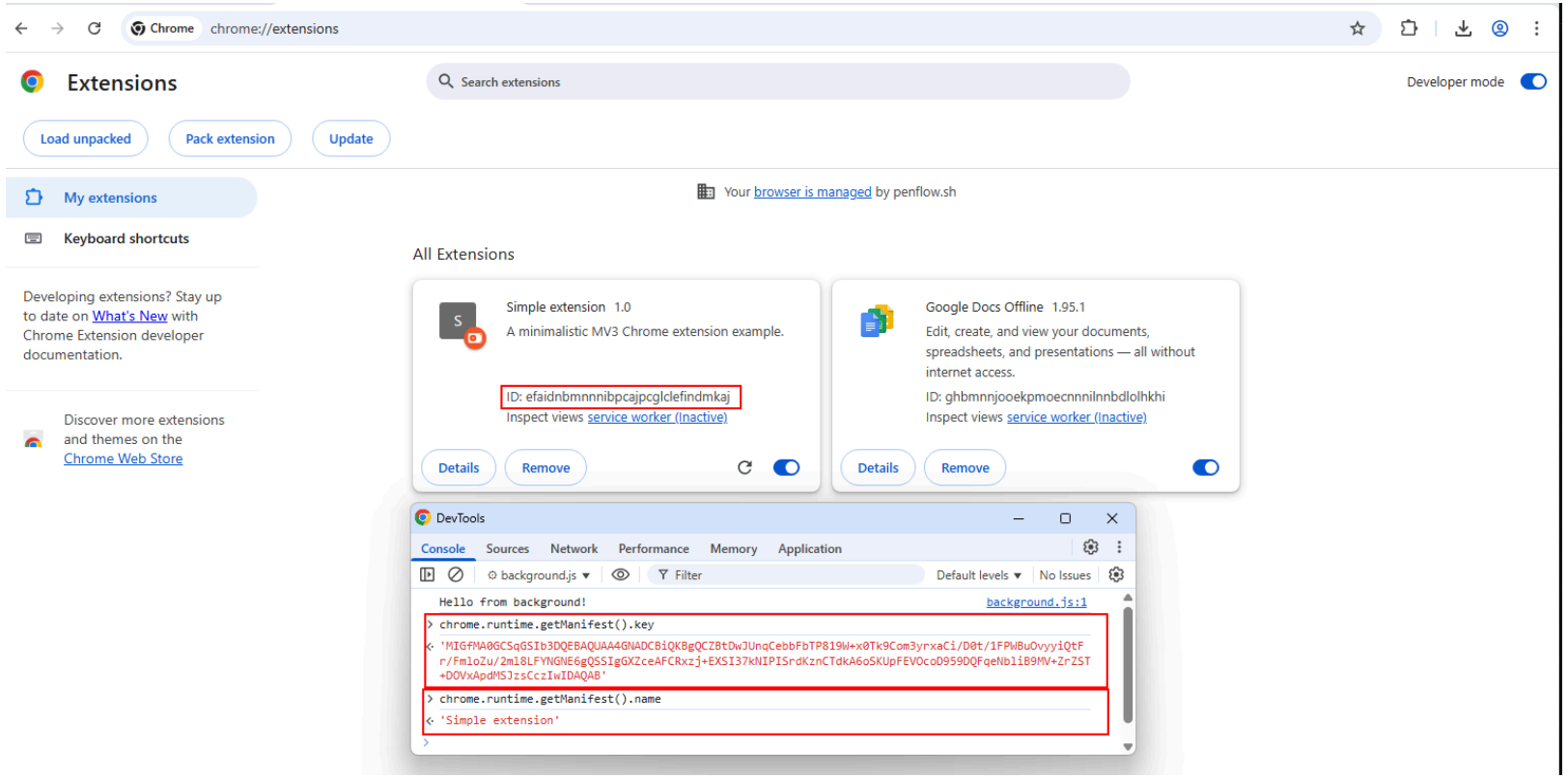
- Extension ID can be precaculated by defining a **public key**
- Since it's present in the manifest, what if we spoof a legitiamete extension id ?



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

SYNACKTIV

- **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

SYNACKTIV

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

SYNACKTIV

Admin user (Detection)

- With admin priviledges, 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

SYNACKTIV

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

References

SYNACKTIV

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

Thanks!

Questions?

SYNACKTIV