Assessing Browser Extension Effectiveness Against Spear Phishing Attacks with ZPhishing Tool in Kali Linux and Browser Exploitation via BeEF on Ubuntu VMWare

Eric Blancaflor[1], Ronald Bernardo[1], Alexandra Angeles[1], Nathaniel Calinisan[1], Bianca Eileen Oregas[1], Francis Andrei Santos[1]

[1]*School of Information Technology, Mapúa University, Philippines*

Abstract:

Introduction: Browser exploitation exploits vulnerabilities in web browsers, posing threats to user privacy and security. In response to these challenges, browser extensions have emerged as potential tools for strengthening defenses against such threats.

Objectives: This paper evaluates the efficacy of browser extensions in defending against spear phishing attacks and browser exploitation techniques targeting home users.

Methods: Utilizing the ZPhisher toolkit in Kali Linux for spear phishing simulations and the BeEF framework on Ubuntu VMWare for browser exploitation, the study assesses various browser extensions' performance in detecting and blocking phishing attempts and preventing exploitation.

Results: The research identifies SafeToOpen Online Security and Criminal IP: AI-Based Phishing Link Checker as effective in phishing detection, while NoScript proves successful in browser exploitation prevention. These extensions demonstrate proactive defense mechanisms, alerting users to threats and blocking malicious connections.

Conclusions: The evaluation of various extensions revealed notable effectiveness in mitigating these threats, with SafeToOpen Online Security emerging as a preferred option for phishing detection and NoScript for browser exploitation prevention. The simulated spear phishing attack and browser exploitation utilizing ZPhishing on Kali Linux and BeEF on Ubuntu VMWare demonstrated that these extensions offer proactive defense mechanisms. The researchers recommend educating users on browser security, ensuring regular extension updates, and integrating machine learning for enhanced threat detection. Browser extensions offer accessible and effective defenses against evolving cyber threats, safeguarding users' privacy and security in the digital realm.

Keywords: Phishing, ZPhisher, Browser Extensions, Phishing detection, Browser exploitation framework (BeEF)

INTRODUCTION

In today's technological era, many digital natives, particularly home users, are vulnerable to sophisticated phishing attacks and browser exploitation techniques. Spear Phishing poses a significant risk to individuals by targeting them with tailored deception tactics. Concurrently, browser exploitation exploits vulnerabilities in web browsers, posing threats to user privacy and security. In response to these challenges, browser extensions have emerged as potential tools for strengthening defenses against such threats. However, the efficacy of these extensions in mitigating risks remains to be determined, necessitating a comprehensive evaluation.

OBJECTIVES

This study's main objective is to assess the effectiveness of browser extensions in mitigating spear phishing attacks and browser exploitation for home users. The study specifically aims to address the following:

- Evaluate the performance of different types of browser extensions in detecting and blocking phishing attempts targeting home users, as well as analyze their capability in identifying and neutralizing browser exploitation attempts aimed at the same demographic.

- Identify the most effective browser extensions based on consistent and accurate phishing detection and exploitation prevention results.
- Provide recommendations for enhancing the usability and effectiveness of browser extensions as accessible mitigations against cyber threats for home users.

SCOPE OF THE STUDY

This study assesses the effectiveness of browser extensions in mitigating phishing attacks and browser exploitation without a preference for a particular browser. Various extensions, including Chrome extensions, Firefox add-ons, and others, will be considered based on their relevance and popularity among home users. The evaluation will prioritize the accuracy of phishing detection and exploitation prevention, focusing on usability for individuals with varying levels of technical expertise. The simulation attacks were conducted utilizing Kali Linux and Ubuntu virtual machines, leveraging the potent capabilities of the BeEF (Browser Exploitation Framework). Specifically, Kali Linux was employed for spear phishing attacks using Zphisher tool, while Ubuntu was utilized for browser exploitation via BeEF. However, the study's findings may be limited by extension availability, compatibility with different browsers, and evolving cyber threats. Additionally, while the study aims to propose recommendations for improving extension effectiveness, it acknowledges that extensions serve as mitigations rather than definitive solutions to the security issues of phishing attacks and browser exploitation.

LITERATURE REVIEW

Phishing Attacks

Phishing attacks pose a persistent threat to internet users and can have severe consequences. These attacks involve scammers using deceptive tactics, such as fake emails and websites, to trick people into revealing their personal information. Phishing is akin to fishing in a lake, but the goal is to catch personal data instead of fish. Phishing is a significant cyber threat that costs billions in damages annually, and it relies on a combination of social engineering and technology to exploit unsuspecting internet users [1]. Given the augmented probability of attacks stemming from the complex Windows environment, the study explores various attack vectors and exploitation methods within browser security research. Differing the vulnerabilities specific to different browser engines and design flaws. Furthermore, the study outlines different strategies for evaluating browser security and conducting research, utilizing multiple attack vectors while considering the latest protective measures and identified vulnerabilities [2]. The study delves into the exploitation of large language models (LLMs), notably GPT-3.5 and GPT-4, for spear phishing, a malicious activity aimed at coercing individuals into divulging sensitive information. Spear phishing involves highly targeted attacks, often personalized to deceive specific individuals or organizations. The research explores how LLMs can aid various stages of spear phishing, including reconnaissance and message generation. By crafting customized spear phishing messages for over 600 British Members of Parliament, the study showcases the potential scalability and cost-effectiveness of using LLMs for such nefarious purposes. The findings underscore the pressing need for robust defenses against spear phishing, particularly considering emerging AI-driven threats [3]. Agazzi (2020) explores the pervasive threat of phishing and spear phishing attacks in cyber espionage, which have accounted for over 91% of cyberattacks since 2012. It delineates the tactics attackers employ in five steps to maximize their success rates. Additionally, the research highlights four layers of protection against these social engineering attacks. The first two layers encompass automated and decision-aid tools, while the third emphasizes users' knowledge and expertise in identifying and mitigating threats. Lastly, the study stresses the significance of implementing multi-factor authentication as an external layer of defense, providing an additional barrier against phishing and spear phishing attempts [4].

Home users are individuals of varying ages and technological proficiency who utilize personal computing devices, smart home technologies, and internet-connected services for domestic purposes. They engage in activities such as online communication, entertainment streaming, financial transactions, and remote work or learning from their residential environments. Home users often interact with many digital platforms and devices, including smartphones, computers, smart TVs, home assistants, and IoT (Internet of Things) devices. Their cybersecurity posture plays a crucial role in protecting personal data, privacy, and digital assets from cyber threats such as malware, phishing attacks, ransomware, and unauthorized access. Therefore, effective cybersecurity measures for home users encompass awareness, education, and proactive defense strategies tailored to their diverse needs and technological literacy levels [5]. Douha et al. (2023) study analyzed the attitudes and preferences of adult smart-home users regarding cybersecurity awareness training and incentives for adopting secure practices. It reveals that cultural factors significantly influence users' willingness to engage in training and their views on its importance across different demographics. Moreover, users show a preference for nonfinancial incentives, indicating alternative approaches may be more effective in promoting cybersecurity behaviors. These insights can guide information security professionals in designing culturally sensitive training programs and assist governments in developing incentives to enhance cybersecurity adoption among smart-home users. Ultimately, the research underscores the necessity of considering cultural nuances in cybersecurity initiatives for smart-home users, fostering a safer digital environment [6].

Phishing Detection and Prevention

Nadeem M et al. (2023) explore the evolution of phishing tactics from basic methods to more advanced forms like spear phishing and assess their effects through real-world examples. User education, email filtering, multi-factor authentication, regular system updates, heuristic analysis, and SIEM systems are vital phishing detection and prevention strategies amidst evolving social engineering tactics and increasingly targeted attacks [7]. The Facebook Spam Detection Extension Tool is a browser extension designed to combat spam and phishing on Facebook by analyzing content and behavioral patterns. It aims to improve the user experience and security on the platform by identifying and filtering out spam and potential phishing attempts, which can involve deceptive messages and unusual behavior. This tool enhances safety for Facebook users by preventing device damage and protecting against phishing threats [8]. The study by Tang (2022) presents the design and development of a machine learning-based framework to detect phishing websites. Phishing, a type of cyber-attack, involves tricking users into divulging personal information by clicking on deceptive links sent via emails or social media messages. While machine learning has been increasingly used for detecting phishing links, existing approaches often need to be revised to include limitations such as reliance on outdated or poorly characterized datasets. To address this, the framework introduced in this thesis integrates multiple detection strategies, including whitelist, blacklist, heuristic rules, and machine learning models to enhance accuracy and flexibility. The study evaluates the performance of various machine learning models and concludes that the Gated Recurrent Units (GRU) model achieved the highest accuracy of 99.18%. Additionally, the framework incorporates expert-driven heuristic rule-based strategies with new HTML-based features. A prototype with a browser extension is developed to provide real-time detection results to users [9]. Ganal et al. (2023) introduce PhisherHunter, a tool designed for automatically detecting phishing websites and preventing user abuse. Phishing websites replicate real ones, tricking users into disclosing personal data. PhisherHunter employs four detection methods, achieving a successful detection rate of 95.4%, primarily through examining newly registered websites. In terms of active defense, the tool automatically identifies hosting companies to halt publication (98% success rate), employs an active honeypot technique to track information (92% success), and uses fake data to poison phishing websites (92% success). These methods aim to mitigate the threat posed by phishing attacks, enhancing online security [10].

Browser Exploitation Framework (BeEF)

A recent study from Fowdur et al. (2024) has discussed a novel browser extension employing Machine Learning (ML), notably Support Vector Machine (SVM), to effectively detect Cross-site scripting (XSS) attacks and various irregularities within recently installed extensions, with notable accuracies of 99.5% for malicious script detection, alongside the development of a Windows application in Java for real-time monitoring of suspicious network activities originating from these extensions [11]. Zonta et al. (2024) investigate the threat posed by malicious browser extensions and links, emphasizing their ability to compromise Internet security by accessing user data without consent and the challenges in combatting these extensions due to their stealthy behavior post-installation. It also evaluates various detection methods, including intrusion detection, machine learning, and deep learning techniques, to address the risks associated with malicious extensions. By stressing the importance of proactive detection in cybersecurity, the study provides insights for developing robust strategies to protect web browsers from evolving threats. Furthermore, it offers a detailed comparison of different detection approaches, which informs our research on assessing the effectiveness of browser extensions in countering phishing attacks and browser exploitation [12]. Malviya et al. (2021) developed a web browser prototype with a classification capability to counter Cross-Site Scripting (XSS) attacks. The browser addresses the absence of real-time XSS mitigation tools by employing machine learning techniques to classify web pages as malicious or non-malicious. The study's classification experiments demonstrate superior accuracy, precision, recall, and F1-score performance compared to alternative methods. Utilizing the open-source WebKit, the study implements the browser and evaluates the minimal overhead generated by the classification module during real-time browsing. This prototype offers a practical solution for researchers and end-users, enhancing browser security amidst the pervasive threat of XSS attacks and contributing to browser exploitation and mitigation techniques advancements [13]. Mimura and Yamasaki (2022) address the issue of cross-site scripting (XSS) attacks, emphasizing their prevalence and the often-overlooked client-side vulnerability aspect. While many investigations have focused on server-side vulnerability, evaluating client-side vulnerability is equally important. The proposed method utilizes the Browser Exploitation Framework (BeEF) to automate the audit process, providing effective client-side attack vectors. In leveraging the RESTful API, the method enables remote testing of client computers, proposing a comprehensive evaluation of the impact of XSS vulnerability beyond merely updating browsers and operating systems. The experimental results demonstrate the effectiveness of the proposed method in assessing client-side vulnerability against XSS attacks [14].

Ubuntu and Kali Linux VMWare

In Burgess and Sezer's (2023) study on browser and web-based threats, the choice of Ubuntu VMware reflects a deliberate effort to establish a secure and controlled research environment. Operating within Ubuntu's Linux-based ecosystem offers advantages such as robustness, flexibility, and access to numerous open-source security tools. This setup enables an in-depth analysis of browser exploitation techniques and vulnerabilities. Ubuntu's compatibility with various web browsers and security frameworks, the researchers developed custom experimental frameworks to simulate real-world attack scenarios and evaluate defense mechanisms effectively. Moreover, Ubuntu VMware likely facilitated ethical hacking and penetration testing, allowing researchers to identify and exploit vulnerabilities in web applications while ensuring data integrity for forensic analysis. It is essential to employ versatile and secure platforms like Ubuntu VMware in security research to address evolving cyber threats effectively [15]. In Softić et al. (2022)'s study examines the vulnerabilities of Windows 10 and its resilience against cyber-attacks. It utilizes CVE data and vulnerability reports to gauge the operating system's security performance. Metasploit and Nmap are employed for penetration testing and intrusion experiments within a simulated environment. Kali Linux serves as the platform for conducting these simulation attacks, focusing on various aspects such as information gathering, scanning, vulnerability selection, and launching attacks to gain access to the operating system. Despite installing the latest Windows 10 version, the study finds that complete protection against attacks is not guaranteed, accentuating the need for further research to identify vulnerabilities and recommend better security solutions [16].

METHOD

The study conducted a simulated attack on a secured virtual machine environment. The tools used are VMWare, Kali Linux, Ubuntu, ZPhisher, and BeEF (Browser Exploitation Framework). Figure 1 and 2 show the process of how the attack was simulated.

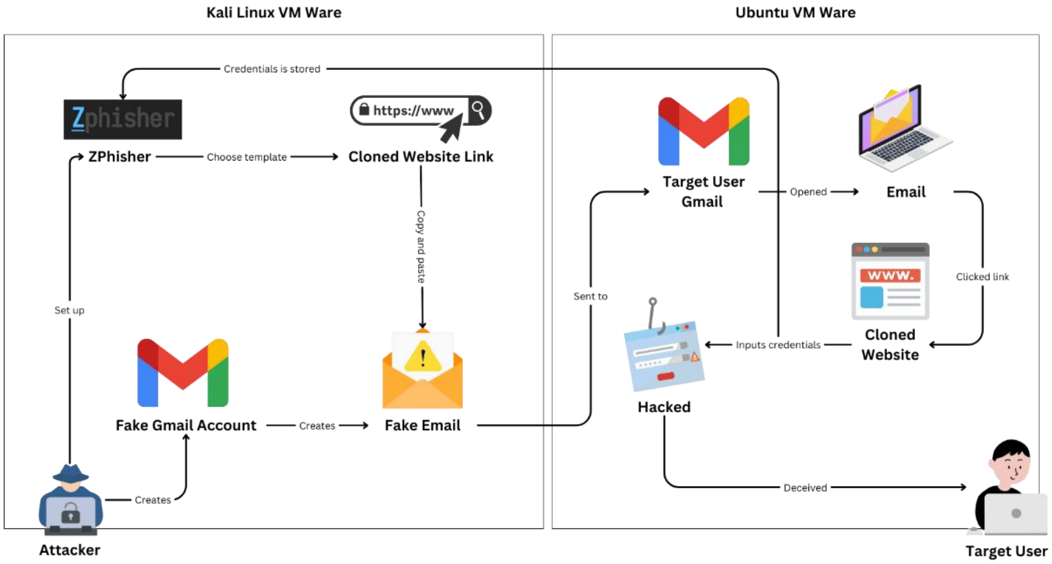


Figure 1. Phishing Attack Process

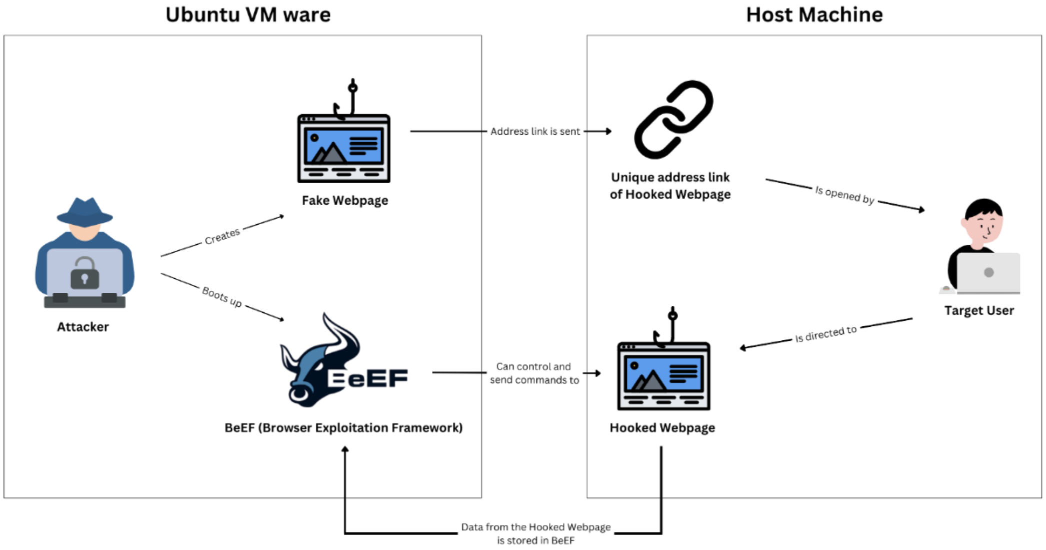
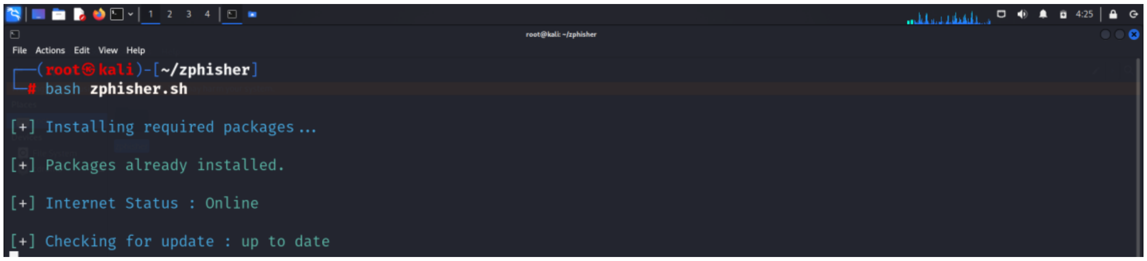

Ubuntu VM ware
Host Machine
Fake Webpage
Address link is sent
Unique address link of Hooked Webpage
Is opened by
Creates
Attacker
Boots up
Target User
is directed to
BeEF
BeEF (Browser Exploitation Framework)
Can control and send commands to
Hooked Webpage
Data from the Hooked Webpage is stored in BeEF

Figure 2. Browser Exploitation using BeEF Process

Phishing Attack Procedure

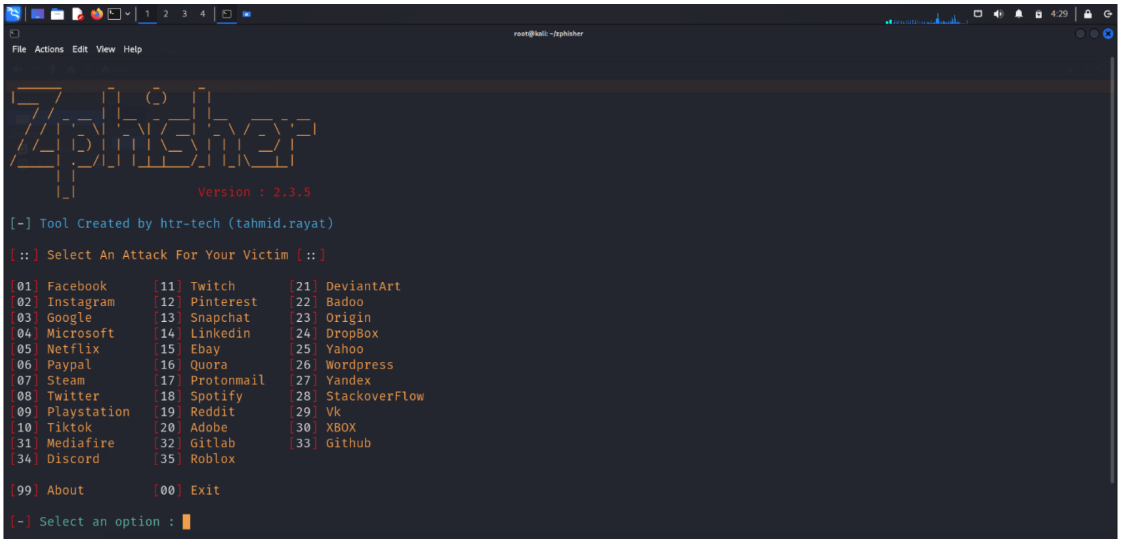
The first step in conducting the attack is to set up the VMWare Kali Linux. The tool that was used for the phishing attack is Zphisher. This tool was chosen by the researchers since it is easy to execute and is compatible with Kali Linux.

From the attacker's perspective, the starting point is to set up the phishing kit by cloning the Zphisher repository from GitHub to a Kali Linux computer and installing the necessary components listed in the Zphisher instructions. After Zphisher is configured, the researchers launch the Zphisher script, bash zphisher.sh, by navigating to the Zphisher directory (see Figure 3). Select the phishing template—such as a Netflix login page—by completing the on-screen instructions (see Figure 4). Begin the phishing attack by running Zphisher. It will produce a phishing link for the clone website and manipulate the target to visit the phishing page (see Figure 5). The researchers create a Gmail dummy account where the malicious link will be sent.



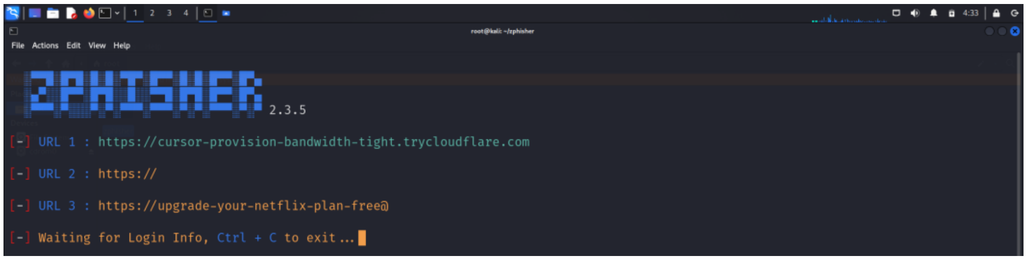
```
┌──(root㉿kali)-[~/zphisher]
└─# bash zphisher.sh

[+] Installing required packages...

[+] Packages already installed.

[+] Internet Status : Online

[+] Checking for update : up to date
```

Figure 3. Script to start Zphisher



```
                                          Version : 2.3.5

[-] Tool Created by htr-tech (tahmid.rayat)

[::] Select An Attack For Your Victim [::]

[01] Facebook      [11] Twitch       [21] DeviantArt
[02] Instagram     [12] Pinterest    [22] Badoo
[03] Google        [13] Snapchat     [23] Origin
[04] Microsoft     [14] Linkedin     [24] DropBox
[05] Netflix       [15] Ebay         [25] Yahoo
[06] Paypal        [16] Quora        [26] Wordpress
[07] Steam         [17] Protonmail   [27] Yandex
[08] Twitter       [18] Spotify      [28] StackoverFlow
[09] Playstation   [19] Reddit       [29] Vk
[10] Tiktok        [20] Adobe        [30] XBOX
[31] Mediafire     [32] Gitlab       [33] Github
[34] Discord       [35] Roblox

[99] About         [00] Exit

[-] Select an option :
```

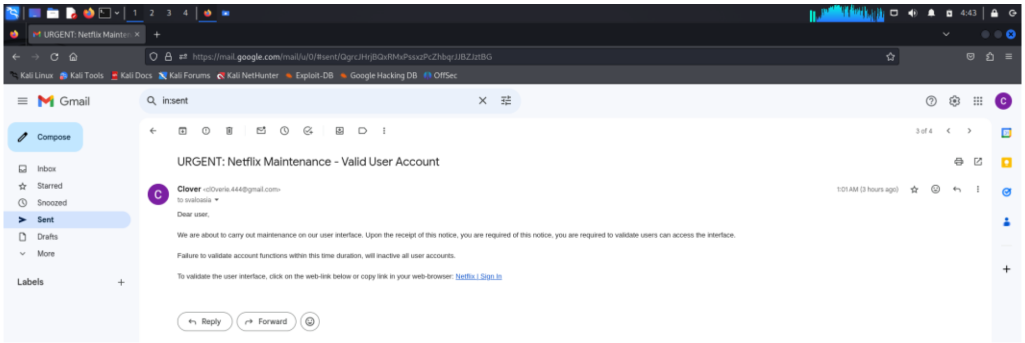
Figure 4. Zphisher Website Templates



```
ZPHISHER 2.3.5

[-] URL 1 : https://cursor-provision-bandwidth-tight.trycloudflare.com

[-] URL 2 : https://

[-] URL 3 : https://upgrade-your-netflix-plan-free@

[-] Waiting for Login Info, Ctrl + C to exit...
```

Figure 5. Zphisher Generated Malicious Link

The researchers set up a fake Gmail account to send the target phishing email. After creating the fake Gmail account, sign in and write a believable phishing email subject line and body (see Figure 6). Next, attach the Zphisher-generated phishing link to the email and send it to the intended recipient's email address.



https://mail.google.com/mail/u/0/#sent/QgrcJHrjBQxRMxPssxzPcZhbqrJJBZJztBG

URGENT: Netflix Maintenance - Valid User Account

Clover <cl0verie.444@gmail.com>
to svaloasia
1:01 AM (3 hours ago)

Dear user,

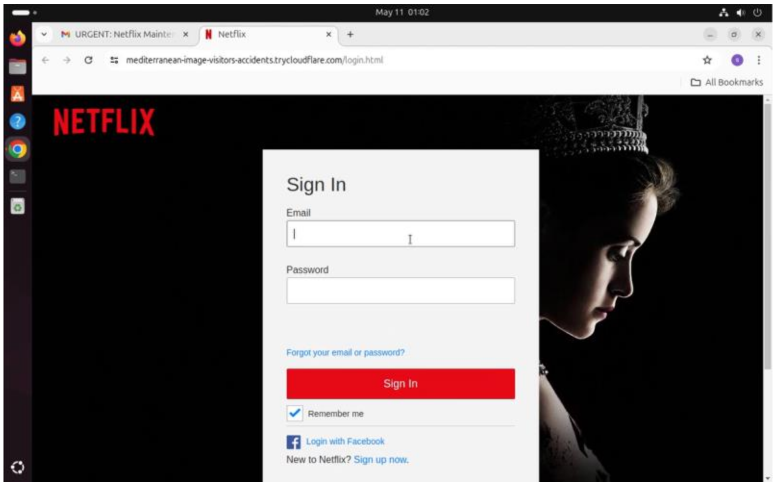
We are about to carry out maintenance on our user interface. Upon the receipt of this notice, you are required of this notice, you are required to validate users can access the interface.

Failure to validate account functions within this time duration, will inactive all user accounts.

To validate the user interface, click on the web-link below or copy link in your web-browser: Netflix | Sign In

Figure 6. Fake Email for Phishing Attack

The victim will receive the phishing email in their inbox. Upon receiving the email, they assess the subject line and message to determine if it looks legitimate. If the email appears convincing, they may click on the malicious link provided in the email. Clicking on the link redirects them to the phishing page created by the attacker using Zphisher (see Figure 7). Believing the phishing page to be legitimate, they will enter their login credentials, such as their Gmail username and password.



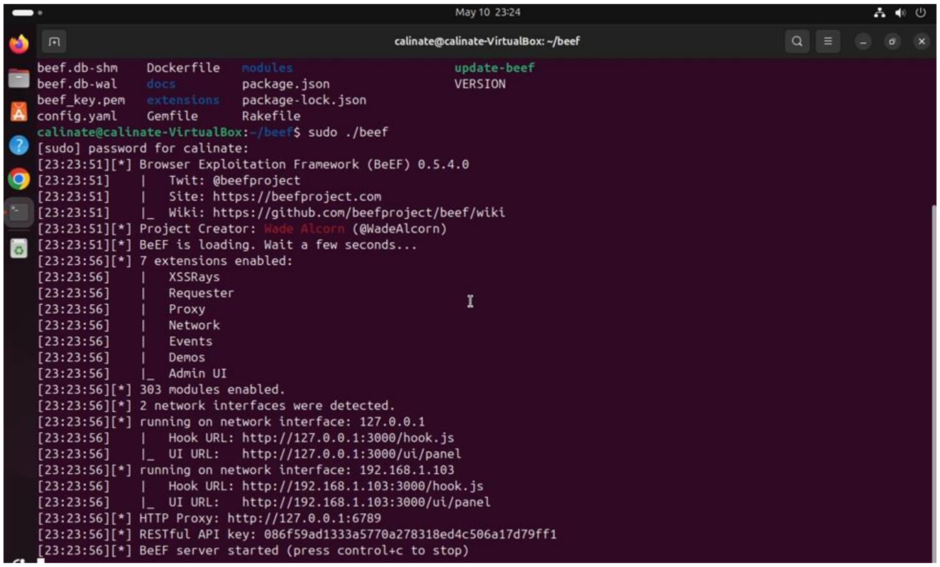
mediterranean-image-visitors-accidents.trycloudflare.com/login.html

Figure 7. Fake Website

The target had no idea that the phishing page stores their login details. Users can be sent to a genuine website after inputting their credentials. The attacker can access the compromised credentials from the Zphisher logs, and they can be used maliciously. This simulation highlights browser extensions' significance in mitigating such attacks by showing how simple it is to carry out a phishing attack using Zphisher.

Browser Exploitation Procedure

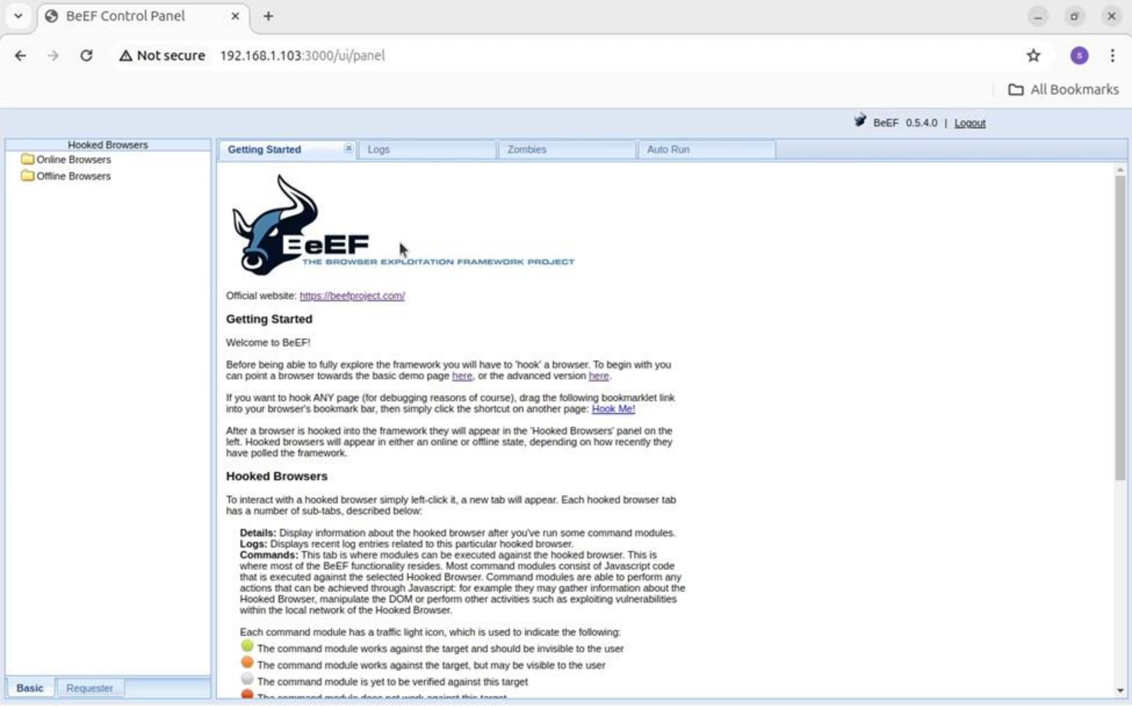
The first step in the browser exploitation attack was conducted in a controlled environment, which is the Ubuntu Virtual Machine. After installing the BeEF, Browser Exploitation Framework, the attacker had to set up their hooked webpage. When everything has been set up accordingly, the attacker is now ready to proceed with the attack. As you can see in Figure 9, the attacker has opened the terminal to start the BeEF, by entering “sudo ./beef”.



```
beef.db-shm    Dockerfile   modules                     update-beef
beef.db-wal    docs         package.json                VERSION
beef_key.pem   extensions   package-lock.json
config.yaml    Gemfile      Rakefile
calinate@calinate-VirtualBox:~/beef$ sudo ./beef
[sudo] password for calinate:
[23:23:51][*] Browser Exploitation Framework (BeEF) 0.5.4.0
[23:23:51]    |   Twit: @beefproject
[23:23:51]    |   Site: https://beefproject.com
[23:23:51]    |_  Wiki: https://github.com/beefproject/beef/wiki
[23:23:51][*] Project Creator: Wade Alcorn (@WadeAlcorn)
[23:23:51][*] BeEF is loading. Wait a few seconds...
[23:23:56][*] 7 extensions enabled:
[23:23:56]    |   XSSRays
[23:23:56]    |   Requester
[23:23:56]    |   Proxy
[23:23:56]    |   Network
[23:23:56]    |   Events
[23:23:56]    |   Demos
[23:23:56]    |_  Admin UI
[23:23:56][*] 303 modules enabled.
[23:23:56][*] 2 network interfaces were detected.
[23:23:56][*] running on network interface: 127.0.0.1
[23:23:56]    |   Hook URL: http://127.0.0.1:3000/hook.js
[23:23:56]    |_  UI URL:   http://127.0.0.1:3000/ui/panel
[23:23:56][*] running on network interface: 192.168.1.103
[23:23:56]    |   Hook URL: http://192.168.1.103:3000/hook.js
[23:23:56]    |_  UI URL:   http://192.168.1.103:3000/ui/panel
[23:23:56][*] HTTP Proxy: http://127.0.0.1:6789
[23:23:56][*] RESTful API key: 086f59ad1333a5770a278318ed4c506a17d79ff1
[23:23:56][*] BeEF server started (press control+c to stop)
```

Figure 8. Start BeEF

As you can see in Figure 8, the next step starts with the attacker opening the UI URL link to access the control panel of BeEF. However, in order to access the control panel of the BeEF, the attacker needs to authenticate themselves in the login page of the BeEF panel. The login credentials can be found within the contents of the "config.yaml". After entering the correct credentials, the webpage displays the dashboard of the control panel of BeEF. Figure 6 shows the dashboard of the control panel of BeEF. As shown in Figure 9, please pay close attention to the hooked Browsers, located in the left side of the interface. This indicates whether a hooked webpage was visited by a browser. As you can see in Figure 10, there are currently no hooked browsers or webpages that have been visited.



Official website: https://beefproject.com/

Getting Started

Welcome to BeEF!

Before being able to fully explore the framework you will have to 'hook' a browser. To begin with you can point a browser towards the basic demo page here, or the advanced version here.

If you want to hook ANY page (for debugging reasons of course), drag the following bookmarklet link into your browser's bookmark bar, then simply click the shortcut on another page: Hook Me!

After a browser is hooked into the framework they will appear in the 'Hooked Browsers' panel on the left. Hooked browsers will appear in either an online or offline state, depending on how recently they have polled the framework.

Hooked Browsers

To interact with a hooked browser simply left-click it, a new tab will appear. Each hooked browser tab has a number of sub-tabs, described below:

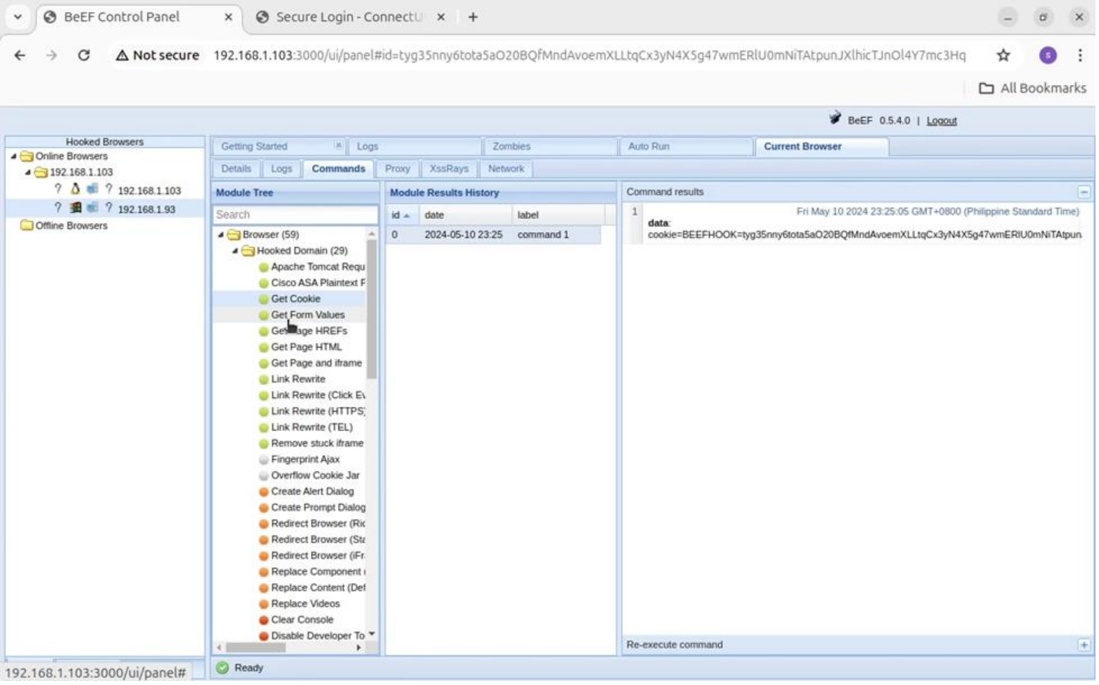
Details: Display information about the hooked browser after you've run some command modules.
Logs: Displays recent log entries related to this particular hooked browser.
Commands: This tab is where modules can be executed against the hooked browser. This is where most of the BeEF functionality resides. Most command modules consist of Javascript code that is executed against the selected Hooked Browser. Command modules are able to perform any actions that can be achieved through Javascript: for example they may gather information about the Hooked Browser, manipulate the DOM or perform other activities such as exploiting vulnerabilities within the local network of the Hooked Browser.

Each command module has a traffic light icon, which is used to indicate the following:

● The command module works against the target and should be invisible to the user
● The command module works against the target, but may be visible to the user
○ The command module is yet to be verified against this target
● The command module does not work against this target

Figure 9. BeEF Interface

The next step is to trick the target to open the hooked webpage and exploit their browser. Once the target user opens the hooked webpage in their browser, it will reflect in the BeEF Hooked Browsers tab. After this step, the attacker may click on the target's browser in the Hooked Browsers tab to execute commands that will exploit or manipulate the target's browser. The list of commands for the selected Hooked Browsers is as seen in Figure 10. Some of the commands that are available to execute by the attacker to the target are Get Cookie, Get data from textfield, redirect webpage, create alert dialogs, detect extensions and popup blocker, and many more. By following all these steps, the attacker has and can successfully attack and exploit the target user's browser.

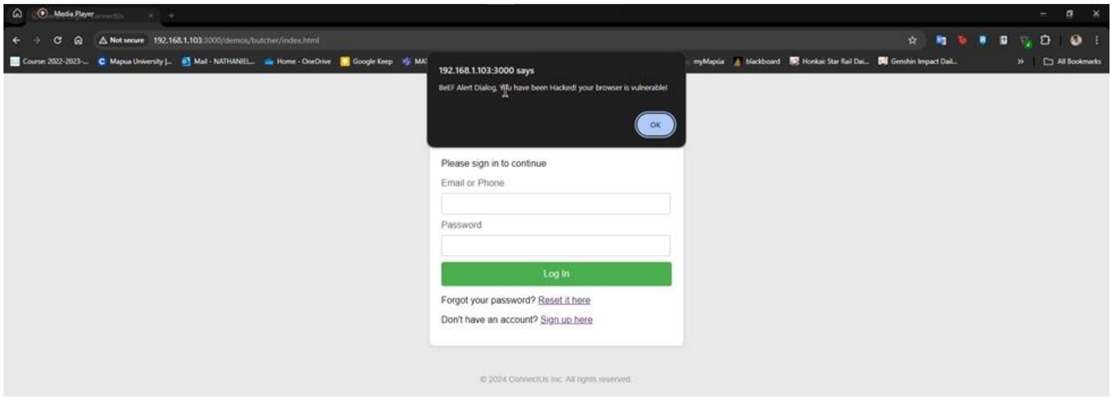


id	date	label
0	2024-05-10 23:25	command 1

```
1                    Fri May 10 2024 23:25:05 GMT+0800 (Philippine Standard Time)
data:
cookie=BEEFHOOK=tyg35nny6tota5aO20BQfMndAvoemXLLtqCx3yN4X5g47wmERlU0mNiTAtpun
```

Figure 10. Commands Tab of Selected Hooked Browser

As for the viewpoint of the target user, figure 11 shows the sample webpage that the attacker created to fool the target user. You can also see in Figure 8 an alert dialog box which was executed by the attacker.



192.168.1.103:3000 says

BeEF Alert Dialog, You have been Hacked! your browser is vulnerable!

Please sign in to continue

Email or Phone

Password

Forgot your password? Reset it here

Don't have an account? Sign up here

© 2024 ConnectUs Inc. All rights reserved.

Figure 11. Victim's Viewpoint

RESULTS AND DISCUSSION

In the defense against simulated spear phishing attacks and browser exploitation, the study identified three browser extensions that showed effectiveness in mitigating these threats. Among these extensions, two were found to be effective against phishing attacks, while one was effective for browser exploitation prevention. Figures 12 show how the extensions will be used to prevent the spear phishing attacks and browser exploitation from luring victims.

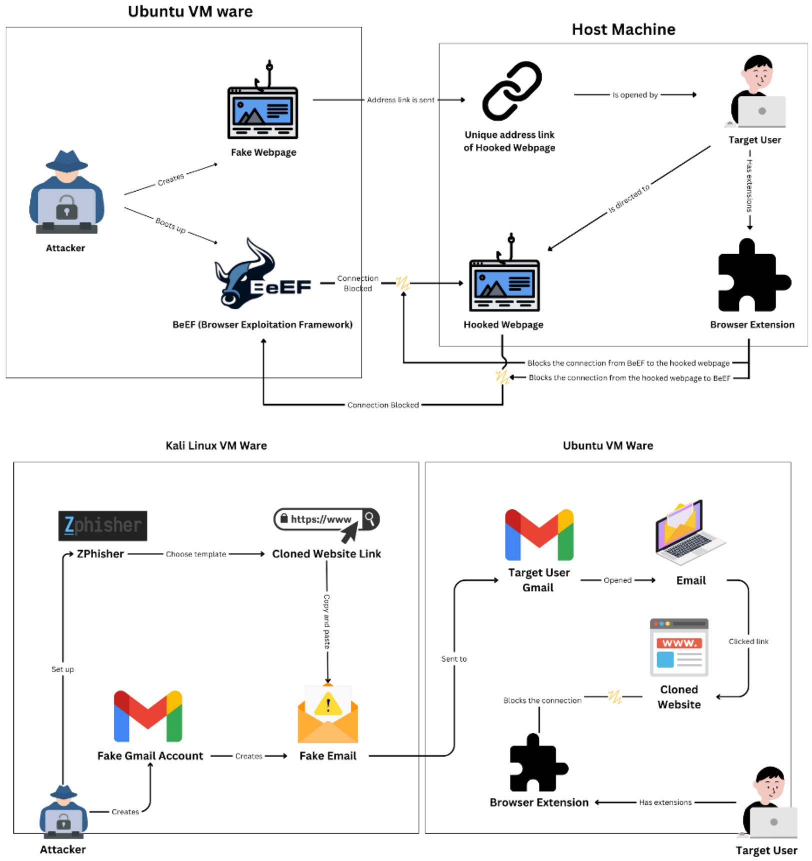
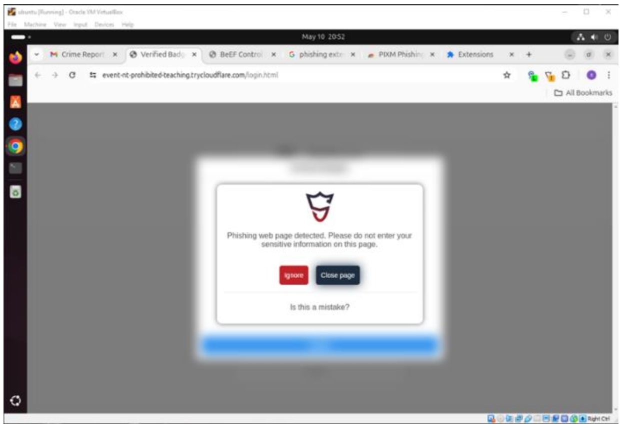


Figure 12. Phishing Attack Mitigation Flow > Browser Exploitation Mitigation Flow

BROWSER EXTENSIONS FOR PHISHING AND BROWSER EXPLOITATION

SafeToOpen Online Security

The first extension, "SafeToOpen Online Security," demonstrated robust performance in detecting phishing sites. Upon encountering a suspicious website, the extension generates an alert popup, informing the user that the site is potentially malicious. The alert popup message can be seen in Figure 13. The user is then presented with the option to ignore the warning or close the page, thereby providing a proactive defense mechanism against phishing attempts.

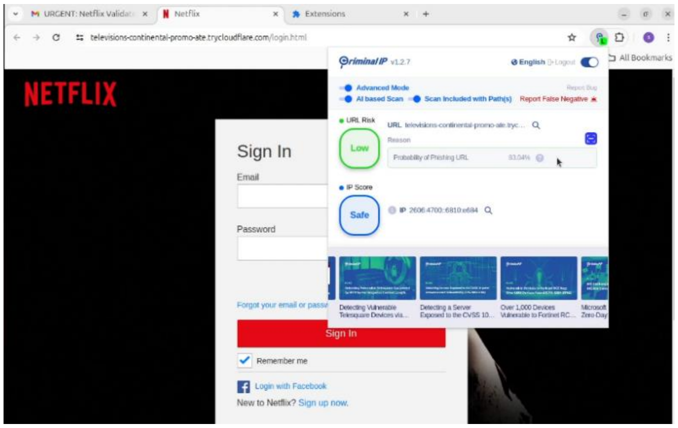


Phishing web page detected. Please do not enter your sensitive information on this page.

Figure 13. SafeToOpen Online Security Extension Popup Alert

Criminal IP: AI Based Phishing Link Checker

The second extension, "Criminal IP: AI Based Phishing Link Checker," also exhibited functionality in identifying potential phishing sites. However, unlike the SafeToOpen extension, Criminal IP does not block access to the website outright. Instead, it provides users with a percentage or probability indicating the likelihood that the site is a phishing site, allowing users to make informed decisions regarding site trustworthiness. Figure 14 displays an example of Criminal IP extension.



URL: televisions-continental-promo-ate.tryc...

Probability of Phishing URL 93.04%

IP 2606:4700::6810:e684

URGENT: Netflix Validat
Netflix
Extensions
televisions-continental-promo-ate.trycloudflare.com/login.html
All Bookmarks
Criminal IP v1.2.7
English
Logout
Advanced Mode
Report Bug
AI based Scan
Scan Included with Path(s)
Report False Negative
URL Risk
Low
Reason
IP Score
Safe
Detecting Vulnerable Telesquare Devices via...
Detecting a Server Exposed to the CVSS 10...
Over 1,000 Devices Vulnerable to Fortinet RC...
Microsoft Zero-Day
NETFLIX
Sign In
Email
Password
Forgot your email or passw
Sign In
Remember me
Login with Facebook
New to Netflix? Sign up now.

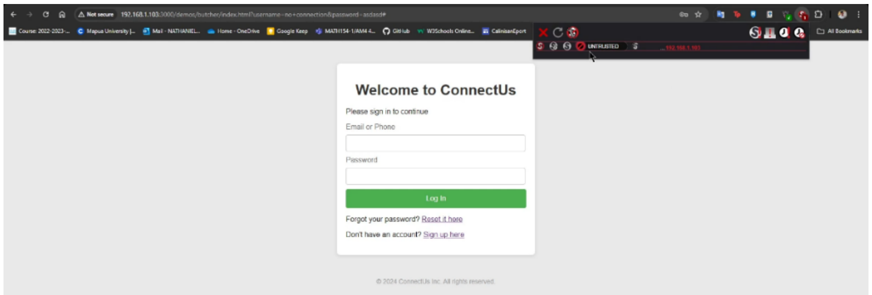
Figure 14. Criminal IP: AI Based Phishing Link Checker Extension

While both extensions offer phishing detection capabilities, the study recommends "SafeToOpen Online Security" as the preferred option due to its proactive approach in alerting users to potential threats and providing options for action.

NoScript

For browser exploitation prevention, the study identified the "NoScript" extension as an effective defense measure, which can be seen in Figure 17. NoScript operates by blocking connections between the hooked webpage and the BeEF framework, thereby preventing the execution of commands and manipulation of the target browser. Consequently, interactions with the hooked webpage are rendered ineffective, as commands from the BeEF framework fail to register without a connection established, as seen in Figure 15.

The registered Hooked Browser is a browser from the same machine (Ubuntu VM) which didn't enable any extensions. Notice that there is no Hooked Browser registered in a Windows Machine, this proves that the extension proved successful.



Welcome to ConnectUs

Please sign in to continue

Email or Phone

Password

Forgot your password? Reset it here

Don't have an account? Sign up here

© 2024 ConnectUs Inc. All rights reserved.

Figure 15. NoScript Extension

The effectiveness of NoScript in thwarting browser exploitation attempts underscores its utility as a defensive tool against sophisticated attack vectors. By preemptively blocking malicious connections, NoScript enhances browser security and mitigates the risk of exploitation.

CONCLUSION AND RECOMMENDATION

In conclusion, browser extensions play a significant role in defending against spear phishing attacks and browser exploitation. The evaluation of various extensions revealed notable effectiveness in mitigating these threats, with SafeToOpen Online Security emerging as a preferred option for phishing detection and NoScript for browser exploitation prevention. The simulated spear phishing attack and browser exploitation utilizing ZPhishing on Kali Linux and BeEF on Ubuntu VMWare demonstrated that these extensions offer proactive defense mechanisms. SafeToOpen Online Security promptly alerts users to potential phishing sites, empowering them to take necessary actions to protect their information. On the other hand, NoScript effectively blocks malicious connections, thereby thwarting browser exploitation attempts.

Overall, based on the research's findings, the following recommendations are proposed to enhance the usability and effectiveness of browser extensions in countering cyber threats for home users such as by educating users about the importance of browser security and the role of extensions in protecting against threats can empower them to make informed decisions while browsing. Developers should prioritize regular updates and improvements to ensure that browser extensions remain effective against evolving cyber threats. Incorporating machine learning algorithms into browser extensions can enhance their ability to detect and mitigate emerging threats by analyzing real-time patterns and behaviors. Ensuring compatibility with various browsers will expand the reach and accessibility of browser extensions, allowing more users to benefit from enhanced security measures. Encouraging user feedback and collaboration within the developer community can facilitate the identification of vulnerabilities and the implementation of robust security measures in browser extensions. Browser extensions can serve as accessible and effective mitigations against the ever-growing spectrum of cyber threats, safeguarding users' privacy and security in the digital landscape.

REFERENCES

[1] A Syed Mustafa, Baby Pn, N Divyashree, Iffath Fathima, and Javeria Fathima. 2024. A Comprehensive Review of Phishing Attacks Techniques, Types and Solutions. Journal of Hacking Techniques, Digital Crime Prevention and Computer Virology 1, 1 (April 2024), 15–24. https://doi.org/10.46610/johtdcpcv.2024.v01i01.002

[2] Stefan Sabin Nicula and Răzvan-Daniel Zota. 2022. An analysis of different browser attacks and exploitation techniques. In Smart innovation, systems and technologies. 31–41. https://doi.org/10.1007/978-981-16-8866-9_3

[3] Julian Hazell. 2023. Spear phishing with large language models. arXiv (Cornell University) (January 2023). https://doi.org/10.48550/arxiv.2305.06972

[4] Alessandro Ecclesie Agazzi. 2020. Phishing and Spear Phishing: examples in Cyber Espionage and techniques to

protect against them. arXiv (Cornell University) (January 2020). https://doi.org/10.48550/arxiv.2006.00577

[5] Elmarie Kritzinger and Sebastiaan H. Von Solms. 2010. Cyber security for home users: A new way of protection through awareness enforcement. Computers & Security 29, 8 (November 2010), 840–847. https://doi.org/10.1016/j.cose.2010.08.001

[6] N'guessan Yves-Roland Douha, Karen Renaud, Yoshiyuki Taenaka, and Youki Kadobayashi. 2023. Smart home cybersecurity awareness and behavioral incentives. Information & Computer Security/Information and Computer Security 31, 5 (June 2023), 545–575. https://doi.org/10.1108/ics-03-2023-0032

[7] Muhammad Nadeem, Syeda Wajiha Zahra, Muhammad Nouman Abbasi, Ali Arshad, Saman Riaz, and Waqas Ahmed. 2023. Phishing Attack, Its Detections and Prevention Techniques. International Journal of Wireless Information Networks 12, (October 2023), 12–25. https://doi.org/10.37591/ijwsn

[8] Luqman Khir Azman and Nurul Azma Abdullah. 2023. Facebook Spam Filtering Too lusing Keyword-Based Technique. Applied Information Technology and Computer Science 4, 2 (November 2023), 46–65. https://doi.org/10.30880/aitcs.2023.04.02.004

[9] Qusay H. Mahmoud. 2022. Design and development of a machine learning-based framework for phishing website detection. Retrieved from https://ir.library.ontariotechu.ca/handle/10155/1446

[10] Samet Ganal, Ecir Küçüksille, and Mehmet Ali Yalçınkaya. 2023. PhisherHunter: Module design for automatic detection of phishing websites and preventing user abuse. Retrieved from https://dergipark.org.tr/en/pub/pajes/issue/80772/1383412

[11] Tulsi Pawan Fowdur and Shuaïb Hosenally. 2022. A real-time machine learning application for browser extension security monitoring. Information Security Journal 33, 1 (October 2022), 16–41. https://doi.org/10.1080/19393555.2022.2128944

[12] Rama Abirami K, Tiago Zonta, and Mithileysh Sathiyanarayanan. 2024. A Holistic Review on Detection of Malicious Browser Extensions and Links using Deep Learning. 2024 IEEE 3rd International Conference on AI in Cybersecurity (ICAIC) (February 2024). https://doi.org/10.1109/icaic60265.2024.10433842

[13] Vikas Kumar Malviya, Sawan Rai, and Atul Gupta. 2021. Development of web browser prototype with embedded classification capability for mitigating Cross-Site Scripting attacks. Applied Soft Computing 102, (April 2021), 106873. https://doi.org/10.1016/j.asoc.2020.106873

[14] Mamoru Mimura and Takumi Yamasaki. 2021. Toward automated audit of Client-Side vulnerability against Cross-Site scripting. In Lecture notes in networks and systems. 148–157. https://doi.org/10.1007/978-3-030-90072-4_15

[15] Burgess, J. and Sezer, S., 2023. Investigating browser and web-based threats (Doctoral dissertation, Queen's University Belfast). Retrieved from: https://pureadmin.qub.ac.uk/ws/portalfiles/portal/456488475/Investigation_of_Browser_and_Web_based_Threats.pdf

[16] Jasmin Softić and Zanin Vejzović. 2022. Windows 10 Operating System: Vulnerability Assessment and Exploitation. 2022 21st International Symposium INFOTEH-JAHORINA (INFOTEH) (March 2022). https://doi.org/10.1109/infoteh53737.2022.9751274