

B.M.S COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU



A Technical Seminar Report based on Technical Activity

Web Browser Security:
Different Attacks Detection and Prevention Techniques

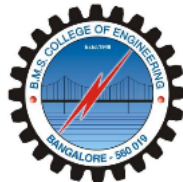
Submitted in partial fulfilment for the award of degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:
SUSHANTH

1BM21CS227

Work carried out at



Internal Guide

Lohith J J
Assistant Professor
BMS College of Engineering

Department of Computer Science and Engineering
BMS College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
2022-2023

B.M.S COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



DECLARATION

I, SUSHANTH (1BM21CS227) student of 4th Semester, B.E, Department of Computer Science and Engineering, BMS College of Engineering, Bangalore, hereby declare that, this technical seminar entitled " **Web Browser Security: Different Attacks Detection and Prevention Techniques**" has been carried out under the guidance of **Lohith J J**, Assistant Professor ,Department of CSE, B.M.S College of Engineering, Bangalore during the academic semester March-July 2023. I also declare that to the best of my knowledge and belief, the technical seminar report is not from part of any other report by any other students.

Signature of the Candidate

SUSHANTH (1BM21CS227)

B.M.S COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the Technical Seminar titled “**Web Browser Security: Different Attacks Detection and Prevention Techniques**” has been carried out by SUSHANTH (1BM21CS227) during the academic year 2022-2023.

Signature of the Guide

Signature of the Head of the Department

Signature of Examiners with date

1. Internal Examiner

2. External Examiner

Abstract

Web browsers play a pivotal role in our daily lives, facilitating access to the vast landscape of the internet. However, this convenience comes with inherent security risks, as cybercriminals continuously devise innovative ways to compromise user data and privacy. This paper delves into the realm of web browser security, shedding light on a multitude of potential attacks and vulnerabilities that plague the online ecosystem.

In this paper, we undertake a methodical investigation aimed at enhancing the security of web browsers. These browsers remain susceptible to a spectrum of exploitative attacks stemming from weaknesses within the webpage's user interface, browser cache memory, extensions, and plug-ins. The potential for malevolent actors to harness these vulnerabilities through malicious JavaScript to compromise user systems is a concerning reality. The study comprehensively covers diverse attack vectors, including but not limited to buffer overflow, cross-site scripting, man-in-the-middle, extension vulnerabilities, advanced phishing, browser cache manipulation, session hijacking, drive-by downloads, and click-jacking.

The primary focus is on illuminating the various attack vectors, including but not limited to Cross-Site Scripting (XSS) attacks, phishing attempts, SQL injections, and Man-in-the-Middle (MitM) attacks. Through an in-depth analysis of these threats, we aim to raise awareness among users about the ever-present dangers they face while navigating the web.

Furthermore, this paper explores cutting-edge preventive strategies that organizations and individuals can employ to fortify their defences against these relentless cyber threats. In today's digital age, safeguarding sensitive data and maintaining online privacy have become paramount concerns. Thus, understanding the intricacies of web browser security and adopting proactive measures is imperative for a safer online experience.

Table of Contents

1.INTRODUCTION	2
1.1 Overview	2
1.2 Motivation	3
1.3 Objective	3
2. LITERATURE SURVEY	4
2.1 Role of The Web	4
2.2 The Web Browser.....	4
2.3 Web Extensions	5
2.4 Electrolysis and Sandboxing Elecrolsis:	5
3. METHODOLOGY/TECHNIQUES OR ALGORITHM USED	6
3.1 Attacks On Browser.....	6
3.1.1 Buffer Overflow:	6
3.1.2 Cross-Site Scripting:	7
3.1.3 Man-in-the-Middle:	7
3.1.4 Extension vulnerability:	7
3.1.5 Extreme Phishing:	8
3.1.5 Browser Cache Poisoning:.....	8
3.1.6 Session Hijacking:	9
3.1.7 Drive-by-Download:.....	9
3.1.8 Clickjacking:	9
3.1.9 SQL Injection:.....	10
4.TOOLS USED	20
5.MODULES IMPLEMENTED AND OUTPUT	20
5.1 Phishing Site Attack.....	20
5.2 Cross Site Scripting Attack	21
5.3 Extension Attack	22
5.4 SQL Injection.....	23
6. Learnings and Takeaways from the Study.....	24
7.REFERENCES.....	24

Web Browser Security: Different Attacks Detection and Prevention Techniques

1.INTRODUCTION

1.1 Overview

In the contemporary landscape of the Internet, security has evolved into a ubiquitous concern. The web and Internet-based social networks have seamlessly integrated into the lives of individuals across the globe. However, the growing prominence of these platforms has brought forth a pressing issue – the escalating frequency of attacks against digital systems.

Recent research shows that **75%** of cyber attacks are done at the web application level[1].

These attacks are orchestrated with the intent to pilfer private and financial information of web users, thereby underscoring the paramount importance of security measures. An ongoing challenge involves the infiltration of malicious content into systems, often without the users' knowledge, thus perpetuating a recurring predicament for host systems. This challenge extends its reach across diverse devices, be it smartphones or desktops, with the perpetrators exploiting a range of malicious entities such as viruses, Trojans, malware, and system vulnerabilities.

Central to these security concerns are the vulnerabilities inherent within systems. Such vulnerabilities serve as pivotal entry points for various forms of attacks, each with distinct characteristics and intentions. Among these factors, the successful delivery and subsequent execution of malware emerge as decisive elements in the fruition of these threats. The utilization of Simple Mail Transfer Protocol (SMTP) further amplifies the ease of executing threats. Mailborne threats, adept at luring recipients into interacting with malicious attachments, are prevalent in this landscape. Intriguingly, the efficacy of the delivery mechanism isn't contingent upon user actions, but rather capitalizes on exploiting applications or system vulnerabilities. Notably, the vulnerabilities in the client browser offer a strategic conduit for malware execution, leveraging the unwitting engagement of victims who encounter malicious web pages.

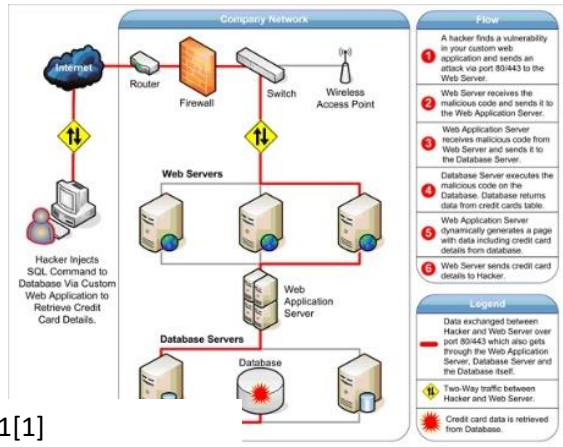


Fig.1.1[1]

In this intricate interplay of vulnerabilities, malware, and exploitation, the broader realm of cybersecurity strives to counterbalance these challenges through proactive measures.

1.2 Motivation

The primary motivation driving this research is the critical need to explore and understand the diverse vulnerabilities present in web security. The modern digital landscape is characterized by an ever-growing number of threats and cyberattacks that exploit these vulnerabilities. These attacks, which include XSS attacks, phishing attempts, SQL injections, and Man-in-the-Middle attacks, pose a significant risk to individuals, organizations, and the broader online community.

Additionally, organizations face substantial challenges in protecting their online presence and user data. Our motivation extends to aiding these organizations in effectively mitigating security risks. Implementing preventive techniques and robust security measures is essential for maintaining trust, safeguarding data privacy, and upholding the reputation of businesses and institutions.

1.3 Objective

This paper aims to investigate web browser security, focusing on different attack types such as XSS, phishing, SQL injection, and Man-in-the-Middle attacks. We seek to identify vulnerabilities in web applications and propose mitigation techniques to address these threats. By doing so, we aim to enhance user awareness about online security risks and empower individuals to protect their personal and sensitive data.

2. LITERATURE SURVEY

2.1 Role of The Web

The Web is used as the file repository for downloading other malicious files via HTTP. By using Trojan downloader vulnerable client browser visits an attack site. Attacker loads malicious script keeping in mind to infect the victim. Spammed Email messages and attack websites are acclimated lure victims to malicious code. Generally modest number of exploits is utilized as a part of attacks in similar ways in order to attack the system and install the malware.

Malicious sites: Malicious websites are created by cybercriminals to steal data and plant malware such as ransomware. These websites often masquerade as legitimate ones and use phishing emails to lure visitors[2].

According to Malicious scripts inquiry the client browser will load the appropriate exploits for that browser. By Trading off a website malicious substance is stacked into the pages for conveyance and execution of threat. Users trust level is adapt with browser configuration to render the page appropriately. HTML provides the IFRAME tag which is most commonly used in methods to compromise a site, which can be utilized to load content into the page. Height and width attributes are most relevant to malicious use. They can be used to control the size of the frame in the host web page in which malicious content is loaded.

2.2 The Web Browser

Web browsers are the underlying execution platform shared between web applications. Major web browsers, including Firefox, Chrome, Internet Explorer, Safari, and Opera, provide extension features that allow user to modify behaviour of the browser as well as enhance its functionality and GUI interface. Network Module gets a site page and plans content to be parsed by the HTML parser. The HTML parser creates a DOM which can then invoke other execution engines like JavaScript engine, CSS. The legitimate flow of processed content between components.

2.3 Web Extensions

Web Extensions is a new browser extension API. Web Extensions must be compatible with multiprocess Firefox (Electrolysis) as well as changes to Firefox's internal code should be less likely to break add-ons.

2.4 Electrolysis and Sandboxing Electrolysis:

Electrolysis: Imagine your web browser is like a cake, and each tab or window you open is a slice of that cake. Electrolysis takes the idea of the cake and turns it into cupcakes. Each cupcake (tab/window) is in its own separate wrapper (process). This way, if something goes wrong with one cupcake (like a bad website), it won't spoil the others, keeping your whole browsing experience safer and smoother.

Sandboxing:

Sandboxing is the practice where an application, a web browser, or a piece of code is isolated inside a safe environment against any external security threat. The idea of sandboxing is to enhance security[3].

Think of sandboxing as a protective bubble around your browser. When you play in a sandbox at the park, you're contained within the sandbox walls. Similarly, sandboxing keeps your browser contained in a safe space on your computer. Even if something harmful gets into your browser, like a sneaky bug, it's stuck in the sandbox and can't escape to mess up the rest of your computer.

In simpler words, Electrolysis makes each tab or window in your browser its own separate space, like cupcakes instead of a cake. Sandboxing wraps a protective bubble around your browser to keep it and your computer safe from any bad stuff that might try to get in.

3. METHODOLOGY/TECHNIQUES OR ALGORITHM USED

3.1 Attacks On Browser

3.1.1 Buffer Overflow:

Buffer overflow is a prevalent cybersecurity vulnerability that occurs when a program or application attempts to write more data into a buffer, a temporary storage area, than it can hold. This overflowed data can overwrite adjacent memory locations, potentially leading to unintended consequences like crashes, system instability, or, more critically, the execution of malicious code. Hackers exploit buffer overflow vulnerabilities to inject and execute malicious code, enabling unauthorized access, control, or data theft. Preventing buffer overflows requires stringent programming practices, input validation, and memory management techniques to ensure that programs can handle data inputs safely and efficiently, guarding against potential breaches and protecting software and systems from compromise.

Here's an example of how an attacker might exploit a buffer overflow vulnerability with a simple scenario involving a login form:

Scenario: Buffer Overflow in a Login Form

Imagine a website's login form that asks for a username and password. However, the website's code has a buffer overflow vulnerability in how it handles usernames.

User Input: The attacker knows about this vulnerability and crafts a malicious username that's intentionally very long, designed to overflow the buffer.

Buffer Overflow: The website's code doesn't properly check the length of the username input before storing it in a buffer. The attacker's excessively long username overflows the buffer and overwrites adjacent memory.

Crafted	Username:	Attacker's	input:
"AA...AAAAAAAABBBBBB BB"			

Overflow Effect: The overflowed data might include crafted code instructions that the attacker wants to execute. **Overwriting Control Flow:** The attacker's input overflows the buffer and overwrites parts of the program's memory, including control structures.

Exploitation: Due to the overwritten control structures, the program might execute the attacker's malicious function instead of following its normal logic.

3.1.2 Cross-Site Scripting:

Cross-Site Scripting (XSS) stands as a significant web vulnerability where attackers inject malicious scripts into web applications, which are then executed in the browsers of unsuspecting users. This exploit occurs when user inputs are inadequately validated and directly rendered on a webpage without proper sanitization. There are three main types of XSS attacks: Stored XSS (malicious code stored on a server and displayed to users), Reflected XSS (malicious code embedded in URLs), and DOM-based XSS (malicious code manipulates Document Object Model elements). The repercussions of XSS can range from session hijacking and data theft to website defacement. Preventing XSS requires input validation, proper encoding, and output escaping techniques, as well as employing security mechanisms like Content Security Policy (CSP) to restrict script execution. With its potential to compromise user data and damage reputation, understanding and mitigating XSS vulnerabilities is paramount in ensuring robust web security.

3.1.3 Man-in-the-Middle:

A Man-in-the-Middle (MitM) attack is a sophisticated form of cyberattack where an unauthorized entity covertly intercepts and potentially alters the communication between two parties without their knowledge. This attack takes place when attackers position themselves between the sender and receiver, often exploiting unsecured Wi-Fi networks or compromised devices. By eavesdropping on the exchanged data, the attacker can gain access to sensitive information, manipulate messages, or even inject malicious content. MitM attacks pose severe risks to data integrity, privacy, and security, affecting various sectors, including online banking, email communication, and e-commerce. Safeguarding against MitM attacks requires encryption, digital certificates, secure protocols like HTTPS, and consistent vigilance to detect and prevent unauthorized interference in digital communications.

3.1.4 Extension vulnerability:

Extension vulnerabilities refer to security weaknesses present in browser add-ons or extensions, often exploited by malicious actors to compromise user privacy and system integrity. These extensions,

intended to enhance browsing experiences, can inadvertently grant unauthorized access to user data or inject malicious code into web pages. Attackers exploit extension vulnerabilities to perform actions such as stealing credentials, redirecting users to malicious sites, or collecting sensitive information without consent. These vulnerabilities can arise from poor coding practices, lack of security reviews, or inadequate permissions management. To mitigate extension vulnerabilities, users should carefully vet and limit the number of installed extensions, keep them updated, and install only from trusted sources. Developers, on the other hand, should follow security best practices, adhere to coding guidelines, and conduct regular security assessments to prevent extension vulnerabilities from becoming a significant threat vector in the ever-evolving landscape of web security.

3.1.5 Extreme Phishing:

Extreme phishing represents a highly sophisticated evolution of the traditional phishing attack, designed to deceive users through advanced tactics and techniques. This form of cyberattack goes beyond generic emails and websites, employing intricate methods to manipulate victims into divulging sensitive information or performing unauthorized actions. Extreme phishing may involve personalized emails, domain spoofing, social engineering, and even exploiting zero-day vulnerabilities in software. By meticulously mimicking trusted entities, including financial institutions and legitimate services, extreme phishing attacks can successfully bypass traditional security measures and target even tech-savvy users. To counter these advanced threats, users need to be vigilant, scrutinize incoming emails, verify website URLs, and avoid clicking on suspicious links. Organizations should implement robust email filtering, multi-factor authentication, security awareness training, and continuous monitoring to thwart extreme phishing attempts and safeguard sensitive data from falling into the hands of malicious actors.

3.1.5 Browser Cache Poisoning:

Browser cache poisoning is a cybersecurity threat that exploits vulnerabilities in web caching mechanisms to compromise user security and privacy. Caching, intended to enhance web page loading speed, can inadvertently lead to unauthorized access and data exposure. Attackers inject malicious content into the cache, which subsequently gets served to unsuspecting users. This can result in the leakage of sensitive information, unauthorized access to user accounts, and the spread of malicious code. Browser cache poisoning often targets public Wi-Fi networks and unsecured connections, making users more vulnerable to these attacks. Mitigation strategies involve implementing secure caching practices, using HTTPS to encrypt communication, regularly clearing browser caches, and staying cautious when using public networks. Staying informed and adopting security-conscious

browsing habits is crucial in defending against the intricate challenges posed by browser cache poisoning.

3.1.6 Session Hijacking:

Session hijacking, also known as session stealing or session fixation, is a cyberattack where an unauthorized entity gains control over a user's active session on a web application or service. This attack is executed by stealing or intercepting the session identifier, which allows the attacker to masquerade as the legitimate user. Session hijacking can lead to unauthorized access to sensitive data, account takeover, and potential malicious actions carried out on behalf of the victim. Attackers often exploit vulnerabilities in session management mechanisms or intercept session identifiers transmitted over unencrypted connections. Mitigating session hijacking involves implementing secure session management practices, using strong and unique session identifiers, enabling HTTPS to encrypt communication, and monitoring for unusual activities that could indicate compromised sessions. User awareness about the risks of public Wi-Fi networks and unsecured connections is also crucial in preventing session hijacking attacks.

3.1.7 Drive-by-Download:

Drive-by download is a malicious technique used by cybercriminals to infect computers with malware without the user's knowledge or consent. This threat is executed when a user visits a seemingly legitimate website that has been compromised by attackers. As the user interacts with the site, the malicious code is automatically downloaded onto their device. Drive-by downloads can install various forms of malware, such as viruses, ransomware, spyware, or adware, posing significant risks to user privacy and system security. Attackers exploit vulnerabilities in web browsers, plugins, or other software to initiate these downloads. Preventive measures include keeping software updated, using reliable antivirus software, enabling browser security features, and being cautious when clicking on links or visiting unfamiliar websites. With drive-by downloads aiming to exploit unsuspecting users, proactive security practices are crucial in safeguarding against this stealthy threat.

3.1.8 Clickjacking:

Clickjacking, also known as a UI (User Interface) redress attack, is a crafty cyber threat that tricks users into clicking on something different from what they perceive. This attack involves embedding a malicious layer over seemingly innocent web content, effectively overlaying deceptive elements on top of legitimate ones. When users interact with what they perceive to be harmless content, they are unknowingly performing actions on hidden, malicious elements. Clickjacking can lead to unintended

actions such as revealing sensitive information, giving unauthorized access, or even unknowingly sharing social media posts.

Attackers leverage transparent iframes, CSS opacity manipulation, and other tactics to create this illusion of benign content. Preventive measures include implementing security headers like X-Frame-Options, Content Security Policy (CSP), and frame-busting scripts that thwart malicious framing attempts. Users should also exercise caution, verifying the authenticity of web content before interacting with it. As clickjacking relies on user deception, staying vigilant and employing security practices is paramount to avoid falling victim to this deceptive online manipulation.

3.1.9 SQL Injection:

SQL injection is a malicious technique used by attackers to manipulate SQL queries that interact with a database. It occurs when untrusted data is improperly included in SQL statements, potentially allowing unauthorized access, data manipulation, or even database breaches. For instance, consider a login form that accepts a username and password and constructs an SQL query to validate the login credentials:

Ex: SELECT * FROM users WHERE username = '\$username' AND password = '\$password';

If an attacker enters ' OR '1'='1 as the username and leaves the password field empty, the query becomes:

SELECT * FROM users WHERE username = " OR '1'='1' AND password = ";

Since '1'='1' is always true, the attacker effectively bypasses the login process and gains unauthorized access. There are many payloads like this.

To prevent SQL injection, input data should be properly sanitized and validated, and prepared statements or parameterized queries should be used to separate SQL code from user input.

4.Prevention Techniques

Buffer Overflow Attack:

Input Validation and Bounds Checking:

Implement strict input validation to ensure data is within expected limits. Use libraries or functions that automatically handle bounds checking to prevent overflows.

Use of Secure Programming Languages:

Choose programming languages that have built-in safety features, like Rust or languages with safer string handling, such as Java.

Memory Protection Mechanisms:

Employ Address Space Layout Randomization (ASLR) to randomize memory addresses, making it harder for attackers to predict target addresses. Implement Data Execution Prevention (DEP) or No eXecute (NX) to mark certain memory areas as non-executable.

Use of Compiler Flags and Options:

Enable compiler options that perform additional checks and insert security mechanisms. Utilize options like "-fstack-protector" to detect and prevent stack buffer overflows.

Secure Coding Practices:

Follow secure coding guidelines and best practices to write robust and safe code. Avoid using unsafe functions and deprecated libraries that are prone to vulnerabilities.

Regular Software Updates:

Keep software, operating systems, and libraries up to date to benefit from security patches and fixes.

Static and Dynamic Code Analysis:

Employ automated tools that analyze code for vulnerabilities, helping identify potential buffer overflow issues.

Use of Sandboxing and Virtualization:

Run applications in sandboxed environments or virtual machines to limit the impact of successful attacks.

Security Audits and Penetration Testing:

Conduct thorough security audits and penetration testing to identify and address vulnerabilities.

Security Training and Education:

Train developers in secure coding practices to raise awareness and reduce the likelihood of introducing vulnerabilities.

Cross-Site Scripting (XSS):

Cross-Site Scripting (XSS) vulnerabilities can be mitigated through a combination of coding practices, input validation, and security measures. Here are some effective prevention techniques:

Input Validation and Output Encoding:

Validate and sanitize user input before rendering it in web pages. Use output encoding (escaping) to convert special characters into their corresponding HTML entities.

Content Security Policy (CSP):

Implement CSP headers to control which sources of content are allowed to be executed on your web page. Enforce a policy that prevents inline scripts and unauthorized content from running.

Use of Libraries and Frameworks:

Leverage modern web frameworks and libraries that automatically handle input validation and output encoding.

Avoiding Inline Event Handlers:

Avoid using inline event handlers like onclick in HTML tags. Use unobtrusive JavaScript by attaching event listeners programmatically.

HTTP-Only Cookies:

Mark cookies as HTTP-only to prevent client-side scripts from accessing sensitive cookie data.

Escape Dynamic JavaScript Values:

Ensure that dynamic JavaScript values are properly escaped to prevent script injection.

Sanitize User-Generated Content:

Use sanitization libraries to filter out potentially harmful content from user-generated input.

Regular Security Audits:

Conduct regular security audits and code reviews to identify and address potential XSS vulnerabilities.

HTTP Secure (HTTPS):

Use HTTPS to encrypt data transmission between the user's browser and the server, preventing attackers from intercepting and injecting malicious content.

Web Application Firewall (WAF):

Implement a WAF to detect and block incoming XSS attacks.

Man-in-the-Middle Attacks:

Mitigating Man-in-the-Middle (MitM) attacks requires a multi-faceted approach that involves secure communication protocols, proper authentication, and user awareness. Here are effective prevention methods:

Use of Secure Communication Protocols:

Utilize HTTPS for web communication to encrypt data between the user's browser and the server, making it difficult for attackers to intercept and manipulate. Employ protocols like TLS/SSL to establish secure connections and protect against eavesdropping.

Server Authentication:

Use trusted digital certificates to authenticate servers. Verify server certificates against known Certificate Authorities (CAs) to ensure the authenticity of the server.

Two-Factor Authentication (2FA):

Implement 2FA to add an extra layer of security, making it harder for attackers to gain unauthorized access.

Public Key Infrastructure (PKI):

Use PKI to manage digital certificates and ensure secure communication between parties.

Avoid Unsecured Wi-Fi Networks:

Refrain from using unsecured public Wi-Fi networks, which are common targets for MitM attacks. Use Virtual Private Networks (VPNs) to encrypt your internet connection when using public networks.

Network Segmentation:

Segregate networks to limit the potential attack surface and reduce the impact of successful MitM attacks.

Be Cautious with Links:

Avoid clicking on links in unsolicited emails or messages that could lead to phishing sites set up for MitM attacks.

Check for HTTPS and Valid Certificates:

Always check for HTTPS and verify the validity of the SSL certificate before entering sensitive information on websites.

Security Awareness Training:

Educate users about the risks of MitM attacks and the importance of secure browsing practices.

Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS):

Implement IDS and IPS to detect and prevent abnormal network activities indicative of a MitM attack.

Use Encrypted Communication Channels:

Implement end-to-end encryption in communication apps to prevent intermediaries from eavesdropping.

Extension Vulnerabilities:

Mitigating extension vulnerabilities requires a combination of cautious practices, regular updates, and security measures. Here are preventive methods to consider:

Source Verification:

Download extensions only from trusted sources like official stores (e.g., Chrome Web Store) to minimize the risk of malicious extensions.

Check Reviews and Ratings:

Read user reviews and ratings before installing an extension to gauge its credibility.

Limit the Number of Extensions:

Install only necessary extensions to reduce the attack surface and potential vulnerabilities.

Regular Updates:

Keep all extensions and browsers up to date to benefit from security patches and bug fixes.

Permissions Audit:

Review the permissions requested by each extension before installation. Avoid extensions that ask for excessive or unnecessary permissions.

Extension Source Code Review:

If feasible, review the source code of open-source extensions to ensure its integrity.

Disable or Remove Unneeded Extensions:

Disable or remove extensions that are no longer in use to minimize potential vulnerabilities.

Use Security Plugins:

Install browser security plugins that detect and warn about potentially harmful extensions.

Browser Updates:

Keep your browser up to date to take advantage of security enhancements and protections against extension vulnerabilities.

Privacy Settings:

Configure browser privacy settings to restrict what data extensions can access.

Extreme Phishing:

Preventing extreme phishing, which involves sophisticated tactics to trick users, requires a multi-layered approach that combines awareness, technology, and user education. Here are effective preventive methods to counter extreme phishing attacks:

User Awareness and Education:

Regularly educate users about the risks and techniques employed in extreme phishing attacks. Teach users how to identify suspicious emails, URLs, and unusual behavior.

Anti-Phishing Software:

Implement robust anti-phishing tools that can detect and block advanced phishing attempts.

Email Filtering and Authentication:

Employ strong email filtering to detect and block phishing emails before they reach users. Implement email authentication protocols like SPF, DKIM, and DMARC to verify the authenticity of incoming emails.

URL Analysis and Detection:

Utilize website reputation services and URL analysis tools to identify potentially malicious websites.

Multi-Factor Authentication (MFA):

Implement MFA to add an extra layer of security, making it harder for attackers to gain unauthorized access.

Security Awareness Training:

Provide regular security training to employees and users to keep them informed about the evolving tactics of extreme phishing.

Behavioral Analysis:

Use behavioral analytics to detect abnormal user behavior that might indicate a phishing attack.

User Behavior Monitoring:

Monitor user actions to identify unusual patterns or actions that deviate from typical behavior.

Strong Password Policies:

Enforce strong password policies to prevent attackers from gaining unauthorized access even if they acquire credentials.

Regular Security Audits:

Conduct regular audits of systems and applications to identify and address potential vulnerabilities.

Vendor Security Assessments:

Evaluate the security practices of third-party vendors who handle sensitive information to prevent potential supply chain attacks.

Incident Response Plan:

Develop a well-defined incident response plan to handle extreme phishing incidents efficiently.

Continuous Threat Intelligence:

Stay updated with the latest phishing techniques and tactics through threat intelligence sources.

Browser Cache Poisoning:

To prevent browser cache poisoning attacks, a combination of security mechanisms and best practices should be employed. Here are some preventive methods to counter browser cache poisoning:

HTTP Strict Transport Security (HSTS):

Implement HSTS headers to enforce secure connections over HTTPS, reducing the risk of caching attacks.

Content Security Policies (CSP):

Utilize CSP to control which sources of content are allowed to be loaded, reducing the likelihood of malicious content injection.

Cache-Control Headers:

Set appropriate Cache-Control headers to control caching behavior and reduce the risk of storing compromised content.

Encrypted Communication:

Use HTTPS for all communications to ensure that data transmitted between the user's browser and the server remains encrypted.

Input Validation and Output Encoding:

Validate and sanitize user input to prevent injection attacks that might lead to cache poisoning.

Regular Security Updates:

Keep server software and web applications up to date to patch known vulnerabilities.

Vigilant Monitoring:

Monitor server logs and network traffic to detect unusual or unauthorized activities indicative of cache poisoning.

Static Content and Dynamic URLs:

Apply caching primarily to static content and avoid caching URLs with dynamic parameters to reduce the impact of cache poisoning.

Implementing Nonce and Tokens:

Use nonce or token-based mechanisms to ensure the integrity of resources and prevent cache poisoning.

Security Audits:

Conduct regular security audits to identify and address potential vulnerabilities in your web application.

Security Headers:

Implement security headers like X-Frame-Options, X-Content-Type-Options, and X-XSS-Protection to enhance overall security posture.

Session Hijacking:

Session Hijacking Prevention: Fortifying user sessions against unauthorized access involves implementing strong session ID generation, employing SSL/TLS encryption through HTTPS, setting appropriate session timeouts, marking cookies with HttpOnly and Secure flags, regenerating session IDs upon login, tracking IPs and verifying user agents, implementing CSRF protection, using tokens or nonces, educating users, and conducting regular security audits to ensure the integrity of session management practices.

Drive-By Download:

Prevent drive-by download attacks by keeping software updated, utilizing antivirus software, enabling automatic updates, implementing web content filtering, educating users about risks, using script blockers, and considering intrusion detection systems and browser extensions for enhanced protection. Stay informed about evolving threats to ensure ongoing defense.

Clickjacking:

Protect against clickjacking by adding confirmation mechanisms for clicks, utilizing frame-busting techniques with JavaScript, implementing the X-Frame-Options header to prevent framing, and educating users about interacting only with trusted components. Regularly audit and review web pages to detect potential clickjacking vulnerabilities and ensure a secure browsing experience.

SQL Injection:

SQL injection prevention involves using parameterized queries (prepared statements) to separate user input from SQL code, input validation to ensure data adheres to expected formats, escaping user input to neutralize special characters, applying the least privilege principle to limit user privileges, utilizing stored procedures for executing SQL on the server side, and considering a web application firewall (WAF) for additional protection. These strategies collectively mitigate the risk of malicious actors exploiting vulnerabilities to manipulate SQL queries and compromise database security.

Ex: `$username = mysqli_real_escape_string($connection, $_POST['username']);`

4.TOOLS USED

HTML

Javascript

PHP

SQL

XAAMP

SET-Kali Linux

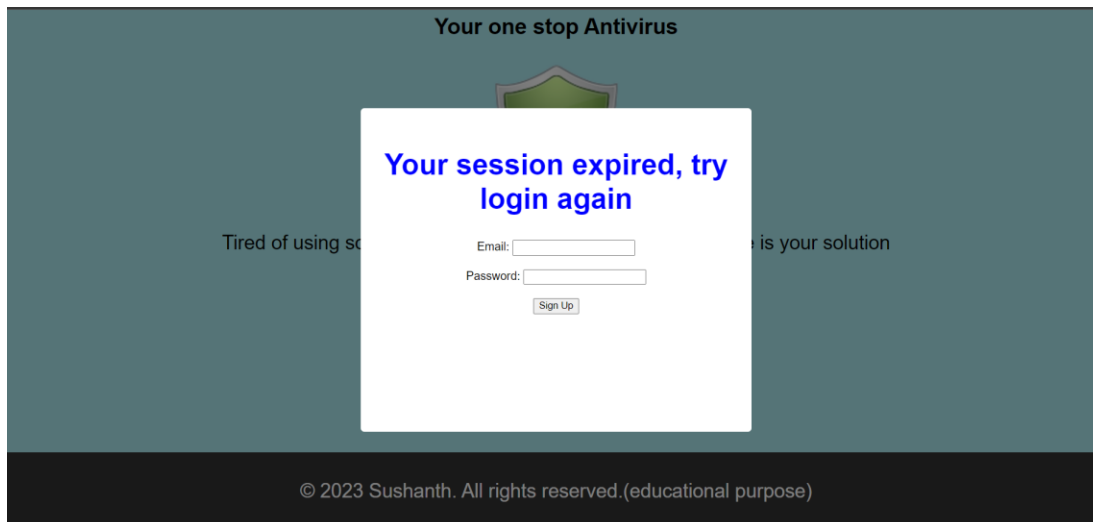
5.MODULES IMPLEMENTED AND OUTPUT

5.1 Phishing Site Attack

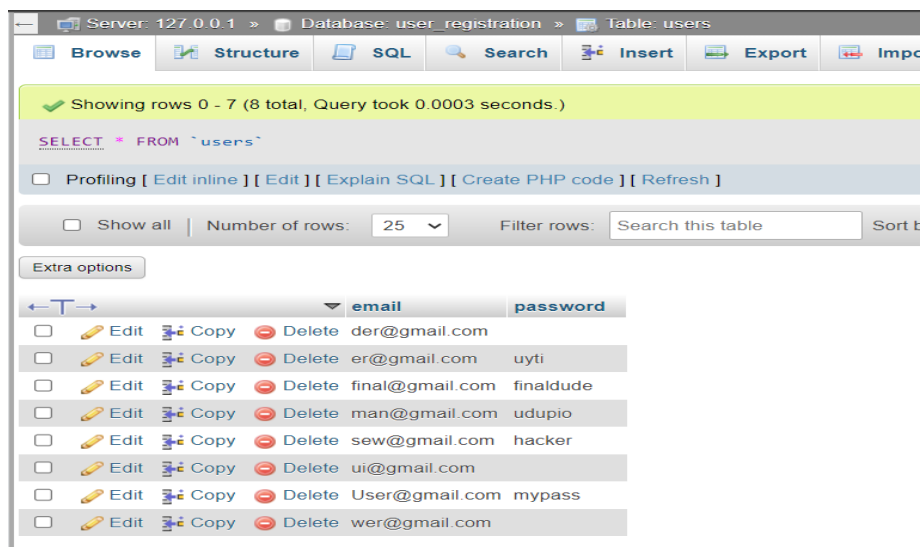


This is the phishing site depicting as a famous Antivirus software company, using SET we can clone any site deceiving user and make him to believe that it is the legit site. This site can be made to opened by user by phishing mails or links.

When user clicks on Download, a popup window will open,

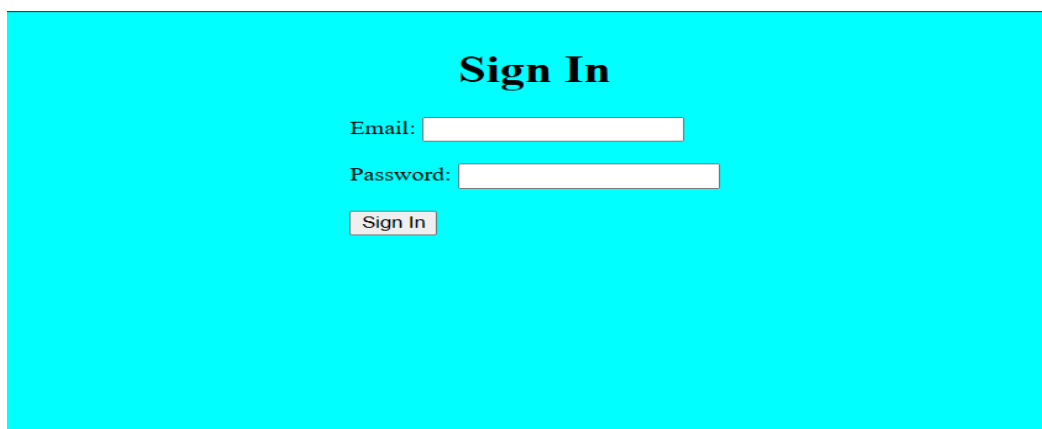


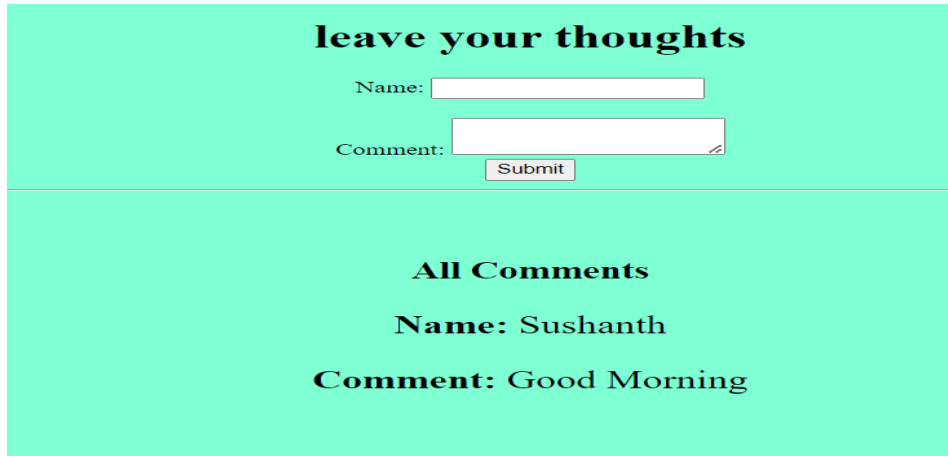
User thinks that he should login, and types his credentials and Submit. Since it is a phishing site, user credentials will be gained by Attacker.



5.2 Cross Site Scripting Attack

Vulnerable Site:





leave your thoughts

Name:

Comment:

All Comments

Name: Sushanth

Comment: Good Morning

If attacker knows about this vulnerability, he may use different payloads.

Ex: `<script>document.body.style.backgroundColor="red";</script>`

If he comments this , then it is stored in database and every user will see changes.



leave your thoughts

Name:

Comment:

All Comments

Name: Sushanth

Comment: Good Morning

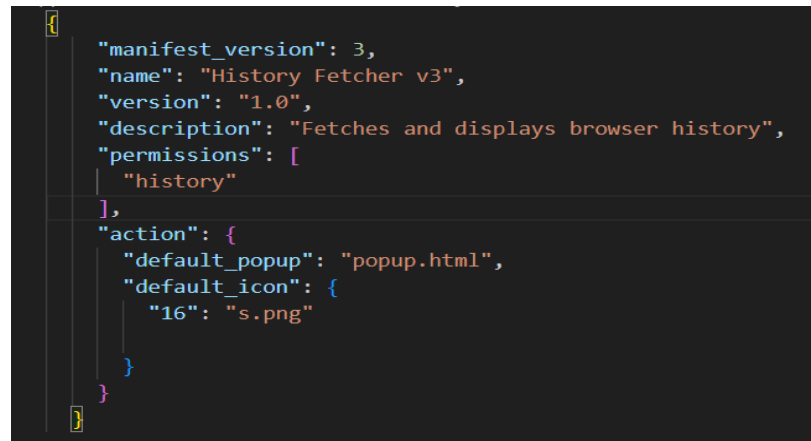
Name: hacker

Comment:

5.3 Extension Attack

Using Chrome extension and web hook, Attacker can harvest user history.

manifest.Json file for harvester extension



5.4 SQL Injection

Payload : 'or'1'='1

If a sign in page is vulnerable to this injection attack , attacker can easily login to page without password or email id

Conclusion: Strengthening Web Browser Security

In conclusion, web browsers are essential tools but can be vulnerable to various attacks. Memory vulnerabilities like heap overflow can be a concern, emphasizing the importance of security updates. Plugins also pose risks, requiring consistent attention. Attacks from malicious websites are a reality, highlighting the need to block pop-ups and educate users. Browser security can be bolstered with multiprocess architecture and sandboxing, enhancing protection by isolating activities. Prioritizing these features and staying informed about security measures can create a safer browsing experience for users.

6. Learnings and Takeaways from the Study

The research paper on "Web Browser Security: Different Attacks Detection and Prevention Techniques" offers valuable insights into the world of web security, highlighting the diverse range of threats that individuals and organizations face. Its key takeaways revolve around the importance of raising awareness among users about these vulnerabilities, empowering them to make informed choices when navigating the digital landscape. Got to know about the different attacks possibilities and preventions.

7. REFERENCES

- [1] <https://www.acunetix.com/websitesecurity/web-application-attack/>
- [2] <https://www.mimecast.com/blog/what-are-malicious-websites/>
- [3] <https://www.browserstack.com/guide/what-is-browser-sandboxing>
- [4] <https://www.youtube.com/watch?v=PPzn4K2ZjfY&pp=ygULd2ViIGF0dGFja3M%3D>
- [5] <https://www.youtube.com/watch?v=2nXOxLpeu80&pp=ygULd2ViIGF0dGFja3M%3D>
- [6] <https://www.ibm.com/topics/phishing>
- [7] <https://www.synopsys.com/glossary/what-is-csrf.html#:~:text=Definition,has%20in%20an%20authenticated%20user.>
- [8] <https://web.dev/same-origin-policy/#:~:text=The%20same%20origin%20policy%20is,from%20multiple%20sites%20at%20once.>
- [9] [\(PDF\) Web Browser Security: Different Attacks Detection and Prevention Techniques \(researchgate.net\)](#)