Browser Exploitation, Attack Prevention and Mitigation

Domain: Browser Security

CDAC, Noida

CYBER GYAN VIRTUAL INTERNSHIP PROGRAM

Submitted By:

Abhigyan Sharma

Project Trainee, (May-June) 2024

BONAFIDE CERTIFICATE

This is to certify that this project report entitled Browser Exploitation, Attack Prevention and Mitigation submitted to CDAC Noida, is a Bonafide record of work done by Abhigyan Sharma under my supervision from 15 May 2025 to 12 June 2025

Declaration by Author(s)

This is to declare that this report has been written by me/us. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be plagiarized, I/we are shall take full responsibility for it.

Name of Author(S):Abhigyan Sharma

TABLE OF CONTENTS

Content	Page No.
1. Introduction 1.1 Problem Statement 1.2 Learning Objective 1.3 Approach	6
2. Browser Exploitation: Background and Challenges 2.1 Introduction to Browser Security 2.2 Common Browser Vulnerabilities 2.3 Security Architecture of Modern Browsers	8
3. Literature Review 3.1 Phishing 3.2 Malware 3.3 ClickJacking 3.4 Cross-Site Scripting (XSS)	9
4. Identification of Threat Vectors 4.1 Drive-by downloads 4.2 Cross-Site Scripting 4.3 Malicious Extensions	11
5. Browser Security Mechanisms 5.1 Comparison of Major Browsers 5.2 Analysis of Privacy Policies	18
6. Security Settings and Hardening Configurations 6.1 Safari 6.2 Chrome 6.3 FireFox	20
7. Implementation of Sandboxing 7.1 Sandboxing in browsers 7.2 Sandboxing for developers	24
8. Deployment of Detection Systems 8.1 Intrusion Detection Systems 8.2 Nessus Vulnerability Scanner 8.3 Web Application Firewall	25
9. Conclusion And Recommendations	30
10. References	31

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who supported me throughout the course of this project.

First and foremost, I extend my heartfelt thanks to my project mentor and guide, Prateek Saraswat, whose expert guidance, constant encouragement, and insightful feedback were invaluable at every stage of the project. Their mentorship helped me navigate complex topics related to browser security and develop a clear understanding of exploitation attack vectors and their mitigation techniques.

I am also grateful to CDAC,Noida for providing me with the opportunity to work on this internship project. The resources, support, and collaborative environment offered during the internship significantly contributed to the successful completion of this project.

This project has been an enlightening experience, and I am confident that the insights gained will serve as a strong foundation for my future endeavors in the field of cybersecurity.

Browser Exploitation, Attack Prevention and Mitigation

PROBLEM STATEMENT:

In today's digital world, web browsers are an essential part of our daily lives — from checking emails to handling online banking and work-related tasks. But this convenience also makes browsers a prime target for cyberattacks. Hackers are constantly looking for ways to exploit browser vulnerabilities to gain unauthorized access to sensitive data or even take control of users' devices.

Despite the security features built into modern browsers, many users still fall victim to browser-based attacks due to outdated software, weak configurations, or a lack of awareness. These attacks not only compromise personal privacy but can also lead to serious security breaches in organizations.

This project aims to explore how these browser exploitation attacks happen and to find practical ways to prevent and reduce their impact. By using techniques like browser hardening, sandboxing, regular updates, and implementing tools like intrusion detection systems and web application firewalls, this project seeks to make browsing safer for everyone.

LEARNING OBJECTIVE:

The objective of this project is to develop a thorough understanding of the mechanisms, techniques, and implications of browser exploitation attacks, and to investigate effective strategies for their prevention and mitigation. The key learning outcomes include:

- Gaining in-depth knowledge of common browser vulnerabilities and the attack vectors leveraged by adversaries to exploit them.
- Analyzing the architecture and security features of modern web browsers to identify potential weaknesses.
- Exploring and implementing robust countermeasures such as browser hardening, sandboxing, secure configuration practices, and timely patch management.
- Studying the deployment and effectiveness of intrusion detection systems (IDS) and web application firewalls (WAF) in monitoring and mitigating browser-targeted threats.
- Understanding the significance of user awareness and designing educational strategies to promote secure browsing behavior.

APPROACH:

The project was carried out through a systematic, step-by-step methodology designed to first understand the nature of browser exploitation attacks and then develop and evaluate effective countermeasures. The following approach was adopted:

Literature Review and Research

- Conducted an in-depth study of existing literature on browser security, attack vectors, and recent exploitation incidents.
- Referred to authoritative sources including the SANS Institute, OWASP, Trend Micro, and NCSC to understand browser vulnerabilities and protection techniques.

Identification of Threat Vectors

- Analysed common browser-based attack techniques such as drive-by downloads, cross-site scripting (XSS), malicious extensions, and phishing.
- Studied how these attacks exploit browser features, plugins, and user behaviour.

Analysis of Browser Security Mechanisms

- Evaluated the built-in security features of modern browsers such as Chrome, Firefox, and Edge.
- Identified security settings and configurations that can be used to harden browser environments.

Implementation of Preventive Measures

- Applied browser hardening techniques, such as disabling insecure features, enabling security flags, and using secure configurations.
- Integrated sandboxing and ensured regular updates to reduce the attack surface.

Deployment of Detection and Monitoring Tools

- Set up and tested intrusion detection systems (IDS) and web application firewalls (WAF) to monitor traffic and block malicious activities.
- Explored the effectiveness of browser security extensions like NoScript and uBlock Origin.

Vulnerability Scanning and Risk Assessment

- Used tools like Nessus and Qualys to perform vulnerability scanning and assess the security posture of browsers under different configurations.

User Awareness and Education

- Developed user training materials and best-practice guidelines to promote secure browsing behaviour.
- Designed example phishing simulation exercises to raise awareness of common social engineering tactics.

IMPLEMENTATION:

BROWSER SECURITY

Web browsers are applications that are used to access the internet. It is imperative that web browser users consider the security and privacy of the web browsers that they choose to use. Privacy is how a user chooses to have their personal information used and controlled. Security on the other hand is how the personal information is protected while using the services the web browser provides. Web browsers work hard to protect their users from hackers, businesses, websites, advertisers, internet service providers, and government agencies.

There are a multitude of web browser security vulnerabilities that can be exploited and interrupt the user's browser experience. The top four common web browser vulnerabilities mentioned by Gergely Kalman, a chief technology officer (CTO) are injection flaws, broken authentication, cross site scripting (XSS), and insecure direct object references. The first vulnerability, injection flaws, is one not many daily users are familiar with. Injection flaws are an effect of a lack of filtering of untrusted input, and even in the case of trusted input you can never be 100% sure. Everything has to be considered untrusted input because if not then you are more at risk of being exploited and it is better to be safe rather than sorry. The next most common web vulnerability is broken authentication. Broken authentication happens when in the process of proving the identity of a user, an interruption occurs. An interruption is when an attacker is able to divert or capture the authentication methods that are put into place. Weaknesses that open the door for interruption could be the use of plain text, encrypted, or weakly hashed passwords, ineffective or missing multi-factor authentication, the allowance of fragile, default, or common passwords, and the allowance of weak credential recovery processes. The next vulnerability is cross site scripting (XSS). Cross site scripting happens when a hacker, or browser user with malicious intent uses a web application to send malicious code to another user's computer, or its the introduction of harmful script while loading a web application. The last of the top four common web browser mistakes is having insecure direct object references (IDOR). Insecure direct object references can lead to vulnerabilities because users are provided with direct access to important items like files and database keys, and hackers can take the reference and cause damage . Authorisation is needed because of the direct access that is being provided to web browser users.

EXISTING LITERATURE ON ATTACK VECTORS

Phishing

Phishing is a well-known attack vector in which the adversary sends emails to targets in order to trick them into giving out personal information, or executing malicious code.

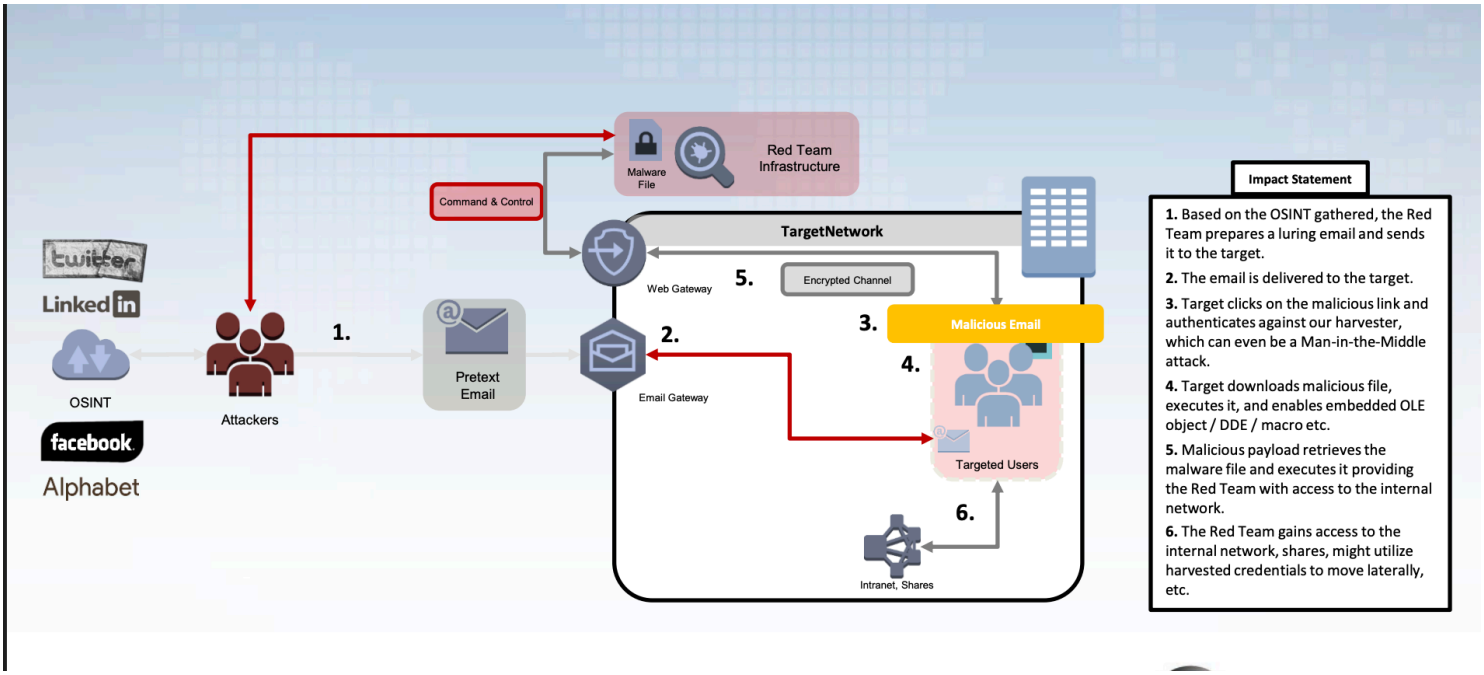
In early 2009, web security researchers began to explore how the growing support for media elements in HTML5—specifically <audio> and <video>—could be misused by attackers. Safari, which relied heavily on Apple's WebKit rendering engine, had recently incorporated these new

media APIs, enabling developers to create rich, embedded audio experiences directly in the browser. However, this shift opened new doors for exploitation.

By mid-2009, a vulnerability now known as CVE-2009-1703 was disclosed. The flaw was subtle yet serious. Safari's WebKit engine allowed webpages—hosted on remote servers—to reference local files on a user's machine using the file:// URI scheme. This behaviour, originally intended to enable local media playback in trusted scenarios, proved problematic. Through a crafted <audio> tag like <audio src="file:///etc/passwd">, a malicious page could attempt to load a sensitive file from the local filesystem.

Although modern operating systems blocked the actual playback of such local files from a remote page, the browser's behaviour in handling these requests gave away unintended clues. For example, JavaScript could monitor the success or failure of loading specific file paths, effectively allowing an attacker to probe the existence of files on a user's machine. This side-channel data leakage became a privacy concern.

More alarmingly, the browser's UI—particularly Safari's address bar and status indicators—did not always reflect that such media content originated from a remote attacker. This allowed attackers to spoof trusted UI elements. They could trick users into believing they were interacting with a local file or a system interface, when in fact they were on a phishing site. This opened the door to sophisticated social engineering attacks, particularly effective against less tech-savvy users.



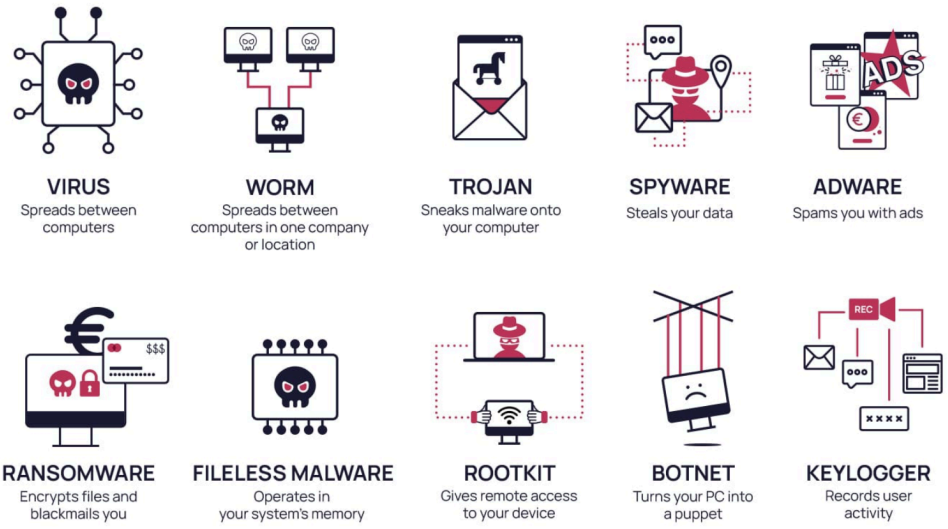
Malware

Malware is a computer program created by cyber-criminals to infiltrate, damage, or control computer systems or mobile devices without your consent or knowledge. What makes malware so dangerous is once your computer or device is infected, it can give the cyber- criminal total control

without you even knowing it. It can silently capture your activities, including who you are communicating with, what you are saying, and your logins and passwords to your most important accounts.

Malware can be introduced into a computer from several sources, including emails, downloads, websites, and external storage devices. The possibilities are endless. No matter the source of the attack, the end results can be disruptive to both individuals and companies, making it important to learn how to protect yourself and your company from the potential for malware attacks.

Major businesses and corporations are often the targets of serious malware attacks. According to a report by IBM, ransomware attacks cost companies an average of **$4.54 million in 2022**, not including the ransom cost. If a company pays the ransom, losses could be even higher, particularly as some hackers will not release the files on receipt of the funds as promised. Small businesses may also be targets of hackers, with even more devastating results. In fact, one in four British small and medium enterprises was targeted by ransomware in 2022, according to a study by Avast.

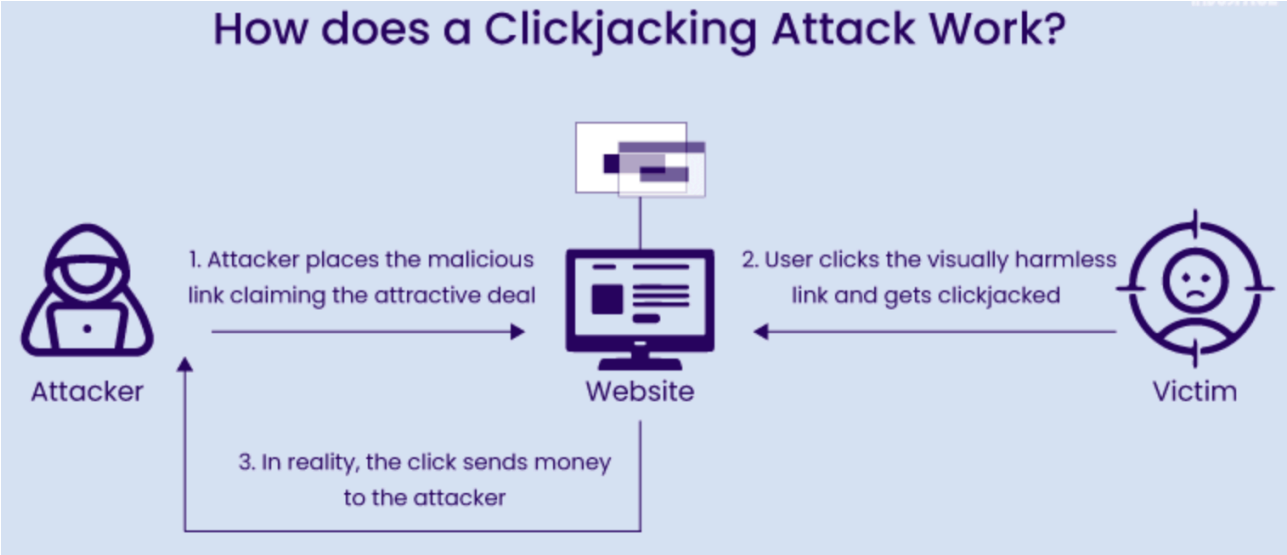


ClickJacking

Clickjacking is an attack that tricks a user into clicking a webpage element which is invisible or disguised as another element. This can cause users to unwittingly download malware, visit malicious web pages, provide credentials or sensitive information, transfer money, or purchase products online. Typically, clickjacking is performed by displaying an invisible page or HTML element, inside an iframe, on top of the page the user sees. The user believes they are clicking the visible page but in fact they are clicking an invisible element in the additional page transposed on top of it.

An update released recently for the WhatsApp desktop app for Windows patches a spoofing vulnerability that could make it easier for threat actors to trick users and achieve remote code execution. According to a brief advisory published by Meta, the vulnerability is tracked as CVE-2025-30401 and it has been patched with the release of WhatsApp for Windows version 2.2450.6. All prior versions are impacted. An attacker could exploit the vulnerability by sending the targeted user a specially crafted file whose MIME type is altered to make it appear as a harmless file. The user would believe that they are opening an image or document file when in reality they would be running an executable or other type of file that triggers the execution of malicious code. “A maliciously crafted mismatch could have caused the recipient to inadvertently execute arbitrary

code rather than view the attachment when manually opening the attachment inside WhatsApp," Meta explained. Attacks involving MIME type manipulation have been known for years, but Meta has not mentioned anything about CVE-2025-30401 being exploited in the wild. However, WhatsApp is a valuable target for threat actors and vulnerabilities affecting the messaging application are known to be exploited in attacks.



Cross-Site Scripting

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.
The Cross-Site Scripting (XSS) vulnerability in Firefox extensions disclosed in early 2024 involved a flaw in how certain Firefox extensions processed untrusted content within privileged contexts. A malicious webpage could send crafted content that was not properly sanitized by the extension, leading to execution of arbitrary JavaScript within the extension's privileged context.

IDENTIFICATION OF THREAT VECTORS

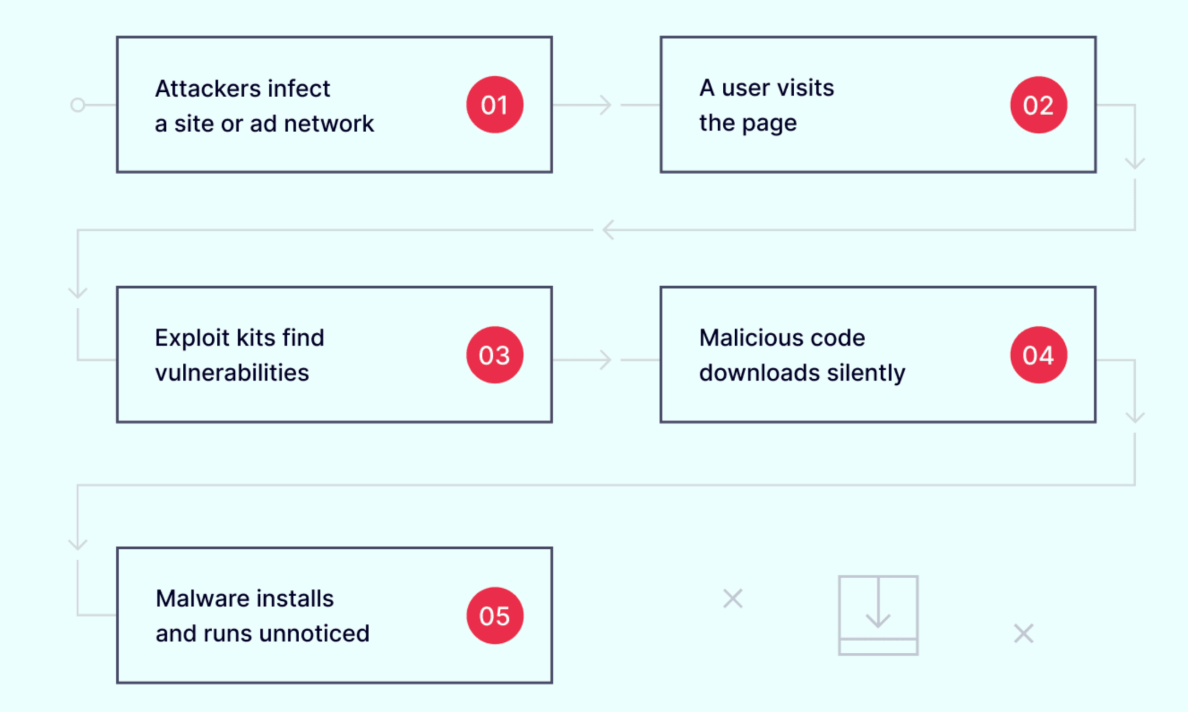
Drive By Downloads

What are Drive By Downloads

A drive-by download attack occurs when a user loads a web page that contains malicious code. No clicks are needed—just visiting the page is enough to trigger an unintentional download. Attackers use hidden scripts and exploit kits to take advantage of security flaws, leading to covert downloads of dangerous software.Most victims don't realise an attack has happened until malware installs and starts causing damage. Web pages with outdated third-party components are prime targets. Once the harmful code executes, it can quickly deliver download payloads to endpoint users. Even legitimate websites can be dangerous if compromised.

How drive-by download attacks work

Threat actors design drive-by download attacks to exploit vulnerabilities in browsers, plug-ins, and outdated software. They begin by compromising a website or ad network, planting rogue code that waits for visitors.
Once a user loads the infected web page, the malicious script automatically scans for security weaknesses in the browser or any third-party software. If it finds a vulnerability, exploit kits take over, silently executing malware code and triggering a hidden download.
The malware installs in the background, often running unnoticed while it steals data or grants cybercriminals remote access. Since this entire process happens without user interaction, infections are difficult to detect before damage occurs.



Who is most at risk?

Anyone browsing the web can be exposed to a drive-by download. However, certain users and organisations face higher risks:

- People using outdated browsers or skipping security updates
- Users with weak browser settings allowing unauthorised downloads
- Companies with lax security policies, making endpoint users easy targets
- Employees with admin privileges who can trigger large-scale malicious installations

Many drive-by download attacks happen through a silent download—when a user visits a breached website, and malicious code installs automatically. Keeping browsers updated and restricting third-party software can reduce the risk of malware

Major drive-by download incidents

- 2016 malicious ads on major news sites - In March 2016, attackers compromised ad networks on sites like The New York Times, BBC, and AOL. Just loading these pages exposed users to hidden

scripts. Exploit kits, including Angler, scanned for browser vulnerabilities and silently installed ransomware—no clicks needed.
- 2016 Chrome on Fedora drive-by download attack - In November 2016, researcher Chris Evans revealed how Chrome's auto-download behaviour and Fedora's unsandboxed "Tracker" tool enabled silent malware infections. Chrome downloaded files without asking, and Fedora's Tracker indexed them automatically. This let attackers execute malicious code without user action. Gstreamer, Fedora's preview tool, had security flaws that made the attack even easier.

How to prevent drive-by downloads

Preventing these attacks requires a layered security approach. Here's what helps:
1. Keep software updated—patch browsers, plug-ins, and operating systems.
2. Use intrusion detection systems to monitor traffic for suspicious activity.
3. Install ad blockers to reduce exposure to malicious advertising networks.
4. Use sandboxing tools to isolate and analyse suspicious downloads.
5. Limit third-party plugins in web pages and applications.
6. Train users to recognise warning signs of compromised websites.
7. Back up data regularly to recover quickly in case of malware installation.
8. Implement endpoint protection solutions that offer real-time malware detection and threat intelligence to monitor and block malicious activities on devices.

Strong security policies and real-time monitoring help reduce the risk of unintentional downloads.

Cross Site Scripting (XSS)

Overview

A cross-site scripting (XSS) attack is one in which an attacker is able to get a target site to execute malicious code as though it was part of the website. A web browser downloads code from many different websites and runs it on the user's computer. Some of these websites will be highly trustworthy, and the user may use them for sensitive operations, such as financial transactions or medical advice. With others, such as a casual gaming site, the user may have no such trust relationship.
In a successful XSS attack, the attacker is able to subvert the same-origin policy by tricking the target site into executing malicious code within its own context, as though it were same-origin. The code can then do anything that the site's own code can do, including, for example:
- Access and/or modify all the content of the site's loaded pages, and any content in local storage.
- Make HTTP requests with the user's credentials, enabling them to impersonate the user or access sensitive data.

Example Pages which are vulnerable to XSS attack

Code injection in the browser

In this example, suppose the website for the user's bank is `my-bank.example.com`. The user is typically signed into it, and code in the website can access the user's account details and perform transactions. The website wants to display a welcome message, personalised for the current user. It displays the welcome in a heading element:

HTML

```
<h1 id="welcome"></h1>
```

The page expects to find the current user's name in a URL parameter. It extracts the parameter value, and uses the value to create a personalized greeting message:

JS

```
const params = new URLSearchParams(window.location.search);
const user = params.get("user");
const welcome = document.querySelector("#welcome");

welcome.innerHTML = `Welcome back, ${user}!`;
```

Let's say this page is served from https://my-bank.example.com/welcome. To exploit the vulnerability, an attacker sends the user a link like this:

HTML

```
<a
  href="https://my-bank.example.com/welcome?user=<img src=x onerror=alert('hello!')>">
  Get a free kitten!</a
>
```

When the user clicks the link:

1. The browser loads the page.
2. The page extracts the URL parameter named user, whose value is <img src=x on_error=alert(“hello!")>.
3. The page then assigns this value to the welcome element's innerHTML property, which creates a new <img> element, which has a src attribute value of x.
4. Since the src value generates an error, the on_error event handler property is executed, and the attacker gets to run its code in the page.

In this case the code just displays an alert, but in a real banking website, the attacker code would be able to do anything that the bank's own front-end code could.

Code Injection in the Server

In this example, consider a website with a search function. The HTML for the search page might look like this:

HTML

```
<h1>Search</h1>

<form action="/results">
  <label for="mySearch">Search for an item:</label>
  <input id="mySearch" type="search" name="search" />
  <input type="submit" />
</form>
```

When the user enters a search term and clicks "Submit", the browser makes a GET request to "/results", including the search term as a URL parameter, like this:

```
https://example.org/results?search=bananas
```

The server wants to display a list of search results, with a title indicating what the user searched for. It extracts the search term from the URL parameter. Here's what this might look like in Express:

JS

```
app.get("/results", (req, res) => {
  const searchQuery = req.query.search;
  const results = getResults(searchQuery); // Implementation not shown
  res.send(`
   <h1>You searched for ${searchQuery}</h1>
   <p>Here are the results: ${results}</p>`);
});
```

To exploit this vulnerability, an attacker sends the user a link like this:

HTML

```
<a href="http://example.org/results?search=<img src=x onerror=alert('hello')">
  Get a free kitten!</a
>
```

When the user clicks the link:

1. The browser sends a GET request to the server. The request's URL parameter contains the malicious code.
2. The server extracts the URL parameter value and embeds it in the page.
3. The server returns the page to the browser, which runs it.

How to avoid cross site scripting vulnerabilities

Fortunately, applications built with modern web frameworks have fewer XSS bugs, because these frameworks steer developers towards good security practices and help mitigate XSS by using templating, auto-escaping, and more. However, developers need to know that problems can occur if frameworks are used insecurely, such as:

- *escape hatches* that frameworks use to directly manipulate the DOM
- React's `dangerouslySetInnerHTML` without sanitising the HTML
- React cannot handle `javascript:` or `data:` URLs without specialized validation

- Angular’s bypassSecurityTrustAs* functions
- Lit's unsafeHTML function
- Polymer's inner-h-t-m-l attribute and htmlLiteral function
- Template injection
- Out of date framework plugins or components
- and more

To safely display data exactly as a user types it in, output encoding is recommended. Variables should not be interpreted as code instead of text. First, when one wish to display data as the user typed it in, start with the framework’s default output encoding protection. Automatic encoding and escaping functions are built into most frameworks.
While not using a framework or need to cover gaps in the framework then one should use an output encoding library. Each variable used in the user interface should be passed through an output encoding function. A list of output encoding libraries is included in the appendix. There are many different output encoding methods because browsers parse HTML, JS, URLs, and CSS differently. Using the wrong encoding method may introduce weaknesses or harm the functionality of the application.

When users need to author HTML, developers may let users change the styling or structure of content inside a WYSIWYG editor. Output encoding in this case will prevent XSS, but it will break the intended functionality of the application. The styling will not be rendered. In these cases, HTML Sanitisation should be used.

There are some further things to consider:

- If you sanitize content and then modify it afterwards, you can easily void your security efforts.
- If you sanitize content and then send it to a library for use, check that it doesn’t mutate that string somehow. Otherwise, again, your security efforts are void.
- You must regularly patch DOMPurify or other HTML Sanitzation libraries that you use. Browsers change functionality and bypasses are being discovered regularly.

Malicious Extensions

Overview

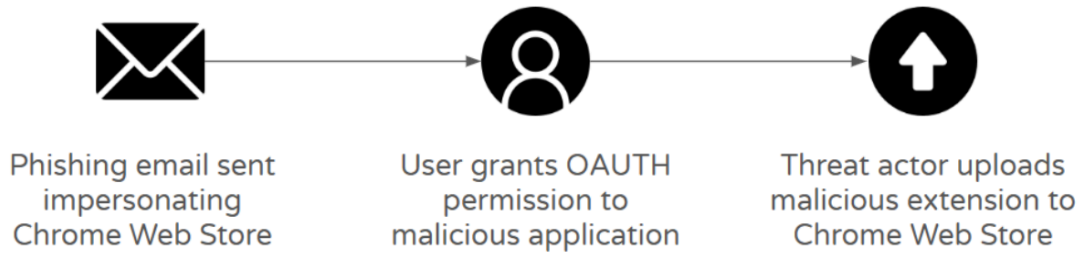
Browser extensions often improve user experience and allow users to work more efficiently. Sources estimate that the Chrome Extension store hosts over one hundred thousand unique extensions:
- Site DebugBear reported 111,933 extensions in August 2024
- chrome-stats lists the number of extensions as high as 145,316

Regardless of the exact number, most users have several extensions installed within their browsers. These can stem from Ad blockers, citation generators, or punctuation or writing aides.
While most extensions provide value to users, there have been several cases of malicious browser extensions being used to target users. There are different ways by which threat actors deploy

malicious browser extensions. The first is compromising existing plugins by exploiting vulnerabilities or compromising developer accounts. This gives a threat actor access to an existing plugin and its code, which can be modified to include malicious capabilities such as keylogging. This is how malicious code was added to the Cyberhaven extension, which is covered in-depth below. Similarly, compromising upstream libraries used by browser extensions may allow a threat actor to deploy code to benign plugins. Lastly, threat actors can design malware that operates as a browser extension. Rilide is an example of an information stealer deployed as a browser extension.

January 2025 Compromised Browser Extensions

The new year started with reports identifying at least 33 compromised Chrome browser extensions. A FieldEffect blog indicates that over 2.6 million users were impacted, and the compromised extensions were used for up to 18 months. One compromised browser extension was from Cyberhaven, a Data Loss Prevention software provider whose extension prevents users from entering data into unauthorized platforms.
Cyberhaven has released extensive details about how the compromise occurred and what they uncovered from their investigation. Access was obtained through a phishing email that targeted the extension's developers. The email claimed to be from the Chrome Web Store and outlined items that violated Google's policy and threatened to remove the extension from the Chrome Web Store. Users who interacted with the email granted OAUTH permissions to the malicious application. Once the malicious application was granted access, the threat actor used it to upload the malicious Cyberhaven extension to the Web Store.



Mitigation Strategies

For Home Users, being aware of what extensions are enabled and the permissions they grant is often the best way to prevent malicious extensions. Consider only installing essential extensions. Before installing any extension, review the permissions it requires and the details in the privacy section of its web store listing. Users can also use information on the web store, such as the number of users, owner, reviews, and last update time, to gain more information about the extension and its overall trustworthiness. Moreover, users should periodically review installed extensions to identify any that are no longer needed and can be removed.
For Corporate IT teams that control web browser settings for employees can use tools like Intune to enforce policies determining what extensions can be installed. Before restricting browser extensions within an environment, it would be beneficial to identify what extensions are currently used within the organization. This will allow teams to determine which extensions are required for business use.

BROWSERS SECURITY MECHANISM

Problem Statement

In this era of technology, with increased use during this pandemic, web browsers have been used more than ever. As more and more people are using web browsers, hackers are given more opportunities to attack. The owners of some of the most popular web browsers have had to increase their security and privacy. Owners of web browsers are not the only ones who should be taking actions to protect the user data. It is important for users to understand the threats to their privacy and security and how to manage their web browsers to protect their own data. It is hypothesized that while Google Chrome is one of the most popular browsers, it will not be the browser that upholds security and privacy as well as others.

The web browsers advertise to take their privacy seriously, compared to other browsers we are going to see how that holds up. Chrome, Firefox, Explorer, Edge, Safari, Opera, and Bing all have private browsing mode. Disparities begin when looking at if they block third-party tracking cookies by default, if they block crypto mining scripts, and if they block social trackers.

FireFox

Firefox is the only browser that takes precaution and performs all four of those actions. In this comparison Chrome comes out as the worst browser because it only does one of the four things, with Explorer being second worst because it only does two of four.

Google Chrome

Google Chrome is one of those web browsers that people are quick to use, and you would think that it would be the most secure due to the fact that it has so many daily users. Unfortunately, it is not. Google is said to be in the data sharing business, for example when logging into your Gmail account you are automatically syncing information into the browser itself and your browser is being tagged using cookies. Chrome is open to all cookies by default and leaving the user in the dark if no research was done prior to use, while the other browsers listed have cookie tracking off by default. Google is aware of how other browsers have handled the cookie problem and still defend their actions. In 2019, Chrome's then director of product management, Ben Galbraith said that "blunt cookie blocking solutions force tracking into more opaque practices". Saying that something is complicated does not mean taking no action and staying neutral by allowing cookie tracking is a good decision either. This could cause Chrome's reputation to plummet, if it hasn't already.

Microsoft Explorer

. Microsoft explorer as a whole is vulnerable so privacy on that browser will forever be an issue. Microsoft does not support Internet Explorer any longer and has warned current users that there is a critical vulnerability. It does seem as though the web browsers whose parent company has a business that could profit from user data is a browser you can side eye.

Bing

Bing happens to be one of those browsers, it is getting some of the flack similar to Chrome. Bing is not widely used but because it owns LinkedIn, and because LinkedIn's online ad division brings in so much revenue there can be an issue with trusting your private data in their browser.

Opera

Opera is another browser that does not have the best record in the category of browser privacy. Opera was sold to a Chinese consortium in 2016 which has led to privacy concerns. It is pretty common knowledge about how China is a communist state and how they do not take privacy as a big issue. With Opera being sold to a Chinese consortium, users can ask how private their information really is when it's owned by country that does not even uphold citizen privacy. In the

past, tech companies in China have been in the news for data sharing and privacy policy controversies.

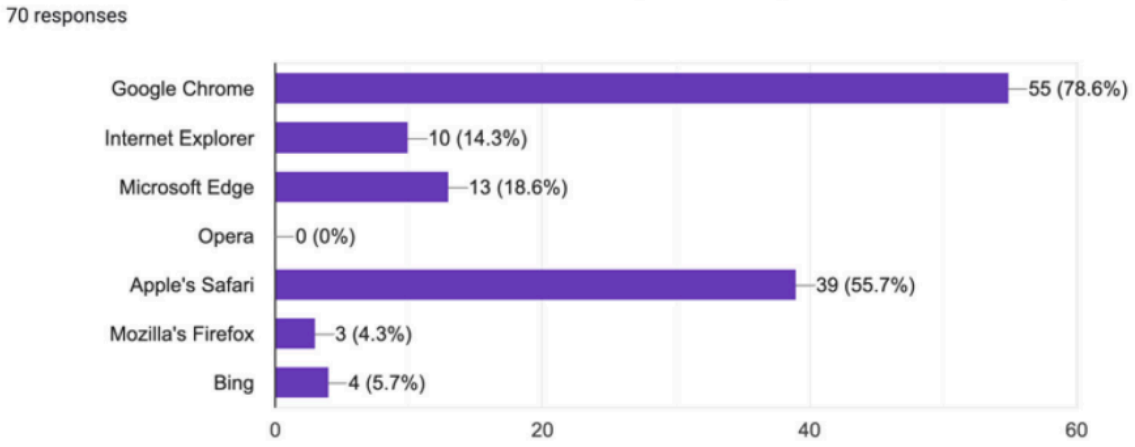
Safari

In the conversation about browser privacy Safari's Intelligent Tracking Program (ITP) has caused quite the disagreement. Four Google researchers have listed five explicit attacks that have exploited ITP's design; this happened because the list created from ITP's algorithm can be used by websites to discover information about websites Safari users have visited. Of course, Apple is in support of their program because they have built it. Apple and Google cannot agree on how to protect their users from cross site tracking, although ITP is more than what Google Chrome has implemented, which is nothing. Safari has also had some security concerns as of in 2021. The browser was under attack in the that year due to two bugs that were causing issues to WebKit, their rendering engine [23]. Safari experienced one of the same exploitations that Chrome experienced, a use after free flaw vulnerability exploitation. The important part, and the only thing that may invalidate the possible zero-day attacks on Safari are that "the bugs affect sixth-generation Apple iPhones, iPads, and iPod touch model hardware…".

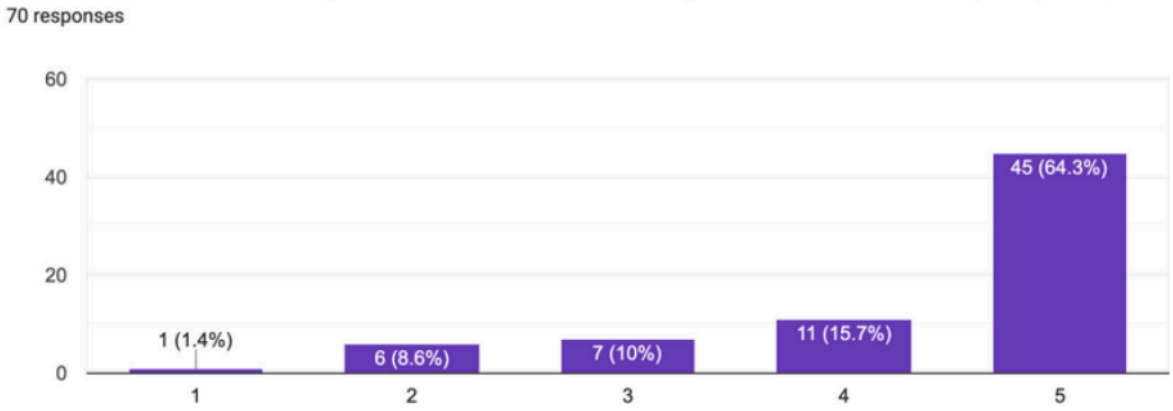
Comparison among the browsers

S. Michael Muchmore, a lead software analyst for software and web applications compared web browsers by conducting a variety of tests and assessing their usability. He conducted a speed test, a memory use test, a storage use test, and a compatibility test using the HTML5test website, through the JetStream benchmark, and the task manager. From his study, Chrome was found to be the most compatible with a score of 528 out of 555, Opera was close to Chrome, and Firefox and Safari were granted scores of 491 and 471 respectively. When it comes to speed, Chrome again won the race. It received the highest benchmark score on Windows 10, Safari of course won on macOS, and Firefox scored very low on both platforms. For the storage test, Opera actually scored the lowest, which was good for that test. It is the "slimmest on both macOS and Windows 10". Lastly, for the memory use test Safari scored the highest. Muchmore says that some browsers use sleeping tabs, which are tabs whose content is removed from memory, and this could have skewed results.

From a survey of users, . Google Chrome was the most commonly used web browser with 78.6% of people saying they currently used it as their web browser, Safari followed up with 55.7%. As far as care for web browser security and privacy goes, 64.3% of the participants said that they cared a lot about their web browser security and privacy. When asked about downloading extensions and plug-ins, 58.6% of participants said that they have downloaded extensions and/or plug-ins before. While the majority of participants said that they have downloaded extensions and/or plug-ins before, a low majority of participants, 34.3%, said that they did not know that their extensions and plug-ins must be updated. Opposite of this, 31.4% stated that they have automatic updates enabled for their browsers with the second majority, 24.3%, stating that they did not know it must be updated. Of 34 out of 70 participants that said their browser had been hacked before, 21 reported using Google Chrome as their browser at the time that they were hacked. When asked about cookies, 40% of participants stated that their cookie settings were set to allow all cookies. Half of the participants answered "whenever I feel like it" when asked how often they delete their browser history. To end the survey participants were asked what steps they take to secure their browser and 55.7% of participants answered saying that they block pop-ups with the next majority of 32.9% of people answering saying that they managed cookies and history.



Web Browser Usage

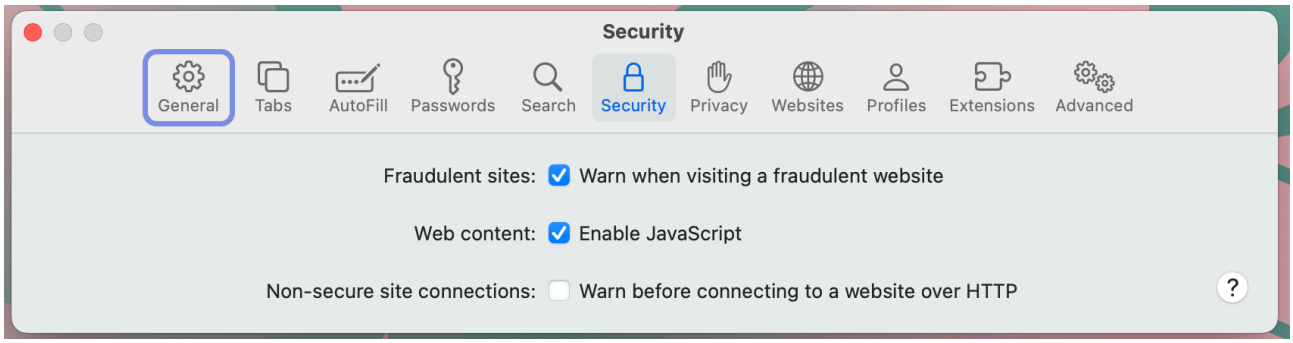


Privacy And Security

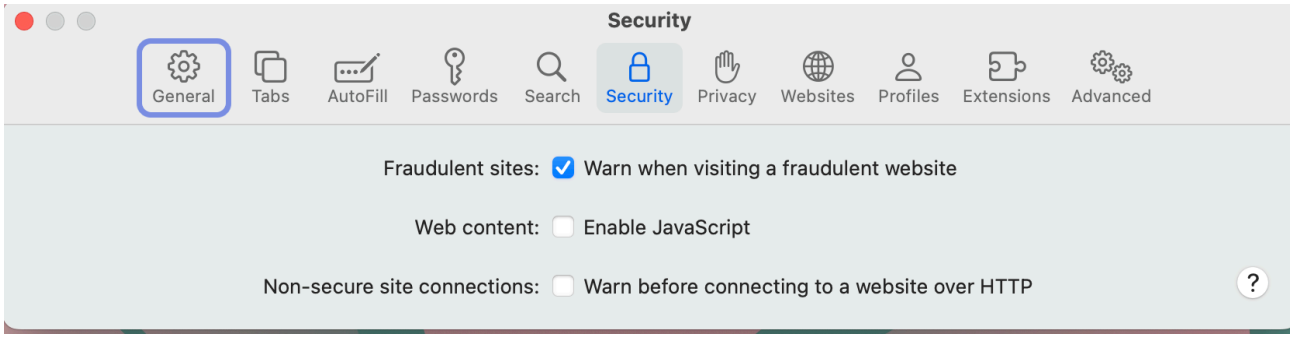
SETTINGS FOR HARDENING BROWSER SECURITY

Safari

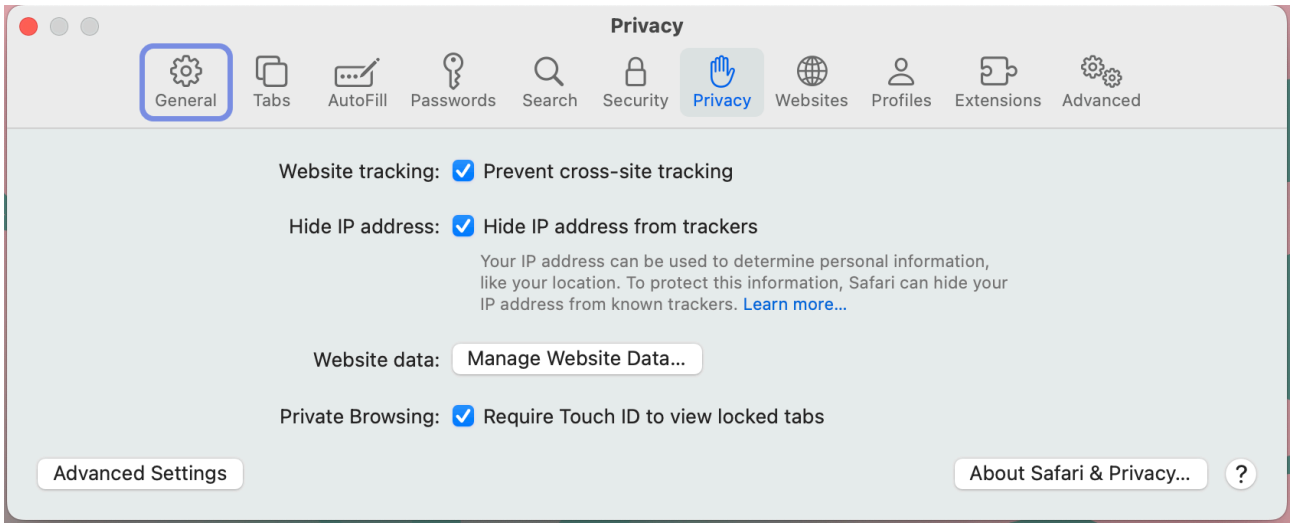
1. Enable Fraudulent Website Warning : Alerts you if you're about to visit a suspected phishing or scam site. To Enable, Open Safari Settings -> select Security tab -> Check “Warn when visiting a fraudulent website”



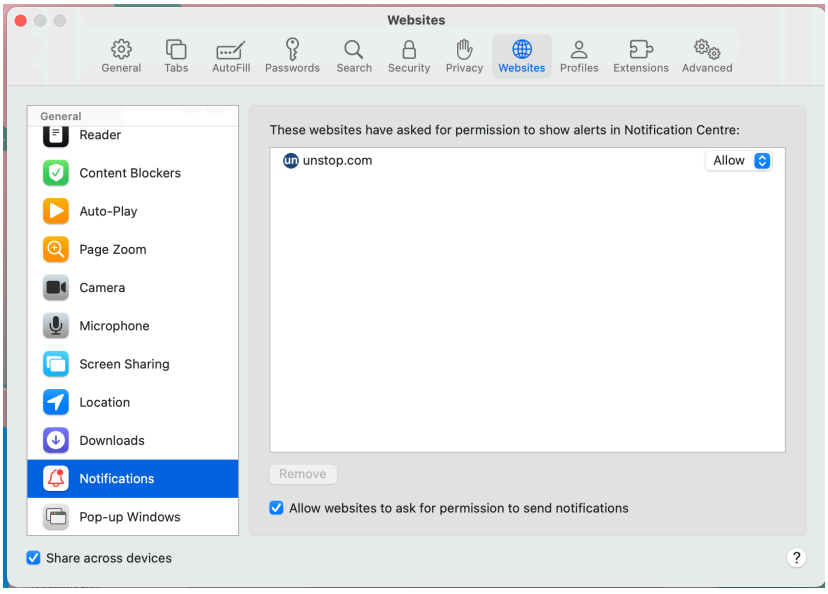
2. Disable JavaScript (Advanced Option): Blocks many web-based attacks but may break most websites. To Enable: Open Safari Settings -> select Security tab -> Uncheck "Enable JavaScript".



3. Prevent Cross Site Tracking: Stops advertisers and third-party content providers from tracking you. To Enable: Open Safari Settings -> select Privacy tab -> Enable "Prevent cross-site tracking".



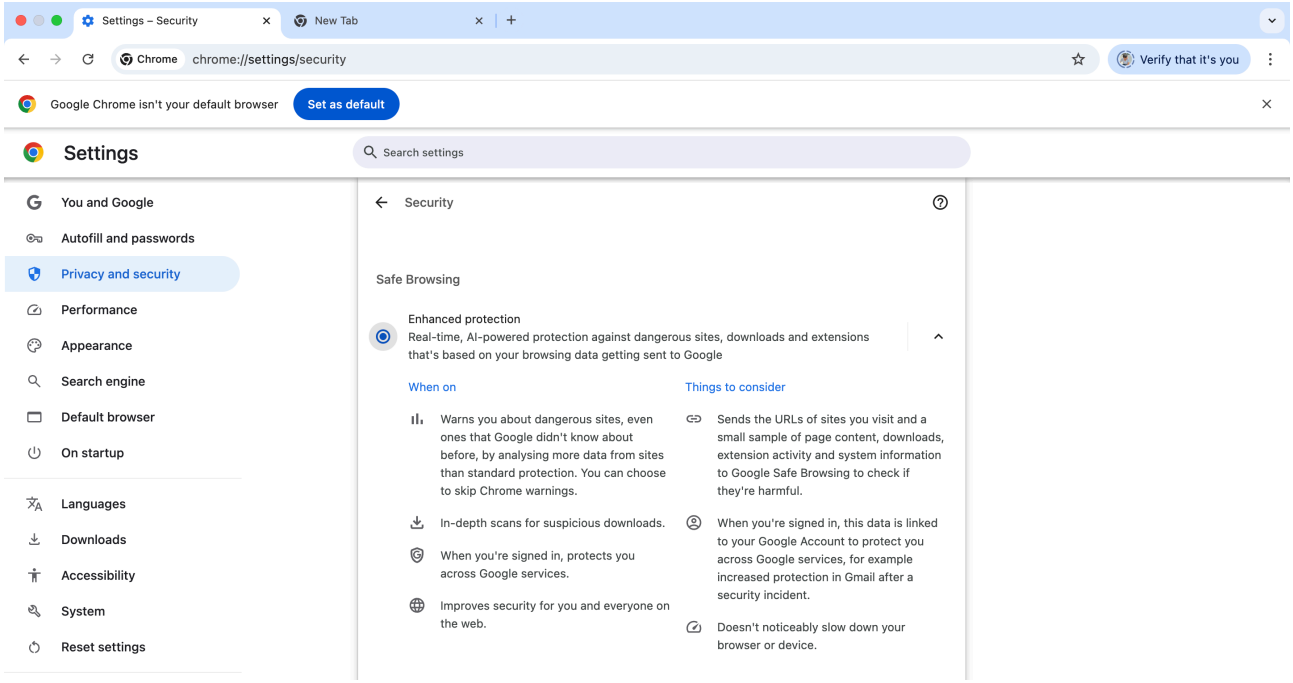
4. Enable Website Sandbox Restrictions: Restricts what websites can access.
Automatic in Safari (no user settings needed), but you can: Go to Safari → Preferences → Websites. Set permissions per website: Camera, Downloads, Notifications, Autoplay, etc.



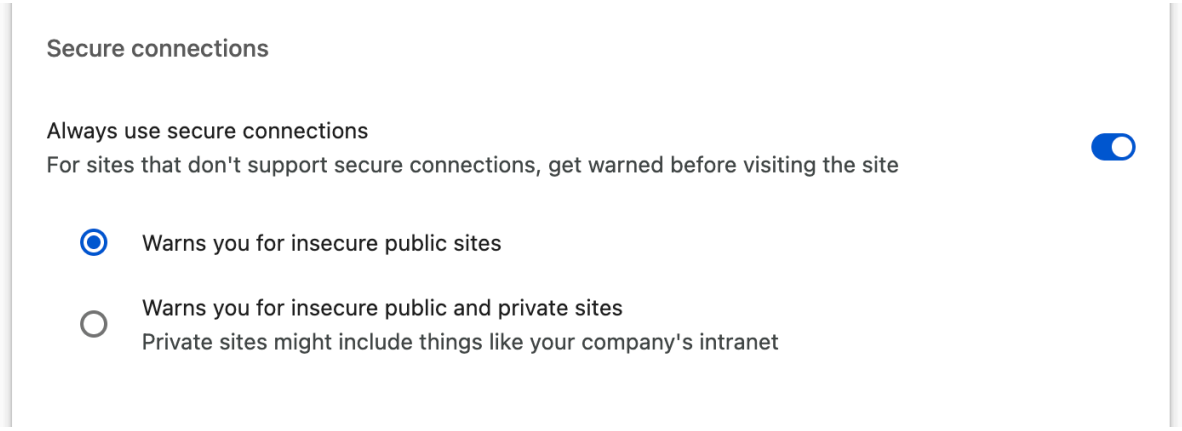
Safari is not compatible with uBlock Origin or NoScript, as these are not available in the Safari Extensions Gallery due to Apple's extension framework restrictions.

Google Chrome

1. Enable Safe Browsing (Enhanced Protection): Warns about dangerous sites, downloads, and extensions.
   To Enable: Go to chrome://settings/security, Under Safe Browsing, choose Enhanced protection

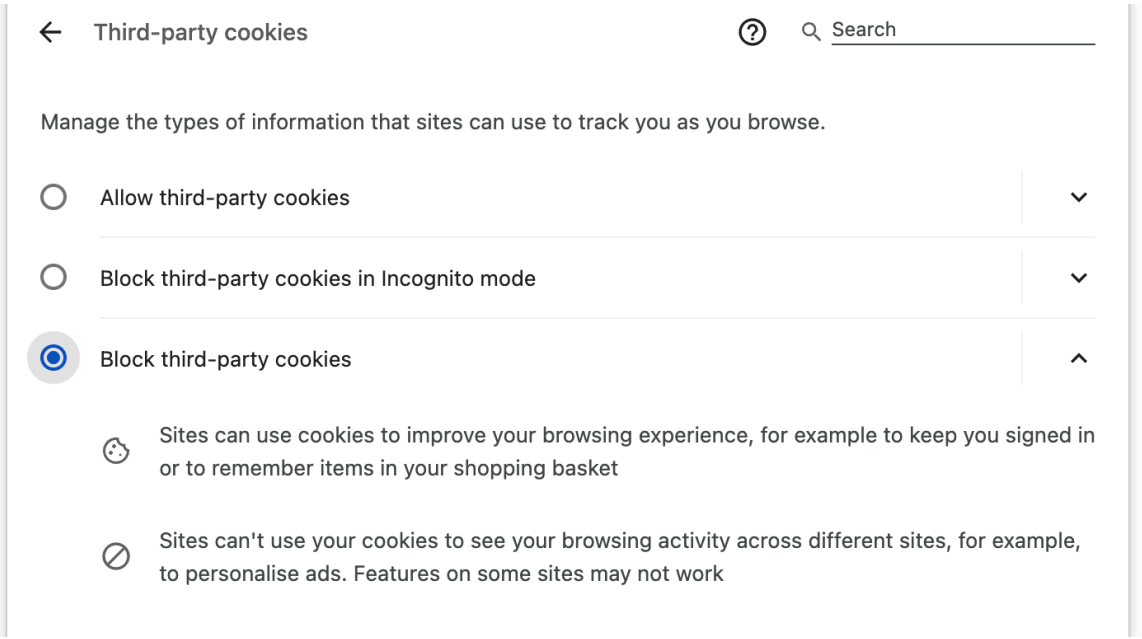


2. Turn on "Always use secure connections": Forces HTTPS connections where possible.
   To Enable: chrome://settings/security, Enable "Always use secure connections"



3. Block Third-Party Cookies: Prevents trackers from following you across sites.
   To Enable: Go to chrome://settings/cookies, Select Block third-party cookies

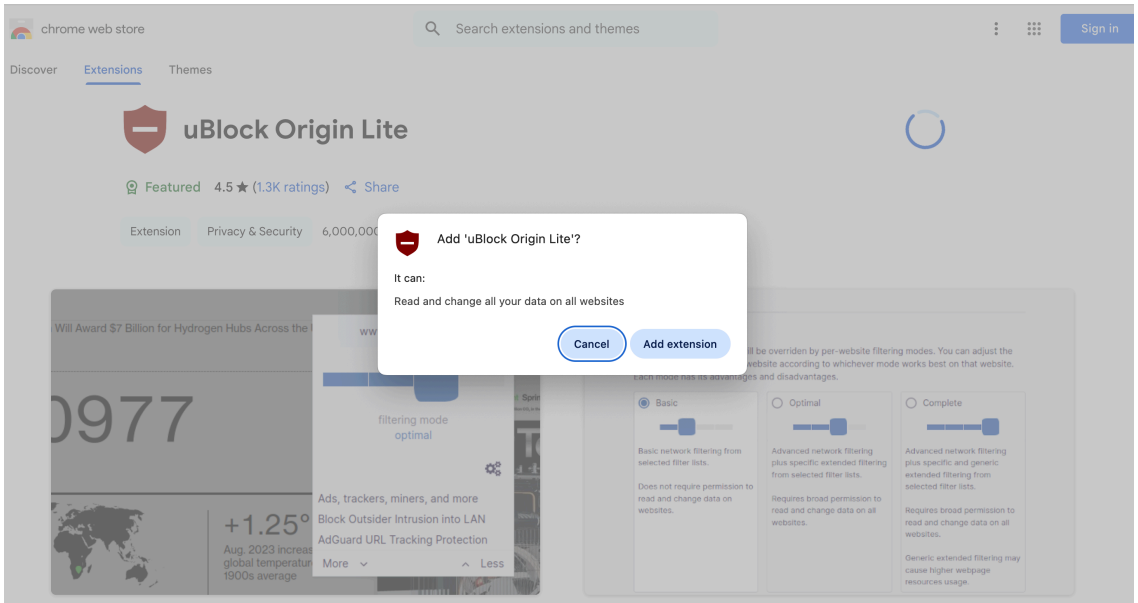
4. Clear Site Data on Exit: Removes cookies and site data when you close Chrome.
   To disable: chrome://settings/cookies, Enable, Clear cookies and site data when you close all windows.



Add Security Extensions in Chrome

How to Install uBlock Origin:
- Go to the official Chrome Web Store page: uBlock Origin – Chrome
- Click “Add to Chrome”
- Confirm by clicking “Add extension”

What it does:
- Blocks ads, trackers, crypto miners, malware domains.
- Extremely customizable.
- Lightweight and open-source.



ScriptSafe
Web Store: ScriptSafe – Chrome
Functionality:

- Allows script blocking per domain
- Blocks JavaScript, iframes, plugins, and more
- Gives fine-grained control

INTEGRATING SANDBOXING TECHNIQUES

Sandboxing security techniques and tools enable you to move suspicious software and files into an isolated environment—a sandbox—where the threat is tested. A sandbox is designed to mimic production environments, but it is deployed safely away from your real assets.
A major advantage of sandbox environments is the ability to isolate threats. Once the threat is isolated, you can test and analyze it, usually by "detonating" the suspicious file and causing it to deploy its malicious payload. The information gathered from the analysis can help you protect your systems from similar threats—essentially turning a zero-day threat into a known factor.

Sandbox security testing proactively detects malware by running suspicious code in a safe and isolated environment, and monitoring the behavior and outputs of the code. This is known as "detonation".

The major advantage of sandbox-based security testing is that it can reliably detect unknown threats. Other methods of testing, both traditional signature-based methods, and modern behavioral analysis based on machine learning (known as featureless detection), are limited in their ability to detect unknown threats.

These traditional methods are only as good as the threat databases and models that support them. The sandbox technique provides an additional layer of defense, making it possible to test payloads that passed other detection techniques, but may still contain threats.

There are three primary ways to implement a sandbox for security testing:

- Complete system emulation—the sandbox simulates the host's physical hardware such as CPU and memory to gain a comprehensive understanding of program behavior and impact.
- Operating system emulation—the sandbox emulates the end user's operating system, but does not accurately simulate system hardware.
- Virtualization / containerization—this method uses a virtual machine (VM) or container to run software in an isolated environment.

How Browsers Like Chrome Implement Sandboxing:

- Each tab runs in a separate OS process (renderer process).
- Processes are started with low privileges (e.g., `seccomp` on Linux, `AppContainer` on Windows).
- They cannot write to disk, access user files, or read clipboard/camera/microphone without permission.

Steps:
1. Isolate Tabs/Extensions: Run each in a separate process.
2. Drop Privileges: Use OS APIs to start processes with restricted permissions.
3. Apply Sandboxing Policies: Restrict file system, IPC, network, and hardware access.
4. Communicate via IPC: Use inter-process communication for necessary data sharing (e.g., between renderer and browser processes).

For Web Developers
Web developers cannot sandbox the browser itself but they can sandbox the content using a <iframe sandbox> tag

<iframe src="malicious.html" sandbox="allow-scripts"></iframe>

INTRUSION DETECTION SYSTEMS (IDS)

Intrusion is when an attacker gets unauthorized access to a device, network, or system. Cyber criminals use advanced techniques to sneak into organizations without being detected.

Intrusion Detection System (IDS) observes network traffic for malicious transactions and sends immediate alerts when it is observed. It is software that checks a network or system for malicious activities or policy violations. Each illegal activity or violation is often recorded either centrally using an SIEM system or notified to an administration. IDS monitors a network or system for malicious activity and protects a computer network from unauthorized access from users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between 'bad connections' (intrusion/attacks) and 'good (normal) connections'.

Working of Intrusion Detection System(IDS)
- An IDS (Intrusion Detection System) monitors the traffic on a computer network to detect any suspicious activity.
- It analyzes the data flowing through the network to look for patterns and signs of abnormal behavior.
- The IDS compares the network activity to a set of predefined rules and patterns to identify any activity that might indicate an attack or intrusion.
- If the IDS detects something that matches one of these rules or patterns, it sends an alert to the system administrator.
- The system administrator can then investigate the alert and take action to prevent any damage or further intrusion.

Intrusion Detection System Evasion Techniques
- **Fragmentation:** Dividing the packet into smaller packet called fragment and the process is known as fragmentation. This makes it impossible to identify an intrusion because there can't be a malware signature.
- **Packet Encoding:** Encoding packets using methods like Base64 or hexadecimal can hide malicious content from signature-based IDS.
- **Traffic Obfuscation:** By making message more complicated to interpret, obfuscation can be utilised to hide an attack and avoid detection.

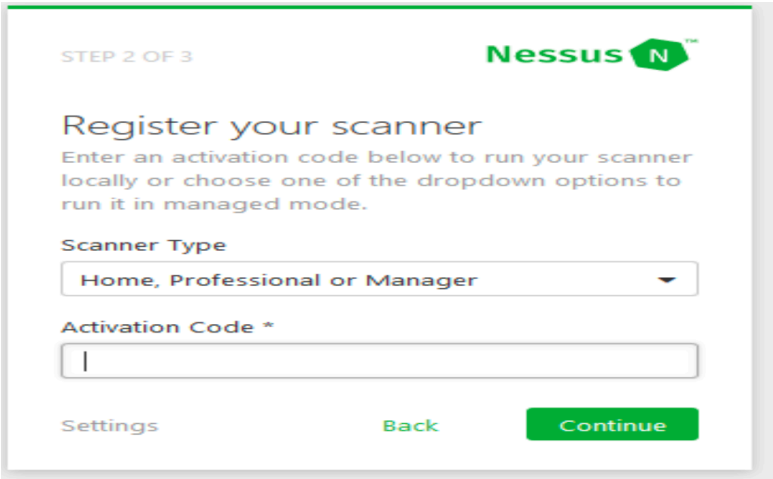
- **Encryption:** Several security features such as data integrity, confidentiality, and data privacy, are provided by encryption. Unfortunately, security features are used by malware developers to hide attacks and avoid detection.

Vulnerability Scanning through Nessus

NESSUS is a proprietary vulnerability scanner developed by Tenable Inc It is a commercial tools. But a free version is available with limited set of capabilities to scan maximum 20 machines per user. It is a remote security scanning tool which scans a system and raises alert when it discovers any vulnerabilities which malicious hackers can use to gain access to any system connected to networks. In this exercise you will learn about NESSUS which is a suite of tools that work together to run tests against target computers using a database of known exploits and weaknesses.

Downloading Nessus

1. Download NESSUS from https://www.tenable.com/downloads/nessus.
2. Download latest release available for windows 64
It is supports Windows Server 2008, Server 2008 R2*, Server 2012, Server 2012 R2, 7, 8, 10, Server 2016 (64-bit).
3. An activation code is also required to make NESSUS work after installation. Get activation code by clicking on GET ACTIVATION CODE on the download page (or by browsing https://www.tenable.com/products/nessus/activation-code)
It requires registration. After registration the activation code is sent to the registered email id.
4. Windows executable file is downloaded, which can be installed in one click.
5. After installation, web client starts with url: https://127.0.0.1:8834
6. Provide userid and password for account creation.
7. Copy activation code, received on email.
8. Nessus completes setup and starts downloading plug-ins and initializes. This may take some time.
9.After initialization, the nessus webclient must open in web browser. Sometimes the " Secure Connection error" may come. To resolve this error, open nessus in private window of your browser.



Scanning Hosts with Nessus

1. Once you have installed and launched Nessus, you're ready to start scanning. The home page ,when nessus starts.
2. First, you have to create a scan. To create your scan:
3. In the top navigation bar, click Scans.
4. In the upper-right corner of the My Scans page, click the New Scan button.

5. Click on new scan to create a new scan task. Next, click the scan template you want to use. Here we use Basic Network Scan which performs a full system scan that is suitable for any host. Use this template to scan an asset, for example, you can perform an internal vulnerability scan on your systems.

6. Click on desired scan type. (The following screenshots are for Basic Network Scan). Prepare your scan by configuring the settings available for your chosen template. The Basic Network Scan template has several default settings preconfigured, which allows you to quickly perform your first scan and view results without a lot of effort. Specify the name of scan and the targets. There may be a single target or a range of targets in a CIDR notation. After Specifying the name of scan and the target IP.

7. Click on Save. The next page will be following.8. 9. Click on blue arrow button to launch the scan.

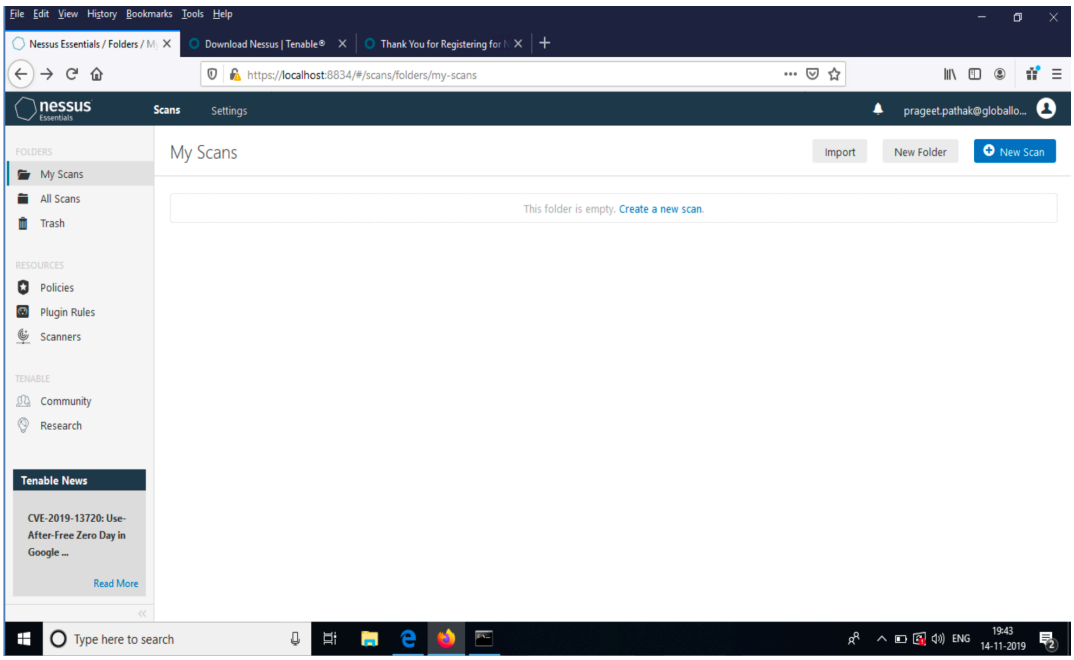
Scan starts. The time it takes to complete a scan involves many factors, such as network speed and congestion, so the scan may take some time to run.

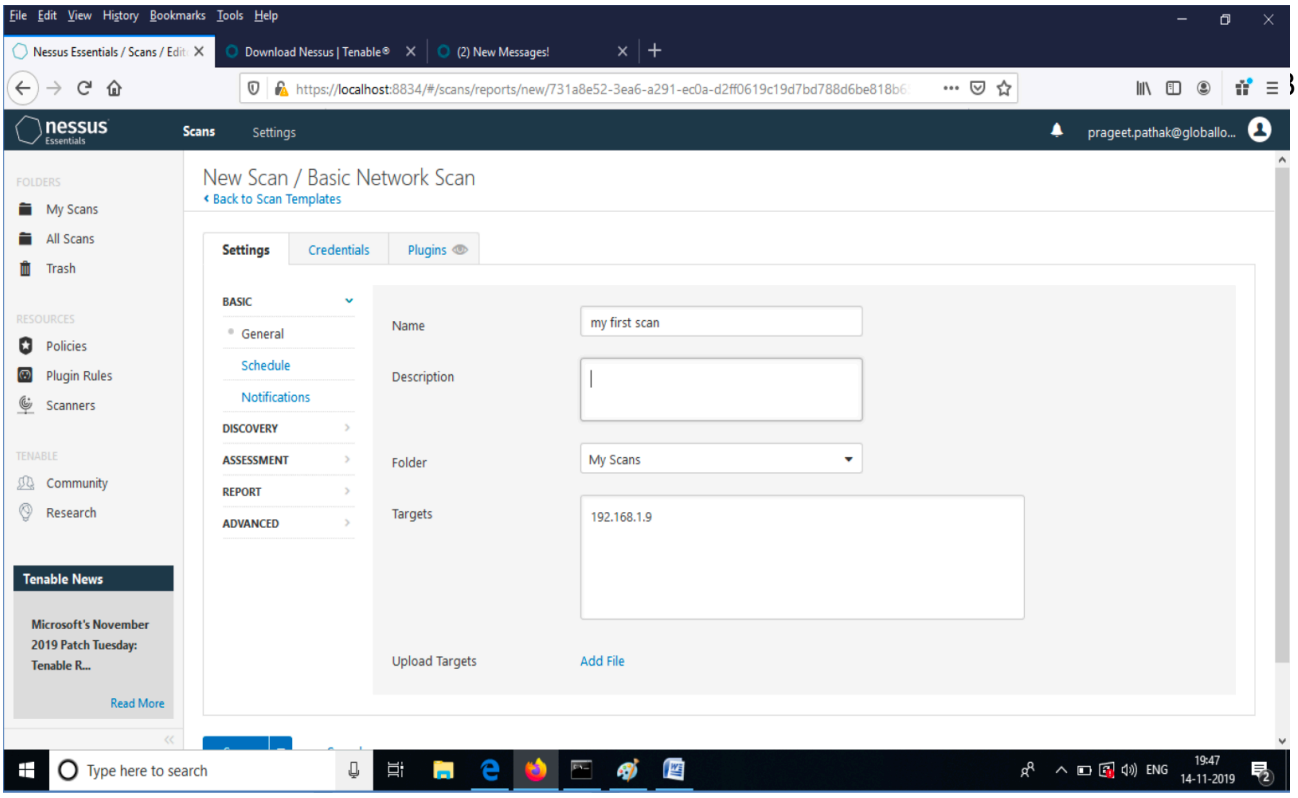
10. By clicking on Vulnerabilities (Bar), the details of vulnerabilities can be found. Viewing scan results by vulnerabilities gives you a view into potential risks on your assets.11. By clicking on any vulnerability details can be obtained.

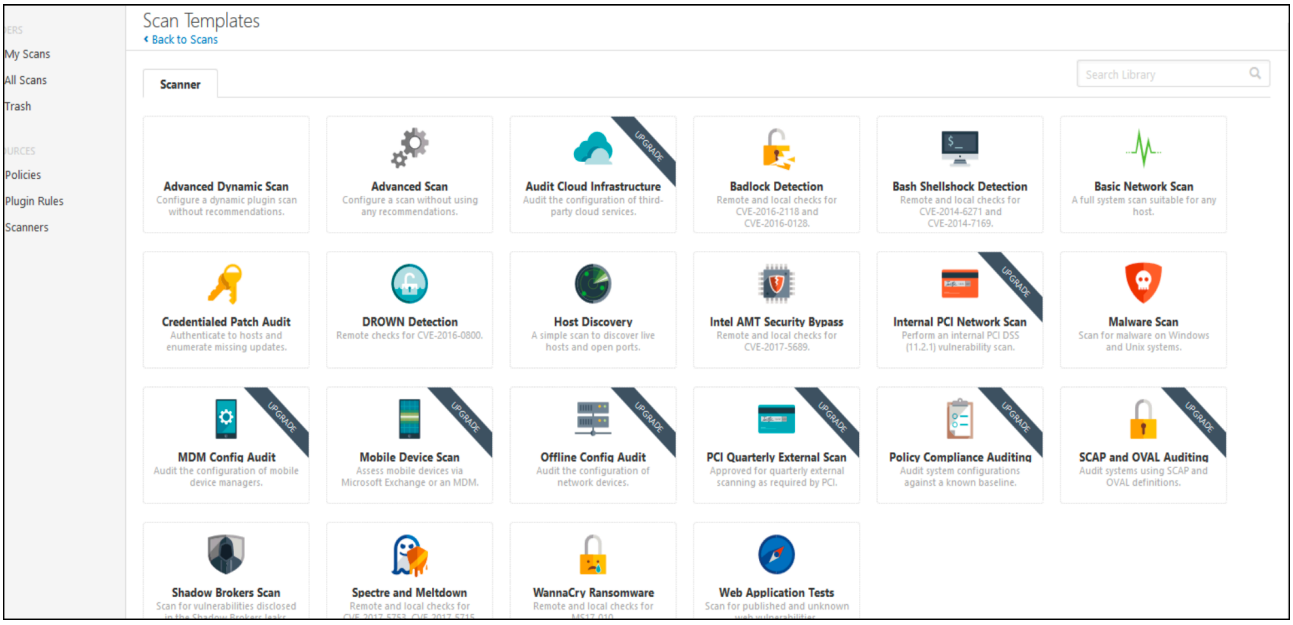
12. To view vulnerabilities:

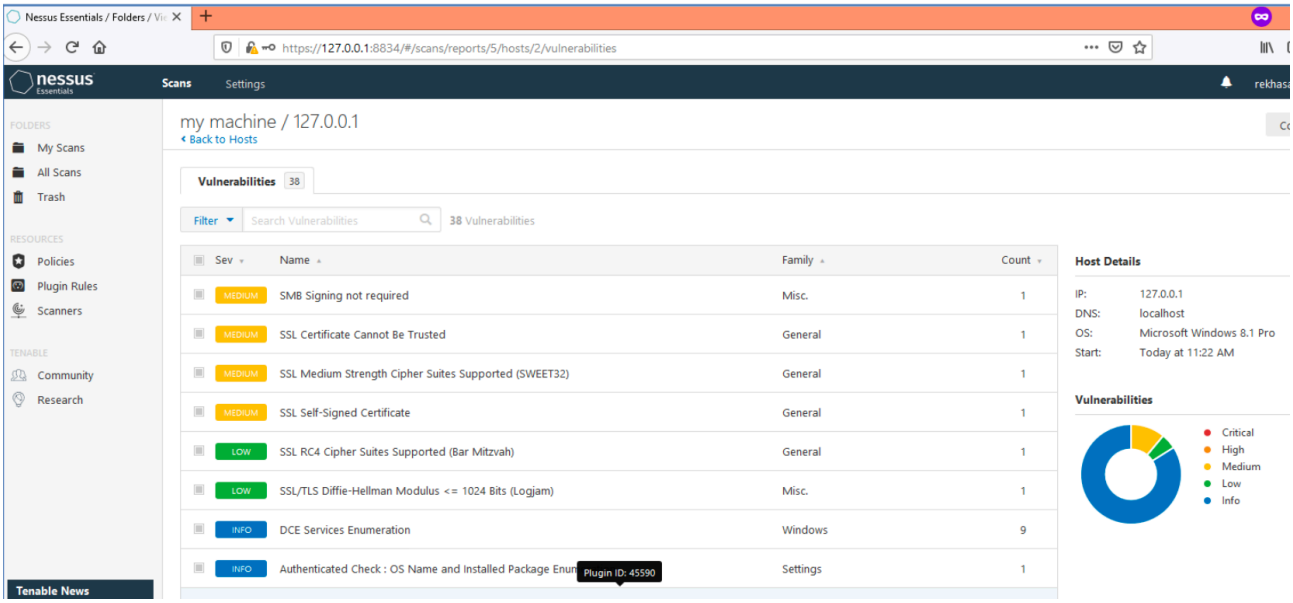
Do one of the following:

13. Click a specific host to view vulnerabilities found on that host.
14. Click the Vulnerabilities tab to view all vulnerabilities.
15. (Optional) To sort the vulnerabilities, click an attribute in the table header row to sort by that attribute.
16. Clicking on the vulnerability row will open the vulnerability details page, displaying plugin information and output for each instance on a host.




File Edit View History Bookmarks Tools Help
Nessus Essentials / Scans / Edit
Download Nessus | Tenable®
(2) New Messages!
https://localhost:8834/#/scans/reports/new/731a8e52-3ea6-a291-ec0a-d2ff0619c19d7bd788d6be818b6
nessus Essentials
Scans
Settings
prageet.pathak@globallo...
FOLDERS
My Scans
All Scans
Trash
RESOURCES
Policies
Plugin Rules
Scanners
TENABLE
Community
Research
Tenable News
Microsoft's November 2019 Patch Tuesday: Tenable R...
Read More
New Scan / Basic Network Scan
Back to Scan Templates
Settings
Credentials
Plugins
BASIC
General
Schedule
Notifications
DISCOVERY
ASSESSMENT
REPORT
ADVANCED
Name
my first scan
Description
Folder
My Scans
Targets
192.168.1.9
Upload Targets
Add File
Type here to search
ENG
19:47
14-11-2019


My Scans
All Scans
Trash
Policies
Plugin Rules
Scanners
Scan Templates
Back to Scans
Scanner
Search Library
Advanced Dynamic Scan
Configure a dynamic plugin scan without recommendations.
Advanced Scan
Configure a scan without using any recommendations.
Audit Cloud Infrastructure
Audit the configuration of third-party cloud services.
Badlock Detection
Remote and local checks for CVE-2016-2118 and CVE-2016-0128.
Bash Shellshock Detection
Remote and local checks for CVE-2014-6271 and CVE-2014-7169.
Basic Network Scan
A full system scan suitable for any host.
Credentialed Patch Audit
Authenticate to hosts and enumerate missing updates.
DROWN Detection
Remote checks for CVE-2016-0800.
Host Discovery
A simple scan to discover live hosts and open ports.
Intel AMT Security Bypass
Remote and local checks for CVE-2017-5689.
Internal PCI Network Scan
Perform an internal PCI DSS (11.2.1) vulnerability scan.
Malware Scan
Scan for malware on Windows and Unix systems.
MDM Config Audit
Audit the configuration of mobile device managers.
Mobile Device Scan
Assess mobile devices via Microsoft Exchange or an MDM.
Offline Config Audit
Audit the configuration of network devices.
PCI Quarterly External Scan
Approved for quarterly external scanning as required by PCI.
Policy Compliance Auditing
Audit system configurations against a known baseline.
SCAP and OVAL Auditing
Audit systems using SCAP and OVAL definitions.
Shadow Brokers Scan
Spectre and Meltdown
WannaCry Ransomware
Web Application Tests
UPGRADE


Nessus Essentials / Folders / Vi
https://127.0.0.1:8834/#/scans/reports/5/hosts/2/vulnerabilities
nessus Essentials
Scans
Settings
my machine / 127.0.0.1
Back to Hosts
Vulnerabilities 38
Filter
Search Vulnerabilities
38 Vulnerabilities
Sev
Name
Family
Count
MEDIUM SMB Signing not required Misc. 1
MEDIUM SSL Certificate Cannot Be Trusted General 1
MEDIUM SSL Medium Strength Cipher Suites Supported (SWEET32) General 1
MEDIUM SSL Self-Signed Certificate General 1
LOW SSL RC4 Cipher Suites Supported (Bar Mitzvah) General 1
LOW SSL/TLS Diffie-Hellman Modulus <= 1024 Bits (Logjam) Misc. 1
INFO DCE Services Enumeration Windows 9
INFO Authenticated Check : OS Name and Installed Package Enu Plugin ID: 45590 Settings 1
Host Details
IP: 127.0.0.1
DNS: localhost
OS: Microsoft Windows 8.1 Pro
Start: Today at 11:22 AM
Vulnerabilities
Critical
High
Medium
Low
Info
Tenable News

WEB APPLICATION FIREWALLS

A WAF or web application firewall helps protect web applications by filtering and monitoring HTTP traffic between a web application and the Internet. It typically protects web applications from attacks such as cross-site forgery, cross-site-scripting (XSS), file inclusion, and SQL injection, among others.

A WAF is a protocol layer 7 defense (in the OSI model), and is not designed to defend against all types of attacks. This method of attack mitigation is usually part of a suite of tools which together create a holistic defense against a range of attack vectors.

By deploying a WAF in front of a web application, a shield is placed between the web application and the Internet. While a proxy server protects a client machine's identity by using an intermediary, a WAF is a type of reverse-proxy, protecting the server from exposure by having clients pass through the WAF before reaching the server.

A WAF operates through a set of rules often called policies. These policies aim to protect against vulnerabilities in the application by filtering out malicious traffic. The value of a WAF comes in part from the speed and ease with which policy modification can be implemented, allowing for faster response to varying attack vectors; during a DDoS attack, rate limiting can be quickly implemented by modifying WAF policies.

Steps to deploy Web Application Firewall (WAF)

1. Create a Cloudflare Account → https://www.cloudflare.com/
2. Add your domain.
3. Update DNS nameservers to point to Cloudflare.
4. Go to Security > WAF in the dashboard.
5. Enable:
6. OWASP Managed Rules
7. Rate Limiting
8. Bot Mitigation
9. Monitor threats in the Firewall Events log.

CONCLUSION & RECOMMENDATIONS:

Conclusion

Web browsers are an indispensable tool in the modern digital ecosystem, making them a primary target for attackers seeking to exploit vulnerabilities. This project explored in detail the mechanics of browser exploitation, the vulnerabilities that make such attacks possible, and the preventive strategies that can be adopted to strengthen browser security.

Through rigorous literature review, simulated attacks, and practical implementation of tools and techniques such as browser hardening, sandboxing, IDS, WAF, and vulnerability scanners like Nessus, a comprehensive understanding of the threat landscape was developed.

Our findings highlight that while browsers like Chrome, Firefox, and Safari have made significant strides in integrating security features, user awareness and proactive configuration play a crucial role in preventing exploitation. Attacks like cross-site scripting, drive-by downloads, and malicious extensions often succeed due to outdated software or lax user settings.

Additionally, the project identified several Indicators of Compromise (IOCs) that can help detect attacks post-exploitation, such as unauthorized extensions, unexpected data traffic, and changes to browser behavior.

Recommendations

Based on the analysis and outcomes of this project, the following recommendations are made:

1. **For Users:**

- Always keep browsers and extensions up to date.
- Avoid installing unknown or poorly reviewed extensions.
- Use security-focused plugins like uBlock Origin and NoScript.
- Enable privacy settings such as "Block third-party cookies" and "Clear site data on exit".

2. **For Organizations:**

- Implement centralized browser configuration policies (using GPO, Intune).
- Use endpoint protection integrated with browser monitoring.
- Regularly perform vulnerability scans using tools like Nessus or OpenVAS.
- Deploy IDS/IPS and WAF at the network level.

3. **For Developers:**

- Sanitize and validate user inputs thoroughly to avoid XSS.
- Use modern frameworks with built-in XSS and CSRF protections.\
- Leverage sandbox attributes (`<iframe sandbox>`) for untrusted content.

4. **For Security Teams:**

- Maintain updated threat intelligence on newly discovered browser vulnerabilities.
- Use sandbox analysis to test unknown downloads or plugins.
- Monitor network logs for signs of drive-by download behavior or command-and-control traffic.

LIST OF REFERENCES:

- Web Browser Security And Privacy: https://par.nsf.gov/servlets/purl/10344954
- Red Teaming - OSINT - Phishing: https://owasp.org/www-chapter-dorset/assets/presentations/2020-04/RT_OSINT_Phishing.pdf
- Malware: https://sosafe-awareness.com/glossary/malware/
- Defending against Malware: https://www.sans.org/newsletters/ouch/defending-against-malware-invisible-enemy/
- Whatsapp Web ClickJacking Chain: https://www.securityweek.com/whatsapp-vulnerability-could-facilitate-remote-code-execution/?utm_source=chatgpt.com
- OWASP Cross Site Scripting: https://owasp.org/www-community/attacks/xss/
- Drive-by Downloads: https://nordlayer.com/blog/what-is-drive-by-download/
- Compromised Browser Extensions - https://blog.pulsedive.com/compromised-browser-extensions-a-growing-threat-vector/
- Sandboxing Security: https://perception-point.io/guides/sandboxing/sandboxing-security-practical-guide/
- Intrusion Detection System: https://www.geeksforgeeks.org/intrusion-detection-system-ids/
- Nessus Lab Manual