Web Browser Security: Different Attacks Detection and Prevention Techniques

Patil Shital Satish
M.Tech 2nd year
CSE, Department
SGGSIE & T
Nanded-431606

Chavan R. K.
Associate Professor
CSE, Department
SGGSIE & T
Nanded-431606

ABSTRACT
In this paper, we present a systematic study of how to make a browser secure. Web browser is vulnerable to different attacks; these attacks are performed due to vulnerabilities in the UI of the web page, Browser cache memory, extensions, plug-in. The Attacker can run malicious JavaScript to exploit user system by using these vulnerabilities. Buffer overflow attack, Cross-site-scripting, Man-in-the-middle, Extension vulnerability, Extreme Phishing, Browser Cache poisoning, Session hijacking, Drive-by-download, Click-jacking attacks are discussed. Browser with electrolysis system and sandboxed processes are discussed to prevent the browser from attack.

General Terms
Chrome process, Sandboxed process, Web Extension, Electrolysis

Keywords
Web application security, Heap overflow, Electrolysis, Sandboxing

1. INTRODUCTION
In today's Internet world, security is a widespread term. Web, Internet-based social networking turn into an essential part for all persons. Security becomes an important issue because the number of attacks against systems is increasing rapidly. Attacks are performed to steal private as well as financial information of a web user. Malicious content loaded into the system without knowledge of a user is a frequent problem for host systems. [40]Nature of problems is same for Smartphone, Desktops. The malevolent substance, for example, infections, Trojans, malware, and vulnerabilities in the frameworks are significant threats.

Vulnerable system or vulnerabilities in the system is a significant factor for the attack. Different Vulnerabilities are used to perform different attack. The Most influential factors in the accomplishment of a threat are the success of delivery of a malware and its execution. By using SMTP execution of threats become easy. Mailborne threats are commonly used to entice the recipient into executing the malicious attachment. The delivery mechanism does not depend upon user action rather most common path is to exploit some application. System framework defenselessness with a specific end goal to pick up execution. Abusing vulnerabilities in the client browser stipulates a component for malware to pick up execution when the victim peruses a malicious page. [3] [23]

2. 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. As shown in figure.

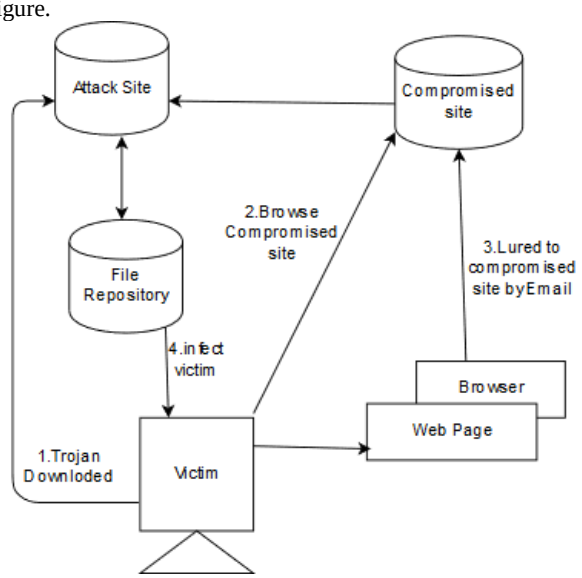

Figure 1. Role of the Web

Malicious sites: Attacker compromises a legitimate site and creates a new site used as a launch area for an attack. 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.

3. 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 behavior 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. [31] [39] Following table shows XPCOM Interface and Possible impact in web browser.

Table 1.XPCOM Interface and Possible Impact

XPCOM Interface	Possible Impact
nsIHistoryListener	Notifies when a new document is open to third party
nsIHttpChannel	Allows access to HTTP GET query parameter
nsIPasswordManager	Might reveal user stored password
nsIRDFDataSource	Write access critical data objects(extension manager)
nsICookieManager	Expose user cookies
nsIDownloader	Download malicious file into user system

3.1 Web Browser Architecture

A Browsing page or frame encloses presentable content and a JavaScript execution environment such as heap or code that interact with content **[47].** Document Object Module (DOM) has control over interaction with content. Nesting of browsing context performed by using IFRAMES. They also read and write persistent storage like cookies .A network requests can issue implicitly in page content that references URL retrieved over the network. Network request also can issue in JavaScript using the XMLHttpRequest (XHR).They communicate by sharing DOM objects. JavaScript language used to display a client-side web page. Attacker attacks on the website by using malicious JavaScript. JavaScript is downloaded into the browser and executed by an embedded interpreter. The centralized repository for extension is known as "Add-On" in Mozilla and Web store in Chrome. .Extensions can directly access private browsing information such as cookies, history and password stores. DOM is responsible for rendering a web page.

DOM Manipulation: The DOM is a Programming interface used to interface with the document .This Programming interface is accessible in various languages as a library. The browser changes all HTML in a page to a tree in light of the nesting. In the event that client need to change any HTML, client can communicate with the DOM Programming interface keeping in mind the end goal to do as such,

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

In first.js reference the body using:

```
onload=function()
{document.getElementsByTagName('body')[0].style.display=
'none';}
```

The getElementsByTagName is a method of the document object. Here manipulating the body element, this is a DOM element. If someone wanted to traverse and find say,

```
onload = function()
{var els = document.getElementsByTagName('*');
for ( var i = els.length; i–; )
 {if (els[i].nodeType == 1&&
els[i].nodeName.toLowerCase() == 'span' ){ alert( els[i] )}}}
```

Traversing the nodeList given back by getElementsByTagName , and looking for a span based on the nodeName property. [41]**.**

Mozilla Platform Browser code is written in different languages like C, C++, and JavaScript. The Large code is partitioned into the different small component and the mechanism of integration of this code is called as Cross-Platform Component Object Model. Each component has unique classID and contractID and they implement one or more interfaces. The functionality of a component specified by using methods and variables which are included in interfaces. Interfaces are reference counted. XPConnect permits JavaScript program access and controls XPCOM objects. It is utilized amongst DOM and JavaScript. All interfaces of an XPCOM objects must be declared in XPIDL. XPIDL compiler is utilized to create both C++ header files and type lib files. The type-lib files are the binary representation of at least one interface.

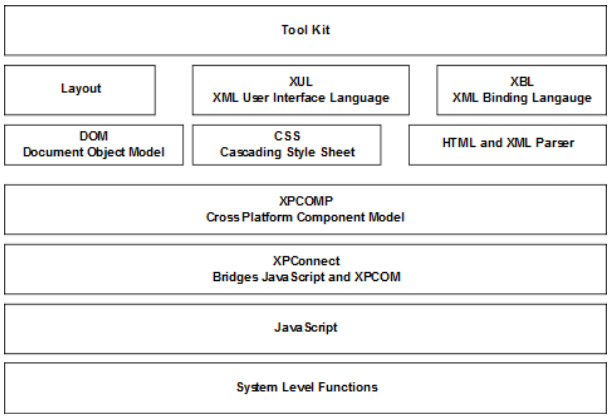

Figure 2.Mozilla Platform

JavaScript is utilized to access and manipulate objects in the DOM to make a dynamic situation for documents. XPConnect is utilized to find DOM object using DOMClassInfo.[43] [44]

4. WEBEXTENSIONS

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

Table 2.Comparison between XUL extensions and Web Extensions

XUL/XPCOM extensions	WebExtensions
Uses two manifest files:1.install.rtf 2.chrome.manifest	Uses Only One manifest file: manifest.json
Extensions can directly manipulate XUL. API: Customazible UI.jsm	WebExtensions does not get direct access UI. API: browser Action API, page Action, commands, context Menus
Get access to the full privileged set of XPCOM APIs and JavaScript code modules through the Components object. Access to browser internal through Browser.	Get access to a limited set of JavaScript API through background scripts. Also get a window global with all the DOM objects available on normal web page.

Gets direct access to web content using Browser .Refactoring the code using frame script for multiprocess.	Compatible by default, code that interacts with web content using the content script.
Localization: using local statements inside the chrome. Manifest then include localized strings in UI elements or in code.	Don't have direct access. Supply localized strings as a collection of JSON files.

WebExtensions should be easier to use than the existing Firefox XPCOM/XUL APIs. [4].Following figure shows structure of WebExtensions.

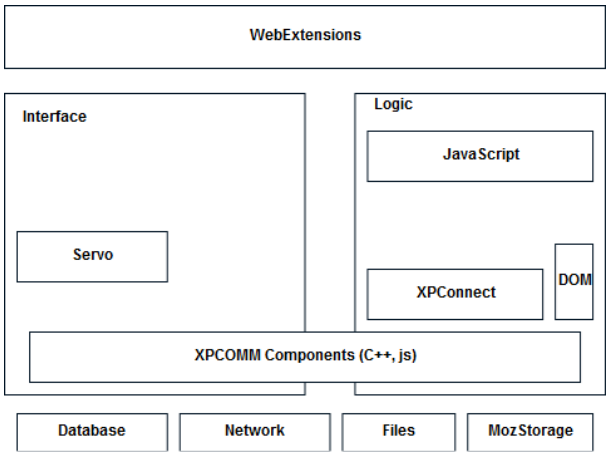

Figure 3.WebExtensions Structure

4.1 Servo: Servo is a trial web browser and the objective is to make another layout engine utilizing an advanced programming language. It is created by Mozilla Research. The model tries to make an exceedingly parallel condition, in which numerous parts like rendering, design, HTML parsing, image decoding are fine-grained, multiple isolated task. The project is composed in the Rust programming language. Two significant components are utilized by Servo depend on prior C++ code from Mozilla. JavaScript support is given by SpiderMonkey, and the 2D designs library .Sky blue is utilized to interface with OpenGL and Direct3D utilizing parallelism and code safety, to accomplish more prominent security and execution versus contemporary programs. Servo is likely to be combining Gecko for making available the Servo API in Firefox.

4.2 Electrolysis and Sandboxing

Elecrolysis: In multiprocess Firefox there are two processes: Chrome process and content process. The Chrome or parent process runs browser chrome or UI as trusted process which controls interaction with the underlying operating system. The parent process is not sandboxed and has regular access to the operating system. It can also access files, devices and network resources. Chrome process should only run trusted code. [56] A child process should run all untrusted web content. The parent process also acts as a broker for privileged resource requests from the child process.

Chrome privileged JavaScript code in one process can communicate with chrome-privileged JavaScript code in a different process by using Message manager. [55] **1) Frame message manager:** FMM enables chrome process code to load a script into a browser frame in a single browser tab in the content process. It is called as frame script and scope is limited within the browser frame. **2) Content frame message manager:** A Content frame message manager is provided for every open tab. It is the content-side end of frame message manager conversations. Messages from Chrome message managers end up when Frame scripts are loaded into the content frame message manager scope. **3) Process message manager:** PMM corresponds to process boundaries. Process boundaries enable code running in the parent (chrome) process to communicate with code running in the child (content) process. Chrome process uses the different message manager such as global frame message manager, window message managers, and browser message managers. This operates on all frames, in all content tabs. If you load a frame script using the global frame message manager, the script gets loaded separately into every open tab. [57]

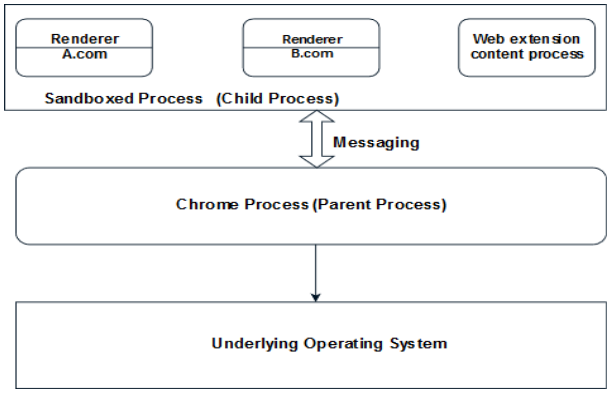

Figure 4 . Electrolysis and Sandboxing

Sandboxing: Sandboxing will be an effective security control; Firefox must be split into two different processes called as parent and child processes. The child processes is responsible for running untrusted web content. Due to this child process can be restricted to limit damage in the event of the compromise. In the Electrolysis project parsing and execution of web content is moved to a content process and sandboxing is based on this process model. A child process is untrusted and intended to run remote web content. [58] A sandbox provides restricted privileges to the child process. Child processes parses and executes html and JavaScript code corresponding to a web page.

The Content process is nothing but a process which runs the html code corresponding to a tab. This web content process is responsible for parsing and executing all the web content currently loaded in the browser tabs that are open. Content processes contain privileged code responsible for the implementation of DOM APIs and code which connects back to the parent to load the resources. The content process is sandboxed and prevented from direct resource access. The Content process only has the ability to execute web content. [33] [59]. Mozilla has several major changes lined up for Firefox, including the Servo rendering engine and the Electrolysis multi-process model. WebExtensions are supported by Mozilla's Add-On repository service and it will replace XUL based extensions.

5. ATTACKS ON BROWSER

5.1. Buffer Overflow: Buffer Overflow attacks are specified by overwriting of memory segments of process. Exceptions, segmentation faults, and other errors are occurred because of overwriting values of the IP, BP, and other registers. These errors bring execution of the application in an unexpected way. [51] Heap Overflow: JEMalloc Memory allocator is used in Firefox, vulnerable to a heap overflow. We plan heap overflow by placing a victim object in the same run

to the vulnerable object. The victim that can help us achieve arbitrary code execution.[61] Heap underflows: when heap objects are very small to store input. Dangling pointers or "use-after-free" error occurs when a program frees an object that is still in use before the due time. Uninitialized reads: when programs read from newly allocated objects such object carry data of old freed object .[62]Stack-based attack: When the submitted data of an input string is evaluated as a command by the application the Format String exploit occurs. It is Very easy to write program for BOF. [7] [15] [18]

```
/* Program for Buffer overflow Attack.*/
 # include <stdio.h>
void f((char*)) { char buffer [10]; strcpy ((buffer,s))}
 void main ((void)) { f (("98765432109876543210")) }
```

This program will result in segmentation fault. A simple mistake can lead to buffer overflow attack.It is very difficult to prevent. [50]

5.2. Cross-Site Scripting: This vulnerability makes it happen for attackers to inject malicious code like JavaScript programs into victim's web browser. Cross-Site Scripting vulnerability allows assailants to infuse malicious code like JavaScript projects into victim's web program. Using this malicious code, the attackers can steal the victim's credentials, like cookies, and passwords. [52] The content of the HTML page can be rewrite by using malicious scripts. Stored XSS Attacks: It is also known as Persistent or Type-I XSS. Stored attacks are the ones where the injected script is permanently stored on the target servers. They can store in the database, in a message forum, visitor log, comment field. Reflected XSS Attacks: It is also known as Non-Persistent, Type- II XSS. In this attack the infused script is reflected off the web server. For example, in the hunt result every reaction that incorporates the info sent to the server as a part of the request. [6] [8] [16]

5.3. Man-in-the-Middle: This attack can be accomplished by using arp poisoning, DNS spoofing methods. A Man-in-the-middle attack also called as bucket brigade attack. MITM is an attack where the assailant access and perhaps modifies the correspondence between two gatherings without their knowledge. [23]Victim believes they are directly communicating with each other. Active eavesdropping is one of the examples of a man-in-the-middle attack. In which the attacker makes autonomous associations with the casualties and retransmit messages between them to make them trust that they are talking specifically to each other over a private connection. Actually the whole discussion is controlled by the attacker. The attacker must have the capacity to remove every single relevant message going between the two casualties and infuse new ones. [1] [11] [24]

5.4. Extension vulnerability: In Firefox extension architecture same JavaScript namespace is shared between all JavaScript extensions installed on a system .Any extension can modify, read, write to other global namespace and introduces namespace pollution problem. In extension reuse, vulnerability attacker uses an existing extension to make API calls and Resource access to hide malicious extension. Extensions interact with web page without any explicit request for MIME type. A browser extension has the same privilege as the Browser itself. The extension additionally has full access to browser and clients working system. Extensions can change the usefulness of the program, behavior of site, access to file framework. An active attacker regulates content loaded via HTTP and reuse it .By replacing this script attacker hijacks extensions privileges and install malware. A JavaScript capacity break is another reason for misuse of extension. [46] [47]

5.5. ***Extreme Phishing:*** This attack support dynamic user interaction. Web Single Sign-On (SSO) systems are significant trend in inline user authentication. OpenID and OAuth are open Web SSO standards rapidly gaining adoption on the Web. In this system one single IDP account is used to sign on multiple RP websites. Web SSO phishing has three distinctive characteristics: 1.Highly concentrated value of IdP account.2.Highly enlarged attack surface area.3.difficulty in detection of attack either by algorithms or by users. A compromised IdP account enable attackers to impersonate the victim on a wide range of RP websites. Second-level context is used Rather than sending emails or phishing URLs. Attacker can host their own legitimate RP website or web page and lure users posting URLs Everywhere. An HTML element contains real popup browser window. Spoofing the EV-SSL symbol and HTTPs URL address in the component should be possible by duplicating a total preview of the symbol and the URL address. [45] [48]

5.6. Browser Cache Poisoning: Clicking through of SSL warnings: While Accessing a website having invalid certificate browser shows SSL warning. At that point the client is accepted to close that website page to ensure against MITM attack. If client disregards notices can be prompt disastrous to the security and protection of the sessions. Attacks against HTTPS: [26] 1. Man-In-The-Script-In-The-Browser attack to avoid enhanced channel -ID based defenses. Attacks via browser cache: 1. Timing attack performed on the browser to sniff browsing history and steal user credentials as well as private information. 2. Attacks by poisoning browser web cache, HTML5 AppCache, HTTP cache .A tool called airpoison is used in the wireless network to move up on to browser cache poisoning via HTTP. 3. Cross-site scripting attack is used to inject malicious content into web page and web storage. 4. Proxy cache poisoning attack uses existing techniques to place poisoning attacks on the forward proxy and reverse proxy. [5] [22]

5.7. Session Hijacking: Session hacking attack is performed at two level, application level and network level here. When establishing a connection with the server using HTTP protocol a unique session ID or current live session is used by client and server. The attacker takes control over a session. Actually attacker hijacks the session from the user and continues the connection to the server pretending to be the user. The Session Hijacking attack is performed to gain unauthorized access to the Web Server. The Attacker compromises a session ID by sniffing or predicting a valid and predictable session token. The attacker utilizes a sniffer to catch a substantial token session. Sometimes the server utilizes a protected encoded association like HTTPS but specific session of the client yet remaining association is sent in plain content. [13] [34] [36]

5.8. Drive-by-Download: In this attack, a victim is lure to a malicious web page of malicious site and that page contains code written in JavaScript programming language. Then attacker waits for their target to browse to the web page. The compromised page will look normal while at the same time the exploits execute and install malware on the victim's computer silently in the background. In drive-by download attack attacker loads the shell code as payload using client-side scripting code into memory and executes the exploit against a vulnerable component. JavaScript is utilized to designate the binary representation of shell code to a variable

that is stored in the address space of the browser. It utilizes heap spraying to make heap area. Once heap memory has been executed then the real exploit launched. [12] [14]

***5.9. Clickjacking*:** Clickjacking attack is called as UI redressing attack. Because this technique is tricking users to click the button or image that will run hidden malicious script from attacker site. The attacker uses to trick a user into clicking on a button or link another page when user was expecting to click on the safe page. So an attacker hijacks the click to attacker website. Since this strategy is deceiving clients to tap the catch or picture that will run hidden malicious script from attacker site. The attacker uses to trap a client into tapping on a catch or connection another page when client was hoping to tap on the safe page. So an attacker hijacks the snap to attacker site. To position an element from the target website clickjacking attack uses two nested IFrames. The Inner IFrame contains the target page and it must be large enough to display entirely. The user will click simply without scrolling the web page where the outer frame is smaller and acts as a window onto the web page. User will think he is clicking on the website he wants to open but actually he is clicking on an invisible website and he cannot see that is underneath his mouse. [9][21]

6. PREVENTION TECHNIQUES

***6.1. Buffer Overflow Attack*:** Stack Buffer Overflow protection techniques involve modification in the arrangement of stack-allocated data. It contains Canary values when this value exploded by stack buffer overflow. It presents that program using more than its allocated size of the buffer. By confirming canary value program can be closed to intercepting it from misbehaving. Also, stops an attacker from allowing taking control over it. [53] Bound Checking is another prevention technique which checks permission to each allocated block of memory. They cannot go apart from the actually allocated space, and tagging assures that memory allocated for storing data cannot contain executable code. The user should use such programming languages that do not give direct memory access like Java, Python, Perl, Lisp over C/C++.If the user is using language that gives direct memory access then use classes that handle memory access like std:: string. Security-related compiler options like DEP, ASLR must be used. It will be helpful for mitigating the impact of overflow. To discover overflow Static code analysis tools like Veracode's service, Fortify, Qualys can be used. [7] [18]

6.2. Cross-Site Scripting: Input Validation is effective XSS attack prevention technique. Input Validation technique should not allow the user to enter incorrect data it should return an error message .Input validation also includes valid use of angular brackets, other characters, quotes. Escaping strategies mention to injecting data in sensitive areas of HTML which offer an attacker the opportunity to affect markup parsing. The Content-Security-Policy (CSP) is an HTTP header. The browser can trust white list of trusted resource provided by CSP. The browser should ignore any source which is not mentioned in whitelist since it is untrusted. Generally, the htmlspecialchars() function is sufficient for filtering output. The user can use htmlentities() if he is using character encoding other than UTF-8. [8] [16]

6.3. Man-in-the-Middle Attack: To prevent DNS spoofing ensure that latest version of DNS software with recent security patches is installed. Also Ensure that auditing is enabled on all DNS server. Most popular email services and online banking applications rely on HTTPS to ensure that communications between our web browser and their servers is in encrypted form. By using DH for key generation and Blowfish for encryption will enhance data security over SSL and HTTPS. ARP poisoning can be avoided by running shell script at the backend. This will keep track of entries in the ARP cache table. Different security measures can be used such as operating systems onto the network should be upgraded, network designing from security point of view, network devices and the computers onto the network should be updated periodically and the patches should be installed regularly. [11]

6.4. Extension Vulnerability: A new browser extension system can be used to protect browser from this attack. Extensions run with least privileges can be exploited by a malicious website to avoid divide extension into three components: content scripts, extension core, and native library. An attacker would need to convince the extension to forward malicious input from the content script to the extension core and from the extension core to the native binary to gain users full privileges. Different components of an extension are isolated from each other by strong protection boundaries: each component runs in a separate operating system process. The content script and the extension core run in sandboxed processes, they cannot use operating system services. The content script is isolated from its associated web page by running in a separate JavaScript heap but both uses the same DOM, prevents JavaScript capability leaks. [17] [20] [28]

6.5. Extreme Phishing: Extreme phishing attack is avoided by utilizing heuristics based phishing detection solution and Web SSO phishing identification procedures. For instance, the goal of a tick activity on the base site page could be catch attention. So it will be utilized to identify contrast if a comparing genuine login window or a fake login window is shown. Web clients ought to be prepared to know about extraordinary phishing. The client ought to give careful consideration to the domain name of a URL shown in the address bar. Web users ought to know about the look and feel of web pages. User ought to separate the parodied Web SSO login windows from genuine ones. One method for identifying distinction between a spoofed Web SSO login window is to expand, drag, or resize. Because a spoofed window can never connect with the website page content area. Browser extensions could be useful for clients to get instinctive data about the domain name continuously.

6.6. *Browser Cache Poisoning:* HTTP Strict Transport Security (HSTS) provides a HTTP response header for a website to force the browser to make SSL connections compulsory for all sub resources on the site. HSTS compliant browsers give users the option to ignore SSL certificate warnings. Public Key Pining (HPKP): allows website to specify their public keys with an HTTP header and instructs browser that does not accept a certificate with the unknown public key. Defenses implemented by browser vendor: Do not cache resources in Web cache or AppCache over broken HTTP connection. Preventing HTTPS sites from loading resources over HTTP. To avoid browser cache poisoning attack the target site checks the integrity of all cached JavaScript sub-resources before loading them into the page, only fresh uncontaminated resources can be loaded into the target sites page. [11] [19] [29]

6.7. Session Hijacking: To prevent the user from session hijacking use Strong Session ID to avoid hijacked or deciphered. SSL and SSH provide strong encryption using SSL certificate. There must be a log out function for every session termination, login for regeneration of Session ID.

HTTPS connection should be used for passing authentication cookies and also reduce the life span of session or cookie. Session hijacking can be prevented at the user level by clear the history, offline contents, and cookies from the browser after every sensitive transaction. To protect from session hijacking there are different tools and techniques are available. By using a sniffer on network attacker can be detected. ANTI-SNIFF-It can detect any sniffer on the network used to capture packets. [27] [35]

6.8. Drive-By-Download: Anomaly discovery depends on the theory that malicious action shows itself through anomalous framework events. Anomaly discovery frameworks screen occasions happening in the framework under investigation. For every occasion, various elements are extracted. During a learning stage, typical component feature values are found out, utilizing at least one show. After this underlying stage, the framework is changed to location mode. In this mode, the component benefits of happening occasions are evaluated concerning the prepared models. Occasions that are too distant from the built up models of typicality are flagged as malicious. [54] [60]

6.9. Clickjacking : To avoid Clickjacking attack provide confirmation window for the click. If it is a different component the user can decline his interaction and report it. Frame busting is another defense against clickjacking attack, which will hinder elements in an IFrame from being displayed on web page. It can be achieved through JavaScript. At page load time it will check if the active page is the top-level in the browser window or not. A new HTTP header called X-FRAME-OPTIONS is added to every authenticated. Server should run in an HTML5 sandbox implementation and it prevents any JavaScript from running on a server. [9]

7. CONCLUSION

Web browser like Mozilla uses JEMalloc memory allocator which is vulnerable to heap overflow .Without security patches, web browsers are vulnerable to different types of attack. A web browser is not totally secure because plug-ins are also vulnerable. Browser based attacks originate from malicious websites. The Attacker can easily deliver malicious code to user's system. The user should block pop-up windows to avoid malicious code to be downloaded on user system. The browser is inherently insecure without multiprocess and exposes the user to different exploits. Multiprocess and OS level sandboxing must become standard and mandatory features and eventually each tab must be contained within a separate process. Multiprocess implementation will have an insignificant effect on RAM usage. The effect on CPU is none, because a multiprocess browser will be able to run on multiple cores. In multiprocess based tabs, layout rendering and JavaScript code should be put into a sandboxed process to reduce kernel attack surface. Web browsers with electrolysis and sandboxing feature restrict access to file system. This protects the user from exploits. Hence, multiprocess and sandbox should become mandatory to protect users from malicious web pages.

8. REFERENCES

[1] Adi, Saltzman, Roi and Sharabani,Active Man in the Middle Attacks: A Security Advisory, A whitepaper from IBM Rational Application Security Group, 2009

[2] Bhargavaand Chen, Daniel,Shastry,DeFreez,Jean-Pierre Haoand Seifert, A first look at Firefox OS security,Nashville, TN USA, 2011

[3] Xiaowei and Xue,Yuan,Li,A survey on web application security ,Nashville, TN USA, 2011

[4] Nicolas, Golubovic, Attacking Browser Extensions.

[5] Yue and Dong, Xinshu and Saxena,Jia,Prateek and Mao, Jian and Liang,Yaoqi and Chen,Zhenkai, Man-in-the-browser-cache: Persisting HTTPS attacks via browser cache poisoning, computers security, 55, (2015)62–80

[6] V and PandianS,Nithya, Lakshmana and Malarvizhi, C,A Survey on Detection and Prevention of Cross-Site Scripting Attack,International Journal of Security and Its Applications,3,9,(2015),139–152

[7] Calton and Beattie,,Cowan, F and Pu, Steve and Walpole,Crispin and Wagle, Jonathan, Buffer Overflow : Attacks and defenses for the vulnerability of the decade,2,(2000)119–129

[8] Gurvinder,Kaur ,Study of Cross-Site Scripting Attacks and Their Countermeasures,International Journal of Computer Applications Technology and Research,10,3,(2014)604–609

[9] A Sankara,Narayanan, Clickjacking vulnerability and countermeasures, New York International Journal of Applied Information Systems, 2012

[10] David, Stefan, Deian and Yang, Petr and Russo, Edward Z and Marchenko, David and Karp, Alejandro and Herman,Brad and Mazieres, Protecting Users by Confining JavaScript with COWL, (2014)131–146

[11] Tarek S and Zaki,Ashraf and Sobh,Elgohary, Mohammed, Design of an enhancement for SSL/TLS protocols, 25, (2006)297–306

[12] Giovanni,Cova, Christopher and Vigna,Marco and Kruegel, Detection and analysis of drive-by-download attacks and malicious JavaScript code, (2010)281–290

[13] Jerry, Louis, Detection of session hijacking, 2011

[14] Manuel and Wurzinger, Egele, Peter and Kruegel, Engin, Christopher and Kirda, Defending browsers against drive-by downloads: Mitigating heap-spraying code injection attacks, (2009)88–106

[15] P Vadivel and Alagarsamy,Murugan,K,BufferOverflow Attack– Vulnerability in Stack,International Journal of Computer Applications,5,13,(2011)1–2

[16] Rohilla, Rakesh,Monika and Kumar,XSS Attack: Detection and Prevention Techniques

[17] Adam and Felt, Barth,Adrienne Porter and SaxenaPrateek and Boodman, Aaron, Protecting Browsers from Extension Vulnerabilities, 2010

[18] Benjamin A and Brodley, Hilmi and Vijaykumar, Kuperman, TN and Jalote, Carla E and Ozdoganoglu, Ankit,Detection and prevention of stack buffer overflow attacks, Communications of the ACM11,48,(2005)50–56

[19] Hodges, Collin and Barth, Jeff and Jackson,Adam, Http strict transport security (hsts), 2012

[20] Gu, Boxuan and Zhang, Xiaole and Champion, Wenbin and Bai, Adam C and Qin, Dong,Feng and Xuan, Jsguard: shellcode detection in JavaScript, (2012)112–130

[21] Marchesini, Sean W and Zhao, John and Smith, Meiyuan, Keyjacking: the surprising insecurity of client-side SSL, Computers Security, 24, (2005)109–123

[22] Jia, Yue and Dong, Yaoqi and Chen,Xinshu and Saxena, Prateek and Mao, Jian and Liang, Zhenkai, Poster: Man-

in-the-Browser-Cache: Persisting HTTPS Attacks via Browser Cache Poisoning

[23] Callegati, Walter and Ramilli, Franco and Cerroni, Marco, Man-inthe-Middle Attack to the HTTPS Protocol, IEEE Security Privacy, 7, (2009)78–81

[24] Eriksson, Mattias and Johansson, TT, An example of a man-in-themiddle attack against server authenticated ssl-sessions, 2003

[25] Fraser,Howard, Modern web attacks, Network Security, 2008, (2008)13– 15

[26] Matthias and Ben-David,Vallentin, Yahel, Persistent browser cache poisoning,2010

[27] Karapanos, Srdjan,Nikolaos and Capkun, On the Effective Prevention of TLS Man-In-The-Middle Attacks in Web Applications, 14, 2014

[28] Barth, Adrienne Porter,Adam and Felt,SaxenaPrateek and Boodman, Aaron, Protecting Browsers from Extension Vulnerabilities, 2010

[29] Jackson, Adam, Collin and Barth, Forcehttps: protecting high-security web sites from network attacks, (2008)525–534

[30] Vallentin, Yahel, Matthias and Ben-David, Quantifying persistent browser cache poisoning, 2014

[31] Jackson,Andrew and Boneh,Collin and Bortz,JohnC,D an and Mitchell, Protecting browser state from web privacy attacks, (2006)737–744

[32] Liang, Wei and Liu, Bin and You,Liangkun and Shi, Mario, Wenchang and Heiderich, Scriptless timing attacks on web browser privacy, (2014)112–123

[33] Jemel, Ahmed,Mayssa and Serhrouchni, Security assurance of local data stored by HTML5 web application, (2014)47–52

[34] Vishnoi, Monika and Tech,Laxman and Agarwal, MIT, Session Hijacking And Its Countermeasures, International Journal of Scientific Research Engineering and Technology (IJSRET), (2013)250–252

[35] Deepak Singh,Jain, Divya Rishi and Tomar, Vineeta and Sahu, Session Hijacking: Threat Analysis and Countermeasures

[36] Kapoor, Shray, Session hijacking exploiting TCP, UDP and HTTP sessions, infosecwriters. com/text resources/.../SKapoorSessionHijacking. pdf, 2006

[37] Ralf and Basin,Rolf and Hauser, David,Oppliger, SSL/TLS sessionaware user authentication revisited, Computers Security, 27,(2008)64–70

[38] Piekarska, Bhargava and Borgaonkar,Marta and Shastry, Ravishankar, Piekarska, Bhargava and Borgaonkar,Marta and Shastry, Ravishankar,What Does the Fox Say? On the Security Architecture of Firefox OS,(2014)172– 177

[39] Securing web browser, http://www.us-cert.gov/publications/ securing-your-web-browser

[40] Attacks on browser, http://www.owasp.org/index.php

[41] See fixed patches in mozilla on bugzilla, http://www.bugzilla.mozilla.org/quickserack=attachment

[42] Mozilla foundation security advisory, https://www.mozilla.org/en-US/ security/advisoris/mfsa2017-01

[43] How Appliction Cache works, https://developer.mozilla.org/en-US/ docs/web/HTML/Using the application cache

[44] All errors in Mozilla browser can see one time at,https://www.mozilla.org/en-US/security/known-vulnerabilities/firefox

[45] Zhao, Rui and John, Stacy and Bussell,Samantha and Karas, Cara and Roberts, Daniel and Gavett,Jennifer and Six, Brandon and Yue, Chuan,The Highly Insidious Extreme Phishing Attacks,(2016)1–10

[46] Privilege escalation vulnerabilities in WebExtensions APIs, https://bugzilla.mozilla.org/showbug.cgi?id=1226423

[47] Pandikumar, T and Girma, Teklish,Analyzing Information Flow in Java based Browser Extensions,(2016)

[48] Chuan,Yue,The Devil Is Phishing: Rethinking Web Single Sign-On Systems Security.,(2013)

[49] Zhao,Chuan and Yi,Rui and Yue,Qing,Automatic detection of information leakage vulnerabilities in browser extensions,(2015)1384–1394

[50] Interger overflow in Websockets during data buffering, https://bugzilla.mozilla.org/showbug.cgi?id=1287266

[51] Buffer overflow rendering SVG with bidirectional content, https://bugzilla.mozilla.org/showbug.cgi?id=1270381

[52] Cross-site reading attack through data and view-source URIs, https://bugzilla.mozilla.org/showbug.cgi?id=1228950

[53] Integer overflow in MP4 playback in 64-bit versions, https://bugzilla.mozilla.org/showbug.cgi?id=1206211

[54] Same origin violation and local file stealing via PDF reader, https://bugzilla.mozilla.org/showbug.cgi?id=1178058

[55] Electrolysis and Accessbility, https://wiki.mozilla.org/Electrolysis/Accessibility

[56] Introduction to Electrolysis, https://wiki.mozilla.org/Electrolysis

[57] Electrolys and multiple content process, https://wiki.mozilla.org/Electrolysis/Multiplecontentprocesses

[58] Sandbox security process model https://wiki.mozilla.org/Security/Sandbox/Processmodel

[59] Hardening the Firefox Security Sandbox https://wiki.mozilla.org/Security/Sandbox/Hardening

[60] Tammo and Dewald,Rieck,Andreas,Konrad and Krueger,Cujo: efficient detection and prevention of drive-by-download attacks,(2010)31–39.

[61] Chariton, Argyroudis, Patroklos and Karamitas, Exploiting the jemalloc Memory Allocator: Owning Firefox's Heap,Blackhat USA,2012

[62] Emery D,Novark, Gene and Berger, DieHarder: securing the heap,(2010) 573—584