FIREFOX CASE STUDY

IT18361728 - H.W.D.T. De Silva

Video link

Contents

Product Overview

A web browser, more often known as a browser, is application software that enables users to access the internet. When a user visits a specific website and types in the URL of a web page, the web browser gets the required resources from the web server of the specified website and then displays the response web page on the user's device. A web browser is not equivalent with a search engine, despite the fact that the two are often used interchangeably among users. A search engine is a website that collates other websites and offers connections to them. However, a user must have a web browser installed on their host device in order to connect to a website's server and see its associated web pages. Browsers are used on a variety of devices, including desktop computers, laptop computers, tablets, and smartphones. Among them is Firefox, one of the most widely used web browsers on the planet. Firefox was initially called Phoenix, but it was then renamed to prevent trademark issues with Phoenix Technologies. On 9th of February in 2004, the current name, Firefox, was selected. Dave Hyatt and Blake Ross founded Firefox as an experimental version of the Mozilla browser on 9th of November in 2004, with the aim of " to ensure the Internet is a global public resource, open and accessible to all." Beginning with version 5.0, a fast release cycle was implemented, resulting in the publication of a new major version every six weeks. This process was intensified in late 2019, to the point that new major releases would occur on four-week cycles beginning in 2020. Firefox 93 is the most recent release of the product, and it was made available for the community use since 5th of October in 2021. According to the latest statistics Firefox is used by 3.67 percent of online users for web surfing and other purposes as of September 2021.

This is considered as one of the biggest open-source teams in the planet, ranking in the top 2% of all Open Hub project teams. Open Hub analyses just recent code modifications while performing this assessment and they admits that 8428 developers had contributed to the project Firefox since the initial release and 1085 developers contributed new code to Mozilla Firefox during the last twelve months. This comprises about 23.9 million lines of source code written in approximately 47 different languages, and it was found out that most of them are written in C++. The majority of volunteer efforts are in the form of testing and bug reporting. Mozilla Firefox's activity has been constant during the past twelve months, and it could be considered as a positive indicator, indicating that development is proceeding at a steady rate and not dropping off.

Firefox is the only major browser that is fully open source, with all opensource participants contributing to its maintenance. Once code is submitted to the Mozilla Repository, all code must be adhered to the Mozilla Public License. The Mozilla Public License Version 1.0 was published in 1998, followed by a significantly modified Version 1.1 the following year. Mozilla Public License 2.0 was introduced more than a decade later, in 2010, and has been considered as the Mozilla Public License choice since that. The Mozilla Public License 2.0 is neither a permissive license that permits unrestricted use of the licensed code nor a strong copyleft license that imposes many limitations on usage. Mozilla Public License 2.0, which is often referred to as a "weak copyleft" license, lies somewhere in the center of aforementioned license types. There are weak copyleft agreements, such as the Mozilla Public License 2.0, require users to publish their modifications to the source code, but approves to share a subset of the code if required. If an author modifies any of the original files, they must provide those modifications together with the code and license them under the Mozilla Public License.

However, if the author separates the Mozilla Public Licensed code from the closed-source code, they could develop an aggregate work from both. It's up to the developer whether to make the new code files public and this approach is often referred with the term "file-based copyleft". The Mozilla Public License 2.0 requires developers to mention that their code is released under the terms of the Mozilla Public License and to disclose the location of the license. However, they state that including the entire licensing agreement is not required. Moreover developers who contributes the project adhering to Mozilla Public License2.0-licensed code may not use the original project's trademarks, names, logos, or other identifying information without the permission of the original project or its contributors. Open-source software writers may use Mozilla Public License 2.0 as a weak copyleft license while still being able to use and benefit from their work in well-known and successful proprietary software projects. There is an explicit patent license in the Mozilla Public License 2.0, however there are also additional phrases to explain the license's scope. An implicit license disclaimer is included, as is a strong defensive termination clause that offers an incentive against patent claims by terminating the copyright license, in addition to terminating the license itself. The Mozilla Public License 2.0 provisions many advantages on consumers of licensed code. For instance, it shields users from some patent litigation brought by contributors by the explicit granting process of patent rights to developers. Additionally, the possibility to mix Mozilla Public License 2.0-licensed code with proprietary works is an advantage. The license encourages businesses to participate in and contribute to the open-source community while maintaining the competitive benefits associated with closed-source software.

On the other hand, since the Firefox's source code is accessible by everyone, professionals like white hat hackers, contributors, and users are more likely to find some existing drawbacks quicker than proprietary products. when a vulnerability is discovered in this project, particularly ones with high severity, a patch is often published within a day or two. To ensure a consistent level of quality, numerous stages of code review are necessary prior to code being included in the core Firefox distribution. Module owners assess the code that contributed to their modules and may request further code evaluations from other peers if the modification has an impact on other modules of the core distribution. "Super-reviews" are used to guarantee that all fixes are compatible with the overall Firebox code base and these reviews are conducted by developers responsible for the Firefox architecture. Thereby Firefox has the ability to deliver a best and reliable product to their users.

Product Assets

There are many assets in Firefox that may be exploited by adversaries for their beneficial which compromises end users of the products in different ways because Firefox is treated as the default browser in many operating systems such as Linux and Ubuntu, thus compromising the assets of the browser poses a significant danger to individual users as well as organizations. Out of all assets Firefox’s most vital assets are browsing and download history, form and search history, user cookies, user cache, active logins and offline website associated data.

Most of the internet users have heard of the term "phishing," which refers to the process where hackers create fraudulent websites that pretending like well-known institutions like banks in order to steal login credentials from legitimate parties. The more information a phisher gathers about their target, the more likely it is that their scam will be successful. Therefore browsing history is a common asset of the Firefox browser adversaries keeps an eye on. Also browser history may be used to violate the privacy of the product user if it is exploited. The adversaries often use a method called history sniffing attack to obtain victims history related information if the user does not erase their browser history. Most people don't want other people to know about the sites they frequently visit, more specifically sites which are related to health issues. Another advantage of accessing a user's browser history is that a phishing or social engineering attack may be launched with that information. If they know the websites the targeted victim frequently visits, they may be able to create a sophisticated phishing attack such as whaling of that site to lure victim into providing their personal information.

When it comes to Firefox’s cache, it is simple yet considered as one of the critical assets that could be leveraged to cause serious impacts if it is compromised by an adversary. It could be performed with the use of simple man in the middle attack and once attacker abuse or alter the cache related to the Firefox browser every time client request content of particular site browser could redirect users with the content of malicious sites that are stored in browsers cache instead of legitimate site and attackers then could carry out further attacks like malware infection, data theft once compromised victim gets lands into their malicious site as a result of cache poisoning in the browser. This process is often known as DNS cache poisoning of a web browser, and this only impacts to the targeted victim since the browser’s assets directly get tampered by the adversaries. Apart from that Firefox uses several cache types such as HSTS, OCSP, HTTP, image, favicon, style sheet, font, DNS, HTTP Authentication, Alt-Svc, TLS certificate for optimize the user experience and prevents sharing them across different sites like cookies due to the security concerns.

Other important asset type is cookies. These tiny text files, sometimes called HTTP cookies or web cookies, hold a small amount of data. Cookies are little text files that a website or its affiliates place on user’s computer or device when users browse different sites. They can't be accessed through a drop-down menu in browser to see what they've done. However, they are critical when it comes to web surfing and may make users experience much more pleasant. Cookies can't contain code, therefore they're completely safe. Viruses and other harmful programs cannot be contained or executed on them. However, cookies could be utilized as the source of malicious activity affecting user's data. Simply being text files makes them highly susceptible to cookie related attacks, which is why they're so often considered as a common attack target misused by adversaries. Even if the user doesn't use a third-party application or spy software, third-party cookies may collect personal information like location and passwords from user's device and exploit it for malicious purposes. Cookie hijacking is the most common phrase for this kind of intrusion and this type of attacks paves the way to get illegal access to websites where legitimate users have signed in without users’ intervention or permission. Apart from that by default, Firefox blocks the trackers and scripts which falls under social media trackers, Cross-site tracking cookies, finger printers, crypto miners. Ads, videos, and other in-page content may all include these trackers. In Firefox, tracking information is disabled in Private Windows. Firefox's new Enhanced Tracking Protection feature keeps a distinct "cookie jar" for each page that users visit. It works this way. Cookies placed in user's browser by a website or third-party content embedded in a website are only permitted to be read by the website that placed them and cannot be shared with any other website. Aside from that, Total Cookie Protection allows non-tracking cross-site cookies, such as those used by popular third-party login services. If Total Cookie Protection detects user's intention to use that provider.   The provider will only be permitted to use a cross-site cookie for the site user is currently visiting. Moreover, because of these short-term exceptions, user's browsing experience remains unaffected while still receiving robust privacy protection. Firefox's Total Cookie Protection manages cookies and other sites' data across all of the open tabs and windows. Together, these characteristics make it impossible for websites to "tag" user's browser, making cross-site tracking almost impossible.

Additionally, the extensions are another major asset when it comes to Firefox which have access to a variety of sensitive user data, including the user's browser history, bookmarks, passwords, cookies, and list of installed extensions, and they are capable of downloading and storing files on the user's device. Although browser extensions and web apps run in different contexts, they may communicate through messages regardless of the browser used. This enables web programs to take advantage of extension-level privileges and steal sensitive user information. It was found that many of the communication interfaces provided by web apps may be abused to get access to privileged capabilities. The findings indicate that interactions between browser extensions and web apps offer significant security and privacy risks to computers, web applications, and, most significantly, to users. These extensions enable web applications to run arbitrary code with the same rights as extensions, circumvent security policies and access user information, save, and retrieve data in the extensions' permanent storage, and trigger the download and storage of arbitrary files on the user device. Furthermore, Firefox browser developing community should conduct more thorough reviews of extensions, paying special attention to the usage of message passing interfaces in extensions to enhance the security of the product.

Example Attacks

As opposed to other types of software, browsers do not operate independently and cannot perform all of their functions. for example, plug-ins and extensions, are often required by browsers in order for them to operate fully. To put it another way, browsers serve as a platform for many components to work together toward a shared objective. This is when things become interesting when it comes to the cyber security domain. Bad actors have the ability to get this information if they have the time or money to do so. With sensitive browser data in hand, cybercriminals have an advantage over their victims, which they often use for financial benefit. When it comes to collecting confidential information, hackers have a lot of choices. Credentials may be stolen through cyberattacks like phishing, MITMs, and drive-by downloads, and browsers are the common thread connecting them all. Because browsers are now so widely used for both business and hackers have an easier time targeting them. Cloud-based SaaS programs have replaced traditional on-premises software in many companies, which caused to access services through web browsers. Thus the likelihood of users being infected with malwares, crypto miners, and other harmful browser-based threats is high due to the prevalence of unverified extensions and out of date plug-ins. Apart from that, browsers will remain a prime target for cyber espionage as a result of the widespread usage of browsers and the sensitive information they carry across the internet.

Phishing

This is the most often used method of launching browser-specific cyberattacks. Phishing occurs when hackers send a harmful email from a legitimate source, fooling the recipient into believing it is safe to access. This email will either include a malicious link or an attachment; clicking on any of these items often initiates a script that gives the hacker full access to the user's machine. Once a hacker gains access, they could compromise the entire host machine in which the browser is installed. When it comes to Firefox, it has a mechanism which is capable of provisioning protection against such phishing attacks and once user visits site it checks against pre identified list of sites which are considered as phishing and associated with malware and if identified as malicious it will automatically warn that to the product user by Firefox browser itself.

MITM

A man-in-the-middle attack is a kind of eavesdropping process in which attackers interject themselves into an ongoing conversation or data transmission. After infiltrating the "middle" of the transfer, the attackers pose as both genuine parties. This allows an attacker to intercept communications and data from either side while also delivering malicious links or other information to both legitimate parties in a manner that may go unnoticed until it is too late. Just like with the phishing attack, Firefox browser provides protection against such attacks by prompting “MOZILLA_PKIX_ERROR_MITM_DETECTED” message. Thus prevents users of the Firefox product being vulnerable to such attacks.

Drive by download

This is a type of attack is carried out triggering malicious scripts which are intended to abuse the features and standards of the web browsers of the users. Once a user landed on a malicious site it automatically downloads malicious files into victim’s host machine without user’s intervention or users are redirected to download pages or else downloading pop ups will appear on the browser. When it comes to Firefox, it utilized a technology called sandboxed iframes to prevent its users from being a victim of drive by download attacks.

Cross site scripting

This is a malicious code, usually in the form of a browser side script, is sent to another end user via a web application during an XSS attack which usually associated with link of a vulnerable web site that the end user used to trust without any hesitation. There web browsers utilize input from users in their output without verifying or encrypting it, which opens them up to these kinds of attacks. XSS may be used by an attacker to deliver a malicious script to a user who is not aware of it. The browser of the end user will run the script regardless of whether it should be trusted or not. The malicious script may access cookies, session tokens, and other sensitive information since it believes the script is from a reputable source which ultimately results to send sensitive information of the particular victim back to the attacker. Unfortunately Firefox is not capable of providing protection against this type of attacks since the feature is still in developing stage.

Cross site request forgery

Another extensively used attack technique is cross-site request forgery. Attackers use this method to execute code or perform website requests on the victim's browser without their permission or knowledge. Typically, this attack takes place when a user is signed into their account on a website that offers account-specific capabilities to that user. Cross-site request forgery allows the attacker to pose as the logged-in user and do illegal operations using the victim's account if they can persuade the victim to click on specially crafted links. Because an attacker has inserted malicious code using an exploit in the website the victim trusts, the victim is often unaware that the links they are clicking are doing harmful operations. For example, if the attacker merely changes the victim's website settings, it may only be bothersome. However, if the website of a bank is infiltrated and the attack is successful, the attacker may be able to conduct money transfers from the victim's account into their own.

Adware

Adware is a kind of malware that appears on browsers with the aim of promoting products or services for other businesses or individuals. Adware usually substitutes the advertisements that a user would normally see on third-party websites. Thereby adversaries tend to gain some profit over that since the traffic is generated to their sites by the compromised users.

Crypto jacking

As a means of operation, cryptocurrency tokens make use of a decentralized database known as a "blockchain". The blockchain is updated on a regular basis with details of all recent transactions. A complicated mathematical method is used to aggregate all of the most recent transactions into a single 'block'. Cryptocurrencies depend on the processing power of the general public to create new blocks of the blockchain every so often. People who provide the bitcoin computer power are rewarded with cryptocurrency. When a device's processing power is used to mine bitcoin, it is said to be crypto jacking. To put it another way, hackers may earn income while their victims remain unaware, they've been infected. Firefox, it has the ability to protect it users against crypto miners and by default it is enabled when users installed the Firefox browser on their systems.

Vulnerability History

CVE-2019-17026

Firefox uses a java script JIT compiler called Ion Monkey and it is the primary component of Firefox's core that performs JavaScript operations. This exploit leverages the use of a type confusion flaw in Firefox's Ion Monkey engine that existed up to Firefox version 72.0.1 and Firefox ESR version 68.4.1 Despite the fact that the problem was addressed in Firefox version 72.0.1 the vulnerability is still theoretically resides in Firefox version 70 and 71. This was initially found by a Chinese company associated with cyber security called Qihoo 360 Technology Co. Ltd. Thanks to the deficiencies raised in heap allocation tactics this particular variation of CVE-2019-17026 only works on Firefox versions up to Firefox version 69. Other than that when it comes to the host operating systems only windows showed that it was susceptible to attack and that includes all the windows versions including windows 10.

The vulnerability was classified as a "type confusion," which falls under memory corruption in which a memory input is originally allocated as one data type and changes that into another type during the execution, resulting in unexpected data processing implications, including the potential to execute arbitrary code on a susceptible machine which uses the vulnerable Firefox versions. When it comes to arrays, they are typically located in different manner according to their items and it makes tampering fairly simple. For instance If an array only includes double values, each array element will be a 4 byte double. If the array only includes integers, each array element will be a 4 byte integer. But if it comprises object, that point it is considered as a mixed type of array which means that each value is tagged to allow integer, double, and pointers instead of one particular type. Thus leverage the type confusion where attackers could take this advantage to treat a double as a pointer or else read a double as a pointer since Ion Monkey doesn't differentiate the arrays based on the data types. This flaw allows for the omission of an array bounds check. The first array is utilized in conjunction with the JIT flaw to perform an out of bound write and tamper the information related to second array. Specifically, it corrupts its length to enable out of bound read write, which is then utilized throughout the rest of the exploit to corrupt the Native Object structure of the third array in order to construct primitive address with arbitrary read write permissions.

There attackers abuse this vulnerability in Ion Monkey compiler by exploiting Firefox. Since just in time compiler is originated from user level it can be easily manipulated by adversaries. The typical method of Ion Monkey is to follow the pointers of web assembly object in order to reach the just in time compiled code and it initiates that segment with read, write and execute permissions where attackers could overwrite arbitrary opcodes. Here attackers used two major shell codes, one is to go through the entire memory address space of the victim machine to determine the vulnerable Firefox process and then the second one is the actual payload which comprises malicious opcode to take over the control.

CVE-2020-15663

When the Firefox browser is initially installed using windows installer on their host machines, some users would choose a different location for Firefox to be installed on their machines which has the permissions to modify the contents by the local user. This may happen if Firefox is installed somewhere other than the usual location of C: Program Files/ Mozilla Firefox directory. A local adversary with a normal account privilege may then overwrite any files in the installation directory if that is the case associated with the compromised machine. An adversary with this permission can already run arbitrary code as the normal user, but in real world scenarios they are looking on gaining access as the SYSTEM user to the compromised machine than the normal user. The vulnerability was initially reported by Mr. Xiaoyin Liu and versions affected were Firefox versions prior to 80.0 and Firefox ESR versions prior to 78.2.

As a standard Windows feature, the Mozilla Maintenance Service runs as a background process. The SYSTEM account has access to this service. In order for the Firefox updater to update write-protected files which are located in C: Program Files/Mozilla Firefox without showing user account control behavior prompts, it must be launched with SYSTEM permission. Local users have the ability to start the Mozilla maintenance service, but their files are located in C: Program Files (x86) Mozilla Maintenance Service and cannot be written to by themselves since they do not have the adequate permission level. When it comes to Firefox updater.exe and updater.ini are copies from the Firefox installation that the maintenance service executes with SYSTEM rights. Despite checking whether updater.exe has an identity string and is signed by Mozilla, the maintenance service tends to ignore the file's version number. As a result, a malicious adversary who already has the access to the victim machine could overwrite the current version of updater.exe with an older version which is more susceptible one and it then transfers old versions of updater.exe and updater.ini to C: Program Files (x86) Mozilla maintenance service update and executes updater.exe in this directory with SYSTEM privileges.

This process is basically performing the version downgrading of Firefox browser instead of upgrading it to the latest version installed to the victim machine and by doing that attacker tends to exploit cryptography related vulnerabilities which were associated with early network protocols that are used in previous versions of Firefox browsers or exploit the vulnerabilities which allows arbitrary code executions to obtain escalated privileges of the compromised systems. Moreover this attack was only get succeeded over machines which runs windows operating systems and other operating systems like ubuntu, Linux, mac didn't get compromised. Also to conduct the attack, adversaries initially had to obtain local or remote access to the compromised system with normal user privileges.

CVE-2020-16044

SCTP stands for Stream Transmission Control Protocol which is typically used for transmitting data that belongs to multiple streams. This is also considered as the next generation TCP, and it is a connection oriented reliable protocol that operates in transport layer. When it comes to Firefox, the developing community came up with an idea to utilize HTTP connections over SCTP to achieve better performance and reliable connection establishment over two end points.

In here while configuring the initiation of SCTP connection between two parties, the initiating party utilizes the COOKIE ECHO chunk to respond to the cookie supplied by the replying party in the State cookie field of the INIT ACK packet and it was typically sent in the same packet where the data chunks reside, but in such cases COOKIE ECHO chunk must precede the data chunks.

When it comes to Firefox, as a result of how Firefox handles browser cookies, there was a use-after-free vulnerability found in cookie echo chunk in SCTP packet in which Firefox used to facilitate a cookie chunk for each corresponding packet transmitted over the internet. If it was misused by an adversary, browser software may provide hackers access to any type of device that the Firefox browser is running on like computer, phone, or tablet. There adversaries utilized specially modified SCTP packet to obtain the chance to exploit the heap of compromised host which ultimately allows for arbitrary code execution. There it was found an attacker may be able to install programs, read, modify, or remove data, or create new accounts with full user rights depending on the privileges associated with the user of the system. Users whose accounts are set up with less administrative user privileges may be less affected than those with full access to the compromised system. Mozilla claims that a malicious COOKIE-ECHO chunk may be created by an attacker to affect the browser's memory. A use-after-free flaw is caused by a program's improper usage of dynamic memory. When a memory region is freed, but the reference to it is not cleared, an attacker may use the mistake to compromise the application, according to the vulnerability's description. When compared to other exploitations this was considered as an easy one and there was no any requirement of form authentication or any other special user interaction and allowed it to be carried out remotely by the adversaries.

The vulnerability was initially reported by Ned Williamson and Firefox android browser versions prior to 84.1.3, Firefox ESR version prior to 78.6.1 and Firefox versions prior to 84.0.2 were affected by the vulnerability according to the official site of the Firefox and named as CVE-2020-16044 on 27th of July in 2020.

CVE-2018-5174

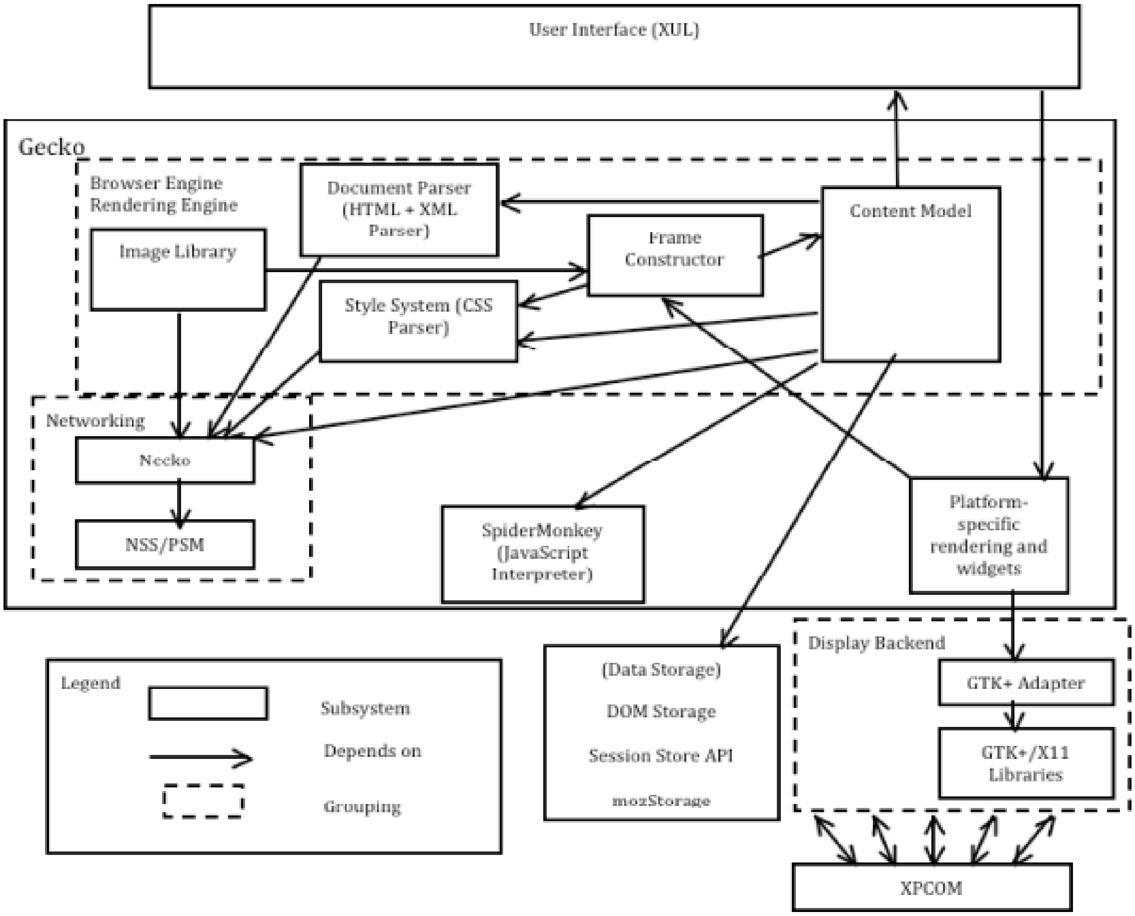
Vulnerabilities in systems or applications may be exploited by malicious actors to evade security measures. Attackers exploit software vulnerabilities by using a programming mistake in a program, service, operating system, or kernel to run code under their control. To prevent from cyber-attacks not only browsers but also operating systems built different protecting mechanisms and smart screen of windows is one such tool intended to protect users from various cyber-attacks. It was introduced in windows 8 and still utilized in windows operating systems with improved versions.

After the windows update which released on 10th of April in 2018 this smart screen did not display any user interface for users if the Firefox browser set “SEE_MASK_FLAG_NO_UI” flag for the files when they are getting downloaded to the host system. There if a file is unknown and may contain malicious code inside, SmartScreen will not prompt the user for permission to run it. If the user is not logged in, Windows will not prompt the user for a destination for the opened file and when Firefox downloads files, it incorrectly sets this flag, leading smart screen to function in less secure manner. Therefore the smart screen user interface will not be shown for specific types of files which belong to known malware categories, and the user will not be able to bypass the block via Firefox and even if a file is run automatically, the user will not be able to see any proof of it. When it comes to new suspicious files that are not recognized as malwares will be allowed to get downloaded via Firefox without any user intervention. When it comes to the attack scenario this could be more severe than normal attacker gaining control over specific target since this paved the way to malicious files like polymorphic malwares, viruses, botnets, spywares, ransomwares and trojans to get downloaded in a concealed manner which could cause critical harm starting from single user to small and medium size organizations which have the vulnerable Firefox version installed on their host systems.

This vulnerability was initially found and reported by Jimmy to the Firefox and uniquely called as CVE-2018-5174 since 1st of march in 2018. Furthermore this vulnerability was only associated with windows operating systems which was running the April 2018 update or later and did not harm any other version of windows or other operating systems. Furthermore Mozilla claims that

Firefox versions prior to 60 and Firefox ESR versions prior to 52.8 were affected by this vulnerability.

Architecture



High level architecture of the Firefox browser

User interface

When it comes to the Firefox architecture the very first layer is its user interface in which the normal users interact with when they are using Firefox browser. This is backed by XML User Interface Language (XUL), UI and XML and they are responsible for providing core components to user interface of the Firefox. Rather of being hardwired into the program itselft, the UI is loaded through an external XUL UI description where XUL is associated with XML (extensible markup language), but it allows for the incorporation of HTML components, including JavaScript, and has a defined set of element types. Proper UI descriptions must take into consideration when it comes to the layouts and components of multiple platforms, such as dialogue boxes. While a single cross platform UI description may work across several platforms, build engineers and UI designers must maintain platform specific copies of some XUL documents to ensure that the browser is displayed in the optimized manner for each platform.

There the cross-platform toolkit (XPToolkit) is the primary UI component and Gecko is the layout engine. Cross platform toolkit defines user interface components, especially things like widgets, utilizing the XML User Interface Language. There is no need for duplicating, since Gecko already has the capability to parse the XML User Interface Language documents into an object model where Gecko actually delegated the XML parsing to the XML Parser component). There cross platform toolkit uses Gecko to parse the XML User Interface Language into the Application Object Model.

After interacting with the application object model, cross platform toolkit creates two layers comprise with frames, widgets and services. There the former part is in charge of the user interface's appearance and design, while the latter part is in charge of executing functions such as printing. These two levels are fully self-independent from each other, and their sole means of communication is the application object model, which is in charge of message transmission.

Browser Engine

Document Parser

The Document Parser component acts as a parser for HTML and XML documents. After receiving URL data from the User Interface via the Content Model, the Document Parser accesses the HTML code via Necko. After parsing the HTML code, it passes it to the Content Model for further processing before passing it to the Frame Constructor for web page rendering. Additionally, the Document Parser includes the Expat Library for XML, which enables it to parse any XML data received from Necko prior to sending the code to the Content Model.

CSS Parser

CSS associated data such as CSS1, CSS2 received from Necko is processed by Gecko's CSS parser and applies the changes to the HTML document and later, the data is transmitted to the Frame Constructor, which uses the user interface to display it correctly. There the additional information about the CSS data is obtained from the Content Model.

Content Model

Before providing the updated code to the Frame Constructor for actual website creation, the Content Model modifies the properties of each retrieved web page obtained from the Document Parser. In order to retrieve the requested site's HTML code and other vital components it begins by interacting with the user interface and gathering URL data. In order to prepare for the display, it manipulates the data with the help of DOM (Document Object Model) structures. For any JavaScript interpretation, it interacts with Spider Monkey, the Image Library, and Data Persistence subsystem components. It also connects with Spider Monkey, the Image Library, and Data Persistence subsystem components. It uses Data Persistence for accessing user settings and bookmarks, for example. The content model alters the DOM tree data and passes it on to the Frame Constructor after getting the necessary data.

Frame Constructor

In order to generate a web page, the Frame Constructor gets all of the essential data from the user interface layer and sends it back to it. Images from the Image Library are also included, as is XML, HTML, and JavaScript parsed from the Content Model. DOM Tree data from the Content Model and any CSS data from the Style System are provided as inputs at the beginning of the process. Using the Platform-Specific Rendering and Widgets component, the Frame Constructor builds the Web Page for the provided platform, according to its best suitability. Before the page is actually shown to the user, the Frame Constructor transmits its built page to the Platform-Specific Rendering subsystem.

Image Library

This is responsible for processing the images associated with the web page before transmitting them to the content model which are obtained from the Necko

Rendering Engine

In order to display web content on the screen, the Rendering engine employs a collection of platform-independent and platform-dependent data structures. For rendering, Gecko employs the Mark II rendering approach, which utilizes placeholders for material that hasn't yet arrived and is controlled by CSS2's contemporary Gecko implementation. When new material is added, the placeholders are changed, resulting in a faster rendering time for the pages that include it.

To put it another way, the rendering engine is in charge of displaying both the user interface and any online content. Graphical primitives like colors, font definitions, and alpha definitions are used by the rendering engine to produce content regardless of the platform it is running on. To render web pages using the graphics primitives, also the rendering engine makes a replica of the current DOM.

JavaScript Interpreter

This is basically the Firefox's Java Script engine, SpiderMonkey, is built in C/C++ and used by the browser. It's basically a fast Java Script interpreter that can handle almost all the spectrum of the language's values. For further rendering, SpiderMonkey uses JavaScript code that was sent from the Content Model and sent back to SpiderMonkey for additional interpretation.

Display Backend

The host operating system and the display backend are inseparably linked. All drawing and windowing functionality offered by Firefox's multiple operating systems is given by the Display Backend, which offers a consistent, cross-platform interface for the User Interface. The User Interface makes use of a variety of widgets to construct the user interface. OS specific graphics adapter and OS graphics libraries are part of the Display Backend. The adapter converts User Interface (UI) calls into ones that the graphics libraries on the host operating system can understand. There is a distinct implementation for each host operating system that makes use of the native graphics libraries given by the operating system. Firefox has a graphics adapter for UNIX-like operating systems that makes use of the GTK+/X11 APIs. Firefox utilizes the native Cocoa framework for its graphics adapter on Mac OS X. Firefox uses the WINAPI and MFC system functions to develop a graphics adapter for Windows.

Data Persistence

The data persistence subsystem's aim is to preserve user, tab, and window data between Firefox sessions. This may be accomplished in a variety of ways by using the various components included in the data persistence subsystem. Additionally, XUL offers a way for storing persistent data that is used to remember the state of windows, toolbars, and more. A property named "persist" is used to specify which properties should be saved. After that, the information is gathered and saved in a file in the same directory as the user's other preferences. XUL enables the state of any element to be saved. Data persistence is located in the second layer, identical to the Gecko subsystem. The data persistence architecture has no outgoing dependencies on other subsystems and just one entering dependence on Gecko. More precisely, it is discovered that the requirement originates with Gecko's Content Model component, which requires data persistence for accessing bookmarks, user preferences, and cached website data.

Cookies, DOM Storage, and Flash Local Storage are all used by Firefox to store user data. It's common for cookies to be used on the internet, although their uses are restricted. Web Applications added DOM Storage as an alternate method of storing additional data while still maintaining security. When there is any event associated with persistent data in browser Gecko and XPToolkit are both involved in the user persistence component.

Session store API

The session store API enables extensions to easily save and restore data across Firefox sessions by providing access to the session store functionality through a simple API. One scenario where enabling this functionality is critical for an extension is that since Firefox 2.0, the user has been able to undo tab closures. To restore the tab successfully, an extension must make use of the session store API's methods for saving any data that will be required for restoration and retrieving the prior data when the tab is restored. nsISessionStore and nsISessionStartup are two critical interfaces for the session store API. The session store API is implemented using two classes: nsISessionStore, which enables extensions and other programmes to store data associated with browser tabs, sessions, and windows, and nsISessionStartup, which manages the session recovery process.

Document Object Model

DOM Storage was added in Firefox to give a bigger, easier-to-use and more secure, alternative to cookie storage. It is a mechanism for securely storing and retrieving string key/value pairs. The DOM Storage is very valuable since there are currently no suitable browser-only techniques for persistently storing substantial quantities of data for any length of time. For example, browser cookies have a limited data storage capacity and offer no structure for persistent data, while other storage options need extra plugins. One of the primary advantages of DOM Storage is that it offers enhanced capabilities. For instance, it enables an individual to "work offline" for lengthy periods of time.

Moz Storage

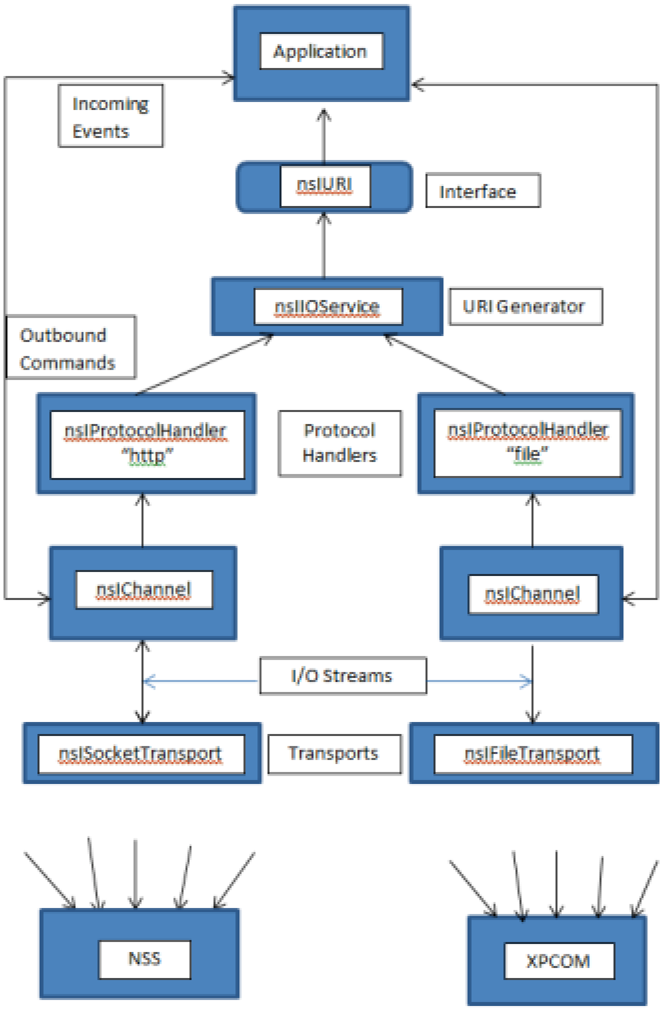
Mozilla Firefox makes use of mozstorage to store information about the browser, such as extensions and other Firefox add-ons. mozstorage is a SQLite-based database API. Web sites can't access it since it's only open to trusted callers. Transactions provide some basic functionality, although concurrency is severely constrained. Because it's not a heavy-duty database system, mozstorage shouldn't be used as one. Gecko (Firefox's core components) and the extensions are the primary interfaces it uses.

XPCOM

The Cross Platform Component Object Model (XPCOM) enables developers to build cross-platform, modular applications. The fundamental objective of XPCOM is to enable developers to break huge software projects down into smaller components and thereby create and build components independently of one another. The components are subsequently reassembled during the application's runtime using reusable binary libraries. Additionally, XPCOM offers the tools and libraries necessary to load and manipulate these modularized components. There, it offers a significant amount of the functionality associated with a development platform, such as component management, file abstraction, object message forwarding, and memory management. Although XPCOM offers its own basic components and classes, the bulk of XPCOM components are given by other components of Firefox, such as Gecko and Necko, which explains its outbound dependencies. XPCOM is utilised by every component of Firefox since it serves as the foundation for handling the system's numerous components and objects. However, its primary use is inside Gecko, since XPCOM offers access to the Gecko library's capabilities as well as the ability to embed or expand Gecko. The modularization of Firefox by XPCOM significantly improves the system's speed, modifiability, maintainability, and development ease.

Necko and NSS

Necko is the name of the project produced by Mozilla and the networking technology used by Mozilla Firefox once it has been implemented as Necko. When the UI subsystem sends out a URL connection request. After then, the request is handled by Gecko and Necko. Requests are sent to a Necko, which decodes them into an instruction set at a lower level. Necko determines the networking protocol, transport number, and Internet connection. Whereas Network Security Services' libraries, APIs, tools, and documentation are used to build secure client and server applications. SSL version 2, version3 TLS as well as the PKCS 5 and 7 protocols are all supported by NSS and whenever Necko loads a website, it checks for Network Security Services so that it can handle any secure data.



High level architecture of the Necko

Necko's major transport interfaces are nsISocketTransport and nsIFileTransport, which enable data to be physically transported. An IO Stream receives the data from these transports.

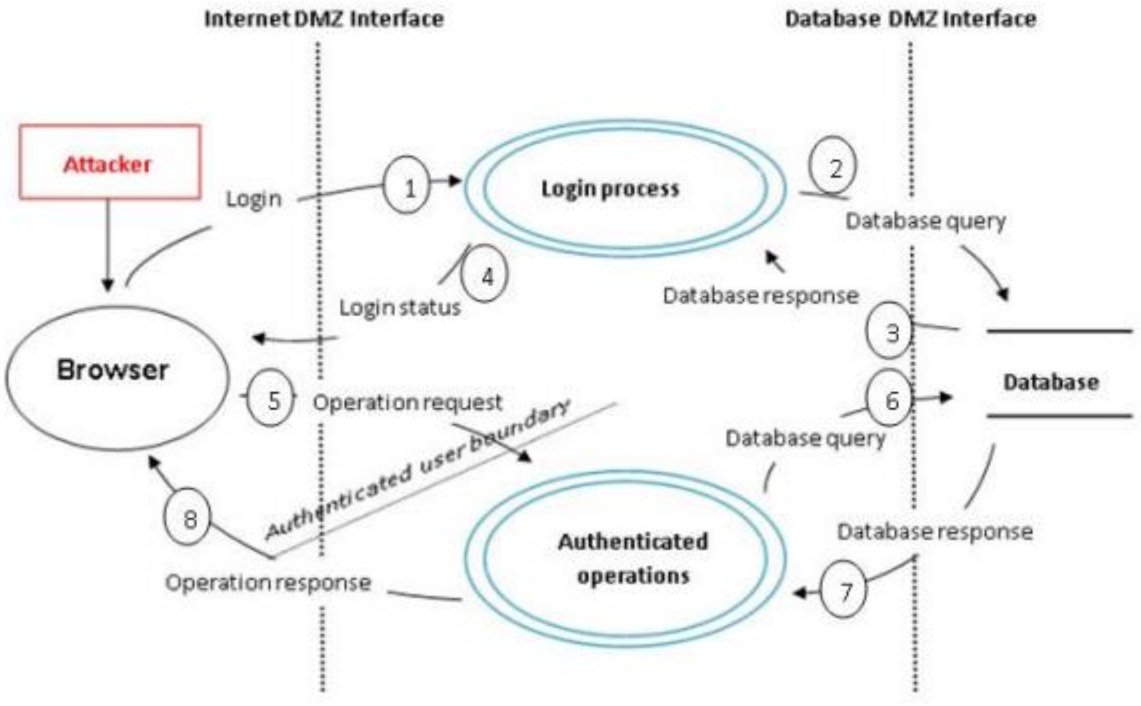
In order to transport data to and from URLs, nsIChannel offers an interface. The ratio of nsIChannels to URLs accessed is exactly one to one. In order to build a channel with the correct protocol, the nsIProtocolHandler is required. The nsIChannel delivers data to the rest of Firefox browser

When the nsIProtocolHandler is called, it generates new channels and assigns them to the proper protocols.

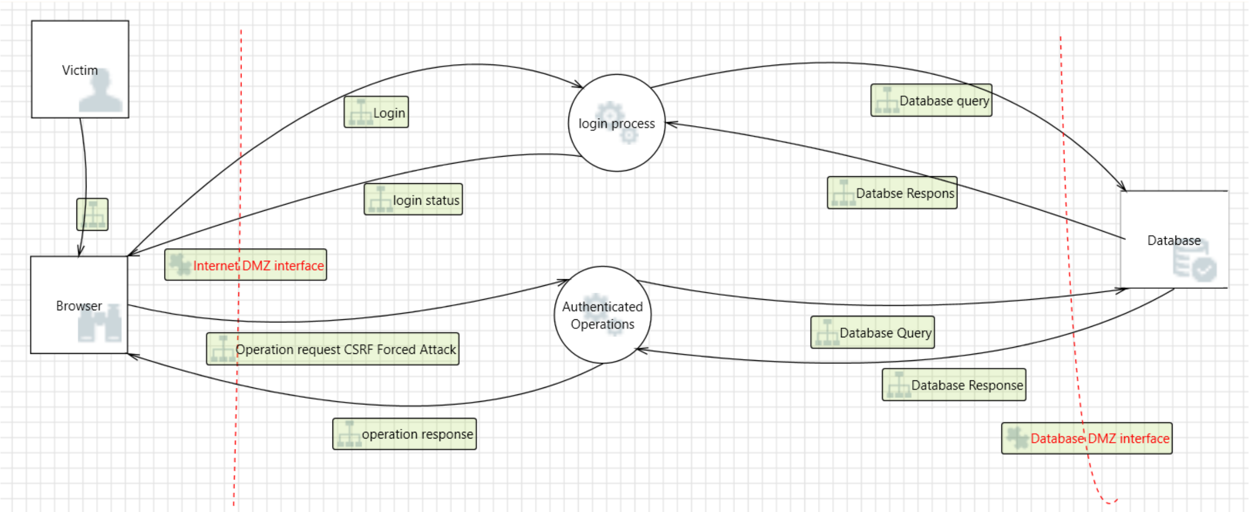
Threat Model

Threat modelling is a structured activity for identifying and evaluating application threats and vulnerabilities. There are three main different perspectives can be taken under this approach and in this paper attacker centric approach was taken from a defensive standpoint, the objectives are to identify potential vulnerabilities, eliminate as many vulnerabilities as feasible, and implement countermeasures to limit the attack risk. It is the security professionals' job to find the weak points that can get exploited by the adversaries in order to get access and compromise the target.

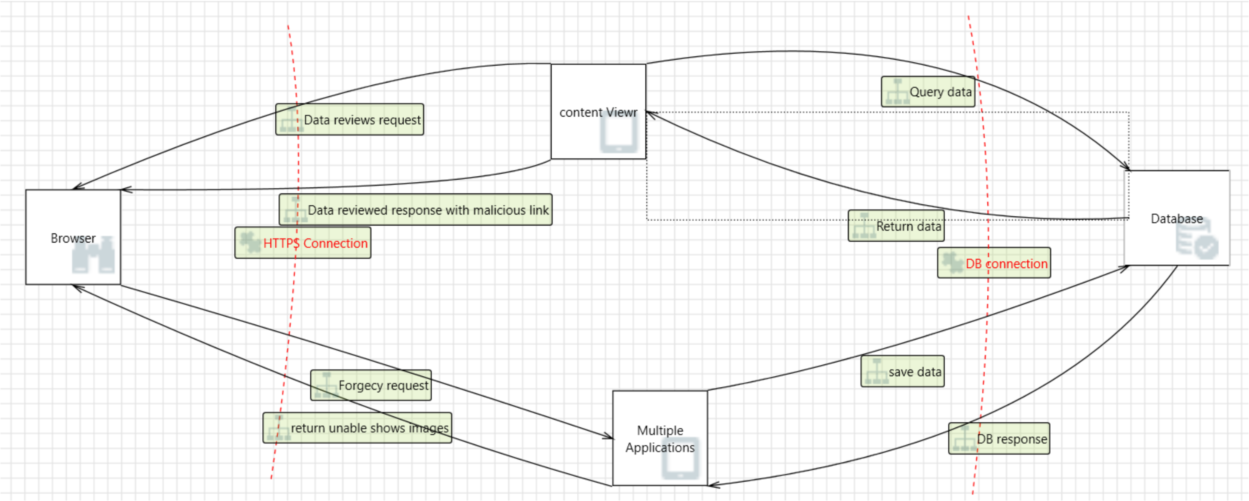
When it comes to browser-based attacks that is carried out against the users of the browsers the attackers' primary goal is to lay foundations for more damaging attacks like CSRF. It might also be a test of their hacking skills by exposing the vulnerabilities of web apps. They may establish a super admin account directly or adjust the firewall settings and appliance preferences of the items on this drive as if they were administrators. These sorts of attacks might be conducted either professionals or amateurs and depending on their level of sophistication attack's impact is determined. Another motive is just for attackers benefit and amusement alone. Inspired attacks are those that use CSRF attacks to change a victim's preferences while simultaneously making the attacker popular and desirous of finding new friends with that fame. Some CSRF attackers may be determined to cause damage to their victims, while others may be amateurs honing their abilities. Another reason is monetary gain, and these types of attacks are driven by a desire for financial gain. This sort of attacker's primary and ultimate purpose is to get financial gain. In order to publish advertisements for items or even transfer money straight from a victim's account, the goal of CSRF attacks is to be used. These are professionals who may make a living from their expertise.



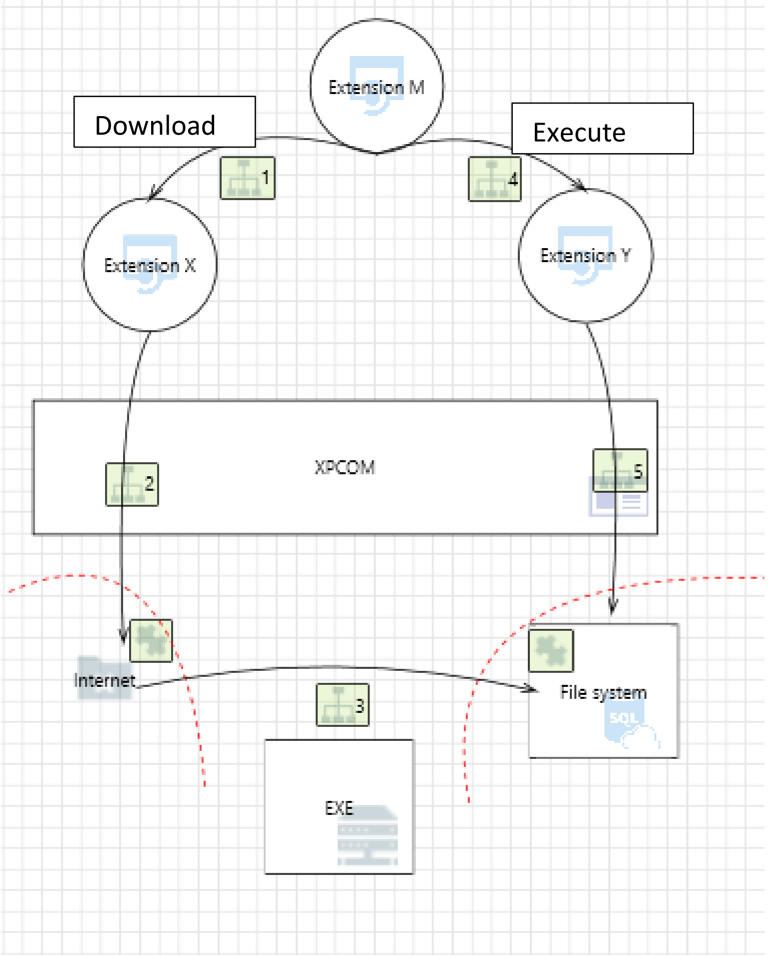
In order to begin, the attacker must investigate the target website in order to discover any feature that might be useful to him (e.g. financial transaction) as shown in the above image. An authorized user account on the targeted website will be required by the attacker. When an attacker has located the weak points, they conduct some tests to see whether their harmful behaviors are successful or not. An attacker develops a malicious link and then tricks the victim into clicking on it, resulting in a CSRF attack.



A fraudulent request is sent to the victim which compromises the browser, which can be seen in the above figure. The victim is attacked while clicking the infected link or reading harmful postings. At the authenticated user boundary, there may be unrestricted authentication area weakness, allowing attackers to get access to the currently logged-in users.



As it can be seen in the above figure, a CSRF attack occurs when the victim examines a malicious post that contains a malicious link embedded in it. The targeted Web application does not properly sanitize user-supplied data before returning it to the user. As a result, a victim's browser may start executing arbitrary HTML or script code



Firefox extensions use the same JavaScript namespace as the browser itself. In other words, any extension installed on a system has unrestricted access to all JavaScript names declared in the extension's global scope. The Mozilla community previously discovered this issue, and it was advised that each extension declare its own namespace to minimize JavaScript name clashes. However, the security ramifications have mostly gone unexplored so far. This common JavaScript namespace enables extensions to read from and write to global variables specified by other extensions, to invoke or override all global functions, and to alter instantiated objects.

While a malicious extension may try an attack simply by making the necessary API calls from its own code, this harmful capability is likely to be identified before the extension is made accessible via Mozilla's online repository. This is because Mozilla requires that all extensions undergo a review process that includes functional testing and source code reviews conducted by human extension vetters. Additionally, the security research community has shown a variety of analytic tools capable of automatically vetting extension code in order to detect and stop suspicious activity.

As can be seen, a malicious extension may use Firefox's common JavaScript namespace to conduct assaults on the system invisibly, bypassing the defenses discussed above. A vulnerability in an extension's extension-reuse mechanism can be defined as a control or data flow from a global JavaScript name to a security-critical API call (e.g., one that provides access to the filesystem or the network, or that allows arbitrary code execution) that could result to compromise the entire machine of the victim. Due to the fact that all extensions have access to global JavaScript names, an attacker can frequently identify a sufficient number of vulnerabilities to write a malicious extension that invokes critical APIs indirectly via other, legitimate extensions, in order to mount a confused deputy-style attack.

According to the image shown above extension M (1)which could be malicious has the ability to exploit the download permission of a legitimate extension X to download malwares in to the file system of the victim machine (2,3) and then same malicious extension M can be used to exploit the execute permission of another legitimate extension like Y (4) to execute (5) the previously downloaded malware to carry out further attacks and compromise the victim

For example, if an extension is malicious but does not call the APIs that allow it to do so, it would be impossible to detect it through human review because it would require an evaluation of the malicious extension in the context of all possible Firefox extensions, or an automated analysis of the entire extension code base, which would be prohibitively expensive or impossible.

Additionally, while it is possible to combine multiple extension-reuse vulnerabilities to create complex attacks, it is frequently sufficient to exploit a single vulnerability to successfully launch damaging attacks, making this attack feasible even when a system has a very small number of extensions installed. For instance, an attacker might simply redirect a user who visits a certain URL to a phishing website or load a web page containing a drive-by download exploit. To simplify, it is believed that any occurrence of an exploitable extension-reuse vulnerability may be leveraged to compromise the security of a Firefox user in this threat model, and hence all such exploits are referred to as assaults.

Inspection Assessment

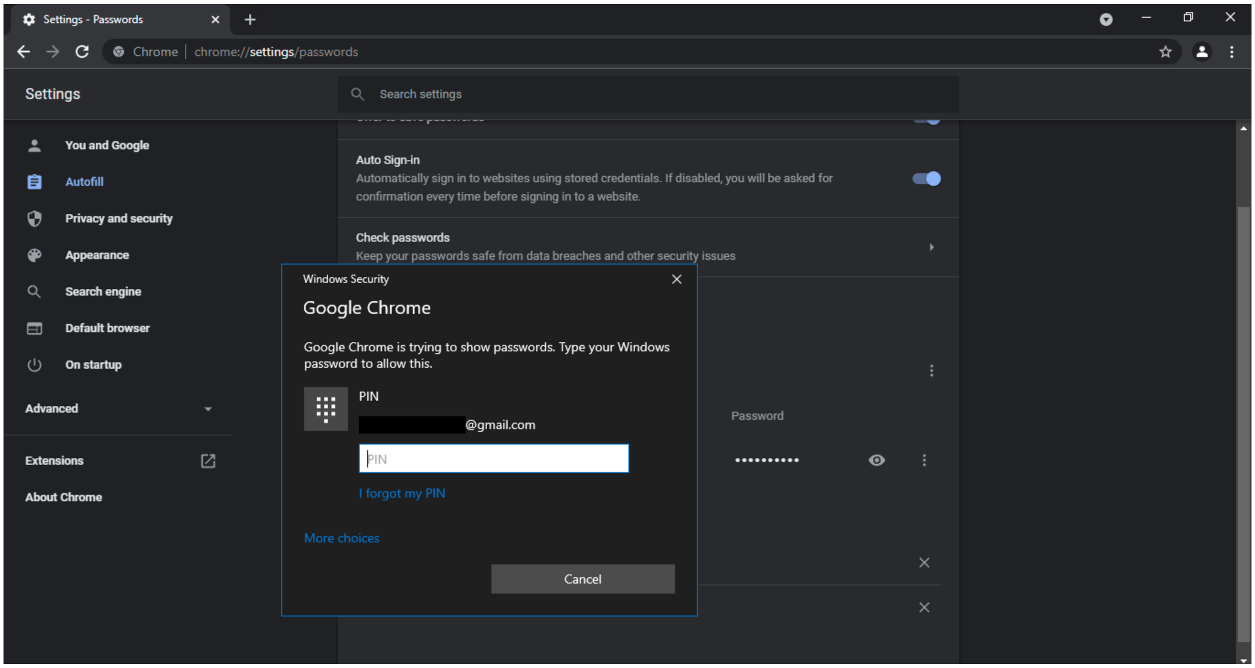
Confidentiality concerns associated with saved user credentials of the Firefox

When a normal user browses web sites, they often create different accounts related do different sites according to user requirements. There most of the users tend to save the corresponding username and password which they used to login to web sites since remembering all the usernames and password is a tedious task for a person. The primary advantage of storing passwords is that they do not need to be entered each time. Thus it reduces the time since users have to go through verification process if the credentials are totally forgot by the users.

While this may seem to be a convenience, it may result in users forgetting their passwords more often and impact a huge risk when it comes to cyber security domain.

There Chrome, Microsoft Edge and Firefox browsers were tested against accessing stored passwords without modifying their default settings.

NOTE: All the browsers were installed on windows 10 machine.



Auto Sign-in

Automatically sign in to websites using stored credentials. If disabled, you will be asked for confirmation every time before signing in to a website.

Check passwords

Keep your passwords safe from data breaches and other security issues

Windows Security

Google Chrome

Google Chrome is trying to show passwords. Type your Windows password to allow this.

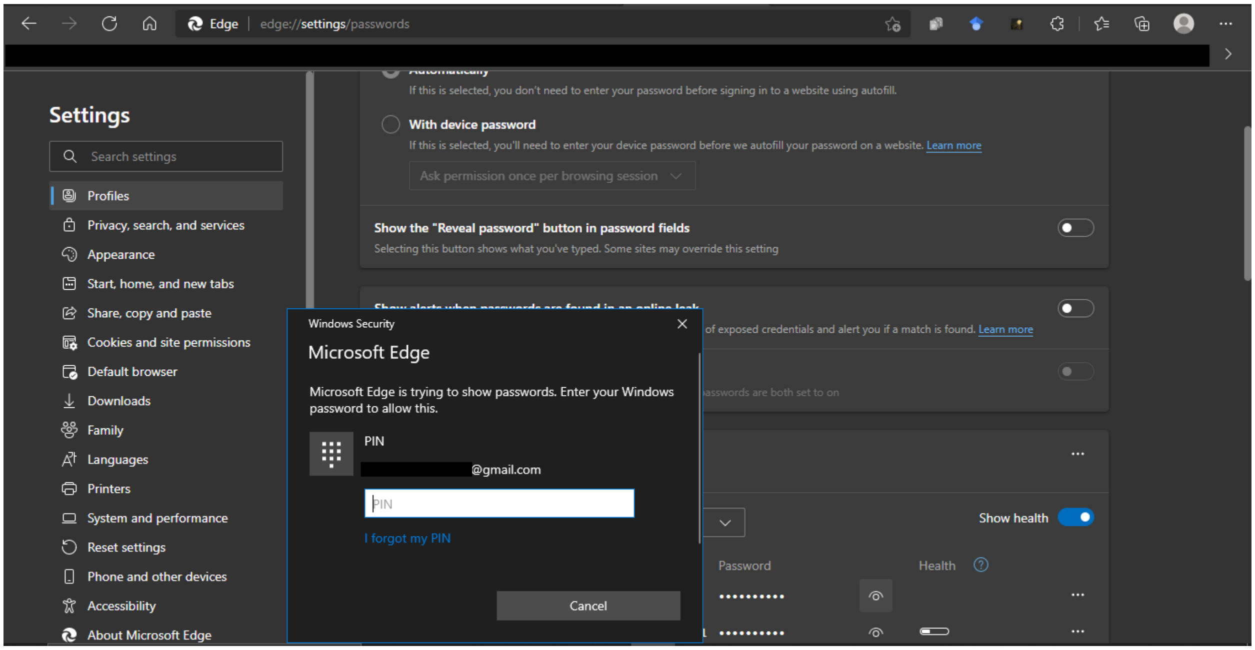
PIN

[redacted]@gmail.com

PIN

Attempting to access stored passwords in Chrome browser

Once the user tries to observe or copy the saved passwords on chrome browser, it prompts a window asking to provide login credentials associated with the host machine in order to reveal them.



[redacted]

If this is selected, you don't need to enter your password before signing in to a website using autofill.

○ With device password

If this is selected, you'll need to enter your device password before we autofill your password on a website. Learn more

Ask permission once per browsing session

Show the "Reveal password" button in password fields

Selecting this button shows what you've typed. Some sites may override this setting

of exposed credentials and alert you if a match is found. Learn more

asswords are both set to on

Windows Security

Microsoft Edge

Microsoft Edge is trying to show passwords. Enter your Windows password to allow this.

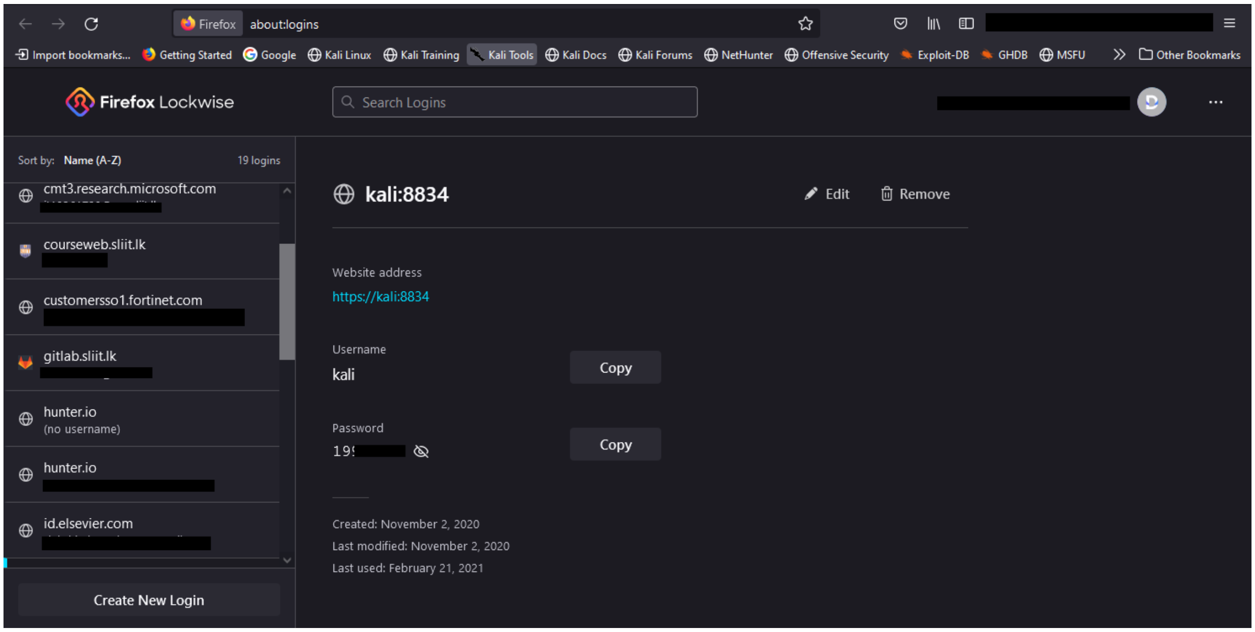
PIN

[redacted]@gmail.com

PIN

Attempting to access stored passwords in Microsoft Edge browser

The same thing was tested with Microsoft Edge browser, and it also prompted a window same as the Chrome browser asking user to provide login credentials associated with the host machine as shown in the above image.



cmt3.research.microsoft.com
[redacted]

courseweb.sliit.lk
[redacted]

customersso1.fortinet.com
[redacted]

gitlab.sliit.lk
[redacted]

hunter.io
(no username)

hunter.io
[redacted]

id.elsevier.com
[redacted]

kali:8834

Website address
https://kali:8834

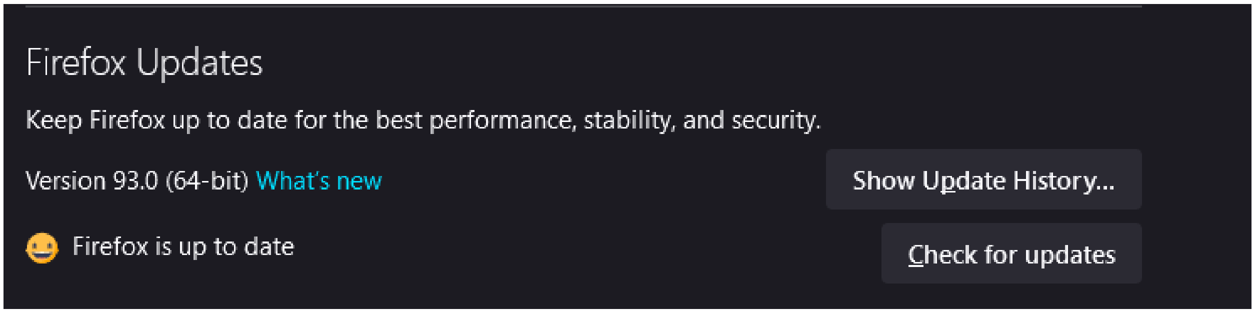
Username
kali

Password
19[redacted]

Created: November 2, 2020
Last modified: November 2, 2020
Last used: February 21, 2021

Attempting to access stored passwords in Microsoft Edge browser

But when it comes to Firefox browser, it did not ask for any kind of verification before accessing the passwords and it revealed the saved passwords and also allowed to copy them as shown in the above image.



Firefox Updates

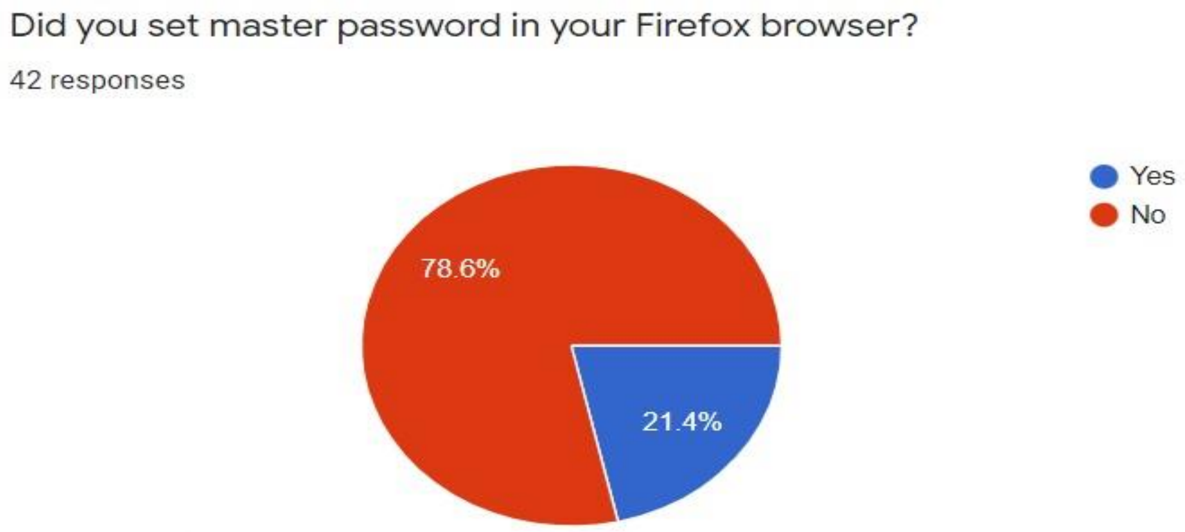
Keep Firefox up to date for the best performance, stability, and security.

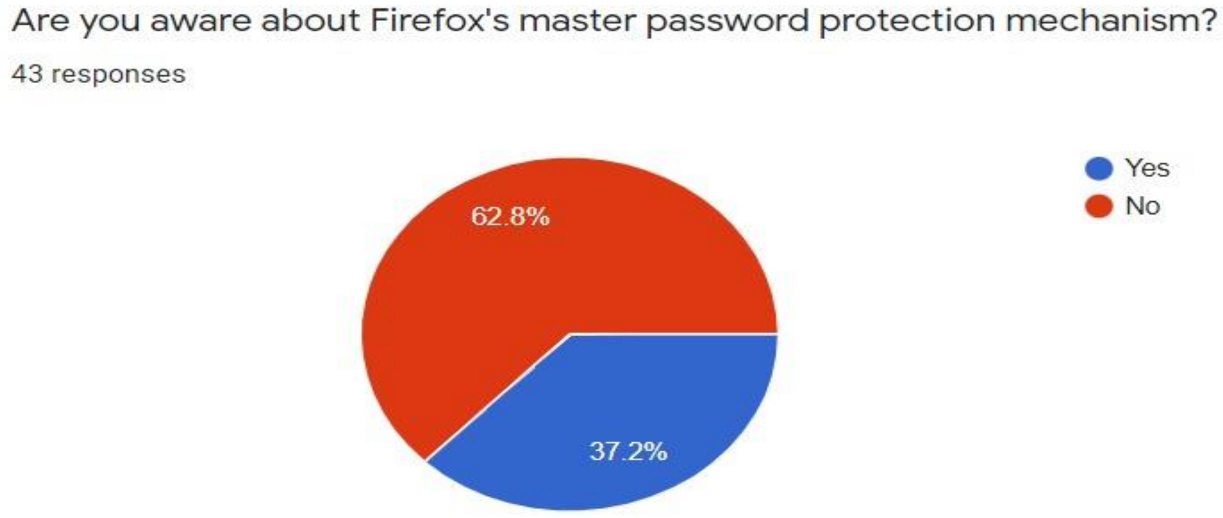
Version 93.0 (64-bit) What's new

Firefox is up to date

Firefox version installed on the host machine

Even though Firefox come up with a solution to add a master password to protect passwords it is not enabled by default, and it is still not addressed by the developers in the latest release as shown in the above image. Furthermore following responses were collected from a group of students of SLIIT, and it proved that most of them are not aware about the master password protection mechanism and even though they aware about it very few amounts of them have set that to protect their stored passwords.

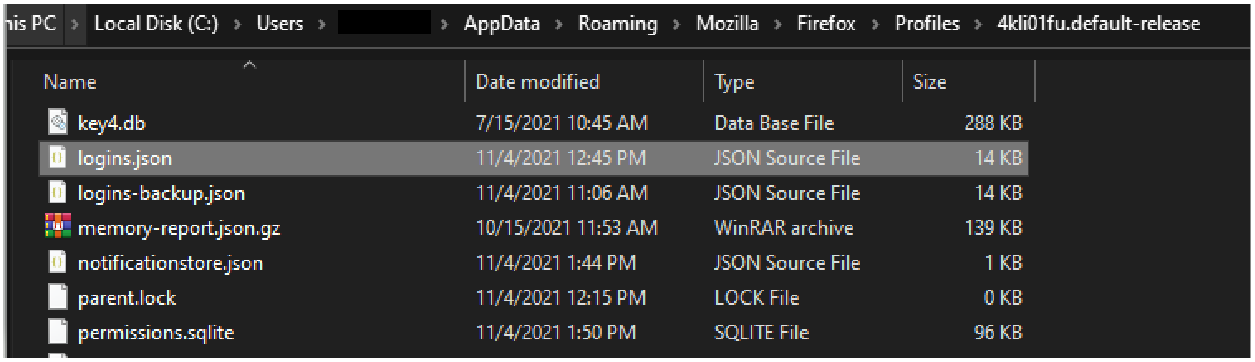




As we all know websites and web applications considered passwords as their primary factor to authenticate and validate their users because it prevents unauthorized access. But when it comes to this issue related with Firefox browser, anyone who has the access to the host machine also have the access to the browser and its stored passwords. This also arise the integrity violation issues since when it comes to the web sites perspective, if the multi factor authentication is not enabled they solely rely on the password for authentication process and there is no way to prove that the legitimate user is login into the site using credentials. Other than that if a user has a bad practice like using the same password for authenticating for different sites also could cause to greater impact to all the other sites as well as it paves the way to attacker to violate confidential, integrity and availability of the sites. For example attacker can log on behalf of the legitimate party and get all the privileges to do actions that the user is normally authorized to do thereby attacker also can leverage those privileges to carry out further attacks. Even though this seems to be a simple problem most of the people are not aware of this issue and majority of them have not set a master password which simply put them in a huge danger. Therefore Firefox need to come up with a solution to protect users saved credentials which is enabled by default to authenticate users before revealing them.

Weak encryption mechanism associated with saved credentials

The value of a hacked system is defined by the value of the data contained on the compromised host and how the adversary may make use of it for malicious purposes. So in this case we can imagine a situation where the user of the Firefox saves all the log in credentials for different accounts of the different sites. There Firefox stores all the login credentials such as hostname, encrypted username, encrypted password in a json file as shown in the mage below



his PC > Local Disk (C:) > Users > [redacted] > AppData > Roaming > Mozilla > Firefox > Profiles > 4kli01fu.default-release

Name	Date modified	Type	Size
key4.db	7/15/2021 10:45 AM	Data Base File	288 KB
logins.json	11/4/2021 12:45 PM	JSON Source File	14 KB
logins-backup.json	11/4/2021 11:06 AM	JSON Source File	14 KB
memory-report.json.gz	10/15/2021 11:53 AM	WinRAR archive	139 KB
notificationstore.json	11/4/2021 1:44 PM	JSON Source File	1 KB
parent.lock	11/4/2021 12:15 PM	LOCK File	0 KB
permissions.sqlite	11/4/2021 1:50 PM	SQLITE File	96 KB

NOTE: normal storage locations of the json file of widely used operating systems

Windows: C:/Users/<PCName>/AppData/Roaming/Mozilla/Firefox/Profiles

Mac: ~/Library/Application Support/Firefox/Profiles

Linux: ~/.mozilla/firefox/Profiles

There the Firefox uses PKCS#11 cryptography standard to encrypt the username and associated password which then uses the device in which the Firefox browser is installed as a "cryptography token" for encryption and decryption. Firefox has developed the NSS library to adopt this standard into their browser. But when it comes to the aforementioned threat scenario attacker can simply obtain the json file from the compromised victim machine and decrypt all the sensitive credentials using a decryption tool without much effort.

Issues associated with site isolation mechanism

Firefox, like other web browsers, downloads and executes code from suspicious websites on the computer of the user. As a solution for that to provide protection against harmful websites and complying with Mozilla's security policies Firefox come up with the idea of site isolation, which is a novel security architecture that separates online content and loads each site in its own operating system process, enhancing current security measures.

As a result of this new security approach, Firefox is better equipped to protect against rogue sites attempting to get sensitive information from other sites the user is visiting. There Firefox browser creates a single, highly privileged, process known as the parent process, which then launches and manages the actions of several web content processes. This process is the most powerful since it can do everything an end-user is capable of doing. and this process-based architecture enables Firefox to isolate potentially dangerous or unreliable code from the rest of the operating system and other users' data. As a result, code with less privileges will have to rely on more privileged code to do tasks that it cannot perform on its own.

Despite the fact that Mozilla is working on Project Fission to put this into practice, and it is still a work in progress and not ready for widespread usage just yet. Firefox I n its current condition, will be far from the maturity needed for site isolation and, and it will take many more years for it to get there. On the other hand, it has a considerable number of cross-site leaks that enable a compromised content process to access the data of another and avoid the site isolation protection mechanism

Issues associated with win32k in Windows platforms

When it comes to the Firefox sandbox protection mechanism and content process, Win32k lockdown is still missing. Historically, Win32k has been the consequence of multiple vulnerabilities, making it a popular target for sandbox escapes and yet Win32k is considered as a risky collection of NT kernel system calls also is a kernel mode driver. However, Firefox has yet to follow Microsoft's lead, since Firefox now only supports this in the socket process, rather than the whole process. Thus it leads to compromise the host by provisioning the full control of the system to the attacker.

Sandbox escapes associated with Linux

Firefox's sandboxing in other operating systems, such as Linux, is noticeably poorer than the Windows operating systems. Generally, the limitations are rather lax, and it's even vulnerable to years-old easy sandbox escape flaws that expose an enormous attack surface from inside the sandbox. The actual problem arises with the X11 system utilized in the Linux and other Unix operating systems that is capable of provisioning framework for GUIs to interact with user via keyboard and mouse. This does not have a proper mechanism to isolate application and fake key events could be injected into the input stream of a one program by any other program and even snapshots of the windows belong to one program can be taken by other program. When it comes to Firefox, the content processes which are used to run web content have a direct connection to the X11 system thus it could be utilized to escape the sandbox protection mechanism of Firefox to execute malicious arbitrary codes. Therefore Firefox need to come up with a solution like provisioning the permission to access X11 only via GPU process whereas render processes or the content process of the Firefox cannot access it.

Restriction issues associated with sys_ioctl arguments

Every user level process has access to a huge number of system calls, many of which go unused over the course of the process. Bugs are discovered and eliminated as system calls evolve. Some user level programs benefited from fewer accessible system calls, whereas others do not. As a consequence, the amount of kernel surface that is available to the program is reduced. Filtering of system calls is intended for usage with such programs. When it comes to Linux operating systems seccomp filtering enables a process to define an incoming system call filter through an API call. The Berkeley Packet Filter (BPF) program is used to implement the filter. To put it another way, this well-defined approach is used for reducing the accessible kernel surface for user level programs. However when it comes to the Firefox seccomp filtering is comparatively less restrictive and still not up to the standards. For example Firefox has very little restriction when filtering ioctl calls. Because the Firefox which calls the ioctl is resides in the user space and the drivers of ioctl resides in the kernel space, the relationship between these two is not as simple as a normal function call. However, due to the security of kernel space operations, the system call process requires a great deal of security processing, which complicates the system call. When it comes to the attacker’s perspective, the non-privileged session can escape to the parent session of the Firefox by using the ioctl to push characters into the terminal's input buffer, allowing an attacker to escape the sandbox protection mechanism of the Firefox. Therefore Firefox needs to come up with a proper filtering mechanism for ioctl calls that are required by the sandbox to function properly which can reduce the attack surface of the kernel level in considerable amount.

Does not support ACG and CIG mechanisms of windows

One of the common attacks which is being exploited in the world is buffer overflow and occurs due to an attempt to write more values than the allocated size of the buffer which is located on the stack. Most of the times this causes to crash the program or to misbehave the program. When a new function is called new stack frame will be allocated and all the local variables and arguments will be pushed on to that particular stack frame along with the return address. When program attempts to write more data than the allocated buffer size it starts to overwrite the values of base pointer as well as the return value. When EIP attempts to read the value of the return address and execute it creates a dangling pointer since it is already filled with garbage values instead of valid memory address. When it comes to attack scenario, attackers take this as an opportunity and inject a malicious script and then overwrite the return address with the memory address which points to the beginning of the malicious script. When EIP reads the return address it will start to execute the malicious script which helps attackers to obtain the control flow and gain unauthorized access to compromised system with special privilege levels. Different solutions were implemented with the time like ASLR, DEP and use of stack canary values. But adversaries executed their malicious code by injecting shellcode into readable memory pages and then exploiting code reuse mechanisms to turn them into executable memory regions by using system functions such as mprotect or VirtualAlloc. Then windows 10 implemented a novel mechanism called arbitrary code guard (ACG) and guaranteed the immutability of all executable memory regions to such attacks. Other than that Another mitigation called code integrity guard (CIG) which is bit similar to ACG in that it applies to the filesystem rather than memory, preventing an attacker from executing a malicious application or library on disc by requiring that all binaries should be signed before importing them into a process. Unfortunately when it comes to Firefox, it still doesn't support both ACG and CIG protection mechanisms on windows platforms and enabling supporting mechanisms for these features will be a significant evolution in Firefox's security posture.

Control Flow Integrity issues

Attackers usually use control flow hijacking techniques like return oriented programming and jump oriented programming concepts to execute malicious opcodes to take over the control of vulnerable machines. These flaws could be exploited by making the Firefox go down a code path the developers never intended. CFI works by making it more difficult for an attacker to carry out their attack. To put it another way, CFI's goal is to enforce what the programmer intended at compile time during the program execution time. Even though this is an essential feature Firefox still struggling to come up with a solution.

Lack of proper hardening mechanisms for JIT engine

All mainstream browsers have a JIT compiler to increase performance. There is an inherent security weakness with JIT though and that is the necessity of memory that is both written and executable permissions associated with JIT engine. In an effort to decrease the security concerns provided by this technology without compromising the speed improvements, browsers have developed JIT hardening methods to make abusing the JIT compiler substantially more difficult.

Firefox tried to strengthen the JIT engine by introducing a so-called "write xor execute JIT" mechanism, however this fails to prevent real exploitation since it is prone to a race window in which an attacker could input malicious shellcode to the memory mapping when it's readable and wait for the engine to convert it to executable. Additionally, owing to the absence of CFI in Firefox, there are also several gadgets available for an attacker to forcibly transition the mapping to executable, such as VirtualAlloc, makeExecutable, ExecutableAllocator or mprotect / in the C library. Therefore Firefox needs to implement protection mechanisms like constant blinding for its JIT engine.

Issues associated with memory allocator

Firefox. Jemalloc is a memory allocator that focuses on speed and efficiency. Because of this lack of security attention, it is very vulnerable to get abused by attackers. It does offer a few beneficial security measures, but they are insufficient in resolving the problems with the allocator's general design introduced by Mozjemalloc. When it comes to the memory partitioning, a heap overflow cannot corrupt an object from a different heap. This is an exploit mitigation technique that uses the memory allocator to split various kinds of objects into distinct, isolated heaps. Firefox's mozjemalloc ultimately added support for partitioning, however since memory is reused between partitions, adversary can trigger an entire page's worth of data to be de-allocated, and eventually reused between partitions. This would allow bypassing the partition protection.