Password Managers: Attacks and Defenses

David Silver, Suman Jana, and Dan Boneh, Stanford University; Eric Chen and Collin Jackson, Carnegie Mellon University

Presented by Sai Tej Kancharla

Contents

Introduction

- With the increase in number of users using multiple accounts on multiple sites, there is a need to store all the authentication details in case the user forgets the login id or password.
- Password Managers are tools which help in storing the user authentication details for various sites securely for easier access to the user.
- All of the Password Managers use autofill policies(automatic or manual) to reduce the wastage of time of user.
- They automatically fill-in the username and password when user enters the login page.

Introduction

- This enables the potential attackers to easily steal our passwords without the users knowledge.
- This paper discusses about the popular Password Managers like the Desktop Browser PMS like Chrome, Firefox and other 3rd party apps like 1Password, LastPass and more
- This paper discusses on ways in which the attacker can steal the data from user without his knowledge
- This paper also gives us some possible ways to prevent suck kind of attacks from happening by strengthening the existing PM and login sites .

Password Managers

- The Password Managers tested were:

Desktop Browser PMs: Google Chrome 34, Microsoft Internet Explorer 11, Mozilla Firefox 29, and Apple Safari 7.

3rd Party PMs: 1Password, LastPass, Keeper, Norton IdentitySafe, and KeePass. All of these besides KeePass provide browser extensions that support password field autofill.

iOS PMs: Mobile Safari's password manager through Apple's iCloud Keychain synchronization service

Android PMs: the default Android browser and Chrome

Password Managers

- All the password managers have Autofill function where they automatically fill the username and password fields in browser automatically.
- There are 2 types of Autofill: Automatic and Manual.
- Automatic Autofill is where the username and password are filled automatically without any user interaction to authorize it.
- Manual Autofill is where the user needs to interact with the system to allow autofill. It can be done by certain ways like clicking, typing, keyboard shortcut or pressing a browser button.
- There are Hybrid Autofill ways where the page is filled automatically on secure paths, and user interaction is required on unsecure paths.

Ways of Attacking Password Managers

- The paper talks about Sweep Attack and various forms of sweep attacks. The Sweep Attack consists of 3 parts:
- The attacker makes the user visit a vulnerable website without users knowledge.
- By tampering with network traffic the attacker injects JavaScript code into the vulnerable webpage as it is fetched over the network.
- The JavaScript code exfiltrates passwords to the attacker.
- They are 3 types: iFrame, Window and Redirect Sweep Attack

iFrame Sweep Attack

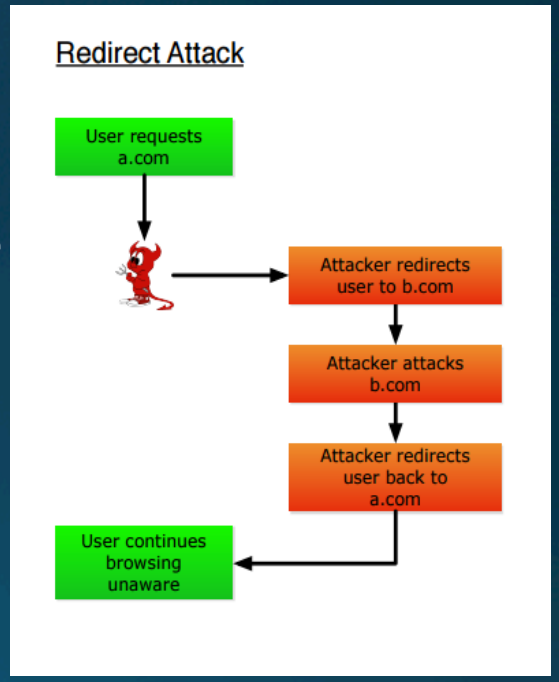
- The attacker directs the user to a landing page which contains iFrames which point to multiple target sites. These iFrames are invincible to the user.
- The attacker injects login form and JavaScript in to each iFrame for stealing the users login credentials.
- As each iFrame loads, the Password Manager fills up the field with the corresponding password.

Window Sweep Attack

- It is also a sweep attack, but instead of using iFrames, it uses windows.
- The attacker can make the landing page open each separate target page in a window, hence making it more noticeable than iFrame attack.
- The windows can be disguised in many ways like minimizing there size, hiding the pages content, closing the window as soon as the password is stolen.

Redirect Sweep Attack

- Redirect Sweep Attack can extract password without using iFrames and Windows.
- Its works on simple logic: when the user requests for a certain page, the attacker responds with a HTTP redirect to vulnerable sites.
- The attacker injects login field into the page and also hides the page from the users view.
- When the PMs autofill the passwords, the JavaScript injected exfiltrates the data and redirects the user to the original page he intended to visit.



Injection Techniques of JavaScript

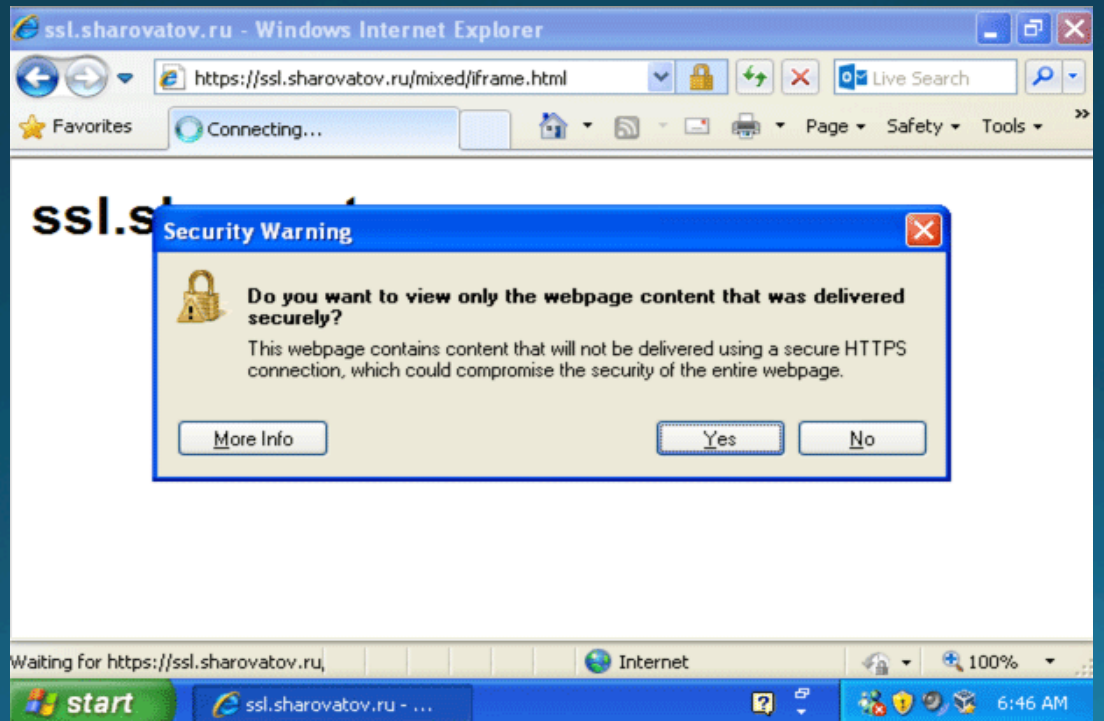
- HTTP Login Page: Websites that serve login pages over HTTP but submit them over HTTPS are vulnerable. While this protects the data while the submission, the attacker can inject the malicious JavaScript at router level and capture data.
- Some sites even serve and submit the login page over HTTP which makes them easier target.
- Embedded Devices: Most of the embedded devices load their login pages over HTTP assuming the channel is protected by network encryptions like WPA/WPA2.
- Some corporate networks also use HTTP to serve login pages as these servers can only be interacted through Virtual Private Network(VPN).

Injection Techniques of JavaScript

- Broken HTTPS: The attacker can exploit sites even when they serve the login pages over HTTPS. Using the Redirect Sweep Attack the attacker serves a self signed certificate to the target site which is redirected from the intended site.
- Even though the self signed certificates produce a warning that the HTTPS is not secure, the user tends click through the warnings and enter the site anyways

Injection Techniques of JavaScript

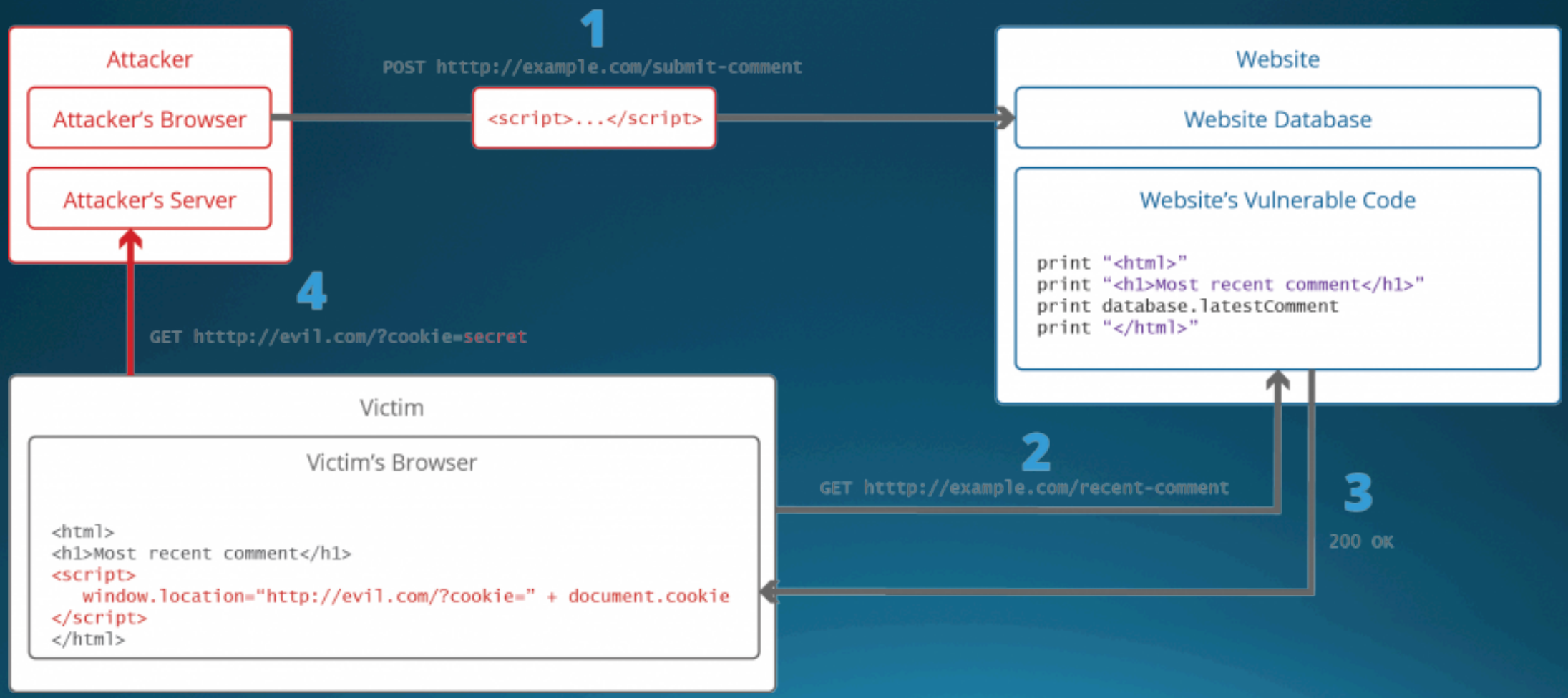
- Active Mixed Content: Any HTTPS webpage containg active content(animated GIFs, JavaScript Applications, Embedded objects and more) which is fetched from HTTP is vulnerable. If the active mixed content is rendered then the HTTPS page is vulnerable.



Injection Techniques of JavaScript

- XSS Injection: Cross-site Scripting (XSS) refers to client-side code injection attack wherein an attacker can execute malicious scripts into a legitimate website or web application.
- By leveraging XSS, an attacker does not target a victim directly. Instead, an attacker would exploit a vulnerability within a website or web application that the victim would visit, essentially using the vulnerable website as a vehicle to deliver a malicious code to the victims browser.
- In order for an XSS attack to take place the vulnerable website needs to directly include user input in its pages. An attacker can then insert a string that will be used within the web page and treated as code by the victim's browser

Injection Techniques of JavaScript



Password Exfiltration

- STEALTH: Here the attacker waits until the PMs autofill the login credentials, then steals the password by loading an attacker controlled page in an invisible iFrame and passing the data as parameters.
- ACTION: The attacker modifies the login forms action attribute so that it submits to an attacker controlled site.
- Since the PMs autofill the passwords on page load and don't warn the user when the submitted form is different then intended, the user will be clueless about the operations.

Securing Password Managers

- Forcing User Interaction: Most of the exploits work mainly based on automatic autofill, so the easiest way to secure the PMs is by making it mandatory for the user to interact with the PM before it autofill's the login credentials.
- Also the PMs should show the domain name for which the autofilling will occur to prevent malicious iFrames accessing the passwords.
- The PMs should not be allowed to autofill in situations like broken HTTPS.

Securing Password Managers

- Secure Filling: The main defense discussed in this paper is Secure Filling. It works as follows:

1. The PM should save Username, Password and also the action present in the login form when they are first saved.
2. When the login form is being autofilled, it should be made unreadable by JavaScript.
3. If the username/password are modified then the autofill aborts and the password field is cleared.
4. Once a autofill is done and all the JavaScript has been run, the browser checks whether the action form in the login page matches with the one it saved originally. If it does not match the password field is erased.

Securing Password Managers

- Limitations of Secure Filling: The secure filling technique does not work with AJAX based login. When the login form's submit button is pressed, these sites use JavaScript to read the form fields, then construct and submit an XMLHttpRequest(XHR) object.
- The XHR is an API used to send HTTP or HTTPS requests to a web server and load the server response data back into the script.
- Data from the response can be used to alter the current document in the browser window without loading a new web page.
- This is not compatible with Secure Filling as JavaScript would not be able to read the password field, hence unable to construct XHR.
- The only workarounds is implementing iFrame or if the browser could provide additional API that allows JavaScript to submit password without reading it.

Securing Password Managers

- Preventing Self Exfiltration attacks: If any page in the victims page supports a public discussion form then the attacker can secure the filling mechanism and submit the login credentials publicly for the attacker to access later.
- For this attack to work, the name of the password field on the login page must be the same as the name of the text field on the public forum page.
- Secure Filling mechanism should only fill a password field whose name matches the name of the field when the password was saved. Also changing the name attribute should cause the auto fill to abort.

Securing Password Managers

- User Registration Pages: HTML does not provide a way to distinguish between password fields on user registration pages and password fields in login forms. Registration pages frequently use JavaScript to evaluate passwords before submission to verify password strength and whether the passwords match.
- It can be solved by allowing the HTML to distinguish between the password fields in login page and registration page. So the PM can allow access to the password field in registration page to JavaScript.

Securing Password Managers

- Server Side Defense: Using HTTPS on both the login page and the page it submits to. Also enabling HSTS (HTTP Strict Transport Security) which prevents the page from loading under HTTP.
- Using Content Security Policy(CSP) to prevent execution of inline scripts. CSP provides a standard HTTP header that allows website owners to declare approved sources of content that browsers should be allowed to load on that page.
- Host the login page in a different subdomain that the rest of the site. This limits the number of pages considered same-origin with the login page, reducing the attack surface.

Conclusion

- The paper allows us to understand how the Password Managers function and how would an attacker access the data using the autofill policy of the PM.
- The paper shows the various ways in which the attack can happen and surveys the password managers individually to know their weakness and exploits possible.
- The paper also provides ways in which the security of the PM can be improved using simple mechanisms.

References

- Password Managers: Attacks and Defenses David Silver, Suman Jana, and Dan Boneh, Stanford University; Eric Chen and Collin Jackson, Carnegie Mellon University

Password Managers: Attacks and Defenses

David Silver, Suman Jana, and Dan Boneh, Stanford University; Eric Chen and Collin Jackson, Carnegie Mellon University

Paper Discussion

- Zhenyu Ning
- CSC 6991 – Advanced Computer System Security

- This paper mainly talks about a specifical man-in-the-middle attack towards Password Manager. In this attack, victims are supposed to be connected to a malicious WiFi network such as free WiFi in a coffee shop. As the attacker can compromise the network traffic, JavaScript injection can be done to steal the passwords stored in Password Managers through some vulnerabilities of them.

- The author mentioned 10 popular Password Managers across 4 platforms and all of them will autofill the password in some situation in which they should not perform this action. For example, when the protocol of the site changed, the form action has changed, the input field of password disabled auto-complete property, the HTTPS certification is invalid, the login form is in an iFrame or a window, Password Managers should not fill the password field automatically instead of warning user or perform some interaction with user. Also, a series of injection techniques were introduced to illustrate the vulnerabilities which could be used in the attack.

- Finally, the author suggests two approaches to strengthen Password Manage, one is forcing user to interaction with Password Manager before it complete the autofill action and another is using dummy value to prevent the password label from reading by JavaScript. But these two policies may still suffer from broken HTTPS.

Paper Discussion

- Hitakshi Annayya
- Now a days there are very much vulnerabilities occur in automatically filling passwords in browser built-in password mangers, mobile password managers etc. Without any interaction with the user attackers can hack the many passwords which are stored in password managers with the rogue WiFi network. This paper discusses the enhancement of the password managers by experimenting with different attack techniques and make a better password manager system to the society.
- The authors start by exploring the 10 existing password managers across four platforms and come to conclusion when to autofill the passwords, and next by showing when can attack can happen and lead to extraction of passwords, and finally how to strengthen the security of credential. An attacker can attack the PM on the least secure page with in the domain (domain and path), login page loaded from one protocol (HTTPS) differ to other protocol (HTTP), modified action form.
- Investigates on many attacks having a man –in –the middle attack networks such as Seep attacks, iFrame sweep attack, window sweep attack.
- Authors find out 2 solutions for many attacks which were discussed in the paper. First , **Forcing user interaction** there should be some user interaction before autofilling a form for a proper defense. This can prevent sweep attacks. Second defense, **secure filling** is that even if an attacker injects malicious JavaScript into the login page, passwords autofilled by the password manager will remain secure so long as the form is submitted over HTTPS.
- Limitations of secure filling:
- 1. Causes compatibility issue with existing sites whose login process relies on the ability to read the password field using JavaScript AJAX based
- 2. Preventing self exfiltration attacks. the attacker is changing the login form's action to another page in the same domain our secure filling mechanism will allow the password to be sent
- 3. User registration pages secure filling proposal is that it cannot improve the security of manually entered passwords.

Paper Discussion

- Lucas Copi
- CSC6991
- 12 October 2015
- Password manager analysis
-
- The paper *Password Managers: Attacks and Defenses* discusses the general methods password managers use for storing and auto-filling passwords into webpages and the security risks associated with these methods.
-
- Attackers can use several methods for confusing password mangers and gaining access to stored passwords. These include: sweep attacks, injection techniques, and password exfiltration. Each method is both explained and its effectiveness demonstrated in the paper.
-
- While the researchers found extreme vulnerabilities in all the password managers they studied, they propose new methods for enhancing security. By carrying out more thorough checks on webpage password forms such as: matching domains, matching actions, and monitoring for malicious javascript code; password managers can better protect user data. The researchers were able to implement these new security measures for the password manager in Google Chrome and found success against the attacks previously discussed in the paper.