分組名單:

姓名	學號
林幼馨	112550021
陳芝瑄	112550024
曾士珍	112550090
蘇宜盈	112550173
楊睿軒	112704039

1. Name of the paper:

Password Managers: Attacks and Defenses

2. Summary:

This paper examines the autofill policies of various password managers, revealing inconsistencies and security vulnerabilities that attackers could exploit, particularly in scenarios such as connecting to malicious Wi-Fi networks. It proposes solutions, such as avoiding autofill under certain conditions and introducing "secure filling" to enhance security. The authors have disclosed their research findings to password manager vendors, resulting in changes to autofill policies aimed at improving security.

3. Strength(s) of the paper:

(1) In-depth Analysis: The paper thoroughly examines the autofill policies of various password managers, revealing inconsistencies and security vulnerabilities. This depth of analysis provides valuable insights into potential weaknesses and areas for improvement in password manager security.

(2) Practical Solutions: It proposes practical solutions to address the identified vulnerabilities, such as avoiding autofill under certain conditions and introducing "secure filling" mechanisms. These solutions are actionable and aimed at enhancing the security of password managers in real-world scenarios.

(3) Engagement with Vendors: The authors have disclosed their research findings to password manager vendors, resulting in changes to autofill policies aimed at improving security. This engagement demonstrates a commitment to real-world impact and collaboration with industry stakeholders to enhance security measures.

(4) Comprehensive Threat Model: The paper presents a detailed threat model outlining various attack scenarios, including remote extraction of passwords and clickjacking attacks. This comprehensive approach helps readers understand the breadth of potential threats facing password managers.

(5) Broader Impacts: The paper discusses the wider impacts of implementing the proposed defenses, including enhanced user security, increased trust in password managers, and mitigation of remote extraction threats. Considering

broader societal implications adds depth to the research and highlights its potential significance in improving cybersecurity practices.

4. Weakness(es) of the paper:

(1) The paper mainly discusses attacks on password managers in web browsers but does not address attacks on standalone password manager applications or other forms of authentication.

(2) The paper discusses various attack scenarios and potential vulnerabilities in password managers. Still, it does not provide concrete evidence or real-world examples to demonstrate the feasibility and impact of these attacks.

5. Reflection:

➢ Learning consolidation

- **Password managers: a survey**

Password Manager Survey

The survey provides a comprehensive overview of desktop browsers, iOS, Android, and third-party apps.

(1) Coverage: Includes mainstream password managers across various platforms.

(2) Third-Party Apps: Examines popular third-party password management applications.

Autofill Policies

Differentiates between automatic and manual autofill methods, highlighting user interaction requirements.

(1) Automatic Autofill: Automatically populates username and password fields upon page load.

(2) Manual Autofill: Requires user interaction before auto-filling.

Autofill Behavior Analysis

Provides a detailed analysis of various autofill behaviors and their security implications.

(1) Domain and Path: Passwords are auto-filled within the original page's domain.

(2) Protocol: Varies among browsers regarding auto-filling on HTTP vs. HTTPS.

(3) Modified Form Action: Examines autofill behavior when the form action differs from the original.

(4) Autocomplete Attribute: Variation in adherence to the autocomplete attribute's disablement of autofill.

(5) Broken HTTPS Behavior: Considers autofill behavior when

encountering broken HTTPS connections.

(6) Modified Password Field Name: Assessment of autofill actions when the password field name is changed.

(7) Additional PM Features: Explores unique features such as iFrame autofill and visibility considerations.

- **Threat Model**

  The attacker only requires temporary control of a network router. Then, they can extract passwords stored in the user's password manager without user interaction.

  Attack Steps:

  (1) Initial Login Phase:
  - A. Users log into various websites without the attacker's interference.
  - B. The password manager records the login credentials used during this phase.
  - C. This step can be performed on a different device from the eventual victim device, primarily if the password manager supports syncing across multiple machines (e.g., Apple's iCloud Keychain).

  (2) Connection to Malicious Network:
  - A. Later, the user connects to a malicious network controlled by the attacker (e.g., a rogue Wi-Fi router in a coffee shop).
  - B. The attacker gains the ability to inject, block, and modify network packets.

- **Remote extraction of passwords from password managers**

  Sweep attacks.

  Goal: Exploit automatic password autofill to steal passwords without user interaction.

  Steps:

  (1) The browser being directed to the vulnerable page.

  (2) JavaScript injection

  (3) password exfiltration

  Effectiveness: Experiments demonstrate the extraction of approximately ten passwords per second.

  Attack amplification: Leveraging password synchronization services to extract passwords from devices that haven't visited the target site.

  Injection techniques

(1) **HTTP login page:** Vulnerable to attack even if the login form is submitted over HTTPS.
(2) **Embedded devices:** Many serve login pages over HTTP, making them susceptible to attacks.
(3) **Broken HTTPS:** Passwords auto-filled even on sites with HTTPS errors.
(4) **Active Mixed Content:** HTTPS pages with active content loaded over HTTP are vulnerable.
(5) **XSS Injection:** Exploiting cross-site scripting vulnerabilities to inject malicious code for password theft.
(6) **Leftover Passwords:** Exploiting old passwords stored in password managers for sites with outdated security measures.

Password Exfiltration Methods

(1) Stealth Exfiltration
A. Wait for the login form to be filled out automatically.
B. Load an invisible attacker-controlled page in an iFrame.
C. Extract credentials as parameters to the attacker's page.

(2) Action Method
A. Change the login form's action attribute to the attacker's site.
B. Submit the form using JavaScript to steal the password.

Clickjacking Attack

(1) Present a benign form overlayed with an invisible iFrame.
(2) The user interacts with the benign form, unknowingly filling in the password.
(3) Password is stolen using any exfiltration technique.
(4) Works when user interaction is needed to fill passwords.
(5) It can be repeated for multiple sites.

- **Strengthening password managers**

Target: ensure that using a password manager results in better security than when users manually enter passwords in a password field.

Defense Measure 1: Mandatory User Interaction

Goal: Ensure password managers are not less secure than manual entry
Attack: Potential risks from automatic password filling.
Defense:

(1) Always require user interaction before autofill, such as keyboard shortcuts, button clicks, or menu selection.
(2) Ensure interactions are protected against clickjacking attacks.
(3) Display the domain name before autofill to inform the user about the auto-filled website.

Implementation:

Chrome: wait_for_username = True

To Minimize Users' Inconveniences: Offer "autofill and submit" functionality triggered by the user, which automatically fills and submits login forms.

Defense Measure 2: Secure Autofill

Goal & Challenge: Prevent malicious JavaScript from accessing auto-filled passwords on insecure web pages.

Mechanism:

(1) Record form actions when storing usernames and passwords.
(2) Make the password field unreadable during autofill to prevent JavaScript from reading it.
(3) Abort autofill if the username or password field is modified during the autofill process.
(4) Before form submission, verify that the form's actions match those recorded during storage.
(5) Prevent JavaScript from accessing the password field to prevent hidden password leaks.
(6) Check that the form's action matches the domain of the action it has stored.

Implementation:

Chrome: modify the PasswordAutofillAgent class.

Limitations of Secure Autofill

(1) Different Login Ways (ex. AJAX):
   ➔ Sol 1: Embed login forms in iframes.
   ➔ Sol 2: Use additional APIs for password submission.
(2) Self-Leakage Attacks: (Leaking passwords to public forums enables attackers to retrieve them.)
   ➔ Sol: Selectively fills password fields only when the field name matches the one saved during password storage
(3) User registration pages
   ➔ JavaScript on registration pages must have access to the password

for evaluation.

➔ Failed to enhance manual password entry security:

➔ Sol: HTML extension for two password field types: one for login (inaccessible to JavaScript) and one for registration (accessible but not auto-filled).

Server-Side Defenses

(1) Use HTTPS on the login page for submission.

(2) Use CSP (Content Security Policy) to prevent the execution of inline scripts.

(3) Host login page separately.

- **Related work**

Vulnerabilities in Password Managers

Belekno and Gasti conducted surveys on password managers, revealing insecure storage methods.

Different threat model: Requires physical device access, unlike attacks discussed in this paper.

Implementation: Survey and analysis likely involved programming languages like Python and Java for data processing and analysis.

Auto-filling of Forms

(1) Attacks exploit crafted forms to steal auto-filled information.

(2) Attacks are distinct from login form attacks; filled information lacks origin specificity.

Implementation: Crafting and testing forms likely involved web development languages such as HTML, CSS, and JavaScript.

JavaScript Injection Attacks

(1) Attackers can steal passwords through injected JavaScript on vulnerable login pages.

(2) Reverse Cross-Site Request (RCSR) vulnerabilities enable phishing attacks leveraging password manager behavior.

Implementation: Exploiting JavaScript vulnerabilities may involve languages like JavaScript and Python for scripting and automation.

Speculated Attacks by RSnake and Saltzman

Speculated attacks leverage auto-filling tools or iFrame injection to steal

passwords.

Implementation: Testing and proof-of-concept development may involve a mix of scripting languages like Python and web development languages like HTML and JavaScript.

XSS Attacks for Stealing Auto-filled Passwords

Suggests using dummy passwords to mitigate XSS attacks.

Implementation: Mitigation strategies likely involve web development languages such as JavaScript, HTML, and CSS.

Browser Bug Databases

Bugs are filed to prevent auto-filling inside iFrames but are ineffective against specific attacks.

Implementation: Bug fixes are likely implemented in programming languages relevant to browser development, such as C++ for Chromium and JavaScript for Firefox.

Secure Password Authentication Systems

Research focuses on designing secure authentication systems to combat phishing and social engineering.

Implementation: Developing secure authentication systems likely involves a mix of programming languages, including Python, Java, and web development languages.

Password Authentication Systems Supporting Autofilling

Aim to prevent phishing attacks without addressing vulnerabilities in existing password managers.

Implementation: Development may involve web development languages and possibly backend languages for server-side processing.

'Password Booth' Proposal by Sandler

Secure browser-controlled mechanisms like secure filling defense, but without consideration of password managers.

Implementation: The proposal likely involves web development languages for UI/UX design and scripting languages for browser interaction.

➢ Improvement and Extension

(1) **Expand Survey Scope:** Include a broader range of enterprise-level

password management solutions, such as HashiCorp Vault or Thycotic Secret Server, to gain a more comprehensive understanding of different use cases and security requirements.

(2) **In-depth Analysis of Autofill Behavior:** Conduct thorough research into autofill behavior across various browsers and operating systems, including different autofill policies, encryption standards, and security features, to provide more nuanced security recommendations.

(3) **Comprehensive Threat Model:** Extend the threat model to categorize attackers as internal and external, considering their different motivations and capabilities. Additionally, further internal threats should be considered, such as insider misuse of privileges or malicious internal users.

(4) **Improved Password Exfiltration Methods:** Develop more sophisticated password exfiltration methods, including leveraging emerging technologies such as machine learning and artificial intelligence for attack vectors. Additionally, it proposes more defenses based on multi-factor authentication and biometric technologies to enhance authentication security.

➢ Broader Impacts

(1) **Enhanced User Security:** The proposed technology significantly improves user security by implementing mandatory user interaction and secure autofill mechanisms in password managers, reducing the risk of password theft from various remote extraction techniques and clickjacking attacks.

(2) **Increased Trust in Password Managers:** Users can have greater confidence in password managers, knowing that they offer better security than manual password entry, which may lead to increased adoption and reliance on these tools for managing credentials.

(3) **Mitigation of Remote Extraction Threats:** By strengthening autofill mechanisms and implementing server-side defenses like HTTPS and CSP, the technology effectively mitigates the threat of remote password extraction, reducing the success rate of sweep attacks and injection techniques.

➢ Observations on Changes in Recent Years

This paper provides numerous comprehensive suggestions for password management and auto-filling, which offer valuable insights. With the increasing cybersecurity awareness and ongoing technological advancements,

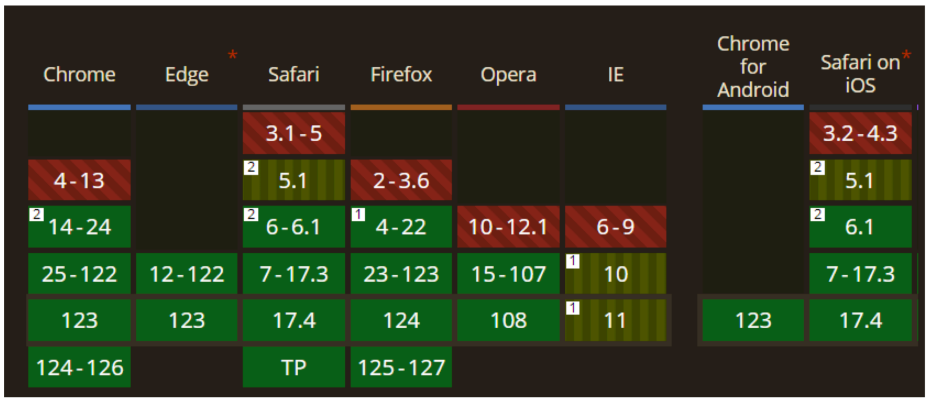
significant browsers and website platforms are actively responding in the real world. For instance:

(1) Since 2018, Google has been flagging HTTP websites as insecure due to the risk of third-party interception of data transmitted between such websites and users or servers. This necessitates several steps to bypass the warning and access the target website.

➔ Such actions align with the recommendations in the paper regarding HTTP warnings and the use of HTTPS for website submission.

(2) Most current browsers support CSP (Content Security Policy), a mechanism browsers provide for websites to set up a whitelist.

➔ Such actions align with the server-side defense mentioned in this paper.



▲ supported browsers and versions

➢ Opinion

In our view, using automatic password autofill functionality presents a dilemma between convenience and risk.

From the perspective of convenience:

(1) The automatic password autofill feature saves time, as users do not have to remember or manually input all account information.

(2) More complex and less predictable passwords can be used because users do not need to remember these passwords but instead have them managed by a password manager.

However, there are risks:

(1) The automatic password autofill feature can make passwords more susceptible to theft by hackers or malicious software, compromising the security of passwords.

Therefore, we should rely on more than just browsers to provide us with secure password management. Instead, we should protect our passwords by regularly changing them, using multi-factor authentication, and avoiding

using the same password across different websites to reduce the risk of password theft.