Preventing the Revealing of Online Passwords to Inappropriate Websites with LoginInspector

Chuan Yue
University of Colorado Colorado Springs
26th Large Installation System Administration Conference (LISA 2012)
Text Passwords: the Dominant Position in Online User Authentication
Password Security

- The *something you know* authentication factor
- Expectations: *strong, protected from being stolen*
- Reality: *weak/shared passwords, various attacks*
Related Features and Mechanisms in Browsers
(Internet Explorer, Firefox, Google Chrome, Safari, and Opera)

- Password Manager
- Phishing Detection and Warning
- Extended Validation (EV) Certificate
Are those password related features and mechanisms in modern browsers sufficient?
Are those password related features and mechanisms in modern browsers sufficient?
Accidental Revealing of Online Password to Inappropriate Websites May Happen!

• We highlight two cases
  – undetected phishing attacks
  – risky password tries

• Modern browsers do not provide sufficient protection
Accidental Revealing of Online Password to Inappropriate Websites May Happen!

- We highlight two cases
  - undetected phishing attacks
  - risky password tries

- Modern browsers do not provide sufficient protection
Outline

• Introduction

• Motivation, Justification, and Related Work

• Design of the LoginInspector

• Implementation and Evaluation

• Security, Usability, and Deployment Analysis

• Conclusion and Acknowledgments
Undetected Phishing Attacks

- Browsers **fail** to detect phishing attacks and give warning
  - Blacklist-based techniques, heuristic-based techniques
  - Not able to detect all the phishing attacks in a timely manner and meanwhile maintain a low false positive rate [4, 13, 29, 39, 48, 49].

- **Passwords for real sites** $\rightarrow$ inappropriate phishing sites!

- LoginInspector takes a **whitelist-based** approach
  - Provide one more layer of protection even if browsers failed
Risky Password Tries

• When users forget passwords for one site, a common practice is to try passwords for other sites they remember.
  – A user study for testing whether this risky practice is common

• Browsers do not and do not have the knowledge to detect

• Passwords for high-security sites $\rightarrow$ inappropriate low-security sites!

• LoginInspector intends to also detect this risky practice
The First User Study on Risky Password Tries

- 30 participants, on campus
- a five-point Likert-scale [58] questionnaire with 7 questions

Q3: Agree or Strongly Agree that sometimes they **forget the password** for a website
Q5: Agree or Strongly Agree that sometimes they **try the password** for one website on another website
Q7: Agree or Strongly Agree that when they try the password for one website on another website, they **hope the Web browser can give them a warning**
Some Closely Related Work

• Password hashing systems
  – E.g., Password Multiplier\textsuperscript{[14]}, PwdHash\textsuperscript{[33]}, Passpet\textsuperscript{[43]}
  – Migrating original passwords to hashed ones is a big burden
  – Cannot log into a website without the tool

• Whitelist-based systems
  – E.g., Antiphish\textsuperscript{[24]} – uses password encryption, less fine-grained
  – E.g., Web Wallet\textsuperscript{[41]} – uses password encryption, special UI
  – Hashing is more appropriate than encryption, users prefer regular login forms than special login dialog boxes
The Key Idea and Functioning of LoginInspector

- Continuously monitor a user’s login actions and securely store domain specific successful login information to an in-browser database.
- For any login attempt that does not have the corresponding successful login record, warn and enable the user to make an informed decision.
High-level Architecture of LoginInspector
The Successful Login Profile Database

• An in-browser database instance
  – Contains a loginprofile table

<table>
<thead>
<tr>
<th>id</th>
<th>domainHmac</th>
<th>recordHmac</th>
<th>timesUsed</th>
<th>firstUsed</th>
<th>lastUsed</th>
</tr>
</thead>
</table>

\[
domainHmac = HMAC(key, d) \quad (1)
\]

\[
recordHmac = HMAC(key, d \parallel u \parallel p) \quad (2)
\]

where, \textit{HMAC} is Keyed-Hashing for Message Authentication\textsuperscript{[27]} with SHA-256 \textsuperscript{[59]} cryptographic hash; \textit{key} is secret key stored in password manager and protected with a master password; \textit{d} is extracted from each login form’s owner document (e.g., https://www.amazon.com or http://en.wikipedia.org).
Login Fields Identification and Protection

• Identification: first password field, then username field
  – Password field: user-assisted identification (“@@” prefix[33]) and automatic identification; Username field: heuristic

• Protection
  – Intercept password keystrokes, generate fake ones, replace back
Login Profile Inspection

• When a user submits a login form
  – Compute a \textit{currentDomainHmac} and a \textit{currentRecordHmac}
  – Run the login profile inspection procedure

\begin{verbatim}
Inspection (currentDomainHmac, currentRecordHmac)
1. if a record with recordHmac=currentRecordHmac exists
2. return ExactMatch; \quad \rightarrow \text{Submit the form using real password}
3. else
4. if a record with domainHmac=currentDomainHmac exists
5. return DomainMatch; \quad \rightarrow \text{Display \textit{Credential Mismatch} warning}
6. else
7. return NoMatch; \quad \rightarrow \text{Display \textit{Initial Visit} warning}
8. endif
9. endif
\end{verbatim}
Warning Generation

- Modal chrome type of dialog box
Admin Report

• Generate/send reports to system administrators if enabled
  – some users may not properly interpret the warning messages
  – only contain the LoginInspector usage information, e.g., a user’s responses to the two types of warning messages in a session

```json
{"userid": "123456",
 "ignored Initial Visit warning": "10 times",
 "ignored Credential Mismatch warning": "6 times",
 "sessionStartTime": "1345846451434",
 "sessionEndTime": "1345846648635"}.
```

– administrators can help individual users or aggregate information
Successful Login Detection, Management, Import/Export

• Successful Login Detection
  – Heuristic approach does not always work well
  – A user-assisted method is useful, a dialog box with “Yes”, “No”
  – Determine if a new successful login record should be added

• Management
  – customize warning messages, remove records, etc.

• Import/Export
  – export records to a file, import from another computer
Implementation and Evaluation

• Firefox Extension
  – Pure JavaScript
  – SQLite\textsuperscript{[62]} database instance
  – Possible for other browsers

• Correctness Evaluation
  – Works correctly on 30 popular legitimate websites, 30 phishing websites, and a new phishing scam\textsuperscript{[60]}

• Performance Evaluation
  – Overhead is low on 30 popular legitimate websites
Correctness Evaluation (1)

- Works correctly on 30 popular legitimate websites
  - Automatic password/username fields identification
  - Correct passwords interception and replacement
  - Correct database operations, login profile inspection, etc.
  - Automatic successful login detection works on 29 sites; the one with an extra link on the failed login page needs user assistance
  - Correct decisions on whether and what type of warning messages should be displayed
Correctness Evaluation (2)

• Works correctly on 30 phishing websites
  – Automatic password/username fields identification on 29 sites; the one with password $type=\text{“text”}$ needs user assistance
  – Correct passwords interception and replacement
  – “Initial Visit” warning message was correctly displayed

• Firefox failed to detect seven of them

• Google Chrome failed to detect eight of them
Correctness Evaluation (3)

• Works correctly on a new phishing scam\textsuperscript{[60]}
  – Email attached HTML file, POST type HTTP request to a hacked legitimate site, very stealthy
    (1) a browser simply loads the phishing webpage as a local file such as file:///C:/Users/.../home.html
    (2) the form is submitted to a legitimate, albeit hacked, website

• Firefox and Google Chrome \textit{did not detect} such scams\textsuperscript{[60]}
Performance Evaluation

- Overhead is low on 30 popular legitimate websites — 2.67GHz CPU
- HMAC calculations completed in 3 milliseconds
- Overhead is mainly on JavaScript invoked SQLite operations
  - **Insert**: average 140.6 milliseconds, with standard deviation 47.2
  - **Update**: average 70.2 milliseconds, with standard deviation 13.1
  - Overhead is incurred only when a login form is submitted
Security, Usability, Deployment Advantages

- **Security**
  - Only store hashed value, does not involve third party
  - Display “active” warnings, send reports to administrators

- **Usability**
  - Does not need to change the original passwords for any site
  - Designed as an auxiliary tool, does not affect the login process

- **Deployment**
  - Can be incrementally deployed, deployment is very simple
Security & Usability Limitations and Suggestions

• The effectiveness of “active” warnings still depends on whether a user can read/understand/pay attention to them
  – a training should target at-risk population, be cost effective

• In the profiling phase, warnings must be carefully ignored
  – perform the profiling in a batch manner, e.g., in an hour
  – system administrators can help regular users build up the profile
  – be cautious about the warnings if they appear again

• The successful login profile is only locally accessible
  – Synchronize to a cloud storage service
Conclusion and Future Work

- Accidental online password revealing may happen
- *Undetected phishing attacks, risky password tries*
- LoginInspector – a profiling-based warning mechanism
- Implemented and evaluated as a Firefox extension
- Future: usability evaluation, password manager integration
Acknowledgments

• Anonymous reviewers, shepherd Mario Obejas
• Jeff Hinson for his important contributions
• Voluntary students and faculty members in user study
• UCCS 2011-2012 CRCW research grant

Thank You!