The security of Mozilla Firefox’s Extensions

Kristjan Krips
Topics

- Introduction
- The extension model
- How could extensions be used for attacks
  - website defacement
  - phishing attacks
  - cross site scripting
- The attacks could result in:
  - loss of sensitive information
  - weakened security
• How are the extensions distributed?

• How could bad extensions be distributed?
  - by hijacking a public Wi-Fi
  - by installing a bad extension on a public computer
  - by using a trusted extension

• Ways to improve the current security model

• Conclusion
Introduction

• **Why?**
  - Firefox is popular
  - it is possible to modify its functionality
  - the growth of attacks against Firefox
  - it is easy to write bad extensions
  - to see how vulnerable the current security model is

• **An extension is a small add-on that enhances the browser with additional functionality**
The extension model

- Extensions work by overlaying the code of Firefox
- The extension code will be merged into Firefox
- An extension is usually made from files of content, skin and locale.
- Extension’s functionality is modified by XUL (XML user interface language) and JavaScript files
- XUL-based applications load the code for their interface from chrome:// URLs.
The basic files of an extension and their functionalities

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>install.rdf</td>
<td>The description of the extension</td>
</tr>
<tr>
<td>chrome.manifest</td>
<td>Contains the locations of chrome files plus their usage instructions</td>
</tr>
<tr>
<td>overlay.xul</td>
<td>Describes UI elements, which are added to the browser window</td>
</tr>
<tr>
<td>overlay.js</td>
<td>Contains scripts, which run in the browser window</td>
</tr>
<tr>
<td>overlay.dtd</td>
<td>Contains translations for overlay.xul, is a part of the locale</td>
</tr>
<tr>
<td>overlay.css</td>
<td>Changes the appearance of UI elements</td>
</tr>
</tbody>
</table>
• Extensions can access the Gecko engine

JavaScript  ←  XPConnect  ←  XPCOM

• XPCOM (Cross Platform Component Object Model)
  - components or reusable cross-platform libraries define:
    - navigation
    - window management
    - managing cookies
    - bookmarks
    - security
    - searching
    - rendering
    - etc.
Possible attack vectors

- **Website defacement**
  - it is possible to change the way a web page is being displayed while it is being loaded
  - even on https pages
  - this is done by modifying DOM (Document Object Model)
• **Phishing attacks**
  - phishing is a type of fraud, which tricks users to give away sensitive information
  - this could be done via directing the user to a fake web site
  - to achieve this the fake web site has to be identical to the real one

- Firefox classifies web pages into three categories:
  - pages with no identity information
  - pages with basic identity information
  - pages with complete identity information

• - To visualize the categorization it provides a colored button on the left side of the address bar since version 3.0
- Another security feature is the padlock icon, which is shown on secure sites. Firefox places the icon on the right side of the status bar.

- It is possible to add an identical padlock icon to the status bar.

- The color of the identity button changes depending on the available identity information.

- It is easy to change the color of the identity button.
• Cross site scripting

- Cross-Site Scripting attacks are a type of injection problem, in which malicious scripts are injected into the otherwise benign and trusted web sites.

- Input rendered in the chrome is a potential XSS injection point
- XSS in chrome is privileged code, so there are no same origin policy restrictions

- It means that some extensions may have security holes, allowing a web page to inject scripts, which could alter the behavior of another web site
Possible attack objectives

- Stealing sensitive information
  - Website defacement can result in loss of usernames and password
  - For example the function behind a login button could have been changed to send the data to the attacker
  - A bad extension could collect the usernames and passwords, which are saved in Firefox and send these to the attacker
• Weakening the security

- There are no security boundaries between extensions

- It is possible to write an extension, which alters the behavior of another extension

- For example the Sage 1.4.3 extension allowed HTML and JavaScript in the <description> tags of RSS feeds to be executed in the chrome security zone.

  - a malicious feed was able to change the settings of an extension called NoScript
How are the extensions distributed?

- Sandbox review system
- It is easy to install extensions from the sandbox
- To make an extension public it needs to be reviewed by two editors
  - Anyone can apply to become an editor
  - This requirement style won’t guarantee that the new editor has the proper knowledge to deal with security issues.
How could bad extensions be distributed?

- Hijacking a public Wi-Fi
  - Usually the connection is not encrypted
  - It is possible to take control of the network
  - Possible to fake an update
  - Every time the browser starts it checks for updates
  - Some extensions have disabled promting the user about the available update
    - thus no human factor
• Installing a bad extension to a public computer

  - Installing an extension is easy
  - It doesn’t require administrative rights
  - Infected public computers could gather much sensitive information

• Using a trusted extension

  - Trusted extensions that are hosted at Mozilla’s official site don’t need reviews for updates.

  - Risk of future updates being unsafe
  - The developer can write a bad extension and publish it at the official web site
Ways to improve the current security model

• At the moment the code added by the extensions is fully trusted

• The extensions shouldn’t have the right to modify the content of an https page.
  - disable all extensions on secure pages
    - would need a restart
  - allow trusted or certified extensions
    - same problem with the restart
  - making it impossible to modify DOM
    - advertisements can’t be blocked
    - may lower the popularity of the browser
• There is a solution for these problems
  - Firefox in safe mode
  - safe mode disables extensions

• Problems with safe mode
  - users like to use extensions
  - switching to safe mode requires a restart
  - users aren’t aware of the vulnerabilities in extensions
Conclusion

• Current extension model has its flaws
  - Extensions can be powerful but all of their code is trusted
  - Website defacement is allowed
  - It is very easy to spread the extensions
  - There are no boundaries between extensions

• The new extension model should
  - limit extension’s rights on secure sites
  - limit the rights of extensions that are not certified or trusted
  - create boundaries between extension
Thanks for the attention

Questions or comments?