Memoirs of a Browser
A Cross-browser Detection Model
for Privacy-breaching Extensions

May 02, 2012, Seoul, Korea

Cristiano Giuffrida - giuffrida@cs.vu.nl
Stefano Ortolani - ortolani@cs.vu.nl

Vrije Universiteit
Amsterdam, The Netherlands

Bruno Crispo - crispo@disi.unitn.it

Università di Trento
Trento, Italy
Google's Chrome Web store used to spread malware

By Dan Goodin | Published 6 days ago

Aprenda tirar vírus do face

Comunidade

Siga os Passos Abaixo

1º) Clique em Instalar Aplicativo.

Install aplicativo

2º) Clique em Permitir ou Continuar.

3º) Clique em Instalar ou Instalar Agora.

Após fazer os passos acima, feche o navegador, abra de novo e confira.

Crooks have found a new venue to push malware: the official [Google Chrome Web Store](http://google.com). It was recently used to hawk Chrome browser extensions secretly hijacking users' Facebook profiles.

*According to* Kaspersky Lab expert Fabio Assolini, one malicious extension hosted on Google's own servers contained hidden code that "can gain complete control" of the user's Facebook profile. The extension then used that access to spread malicious messages and register Facebook Likes for certain items, also inviting fellow users to install it. The same operators advertised a service that delivered Likes of companies looking to promote their profiles. It costs about $27 per 1,000 Likes.
Mozilla pulls password-sniffing Firefox add-on

Mozilla has issued a warning that a Firefox add-on available from the official Mozilla Add-Ons website was secretly sending users' stolen passwords to a remote location.

"Mozilla Sniffer" was uploaded to the Firefox add-on site on June 6th, but was only determined at the start of this week to contain code that sent the contents of website login forms to a remote location.

In other words, if you installed this add-on (and according to Mozilla about 1800 people did) then everytime you entered your password on a website you were potentially handing over your password to crackers.
Twofer Malware: Rogue Internet Explorer Plugin Infests Firefox
22 September 2011

Fake IE plugin & false Firefox add-on in one malicious package

A new piece of malware has emerged to help even out the debate between fans of Firefox and aficionados of Internet Explorer: a rogue IE plugin that also infects Firefox.

The malicious browser helper object (signed by Bitdefender as Trojan.Tracur.G) is downloaded and installed by users who believe they are upgrading Flash Player for Internet Explorer. Once on the system, the rogue BHO will drop – oh, the irony - a rogue Firefox add-on (identified by Bitdefender as Trojan.JS.Redirector.KY) that will monitor users’ on-line habits. At this point, switching browsers won’t help. Both Internet Explorer and Firefox browsers are infected.

The fake Firefox add-on (identified as Trojan.JS.Redirector.KY) “screens” any new web pages loaded into the browser and what kind of information (images, videos, maps, advanced searches) is fetched for the user. If the user types the URL of a search engine, such as Google, Yahoo, Aol, Bing or Ask, it would inject a piece of JavaScript code in the head element of the results page. The JavaScript snippet will hijack the users’ search toward malicious or fake web pages. From these sites, victims can end up downloading additional malware or adware.

By now, it should be a golden rule to stay away from software offered by non-official sites. Double-check, or triple-check, the location’s credibility before downloading a program, application or add-on. If you want to download, make sure the program starts with the URL of the official site.
Also... too many releases to deal with
• **Web browsers are:**
  • complex and variegated
  • constantly updated
  • interpreters that execute code

• **Browser extensions are:**
  • pieces of executable code
  • trustworthy just because someone says so
  • cross-os. Even cross-browser sometimes...
Existing Approaches

- **Existing approaches are:**
  - tailored to a single browser and extension paradigm [Kirda06]
  - tailored to a single extension paradigm [Li07]
  - tailored to a single version [Dhawan09]

- **A desirable approach should be:**
  - cross-version
  - cross-browser
  - not dependent on the extension paradigm
Outline

- What are privacy-breaching extensions?
- The malicious behavior from a system perspective.
- Modeling the malicious activity.
- Detection as a classification problem.
- Results and Conclusions.
Privacy-breaching Extensions (1/2)

- **Privacy-breaching** extensions are extensions that are after private user data.
- “Being after” as in “Harvesting and Leaking”
- “Harvesting and Leaking” as in “Malicious Activity”

- **What is private?** It depends on the malicious extension!

<table>
<thead>
<tr>
<th>PB Extension</th>
<th>Private Data</th>
<th>Private INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Sniffer</td>
<td>Form Values</td>
<td>Form Filling/Submitting</td>
</tr>
<tr>
<td>Keylogger</td>
<td>User Keystrokes</td>
<td>Typing of Keystrokes</td>
</tr>
<tr>
<td>Search Engine Logger</td>
<td>Search Queries</td>
<td>Search Fields Filling/Submitting</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Privacy-breaching Extensions (2/2)

- **What about legitimate extensions?**
  - They can be legitimately interested in private data, for instance user keystrokes.

- **Real Question:**
  - given some private input, can we discriminate the malicious activity from the legitimate activity?
What is an “Activity”?

- The activity depends on the input and the extension installed!

- From a system perspective, an activity is a set of system interactions between different components.

**Key Intuition**

- **Legitimate** and **malicious** activity can be discriminated in terms of:
  - Components invoked
  - Memory activity of the component in question.
Activities Example - Typing a Keystroke

**Legitimate Activities**

- Normal Activity
  - Typing a keystroke
    - Graphic Library
    - JS Engine
    - HTML Engine

- Shortcut Manager
  - Typing a keystroke
    - Graphic Library
    - JS Engine
    - HTML Engine

**Malicious Activities**

- Keylogger
  - Typing a keystroke
    - Graphic Library
    - JS Engine
    - HTML Engine

- Keylogger leaking to the network
  - Typing a keystroke
    - Graphic Library
    - JS Engine
    - HTML Engine
    - Network Engine
We model the activity of the browser via a set of Memory Performance Counters (MPCs).

Q: How can we identify the browser components accountable for an activity?
   A: by the code regions, i.e. libraries, whose instructions performed the memory write!

Q: How can we measure the component’s contribution to the over-all memory activity?
   A1: we count the bytes written by the code region in question!
   A2: we discriminate memory accesses based on the accessed memory region!

We have a MPC for each combination of code region and memory region.
• Typing a keystroke is reflected in the MPCs of all the code regions involved.

• Effect of typing the keystroke ‘a’ on the search field of Firefox 5.0.1
Good and bad MPCs

- Given some private input, the original problem was how to discriminate between legitimate and malicious activities, i.e.:

- Since we model an activity in terms of MPCs, the (easier and discrete) problem is now:

- Underlying Intuition:
  - a privacy-breaching extensions is bound to store all the private input we provide.
  - a legitimate extension will only partially interact with the private input.

- Vetting an unknown extension means classifying whether its MPCs are malicious or legitimate.
Detection via Classification

- Detection of a malicious extension is modeled as a supervised **binary classification** problem solvable via SVMs.

- The training set is made of both **positive** and **negative** examples:
  - **Positive**: extensions performing malicious activities (keystroke logging, etc...)
  - **Negative**: extensions performing legitimate activities (shortcut managers, etc...)

- Each example is modeled via a set of features:
  - Each feature represents the contribution of a single MPC to the overall behavior.
  - Huge feature space (Firefox ~300 different features, hence ~300 dimensions).

- Cross-browser and cross-version approach:
  - **Cross-browser**: we just need to create a new synthetic training set.
  - **Cross-version**: just retrain the model with new MPCs.
Results on keylogging extensions

- We analyzed a data set of 30 publicly available browser extensions:
  - We selected extensions with keylogging behavior.
  - Of the tested extensions, 100% were successfully detected in just 5 seconds.

- False positives were tested against a data set of the 30 most-used browser extensions:
  - 10 extensions per browser.
  - Of the tested extensions, 94% successfully classified.

- We also tested 9 extensions which behavior would appear similar to a keylogging behavior.
  - 3 extensions per browser, examples are shortcut managers, etc...
  - 100% accuracy.

- The infrastructure imposes a measurable slowdown (8X) limited to the detection time.
- No overhead introduced during the normal use of the browser.
Conclusions

• We presented a cross-browser model to detect privacy breaching extensions.
• We implemented the model on the three most adopted web browsers.
• We evaluated it against real world keystroke logging extensions.
• No false positives against extensions intuitively hindering the detection process.

• Future works:
  • We plan to apply the model to other classes of privacy breaching malware.
  • And to extend the approach to other plugin-based applications:
    • i.e., Word, OpenOffice, Thunderbird, etc...
Thanks for your attention!
Any questions?


Classification Example
Classification Example

Negative Examples
Positive Examples
Classification Example
Classification Example