A Misuse Pattern for the Web Browser: Modification of traffic

Paulina Silva
Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile
pasilva@alumnos.inf.utfsm.cl

Raúl Monge
Departamento de Informática
Universidad Técnica Federico Santa María
Valparaíso, Chile
rmonge@inf.utfsm.cl

Eduardo B. Fernandez
Department of Computer & Electrical Engineering and Computer Science
Florida Atlantic University
Florida, USA
fernande@fau.edu

ABSTRACT

Currently, most software development is focused in creating systems connected to the Internet, which allows to add functionality within a system and facilities to their *Stakeholders*. This leads to depend on a *web client*, such as *Web Browser*, which allows access to services, data or operations that the system delivers. However, the Internet influences the attack surface of the system, and unfortunately many stakeholders and developers are not aware of the risks to which they are exposed. The lack of security education among software developers and the scarce and scattered documentation for browsers (and standardization) could become a big problem in large architectural developments that depend on browsers to perform their services. We are studying some security attacks in the web browser by describing them in the form of misuse patterns. A misuse pattern describes how an information misuse is performed from the point of view of the attacker. It defines the environment where the attack is performed, how the attack is performed, countermeasures to stop it, and how to find forensic information to trace the attack once it happens. We are building a catalog of misuse patterns and we present here one we call Modification of traffic in the Web Browser. A catalog of misuse patterns will help designers to evaluate their designs for possible threats.

Keywords

Browser, Web Client, Misuse Pattern, Security, Man-in-the-Browser

1. INTRODUCTION

The current scenario of attacks in the browser has changed considerably, if compared to those browsers in the 90s. Every day Browsers are more robust and difficult to exploit; therefore, the same attack types, such as drive-by downloads or code-based execution that could subvert a system, are less common every time. A new form of attack has emerged and is fairly easy to achieve, because it is based on deceiving the user to perform what the attacker wants. Once the user is tricked, the attacker can achieve total control over the browser or the host, without having to crack the system [1, 2] that hosts the browser. The development of critical systems that interact daily with different users on the network should focus on these attacks because they threaten the confidentiality, integrity and availability of the user's data (personal) as well as the Stakeholders involved.

The new type of attacks described above are called "social engineering attacks", in [3] they are defined as: The act of manipulating someone to perform actions that are not part of the best interests of the victim (person, organization, stakeholder, etc). An attack of this kind can take many forms, there is the possibility of a physical or digital encounter with the victim. Based on social engineering, this attack is one that takes advantage of human behavior and trust of the victim. In the context of web browser, the deceived user is the first and last line of defense against such attacks, the abuse of trust of the user may open the doors of the browser's host, causing damage to both the user and the external systems with which it interacts.

According to studies [4, 2, 5], the browser is the first line of defense against multiple Web threats. However, this is affected by the lack of education of users who use browsers and the constant evolution of threats [4]. This is why many browser manufacturers have created defense mechanisms such as [6] that act when the user requests a page, using black or white list, reputation systems [1] with warning alerts, among others, so the user can at least avoid the page or choose to enter the malicious site anyway (but no granting access to the page without knowing of the threat).

2. MISUSE PATTERN: MODIFICATION OF TRAFFIC IN THE WEB BROWSER

We present a misuse pattern that describes a threat found in the Browser Infrastructure we have obtained in a previous work. This threat happens when an attacker is able to compromise the response received from the Provider contacted. An attacker could try to replace some parameters from the response received, delivering a different content from the original to the Browser User (BU).

Intent

An attacker could modify or give something different than expected when the web Browser User receives the response

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

from the Server or Provider to the Host; by doing so, the browser could interpret the information in a different way than if it had received the original traffic.

Context

A web browser fetches resources from a Provider to satisfy the access of a Browser User. A Provider has resources in the form of web pages or other web services. The Provider is generally a Web App or Web server, that allows input and output of data to other applications, and usually they are built using HTML, Javascript and CCS. If a server wants to deliver a service, and others want to connect to it, all communication will be done with the HTTP protocol. A Provider, depending on the type, can receive many requests for resources from various host. Depending on the type of request, they may or may not be allowed. For those allowed, the Provider generates a response to the Host, which may come back (or not) to the Browser Client that generated the request.

Problem

How could an attacker fool the Browser User and Host, by modifying traffic between the entities involved in communication? It is possible that an attacker is hidden in the middle of the communication between the Host and the Provider, resulting in modified content that may affect the Web Content Renderer or the Browser Client. This, in turn, could bring different results, like stealing private information of the Browser User that the Browser Client uses to customize the navigation, such as the authentication token of a Providers. Depending on the type of attacker, it is possible that it may even affect the host where the browser is, leaving the attacker the possibility of performing malicious acts with the host resources.

The attack could take advantage of the following vulnerabilities:

- The **origin** that defines the Same Origin Policy (SOP) which the browser complies, differs in every type of web browser [7, 8, 9, 10, 11].
- The **origin** is not enough as an isolation mechanism between the different resources (web pages, scripts, css and others) [12, 13, 14, 15].
- Anyone can create a software component like an extension or plugin for some type of web browser and pass it off as something harmless, consequently a user will not notice the threat and will install it. This could lead to a very known attack named Man-in-the-Browser [16, 17, 15, 18]. Also, an installed malware (if the user with the required privileges was fooled to make the installation) could affect not only the traffic but also the logs of the systems, so it can erase its trace from the system.
- It is possible to affect the Browser Client, and in consequence the Host, without having to find a vulnerability in the system or browser. With social engineering methods it is possible to trick the user, because the Browser User is the weakest link in the system.
- The architecture to extend the browser functionality through extensions, plugins and other, depends on the manufacturer, and probably it has a large attack surface.

The attack can be facilitated by:

- There are many tools for social engineering attacks, that tricks the Browser User into accepting the installation of extensions or malicious plugins more easily.
- Any script can be used to exploit the interpreter of the web browser. Often it is also possible to use the same scripting language elements to pass through certain safety barriers provided by the SOP because the language is based on prototypes (ECMAscript).
- Manufacturers of browsers still do not have many defense mechanisms that allow an effective identification of malicious resources.
- Encryption methods can do nothing against an attack that modifies the traffic before sending or after receiving the message.

Solution

Modifying the traffic in the Web Browser gives attackers the ability to listen, modify and record private information between the Browser User that uses the Host and the Provider that gives the service.

The attacker can safely modify the content of the packets intercepted while acting as a “Man-in-the-Midddle”, but at a different level of abstraction. As the study [16] discusses, this approach would usually be commenced with a phishing attack to trick the user into bridging the gap.

This attack compromises the integrity and confidentiallity even if the browser is communicating with a Provider through a secure channel. This is a consequence of the Browser User letting a social engineering attack do its job, by surfing every website without distrust. A single e-mail wishing for the user to click in a URL address could lead into installing a binary, extension or script in the Host.

After a social engineering attack is successful the attacker can take its time, because whatever is installed in the Host, the attacker has installed it with the user’s permission, so the social engineering attack is done. Therefore every action done in the Host or Browser Client, will be identified by the Host as an action done by the Browser User, a user of the Host.

Structure

The structure of the solution used is the same as in our previous work (Browser Infrastructure Pattern). The Attacker class is any entity that could undertake a risky action against the integrity and confidentiality of the browser, the user, Host and Provider (Figure 1). The attacker is able to intercept both the response sent to the Browser Client from the Provider using the Host as a receiver or by having the browser make changes to the input before the messages goes to the Provider, using scripts or other resources.

Dynamics

In Figure 2 a series of required steps is shown, for one of the many misuses that can be made for the use case **Make Request**. The attacker is located between the Browser Client and the Host, intercepting the original request or response and modifying the traffic to its taste; usually an attack basd

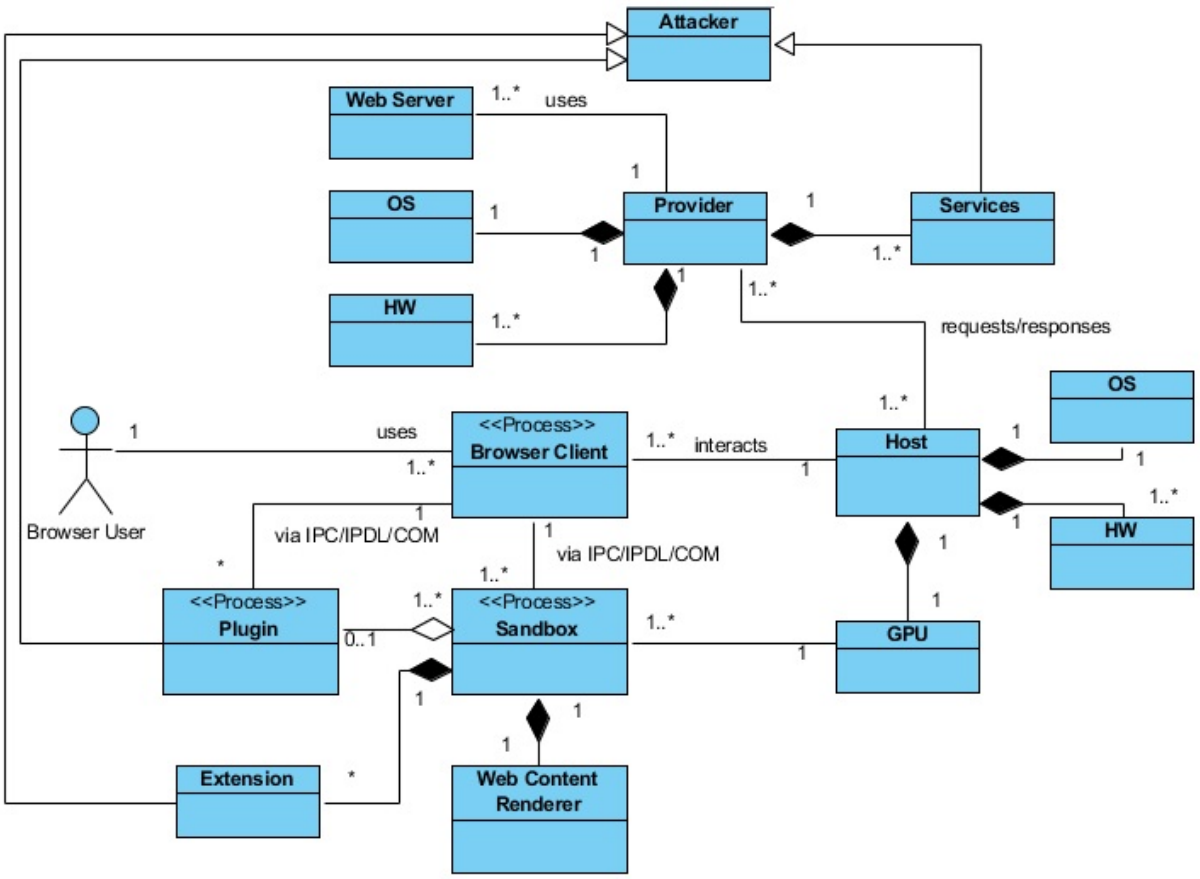

Figure 1: Class Diagram for the Misuse Pattern.

on this misuse is called Man-in-the-Browser (MITB) [15, 18, 17, 16]. This could also happen when the Browser User has allowed the installation of plugins, extensions or external programs in the Host and Browser Client.

Summary

The attacker instercepts the traffic between the Host and Browser Client.

Actor

Attacker

Precondition

For the attack to go unnoticed, the Browser User must have been tricked by a social engineering attack or the attacker must have directly installed a malicious component or process before in the Host.

Description

1. An attacker uses a social engineering technique or vulnerability in the system, to create an entity between the Browser Client and Provider and their communication channel, normally a Plugin, binary or Extension.

2. A Browser User wants to request a resource from a URL, so the first steps are similar to **Make Request**.

3. At the time the Browser Client makes a system call to send the message to the Provider, a plugin, an extension or a program in the Host will intercept the message before the system call is done, as the Browser Client has been tapped to perform that action.

4. The attacker then receives all traffic from the Browser Client, which could be modified or listened.

5. Finally the victim is compromised.

Postconditions

The victim will be fully compromised and it probably will not be possible to detect the modification of a message, it is also possible that the log of the Host will be compromised.

Known Uses

The browser is a software that has different implementations, so the number of attack vectors are significant. Some of these are:

- An extension based on the Google Chrome architecture or the Firefox WebExtension API could intercept the data before it reaches the Browser Client or Host [11]. It could also be possible that a vulnerability in the extension or plugin is used by an attacker and takes advantage of its functionality to attack [15, 18]. Since the plugin, extension or process are elements the Host

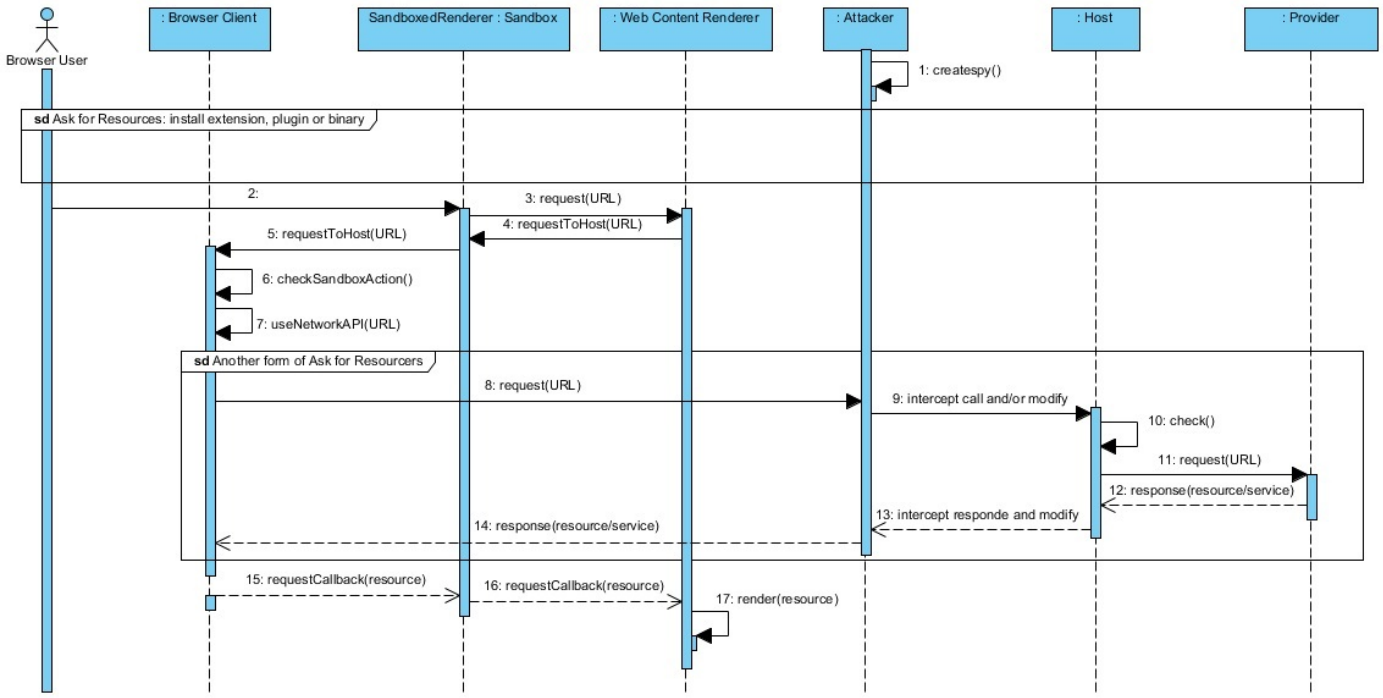

Figure 2: Sequence Diagram for the misuse: Modification of traffic in the Web Browser.

trust, it is possible that the attack is undetectable and the encryption methods do not serve as a mitigation measure.

- This type of attack can be used as base for more advanced attacks. For example, the browser could have *cross-origin javascript capability leak* vulnerabilities, when different security models such as the one used by Javascript and the DOM interfere with each other. As a consequence, a *cross-origin* request can perform even when the SOP was supposed to stop such attack [13].
- If neither an extension or plugin is used, a troyan (a downloaded and executed binary) can be another way to get the Host and all the system compromised.

Consequences

The misuse has the following consequences for the attacker:

- Objectives: they may be different, we highlight vandalism, impersonate another person or monetary gain. While the attacker may be between the host and the traffic that is sent to the Provider, confidentiality and integrity of the data is completely lost. User privacy can no longer be assured.
- Silent: Since the attacker has managed to come between the system calls, that are made to the host, to send data to the Provider, the Host will not recognize or log the anomaly. Calls made to Host are perfectly legal and nothing out of the ordinary, so it will not be seen as something suspicious.
- The attacker could perform actions that affect the integrity of the Host.

Possible sources of error:

- If the Browser User is able to avoid or ignore the social engineering attack carried out at the beginning, this misuse can be subverted. Also, this should consider that the user does not encounter pages with malicious content, which may affect other parts of a browser, but that would cause the same effect as the misuse presented here.

Countermeasures

To prevent this kind of misuse we recommend taking the following preventive measures:

- Reputation services such as SmartScreen [19] from Internet Explorer and Download Application [1] from Google Chrome, can help to identify pages, web content or resources that could contain malware when is installed as plugins, entensiones or process in the Host of the User Browser.
- Providing education about the dangers while surfing the Internet and clarifying the users that they are the last line of defense against such attacks.
- Whitelist and Blacklist are installed in the browser as a preventive measure. They help avoid malicious pages or known malware while the user is browsing, they also are updated in a hourly-basis.
- Browsers like Google Chrome and Internet Explorer offer Sandboxing. This defense mechanism limits the actions of the attacker, which may affect the integrity of the system.

Forense Evidence

Where is it possible to find evidence? Depending on what is desired by the attacker, actions may differ. However the internal log of the browser could help in the audit of the system. This works until an attacker finds a vulnerability in the Sandbox or other component of the browser, in which case he can completely erase its tracks. Also, malware detection such as antivirus systems could help identifying tracks of misuses.

Related Patterns

- The Browser Infrastructure pattern made in a previous work.
- The above pattern, has a class called Browser Client that acts as a Reference Monitor [20].

3. CONCLUSIONS AND FUTURE WORK

We have presented a web browser threat as a form of misuse pattern that systematically describes how one misuse is performed. The aim is to understand and visualize the misuses of the browser that communicates with other systems, mainly to teach developers who have little (or none) security expertise. Through the list of threats done in our previous work, it is possible to detect or infer misuse activities that may appear in one or more misuse cases, which could lead to a violation of the system.

With this misuse pattern we intent to initiated a catalog. This would help to condensate the obtained knowledge using patterns so they can be used as guidelines to communicate relevant concepts, as well as evaluate the existent relationship between the browser and a developed system, to see what kind of interactions they have.

Future work will be related to the creation of a Security Reference Architecture for the *Web Browser* using the same methodology presented here. Other patterns related to Browser Infrastructure pattern will be obtained in order to complete the RA we already begun, such as the Web Content Renderer and Browser Kernel pattern. An example of the type of work to be carried out can be seen in [21]. This study was focused on carry out activities to build secure software and evaluate the safety levels of a system already built.

We plan to build more Misuse Patterns for the Browser Infrastructure Pattern, to continue the study of the possible threats in the *Browser*, as a way to educate Developers and Stakeholders. While at the same time these patterns will allow the construction of the Security Reference Architecture. In the same line, in addition to finding potential threats existing in the system, we need to find countermeasures or security defenses to prevent or foresee such threats through security patterns on the reference architecture built. This is possible to perform under the same exercise already conducted in this work, looking for threats at each action for each use case of the Browser.

4. REFERENCES

[1] M. Rajab, L. Ballard, and N. Lutz, "CAMP: Content-agnostic malware protection," *Proceedings of Annual . . .*, 2013. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.295.6192\&rep=rep1\&type=pdf

[2] N. S. S. Labs and A. R. Abrams, "Evolutions In Browser Security," no. October, pp. 1–20, 2013.

[3] J. Talamantes, "The social engineer's playbook." [Online]. Available: http://www.thesocialengineersplaybook.com/

[4] R. Abrams, J. Pathak, and O. Barrera, "Browser security comparative analysis: Phishing protection," 2013.

[5] ——, "Browser Security Comparative Analysis: Socially Engineered Malware Blocking," 2014.

[6] J. Drake, P. Mehta, C. Miller, S. Moyer, R. Smith, C. Valasek, and A. Q. Approach, "Browser Security Comparison," *Accuvant Labs*, 2011.

[7] W3C, "Same Origin Policy," W3C, Web page, 2010. [Online]. Available: https://www.w3.org/Security/wiki/Same_Origin_Policy

[8] C. Reis and S. D. Gribble, "Isolating web programs in modern browser architectures," *Proceedings of the fourth ACM european conference on Computer systems EuroSys 09*, vol. 25, no. 1, p. 219, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1519065.1519090

[9] C. Jackson and A. Barth, "Beware of finer-grained origins," *Web 2.0 Security and Privacy*, 2008. [Online]. Available: http://seclab.stanford.edu/websec/origins/fgo.pdf

[10] M. Crowley, *Pro Internet Explorer 8 & 9 Development: Developing Powerful Applications for The Next Generation of IE*, 1st ed. Berkely, CA, USA: Apress, 2010.

[11] S. D. Paola and G. Fedon, "Subverting Ajax," *23rd Chaos Communication Congress*, no. December, 2006. [Online]. Available: http://events.ccc.de/congress/2006/Fahrplan/attachments/1158-Subverting_Ajax.pdf

[12] M. Silic, J. Krolo, and G. Delac, "Security vulnerabilities in modern web browser architecture," *MIPRO, 2010 Proceedings of the 33rd International Convention*, 2010.

[13] A. Barth, J. Weinberger, and D. Song, "Cross-Origin JavaScript Capability Leaks : Detection , Exploitation , and Defense," *Opera*, vol. 147, pp. 187–198, 2009.

[14] M. V. Yason, "Diving into IE 10's Enhanced Protected Mode Sandbox."

[15] L. Liu, X. Zhang, G. Yan, and S. Chen, "Chrome extensions: Threat analysis and countermeasures," *. . . of the Network and Distributed Systems . . .*, 2012. [Online]. Available: https://www.cs.gmu.edu/~sqchen/publications/NDSS-2012.pdf

[16] T. Dougan and K. Curran, "Man in the Browser Attacks," *International Journal of Ambient Computing and Intelligence*, vol. 4, no. 1, pp. 29–39, 2012.

[17] N. Utakrit, "Review of Browser Extensions, a Man-in-the-Browser Phishing Techniques Targeting Bank Customers," *Proceedings of the 7th Australian Information Security Management Conference*, pp. 110–119, 2009. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-84864552184\&partnerID=40\&md5=3d08a9c7c4ba9dbe5e04fb831ad5257b$\backslash$nhttp://ro.ecu.edu.au/ism/19/

[18] A. Barth, A. P. Felt, P. Saxena, and A. Boodman, "Protecting Browsers from Extension Vulnerabilities," *Ndss*, vol. 147, pp. 1315–1329, 2010. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.154.5579\&rep=rep1\&type=pdf

[19] R. Colvin, "SmartScreen," 2010. [Online]. Available: http://blogs.msdn.com/b/ie/archive/2010/10/13/stranger-danger-introducing-smartscreen-application-reputation.aspx

[20] E. B. Fernandez and R. Pan, "A pattern language for security models," *proceedings of PLOP*, vol. 1, pp. 1–13, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.90.5898

[21] E. B. Fernandez, R. Monge, and K. Hashizume, "Building a security reference architecture for cloud systems," *Requirements Engineering*, jan 2015. [Online]. Available: http://link.springer.com/10.1007/s00766-014-0218-7