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 of software development creates systems that are 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 the *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 of software developers, the scarce and scattered documentation for browsers (and standardization), could become a big problem in large architectural developments which depend on browser 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 called: Modification of traffic in the Web Browser. A catalog of misuse patterns will help designers to evaluate their designs with respect to possible threats.

Keywords

Web Browser, Web Client, Modular Architecture, Browser Architecture, Reference Architecture, Browser Infrastructure Pattern

Introduction

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

The new type of attacks described above are called "social engineering attacks", in [**?**] they are defined as: The act of manipulating a person 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 victim to whom deception is done. Based on social engineering 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 to host which host the browser, achieving damage to both the user and the external systems with which interacts.

According to studies [3, 2, 4], they affirm that th 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 [3]. This is why many browser manufacturers have created defense mechanisms such as [5] that act when the user request for a page, using black or white list, reputation systems [1] with warning alerts, or other, so the user can at least avoid the page or make him to choose to enter the malicious site (but no making him enter the page without knowing).

In the Top Ten from OWASP [6] (Open Web Application Security Project) - the ten most important security risks in Web applications - it can be distinguished in 2013 the risks directly related to security threats in the browser, such as: Injection (A1), Session management and Broken authentication (A2), XSS (A3), CSRF (A8) and Using components with known vulnerabilities (A9), are the risks that organizations may suffer in their systems when some attacks are made on the browser.

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

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

In this section 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 contacted Provider. An attacker could try to replace some parameters from the response recived, delivering to the Browser User (BU) a different content from the original.

1.1 Intent

An attacker could modify or give something different than the expected when the web browser user (Browser User) receives the response from the Server or Provider. Performing this action, the browser could interpret the information in a different way than if he had received the original traffic.

1.2 Context

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

1.3 Problem

How could an attacker to 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 the modified content that may affect the Web Content Renderer or the Browser Client. This in turn could bring different results, from 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 may 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 mechanisms between the different resources (web pages, scripts, css and others) [12, 13, 14, 15].
- Anyone can create an extension or plugin, etc. to some type of web browser and pass it off as something harmless, consequently a user will not notice of the threat and will install it. This could lead a very known attack named as Man-in-the-Browser [16, 17, 15, 18]. Also, a installed malware (as the user with the required privileges was fooled to make the installation) could affect not only to the traffic but also to 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 to make into the system or the browser. With social engineering methods it is possible 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 manufacturer. And probably it has a large attack surface.

The attack can be facilitated by:

- There are many tools for social engineering attacks, allowing the attack to make the Browser User accepts 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 effectively identify when a resource may be malicious.
- Encryption methods can do nothing against an attack that modifies the traffic before sending or after receiving the message.

1.4 Solution - 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 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.

1.5 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 between the Browser Client and Host, intercepting the original request or response and modifying the traffic to its taste; usually an attack basd on this misuse is called Man-in-the-Browser (MITB) [15, 18, 17, 16]. This as well could happen when the Browser User has allowed the installation of plugins, extensions or external programs in the Host and Browser Client.

1.5.1 Summary

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

1.5.2 Actor

Attacker

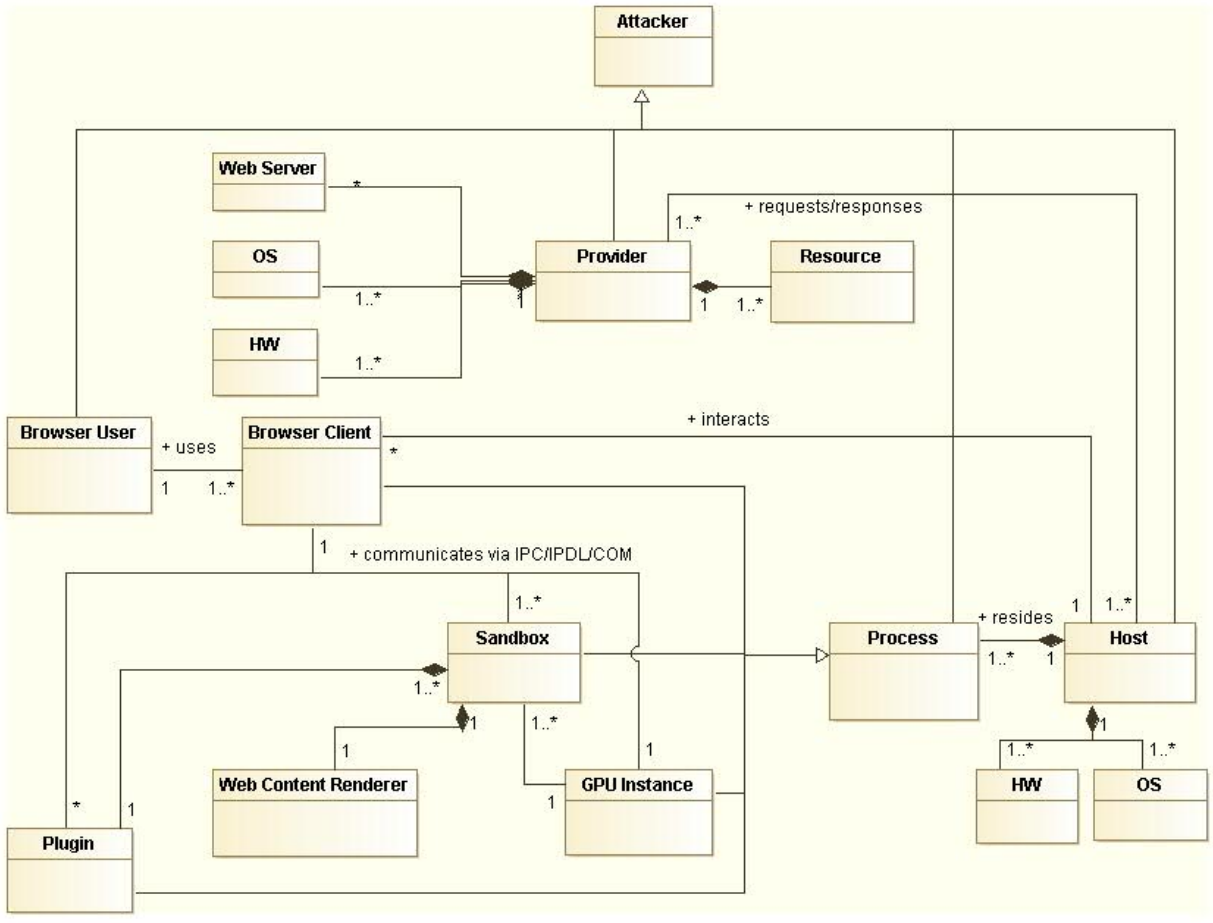

Figure 1: Diagrama de Clases para el patrón de Misuse.

1.5.3 Precondition

For the attack to go unnoticed, it is necessary that the Browser User had been tricked by a social engineering attack or the attacker may have directly installed a malicious component or process before in the Host.

1.5.4 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.

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.

1.5.5 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.

1.5.6 Known Uses

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

- Una extensión basada en la arquitectura de Chrome o la API WebExtension Firefox, podría interceptar los datos antes que llegue al Browser Client [11] o dado a una vulnerabilidad del mismo elemento un atacante se está aprovechando de su funcionalidad para realizar ataques [15, 18]. Dado que el Plugin, la extensión o el process son elementos que el Host confía, es posible que el ataque sea indetectable y los métodos de cifrado no sirven como medida de mitigación.

- Éste tipo de ataque puede ser usado como base para otros ataques más avanzados. Un ejemplo es cuando el *Browser* posee vulnerabilidades *cross-origin javascript capability leaks*, donde los diferentes modelos de seguridad usados por Javascript y el DOM pueden interferir entre sí, causando que una petición *cross-origin*

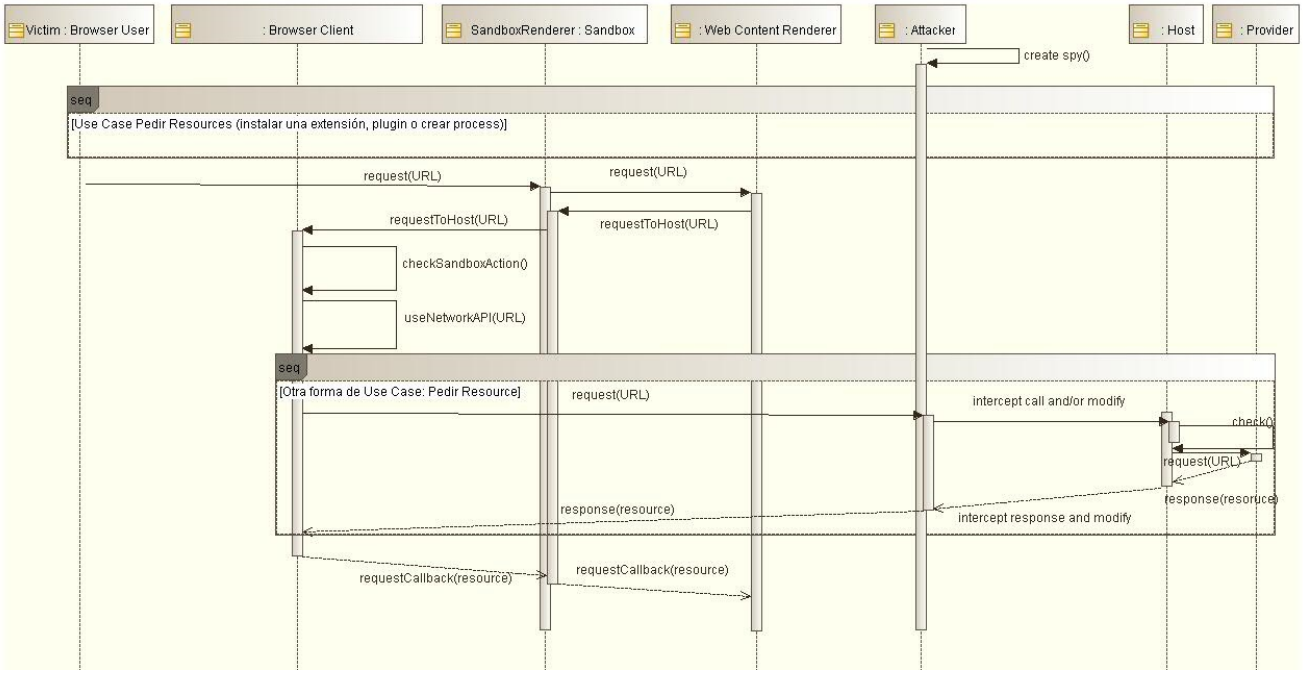

Figure 2: Diagrama de Secuencia para el Mal uso: Modificación de tráfico en el *Web Browser*.

se pueda realizar aún cuando SOP debería ser capaz de detener tal ataque [13].

1.6 Consecuencias

El mal uso tiene las siguientes consecuencias para el Attacker:

- Objetivos: pueden ser diversos, destacándose el vandalismo, personificar a otra persona u obtener una ganancia monetaria. Mientras el atacante se pueda interponer entre el Host y el tráfico que se envía al Provider, la confidencialidad e integridad de los datos está completamente perdida. La privacidad del usuario ya no se puede asegurar tampoco.
- Silencioso: Dado que el atacante ha logrado interponerse entre las llamadas de sistema que se realizan al Host para enviar los datos al Provider, el Host no reconocerá ni logueará ninguna anomalía. Las llamadas hechas al Host son totalmente legales y nada fuera de lo normal para éste.
- El atacante podría realizar acciones que afecten la integridad del Host.

Posibles fuentes de fallo:

- Si el Browser User es capaz de evitar o ignorar el ataque de Ingeniería Social realizado al comienzo, no existiría este mal uso. Esto debe considerar que el usuario también no se encuentre con páginas o contenido malicioso, que podrían afectar otro componente de *Browser*, pero que causarían en el mismo efecto del Mal Uso señalado.

1.7 Contramedidas

Para prevenir este tipo de mal uso se recomienda tomar las siguientes medidas preventivas:

- Servicios de Reputación como Smart Screen de Internet Explorer y Download Application de Google Chrome, ayudan a identificar páginas y contenido/resources que podrían contener malware que se instale como plugins, entensiones o process en el Host del Browser User.
- Entregando educación sobre los peligros de navegar por Internet y aclarar al usuario que él es la última línea de defensa contra éste tipo de ataques.
- White y Black list instaladas en los browser son una medida preventiva para evitar la navegación en páginas o recursos maliciosos ya conocidos.
- Navegadores como Google Chrome e Internet Explorer ofrecen el Sandboxing. Éste mecanismo de defensa limita las acciones del atacante, que pudieran afectar la integridad del sistema.

1.8 Evidencia Forense

Where it is 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.

1.9 Patrones relacionados

- En el patrón Browser Infrastructure, el Browser Client actúa como el Reference Monitor explicado en [19].

Conclusions and future work

We have presented a web browser threats as a form of misuse patterns which describes in a systematic way how one

misuses is performed. The aim is to understand and visualize the misuses of the system, in this case the browser, to teach developers which use the web browser as a client to the systems they are developing. 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 to 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 AR already begun, such as the Web Content Renderer pattern and Browser Kernel. An example of the type of work to be carried out can be seen in [20] where this study carries 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.

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

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

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

[6] "Top 10 2013 - OWASP." [Online]. Available: https://www.owasp.org/index.php/Top_10_2013

[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$\backslash$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] 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

[20] E. B. Fernandez, R. Monge, and K. Hashizume, "Building a security reference architecture for cloud systems," in *Proceedings of the WICSA 2014 Companion Volume*. ACM, 2014, p. 3.