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
ed@cse.fau.edu

ABSTRACT

Currently a lot of software developments create systems that are connected to the Internet, which allows to add functionality within a system and facilities to their *Stakeholders*. This leads to depend in a *web client*, as the *Web Browser*, which allows access to services, data or operations that the system delivers. Nevertheless, the Internet influences the attack surface of the new system, and unfortunately many stakeholders and developers are not aware of the risks they are exposed. The lack of Security Education in Software developers of a project, the low and scattered documentation of each browser (and standardization), could become a great flaw in big architectural developments which depends on the browser to do 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

1. PATRÓN DE MAL USO: MODIFICACIÓN DE TRÁFICO EN EL *WEB BROWSER*

En esta sección presentaremos un patrones de mal uso que describe la amenaza encontrada en el patrón Browser Infrastructure que hemos obtenido en este trabajo. Una de las amenazas es que un atacante haya sido capaz de comprometer la respuesta obtenida desde el Provider contactado. El atacante podría tratar de reemplazar algunos parámetros de la respuesta del provider, entregando al Browser User un contenido distinto al original.

1.1 Intent

Un atacante podría cambiar contenido o dar uno diferente al esperado cuando el *Web Browser* (Browser User) recibe la response del Servidor o Provider. Realizando esta acción, el navegador podría interpretar la información de una manera distinta a que si hubiera recibido el tráfico original.

1.2 Contexto

Un navegador web obtiene resources desde un Provider para poder satisfacer un Browser User que lo necesita. Un Provider posee resources en forma de páginas web, servicios web u otro tipo. El Provider generalmente consiste en una Web App o Web server, que puede permitir entrada y salida de datos a otras aplicaciones, normalmente éstos están construidos usando HTML, Javascript y CCS. Sea que éste desea entregar servicios, como también otros deseen conectarse a él, todo tipo de comunicación estará encima del protocolo HTTP. Un Provider, dependiendo del tipo, puede llegar a recibir muchas peticiones de diversos Host para obtener recursos de éste. Dependiendo del tipo de peticiones, éstas pueden o no ser permitidas. Aquellas que son permitidas, el Provider generará una response a la *request* del Host, la cuál terminará llegando de vuelta (o no) al Browser Client que generó el *request*.

1.3 Problema

¿Cómo podría un atacante engañar al Browser User y Host, por medio de la modificación de tráfico entre las entidades participantes en la comunicación? Es posible que un atacante esté escondido entre medio de la comunicación que hay entre el Host y el Provider, lo que resulta en que el contenido modificado podría afectar dentro del Web Content Renderer o el Browser Client. Esto en consecuencia podría traer diferentes resultados, desde el robo de información privada del Browser User que utiliza el Browser Client para personalizar la navegación como también los token de autenticación hacia los Providers que el Browser User ha utilizado previamente. Dependiendo del tipo de atacante, es posible que éste pueda incluso afectar al Host del *Browser*,

WOODSTOCK '97 El Paso, Texas USA

DOI: 10.475/123_4

dejando que el atacante pueda realizar actos maliciosos con los recursos del Host. El ataque puede sacar ventaja de las siguientes vulnerabilidades:

- El **origen** u **origin** que define el Same Origin Policy que hace cumplir el *Browser* difiere en cada tipo de *Web Browser* [10, 11, 12, 13, 14].
- El **origin** no es suficiente como método de aislación entre resources (páginas web, scripts, css y otros) [15, 16, 17, 18].
- Cualquiera puede crear una extensión o plugin, etc. para algún tipo de *Browser* y hacerlo pasar como algo inofensivo, pero el usuario no se daría cuenta que un ataque Man-in-the-Browser podría estar ocurriendo [19, 20, 18, 21]. Éste programa, posiblemente malicioso, podría afectar el tráfico sin que queden rastros, pues tiene permisos del usuario al ser éste mismo quien autorizó su instalación.
- Es posible afectar al Browser Client, y en consecuencia al Host, sin tener que buscar vulnerabilidades en el *Web Browser*. Solamente utilizando métodos de ingeniería social, sobre el usuario del Host basta para lograr un ataque, pues es **el eslabón más débil**.
- La arquitectura para extender la funcionalidad del navegador, a través de extensiones, plugins y otros, depende demasiado del manufacturador. Y probablemente posee una gran superficie de ataque.

El ataque puede ser facilitado por:

- Existen muchas herramientas para realizar ataques de ingeniería social, lo que permite hacer que el Browser User acepte realizar la instalación de extensiones o plugin maliciosos más fácilmente.
- Cualquier script basado en texto puede ser usado para explotar el intérprete del navegador web. Muchas veces también es posible utilizar los mismos elementos del lenguaje de scripting para poder pasar ciertas barreras de seguridad entregadas por el SOP, pues el lenguaje está basado en prototipos.
- Los manufacturadores de Navegadores no poseen aún mecanismos de defensa que permitan identificar efectivamente cuando un recurso puede ser malicioso.
- Los métodos de cifrado no pueden hacer nada contra un ataque que puede modificar el tráfico antes de enviar o después de recibir el mensaje.
- No existen mecanismos de defensa estandarizados, pues cada manufacturador realiza lo necesario para la marca a la que se acopla. Si el Host posee muchos browser, la superficie de ataque es bastante mayor.

1.4 Solución - Estructura

La solución estructural usada es la misma creada por el patrón Browser Infrastructure que se revisó en el capítulo 4. La clase Attacker es cualquier entidad que podría llevar a cabo una acción arriesgada contra la integridad del *Browser*, usuario, Host y Provider (Figura 1). El atacante es capaz tanto de interceptar las response que el Provider envía al Browser Client usando al Host como receptor, o haciendo que el *Browser* realice modificaciones en el input que va hacia el Provider, utilizando scripts u otro resources.

1.5 Dinámica

En la figura 2 se muestra la serie de pasos necesarios, para realizar uno de los tantos mal usos que se pueden realizar durante el caso de uso Realizar *request*. El atacante queda entre el Browser Client y el Host, interceptando la realización del *request* original y modificando el tráfico a su gusto; usualmente un ataque basado en este mal uso se le llama Man-in-the-Browser (MITB) [18, 21, 20, 19]. Esto podría perfectamente suceder cuando el Browser User ha permitido la instalación de plugins, extensiones o programas externos en el Host y Browser Client.

1.5.1 Sumario

El atacante intercepta el tráfico entre Host y Browser Client.

1.5.2 Actor

Atacante

1.5.3 Precondiciones

Para que el ataque pueda pasar desapercibido, es necesario que el Browser User o el usuario detrás del browser haya caído primero en un ataque de ingeniería social o el atacante haya podido instalar directamente un proceso o componente malicioso en el Host directamente.

1.5.4 Descripción

1. Un atacante utiliza alguna técnica de ingeniería social o vulnerabilidad en el sistema, para crear una entidad que se encargará de estar entre medio del Browser Client y el Provider. Para esto realiza el caso de uso **Pedir Resources** para que el proceso, plugin o extensión se aloje en el Host.
2. Un Browser User desea hacer un *request* a cierta URL, por lo que los primeros pasos de **Realizar Request** son similares.
3. En el momento en que el Browser Client va a realizar la llamada al sistema para enviar el mensaje del Host al Provider, el Browser Client llamará al plugin, extensión o process malicioso, pues el Browser Client ha sido intervenido para que realice esa acción de este modo.
4. El atacante entonces recibirá todo el tráfico del Browser Client, el cuál podría modificar a su antojo.
5. Finalmente la víctima está totalmente comprometida.

1.5.5 Post condiciones

La víctima quedará completamente comprometida y probablemente no sea posible detectar la alteración del mensaje, pues también es posible que se modifiquen los log del Host.

1.5.6 Usos Conocidos

El *Web Browser* es un software que posee diversas implementaciones, por lo tanto la cantidad de vectores de ataque es significativa. Algunos de éstos son:

- Una extensión basada en la arquitectura de Chrome o la API WebExtension Firefox, podría interceptar los datos antes que llegue al Browser Client [14] o dado a una vulnerabilidad del mismo elemento un atacante se está aprovechando de su funcionalidad para realizar

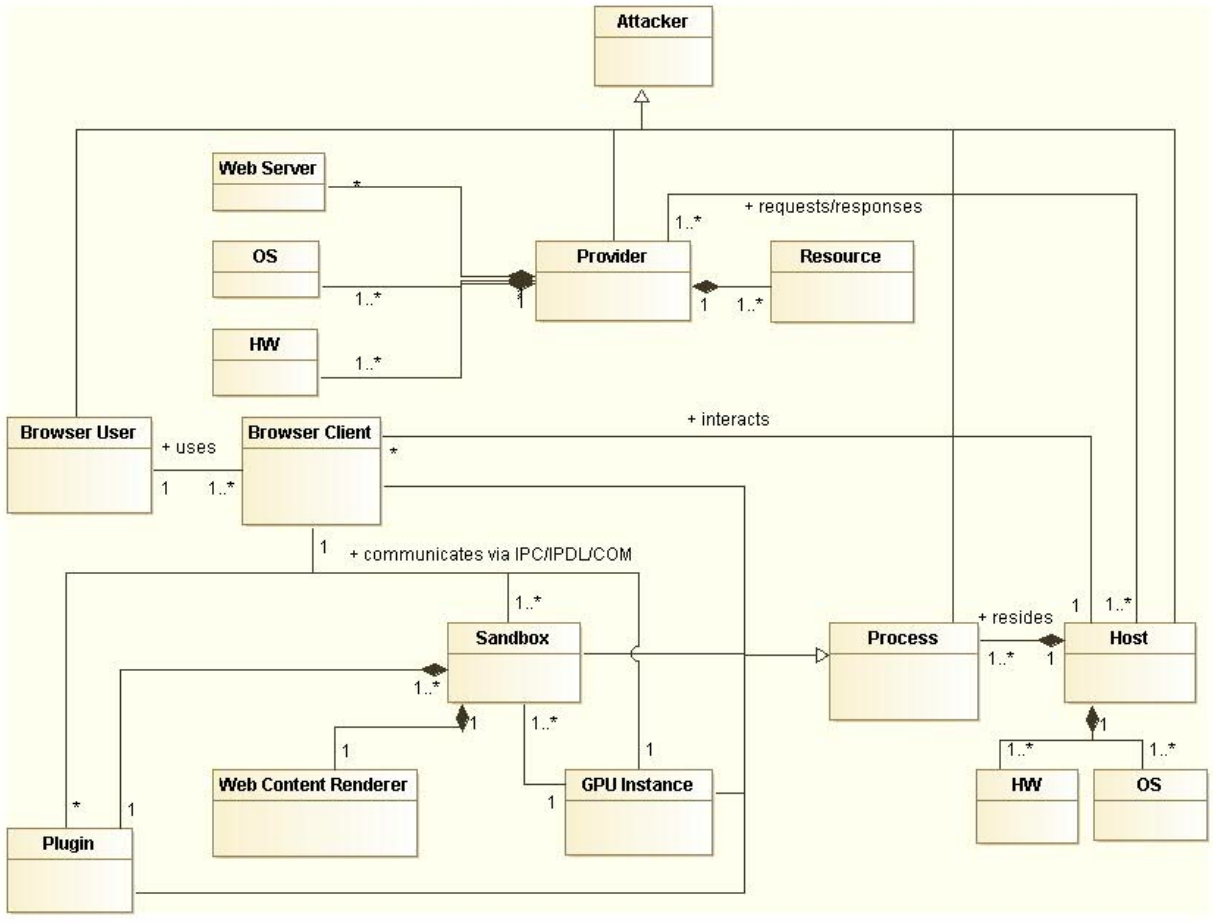

Figure 1: Diagrama de Clases para el patrón de Misuse.

ataques [18, 21]. 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* se pueda realizar aún cuando SOP debería ser capaz de detener tal ataque [16].

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.

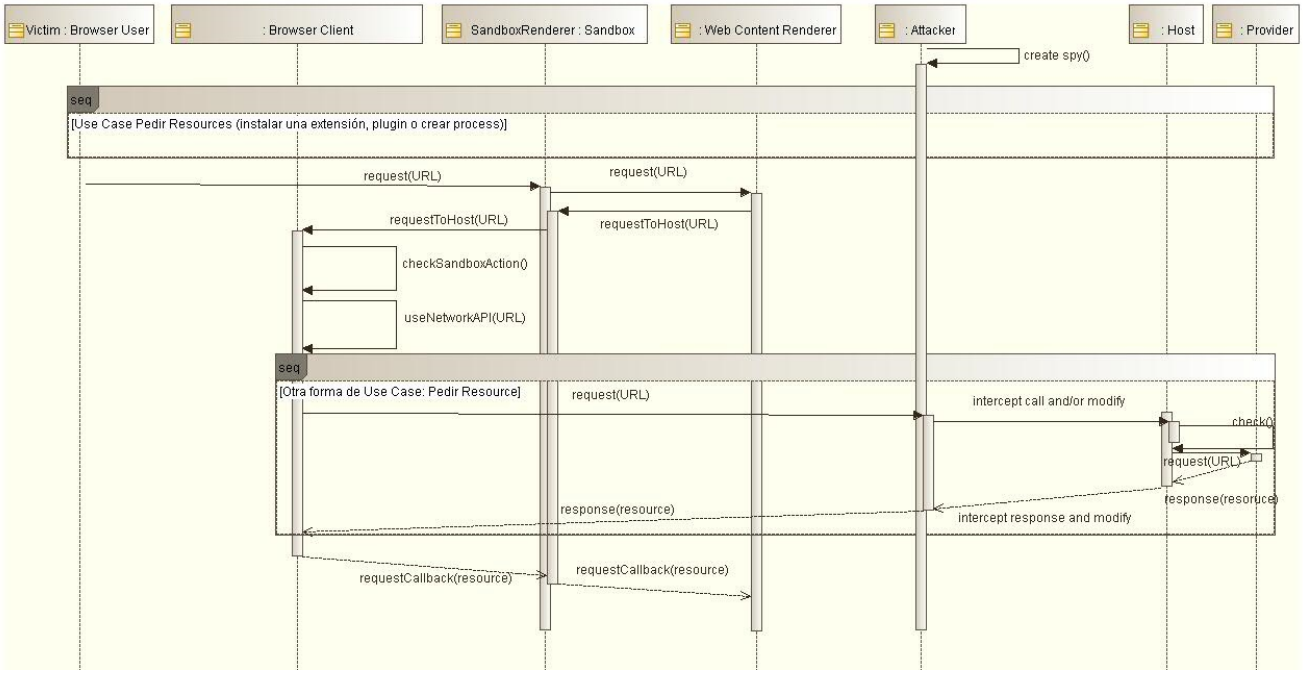

Figure 2: Diagrama de Secuencia para el Mal uso: Modificación de tráfico en el *Web Browser*.

- 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

¿Dónde es posible encontrar evidencia? Dependiendo de lo deseado por el atacante, las acciones que cometerá pueden diferir. Sin embargo el log interno del browser debería de poder servir para auditar el sistema, esto gracias a que mientras no se haya encontrado una vulnerabilidad en el Sandbox del *Browser*, el atacante no puede borrar completamente sus huellas.

1.9 Patrones relacionados

- En el patrón Browser Infrastructure, el Browser Client actúa como el Reference Monitor explicado en [22].

Conclusions

We have presented some web browser threats as a form of misuse patterns which describe in a systematic way how one misuses is performed. El objetivo de éste capítulo es conocer y visualizar los mal usos del sistema, en este caso del Browser, para poder educar a los desarrolladores de proyectos que basan sus sistemas en el uso de éste. A través del listado de amenazas es posible detectar o inferir actividades de mal uso que pueden aparecer en uno o más casos de uso, que podrían resultar en una vulneración del sistema.

iniciar un pequeño catálogo de Patrones de Mal Uso. Esto permitirá condensar el conocimiento obtenido en el punto anterior a través de documentos semi-formales, lo que permitirá generar una guía para comunicar los conceptos relevantes que pudieran afectar la relación existente entre un desarrollo de software y el navegador.

Un catálogo de Misuse Patterns podría ser de gran valor en el Desarrollo de Sistemas que interactúan con el navegador, pues provee a desarrolladores un medio para evaluar los diseños de sus sistemas, al analizar las posibles amenazas del Browser que pudieran afectar al software que está siendo construido.

Future Work

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 [23] 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.

As for *Web Browsers*, attacks based on social engineering does not seem to decrease at any good time, because there is no current technology that can detect a 100% without no false positives the potential threats they can bring. Technologies such as CAMP (Content-Agnostic Malware Protection) appear to be part of the solution, but are still far from perfect.

2. REFERENCES

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[2] E. Fernandez, M. Larrondo-Petrie, T. Sorgente, and M. VanHilst, "A methodology to develop secure systems using patterns," *Chapter*, vol. 5, pp. 107–126, 2006.

[3] E. B. Fernandez, N. Yoshioka, H. Washizaki, J. Jurjens, M. VanHilst, and G. Pernu, *Using Security Patterns to Develop Secure Systems*, H. Mouratidis, Ed. IGI Global, 2011. [Online]. Available: http://www.igi-global.com/chapter/using-security-patterns-develop-secure/48405

[4] A. Grosskurth and M. W. Godfrey, "A reference architecture for web browsers," 2005, pp. 661–664, uRL: `http://grosskurth.ca/papers.html#browser-refarch`.

[5] M. Larrondo-Petrie, K. Nair, and G. Raghavan, "A domain analysis of web browser architectures, languages and features," in *Southcon/96. Conference Record*, Jun 1996, pp. 168–174.

[6] A. Grosskurth and M. W. Godfrey, "Architecture and evolution of the modern web browser," uRL: `http://grosskurth.ca/papers.html#browser-archevol`. Note: submitted for publication.

[7] M. W. Godfrey and E. H. S. Lee, "Secrets from the Monster: Extracting Mozilla's Software Architecture," *In Proc. of 2000 Intl. Symposium on Constructing software engineering tools (CoSET 2000*, pp. 15–23, 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.9211

[8] A. Systems and N. Lwin, "Agent Based Web Browser," *2009 Fifth International Conference on Autonomic and Autonomous Systems*, 2009.

[9] E. Fernandez-Buglioni, *Security patterns in practice: designing secure architectures using software patterns*. John Wiley & Sons, 2013.

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

[11] 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

[12] 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

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

[14] 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

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

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

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

[18] 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

[19] 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.

[20] 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/

[21] 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

[22] 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

[23] 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.