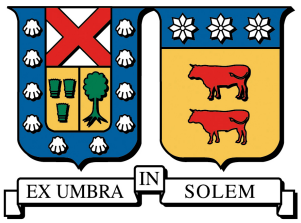
Universidad Técnica Federico Santa María
Departamento de Informática
Valparaíso, Chile



????

Paulina Andrea Silva Ghio

Memoria para obtar al título de: Ingeniera Civil Informática

Profesor Guía: Raúl Monges
Profesor Correferente: Javier Cañas

13 de abril de 2015

Agradecimientos

Resumen

El Web Browser es una de las aplicaciones más usadas - *killer app* - y también una de las primeras en aparecer en cuanto se creó el Internet (Década de los 90). Por lo mismo, su nivel de madurez con respecto a otros desarrollos es significativo y permite asegurar ciertos niveles de confianza cuando otros usan un Web Browser como cliente para sus Sistemas.

En la actualidad, muchos Desarrollos de Software usan ésta técnología dado que grandes compañías como Google, Microsoft, Mozilla, entre la más importantes, dedican la mayoría de su tiempo en asegurar la calidad de sus Navegadores Web, lo que se traduce en un gasto menor de esfuerzo en el Desarrollo. Sin embargo, dado que la misma Internet es considerada como una red no confiable, estos sistemas que usan al navegador como Cliente Web, introducen nuevas amenazas de seguridad que deben ser tomadas en cuenta en el Desarrollo del sistema.

Esta Memoria incursionará en el ámbito de la seguridad del Web Browser, con el objetivo de obtener documentos formales que servirán como herramientas a personas que Desarrollen Software y hagan un fuerte uso del Navegador para las actividades del sistema desarrollado.

Abstract

Índice general

Capítulo 1

Introducción

Mucho antes que aparecieran los grandes sistemas que ahora conocemos, el Web Browser era la herramienta usada por todos para navegar, las ya viejas páginas estáticas, y que permitía un sin número de acciones a aquellos que necesitaban conectarse al Internet. En la actualidad el browser sigue siendo la herramienta predilecta por todos, desde comprar tickets para una película, realizar reuniones por videoconferencia y muchas otras tareas que invitan a nuevas formas de interactuar y comunicar.

La Web 2.0, que se inició con el uso intensivo de tecnologías como AJAX, ha permitido una nueva simbiosis entre el usuario y el Web Browser. Haciendo de éste último, una herramienta casi indispensable para todo tipo de tareas computacionales, tal que su existencia a penetrado completamente en las labores diarias de todos nosotros. En este mismo instante, la Internet a evolucionado nuevamente obteniendo un nuevo nombre, Web 3.0, donde se realiza el uso de inteligencia artificial y sistemas de recomendación para generar nuevo tipo de contenido media para el usuario.

En esta memoria se quiere presentar al Departamento de Informática (DI) de la UTFSM[1] Casa Central, sobre la investigación que contempla un estudio completo del Navegador Web.

1.1. Web Browser

Entre 1989 y 1990, Tim Berners-Lee acuño el concepto de World Wide Web y con ésto realizó la construcción del primer Web Server, Web Browser y las primeras páginas Web. El fin de su construcción era poder

1.2. Motivación: ¿Por que estudiar el Browser?

En el pasado era bien visto que cada Desarrollo de Software, pudiera levantar todo su sistema sin tener que depender de externos para funcionar. Este pensamiento, sin

[1]Universidad Técnica Federico Santa María

embargo, ha cambiado mucho en el último tiempo tanto debido a factores sociales, económicos, técnicos y otros.

Actualmente las nuevas formas para Desarrollar Software, tanto procesos como métodos, tratan de ajustarse lo más posible a lo que su Cliente le pida, para poder cumplir con las metas del Proyecto a Desarrollar en el tiempo estipulado. Muchas veces esto conlleva a reducir gastos que terminan por afectar: la calidad del Software y en consecuencia la Seguridad tanto del sistema, como la del cliente que lo utiliza.

Con la aparición de la Web 2.0 y 3.0, con el uso de AJAX, inteligencia artificial y sistemas de recomendación, permitieron nuevas formas de interacción entre usuarios y sistemas, lo que causó que el browser fuera usado extensivamente en los nuevos Desarrollos de Software dado que:

- Permite disminuir los costos de construir un programa Cliente para el usuario del sistema.
- Actualmente la Seguridad implementada en los Web Browser es bastante buena, dado que existen en la actualidad grandes compañias se aseguran de eso. Esto permite que usar el Browser sea una opción bastante económica para el desarrollo de otros sistemas (y concentrarse solo en eso).
- El browser es una herramienta indispensable. La mayoría de los sistemas que lo usan en la vida cotidiana son de tipo: online bank, declaración de impuestos, compras, y muchos más.

Sin embargo los sistemas que dependen del uso del Browser, deben de tener en cuenta de las posibles anemazas de seguridad a las que se enfrentan por el solo hecho de usar el Navegador. Para un proyecto de gran envergadura sería un error no tener en consideración los posibles peligros que trae el uso del Browser, y es el deber de todo integrante del equipo de Desarrollo del sistema tener conocimientos sobre la seguridad de éste.

En [1] hace referencia a un hecho, que personalmente la autora de esta memoria considera es una realidad incluso en la actualidad de Chile, esta es la falta de cursos o educación en temas de Seguridad a los estudiantes de Ingeniería de Software. De tal manera que éstos estudiantes no se les enseñan Principios de Diseño en Seguridad ni técnicas que permitan una segura implementación de código, de manera que produzcan un código que pueda aguantar ciertos tipos de ataques. Más aún la falta de este conocimiento, hace que los estudiantes no piensen de la seguridad como una **Propiedad Sistémica**, si no más como un requerimiento que puede o no ser tomado en cuenta al comienzo del Desarrollo.

Este trabajo desea ayudar, a quién lo necesite, con el conocimiento necesario para entender el funcionamiento y construcción del Cliente - el Web Browser-, los beneficios detrás de la Seguridad implementada en el Browser y de los peligros existentes de los que nos protegen. De esta manera se espera que alguien que lea este trabajo, tanto Estudiantes como Desarrolladores de Softwares, obtengan el conocimiento necesario al momento de trabajar junto con el Navegador Web.

1.3. Contribuciones

El Objetivo General de esta Memoria es generar un cuerpo organizado de información sobre la Seguridad en el Web Browser, de tal manera que se pueda sistematizar, organizar y clasificar el conocimiento adquirido en un documento con formato fácil de entender, tanto para Profesionales como Estudiantes del área Informática.

Particularmente, este trabajo busca cumplir con los siguientes Objetivos Específicos:

- Comprender los conceptos relacionados al navegador web, sus componentes, interacciones, seguridad y ataques a los que puede estar sometido, mecanismos de defensa y otros, a través del Estado del Arte a construir.
- Construir una Arquitectura de Referencia del Web Browser y Patrones de Mal Uso que representan ciertas amenazas a las que está sometido el Navegador. Esto permitirá condensar el conocimiento obtenido en el punto anterior a través de documentos formales.
- Clasificar los ataques y mecanismos de defensa (mitigación) de los navegadores Web.
- Profundizar el conocimiento en ataques relacionados con métodos de Ingeniería Social.

1.4. Estructura del Documento

El presente documento trata del trabajo de Memoria que se divide en las siguientes partes:

- En el capítulo 2 se presentará la información necesaria para poder construir un completo Estado del Arte sobre el **Browser**, de tal manera de entender su funcionamientos, componentes e interacciones que realiza con otras entidades.
- Luego de tener un extenso conocimiento de lo que actualmente es conocido como **Web Browser**, el capítulo 3

1.5. Definiciones Básicas

Para empezar este estudio es necesario introducir ciertas nociones y lenguaje que se usarán durante todo el documento. Estos conceptos son ampliamente usados en la Seguridad y Desarrollo de Software, por lo que son extendibles para lo que se verá en este estudio en **Web Browser Security**.

- Seguridad - *Security*:
  Es una Propiedad que podría tener un sistema, donde asegura la protección de los recursos en contra de ataques maliciosos desde fuentes externas como internas. La Seguridad también involucra controlar que el funcionamiento de un sistema sea como debería ser, y que nada externo o interno genere un error.

- Error - *Error*:
  Es una acción de caracter humano. Éste se genera cuando se tienen ciertas nociones equivocadas, que causan un Defecto en el Sistema o Código.

- Defecto - *Defect*:
  Es una caracterítica que se obtiene a nivel de Diseño, cuando una funcionalidad no hace lo que tiene que realmente hacer. Según la IEEE CSD o Center for Secure Design un defecto puede ser subdividido en 2 partes: **flaw** y **bug**, donde la primera tiene que ver con un error de **alto nivel**, mientras que un bug es un problema de implementación en el Software. Una falla es más notoria que un bug, dado que ésta es de caracter abstracto, a nivel de diseño del Software [2].

- Falla - *Fail*:
  Es un estado en que el Software Implementado no funciona como debería de ser.

- Vulnerabilidades - *Vulnerability*:
  Es una debilidad inherente del sistema que permite a un atacante poder reducir el nivel de confianza de la información de un sistema. Una vulnerabilidad convina 3 elementos: un defecto en el sistema, un atacante tratando de acceder a ese defecto y la capacidad que tiene el atacante para llevarlo a cabo. Las vulnerabilidades más críticas son documentadas en la *Common Vulnerabilities and Exposures* (CVE) [3]

- Superficie de Ataque - *Attack Surface*:
  Es el conjunto de todas las posibles vulnerabilidades que un sistema puede tener, en un cierto momento, para una cierta versión del sistema, etc.

- Amenaza - *Threat*
  Es una acción/evento que se aprovecha de las vulnerabilidades del sistema, debilidades, para causar un daño, y que dependiendo del recurso al que afecte el daño puede o no ser reparable.

- Ataque - *Attack*
  Es el éxito de la amenaza en el aprovechamiento de la vulnerabilidad (explotación de ésta), de tal forma que genera una acción negativa en el sistema y favorable para el atacante.

- *Exploit*:
  Usar una pieza de software para poder llevar a cabo un ataque sobre un objetivo, intentando **explotar** la vulnerabilidad de éste. Este tipo de acción permite en consecuencia obtener control en el sistema computacional, en donde la vulnerabilidad permitió su acceso.

- Ingeniería Social - *Social Engineering*
  El acto de manipular a las personas de manera que realicen acciones o divulguen información confidencial. El termino aplica al acto de engañar con el propósito de juntar información, realizar un fraude, u obtener acceso a un sistema computacional. La definición anterior encontrada en Wikipedia es extendida por el autor del libro "The Social Engineer's Playbook"[4], donde agrega que además la Ingeniería Social involucra el hecho de manipular a una persona en realizar acciones que finalmente no son para beneficiar a la víctima. Un ataque de éste tipo también puede llegar a ser realizado tanto **cara a cara**, como de forma indirecta. Pero el autor del libro indica que siempre hay un **contacto** previo con la víctima.

- Confidencialidad - *Confidentiality*
  Característica o propiedad que debe mantener un sistema para que la información privilegiada de alguna entidad que depende de tal sistema, no sea develada a nadie más que al que le pertenece la información.

- Integridad - *Integrity*
  Característica o propiedad que asegura que la información no será modificada/alterada nada más que por la entidad a quién le pertenece y con el previo consentimiento de éste.

- Disponibilidad - *Availability*
  Característica o propiedad que permite que la información esté disponible para quién lo necesite, en el momento que sea. La imposibilidad de obtener data en un cierto instante de tiempo, conlleva a la perdida de esta propiedad.

- *Phishing*
  Técnica de Ingeniería Social. Mediante el uso de correo elentrónico, links (url's), acortamiento de urls y otras herramientas, se busca que una victima visite un sitio o aprete un link de manera que se de la **autorización explicita** del usuario para descargar código malicioso o enviar datos a un servidor malicioso. El objetivo de esta técnica es poder obtener información valiosa de la victima o relizar algún daño en el cliente web.

- *Malware*
  Software creado para realizar acciones maliciosas en la data o sistema de un usuario. Puede ser instalado tanto de forma discreta como indiscreta, siendo

la segunda opción causada a través de un ataque previo a cierta vulnerabilidad que permitió la instalación del malware, sin el consentimiento del usuario privilegiado del sistema.

- *Man-in-the-Middle*
  Ataque que causa una pérdida en la Confidencialidad de la data que es revelada. La causa de este ataque puede ser tanto:
  - Por técnicas de Ingeniería Social, entregano un certificado malicioso que el usuario acepta con o sin intención.
  - A través de vulnerabilidades del sistema que debieron ser explotadas antes para causar el ataque MiTM.

1.6. Desarrollo de Software y Secure Software/Software Security

Ningún Desarrollo de Software es igual al anterior. Cada vez que un nuevo Proyecto surge es necesario ver qué tipo de Proceso es el que se usará, qué personas serán parte del grupo de trabajo, qué condiciones económicas está expuesto, qué tipo de Stakeholder están pendientes de que el Proyecto salga exitoso y un sin numero de variables, no menos importantes, a considerar. Por lo tanto, dependiendo de lo anterior los sistemas podrían llegar a ser simples o muy complejos y por consiguiente es necesario tener ciertas metodologías que aseguren que se cumplan con todos los Requerimientos Funcionales como No-Funcionales del Sistema a construir. Sin embargo, por muy buen Plan de Proyecto que se tenga, Jefe de Proyecto, Analista, Arquitecto, Programadores, Testers, etc. el proyecto podría peligrar antes o después de terminado, simplemente por que tuvo **Fallas de Seguridad**.

Un problema recurrente para todos los sistemas distribuidos en la Internet, es la gran cantidad de vulnerabilidades y amenazas a las que se enfrentan día a día. En el área de la Ciberseguridad se dice que atacantes como defensores de la seguridad, están constantemente realizando un *Juego del Gato y el Ratón*. Esto es dado a que constantemente los atacantes se ingenian nuevas formas para cometer delitos informáticos, a través de las vulnerabilidades en los sistemas que están sobre la Internet, y que las grandes compañías de seguridad, tratan de detectar - y crear una solución - lo antes posible. El fenómeno descrito, aparece en la literatura como *Zero-day attack*, y es en momento clave, donde un atacante explota una vulnerabilidad - hasta ese momento desconocida - de algún sistema (importante o no), y que si no es parchado lo antes posible puede comprometer no solo a sistemas, si no también a los usuarios que hacen uso de éste.

Si bien el **Zero-day Attack** es un evento que podría no ocurrir tan repetidamente, dado que se produce por el largo estudio llevado por el atacante, sobre el sistema a vulnerar, existen otras formas de comprometer a un sistema. Muchas veces

al Desarrollar sistemas, se prefiere utilizar API's[2] de otros sistemas para poder incluir funcionalidades ya implementadas, fomentando así el Reuso de piezas de Software. Si bien lo anterior es una buena práctica, si el sistema no cuenta con las medidas de seguridad necesarias, estas piezas podrían ser causa de amenazas de seguridad que terminarían por corromper el sistema y en consecuencia podría causar una pérdida monetaria a los Stakeholders. Por lo mismo, lidiar con las preocupaciones de seguridad es un factor que puede ser clave para el Desarrollo de Software: Una vez que el sistema este en *Deployment*, las consecuencias de no tener en cuenta la Seguridad desde el inicio del Desarrollo de Software pueden ser muy costosas [5], incluso pudiendo afectar la Confidencialidad, Integridad y Disponibilidad de los datos de quienes lo usan [6].

La literatura del área de **Secure Software** indica que los practicantes del Desarrollo de Software deben entender, en gran medida, los problemas de seguridad que podrían llegar a ocurrir en sus sistemas. No basta con saber como unir las piezas, no basta con que cada pieza de por si sea segura, si los componentes del sistema no actuan de forma coordinada, probablemente éste no será seguro [7], dado que la seguridad es una Propiedad Sistémica que necesita ser vista de manera holística. El problema de la mayoría del Software construído es que contiene numerosos **defectos** y **errores**, generando así **vulnerabilidades** que son encontradas y explotadas por los atacantes, de tal manera que generan el compromiso del sistema completo [1]. Por esto mismo, es imperante que sean entendidos los Requerimientos de Seguridad del Sistema a construir desde el inicio de su construcción y que todos los involucrados también los entiendan perfectamente.

Un sistema seguro o **Secure Software**, no tendrá, en lo posible, vulnerabilidades que puedan ser explotadas. Sin embargo, hay que tener en cuenta que dado que el Desarrollo de Software - incluso los sistemas seguros - dependen de personas, procesos, y tecnologías, será imposible tener un 100 % de confiabilidad en la Seguridad implementada, pues los **fallas humanas** siempre existirán. Si el sistema seguro no pudiera llegar a resistir un ataque, lo primero que debe intentar es aislarse del resto y degradar con gracia, de manera que no afecte el rendimiento del sistema. Finalmente, si un ataque tuvo a lugar lo importante es que el equipo detrás del sistema no se quede inmovil, luego de recuperarse del compromiso es necesario tener inmediatamente un parche que solucione la vulnerabilidad que se generó. Más aún, si esta vulnerabilidad se encontrara en un componente externo (*third party code*), el sistema que llegara a hacer uso de esa pieza de Software tiene que tener medidas para que no afecte la totalidad del sistema.

[2]Application Programming Interface

Capítulo 2

Estado del Arte: El Web Browser

2.1. Organizaciones

2.1.1. OWASP

Desde el 2066 la Open Web Application Security Project ha estado entregando pautas de còmo desarrollar aplicaciones web, en un ambiente que constantemente está cambiando. Su objetivo principal es buscar y combatir las causas de inseguridad en el desarrollo de Software, proporcionando una gran cantidad de documentación y herramientas aquellos que lo necesiten y no sean expertos en seguridad. Para lograr su cometido, todos los años la OWASP entrega una lista de los Riesgos y Threats más críticos sobre la seguridad de las Aplicaciones Web, de manera de preveer que Desarrolladores de Sistemas o programadores generen vulnerabilidades en el Software que crean.

2.1.2. IEEE CSD

2.2. Same Origin Policy

Este importante concepto nace a partir del Modelo de Seguridad detrás de una Aplicación Web, al mismo tiempo que es el mecanismo más básico que el Browser tiene para protegerse de las amenazas que aparecen en el día a día, haciendo un poco más complicado el trabajo de realizar un *exploit*. **Same Origin Policy** o **SOP** define lo que es un **Origen**, compuesto por el *esquema*, el *host/dominio* y *puerto* de la URL. Esta política permite que un Web Browser aisle los distintos recursos obtenidos por las páginas web y que solo permita la ejecución de *Script* que pertenezcan a un misno **Origen**.

SOP puede distinguir entre la información que envía y recibe el Web Browser, y solo se aplicará la política a los elementos externos que se soliciten dentro de una página web (recepción de la información). Esta imposibilidad de recibir información

de un **Origen** diferente al del recurso actual, permite disminuir la superficie de ataque y la posibilidad de explotar alguna vulnerabilidad en el sistema donde reside el Browser. Sin embargo, **SOP** no pone ninguna restricción sobre la información que el usuario puede enviar hacia otros.

2.3. HTML: HyperText Markup Language

HTML [8] es conocido por ser un *Simple Markup Language* o lenguage de marcado simple, usado principalmente para crear documentos de hypertextos que son posibles de portar desde una plataforma a otra, sin problemas de compatibilidad. Un documento HTML sigue el estandar dado por SGML o *Standard Generalized Markup Language*, que entrega una semántica apropiada para representar una gran variedad de información y aplicaciones. Un documento HTML consiste de un árbol de elementos y texto, cada uno de esos elementos es denotado por un tag inicial y uno final; estos tags pueden ir aninados y la idea es no se superponen entre ellos. Un HTML User Agent o Browser consume el HTML y lo parsea para crear un árbol DOM, que es la representación en memoria del documento HTML. Para poder crear una Aplicación interactiva y segura con HTML, es necesario evadir la creación de vulnerabilidades por donde los atacantes podrían comprometer la integridad del sitio o del usuario que descarga el recurso HTML del sitio atacado. Típicos errores que deben ser evitados cuando se usa scripting con HTML, es que los scripts en HTML tienen una semántica run to completion, esto quiere decir que el browser ejecutará el script mucho antes de que se realice el parsing del documento o gatillar un evento; este tipo de comportamiento es el que los atacantes aprovechan para realizar sus ataques.

2.4. Webworkers

2.5.

2.6. HTTP

2.6.1. Comunicación en HTTP

postMessage

XMLHttpRequest

WebSockets

Capítulo 3

El Web Browser

Capítulo 4

Documentación Semi-Formal de la Seguridad del Web Browser

4.1. Secure Software Development y Secure Software Design

La filosofía detrás de *Secure Software Developmet* es que detrás de cada etapa de desarrollo del software, se tengan en cuenta que los prinicipios de Seguridad: Confidencialidad, Integridad, Disponibilidad, Auditoría. Para cumplir este cometido es que se deben llegar a políticas y reglas que aseguren la Seguridad como una propiedad sistémica.

Varias comunidades tienen diferentes enfoques y técnicas de cómo asegurar la Seguridad en los sistemas, muchas pueden incluso tener similitudes y hasta trabajar juntas. En este trabajo, el enfoque tomado es aquél que busca entregar la propiedad de seguridad a través del entendimiento de un sistema a un alto nivel, identificando las amenazas durante la elicitación de requerimientos, de manera que se pueda extraer las posibles amenazas que podrían existir y utilizando elementos de diseño para hacer cumplir los principios de seguridad necesarios por el sistema; este enfoque es el que se dedica la comunidad de *Secure Software Design*.

Fernandez [7] sostiene que para construir un sistema seguro es necesario realizarlo de manera sistemática de tal manera que la seguridad sea parte del integral de cada una de las etapas del Desarrollo de Software - de inicio a fin. El enfoque que propone es ingenieril y por tanto es aplicable incluso para sistemas *legacy*, donde es posible hacer ingeniería inversa para comprobar si existen o no los requerimiento de seguridad implementados, de manera que permite generar un estudio con la intención de comparar y mejorar nuevos sistemas. En su libro [**?**] presenta una completa metodología para construir sistemas seguros a partir de patrones de diseño, a los cuales nombra como **Security Patterns**.

Como parte de la metodología propuesta, se plantea que para diseñar primero se deben entender las posibles amenazas a las que está expuesto el sistema. La

identificación de Amenazas [Bra08a] [Fer06c] es la primera tarea que presenta la metodología, que considera las actividades en cada caso de uso del sistema.

4.2. Arquitectura de Referencia - *Reference Architecture*

Una arquitectura de Referencia, de acuerdo a la *Open Security Architecture* o OSA[9], es considerado un elemento que describe un **estado de ser** y debe representar aceptadas buenas practicas.

En este trabajo lo que se pretende hacer con la Arquitectura de Referencia, es dar a entender los componentes y elementos que la mayoría de los Web Browser tienen. Se sabe que el Browser es un pieza de Software que ha sufrido varios cambios desde 1990, por lo tanto entre los desarrolladores de ésta herramienta ya existen conveniones de qué elementos funcionan mejor. Por consiguiente, no es de extrañar que diferentes browsers estén construidos de formas muy similares, incluso existan ciertos **patrones** que pueden explayarse de buena manera mediante una Arquitectura de Referencia, que manifeste los componentes, mecanismos de comunicación, funcionamientos, etc.

4.3. Misuse Patterns

En [10] se

Capítulo 5

Conclusiones

5.1. Conclusiones

5.2. Trabajo Futuro

Bibliografía

[1] Karen M Goertzel, Theodore Winograd, Holly L McKinley, Lyndon J Oh, Michael Colon, Thomas McGibbon, Elaine Fedchak, and Robert Vienneau. Software security assurance: A state-of-art report (sar). Technical report, DTIC Document, 2007.

[2] IEEE Cyber Security. Center for secure design.

[3] The MITRE Coporation. Common vulnerabilities and exposures.

[4] Jeremiah Talamantes. The social engineer's playbook.

[5] Carnegie Mellon University Computer Emergency Response Team. Early identification reduces total cost (segment from cert's podcasts for bussiness leaders).

[6] Mike Hicks. Interview to **Kevin Haley** (from **Symantec**), 2014. Mike Hicks (Profesor of Software Security course in Coursera.org).

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

[8] W3C Working Group. Html5 specification.

[9] Open Security Architecture. Definitions by osa.

[10] E.B. Fernandez, N. Yoshioka, and H. Washizaki. Modeling misuse patterns. In *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, pages 566–571, March 2009.

Apéndice A

Anexos