PROYECTO DE TESIS

[X] Magíster en Ciencias de la Ingeniería Informática

[ ] Doctorado en Ingeniería Informática

1 Título del Proyecto de Tesis	**Towards a Secure Reference Architecture of the Browser**
2 Nombre del Alumno	**Paulina Andrea Silva Ghio**
3 Número Teléfono - Celular	+56 9 79010117
4 e-Mail	**paulinasilvaghio@gmail.com, pasilva@alumnos.inf.utfsm.cl**
5 Fecha de Ingreso al Programa	**2013**
6 Pregrado (Título o Grado Institución, Año)	Ingeniería Civil Informática
7 Profesor Guía de Tesis	Raúl Monge
8 Fecha Presentación Tema de Tesis	Diciembre 2015
9 Fecha Aprobación Tema de Tesis	
10 Fecha tentativa de Término	**Marzo-Abril 2016**
11 Comisión interna de graduación	

RESUMEN

Debe ser suficientemente informativo, y contener una síntesis del proyecto, sus objetivos, resultados esperados y palabras claves.

(Su extensión no debe exceder el espacio disponible).

El *web browser* es una de las aplicaciones más usadas - killer app - y también una de las primeras en aparecer en cuanto se creó la 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.

Al tener sistemas que se interconectan con el *web browser*, *stakeholder* como desarrolladores deben estar al tanto de los posibles riesgos que podrían enfrentar. La insuficiente de educación en seguridad de los desarrolladores de software, la escasa y dispersa documentación sobre este aspcto en los navegadores (así como su estandarización), podría llegar a ser un flanco débil en el desarrollo de grandes arquitecturas que dependen del *browser* para realizar sus servicios. La elaboración de una Arquitectura de Referencia del *web browser*, utilizando patrones arquitecturales, podría ser una base para el mweb browserejor entendimiento de los mecanismos de seguridad y su arquitectura, que interactua con un sistema web mayor. Ésto mismo, entregaría una unificación de ideas y terminología, al dar una mirada holística sin tener en cuenta detalles de implementación tanto del *browser* como el sistema con el que interactúa.

El desarrollo de una Arquitectura de Referencia de Seguridad del navegador permitirá abstraer los mecanismos de defensa existentes en el *browser*, lo que permitirá entregar a los desarrolladores una descripción comprensible sobre estas técnicas de defensa. Los conocimientos que este trabajo desea entregar se enfocan principalmente en los mecanismos de defensa que posee el navegador web y que a través de patrones de seguridad se desea explicar.

Los resultados de este trabajo dentro del ámbito de la seguridad del navegador web, es generar un cuerpo organizado de información sobre este componente y su seguridad, de tal manera que se pueda sistematizar, organizar y clasificar el conocimiento adquirido en un documento, con formato semi-formal, tanto para profesionales como estudiantes del área Informática que estén insertos en el área de desarrollo de software. Se espera que este trabajo sirva como una guía inicial para aquellos que poseen escasos conocimientos de seguridad.

ABSTRACT

The web browser is known as one of the most used applications - or killer app - and also was the first introduced when the Internet was created (1990s). Which is why, its significant maturity level is above in comparison with other developements and can assure a certain level of trust whenever it is used as a client with other systems.

Having systems which are interconnected with the web Browser, stakeholder and Developers should be aware of the potential risks they could face. 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. A Reference Architecture of the web browser, using Architectural Patterns, could be a base for understanding the security mechanisms and its architecture, which interacts with a bigger web system. This would give an unification of ideas and terminology, giving a holistic view regardless the implementation details for both the browser and the system it communicates to.

The development of a Security Reference Architecture for browsers will allow the abstraction of existent security mechanism in the browser, which can deliver to developers a understandable description of this defense techniques. We have focused on abstracting defense mechanism of the browser into security patterns, to deliver in a easy way the knowledge.

The results of this work in the scope of web browser security, is to create an organized body of information on this componenet and its security, so it can systematize, organize and classify the adquired knowledge in a document, with a semi-formal format, for professionals or students in the computer area and developing software. It is expected this work to serve as an initial guideline for those with little knowledge in security.

I. FORMULACIÓN GENERAL DE LA PROBLEMÁTICA Y PROPUESTA DE TESIS

Debe contener la exposición general del problema, identificando claramente qué aspectos relacionados con la informática son los más relevantes. Además, deberá contener el marco teórico, la discusión bibliográfica con sus referencias y, finalmente, su propuesta de tesis.

(La extensión máxima de esta sección es de hasta 5 páginas. En hojas adicionales incluya la lista de referencias bibliográficas citadas)

Contexto General

Entre 1989 y 1990, Tim Berners-Lee acuñó el concepto de *World Wide Web* y con esto realizó la construcción del primer *web server*, *web browser* y las primeras páginas *web*. Mucho antes que aparecieran los grandes sistemas que ahora conocemos, el *web browser* permitía navegar páginas estáticas y realizar una serie de acciones limitadas a las tecnologías de ese tiempo. En la actualidad el *browser* es 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.

Durante la conocida guerra de navegadores, en la década de los noventa, los browsers tuvieron solo el objetivo de poder adquirir la mayor cantidad de usuarios posibles, entregando mejores funcionalidades que sus competidores. Debido a esto era habitual encontrar muchos parches que solucionaban problemas de seguridad, dada la cantidad de errores de programación y deficiente estructura del navegador, considerando que no había una pronta preocupación por integrar seguridad en el software. Además, la nula documentación que existía debido a la gran competitividad, muchas veces hacía que las extensiones hechas por *third-parties* creaban más agujeros de seguridad, que nuevas funcionalidades. El inicio del software libre u o*pen source*, cambió el escenario y las circunstancias, pero aún así, existen navegadores propietarios que no exponen su arquitectura tanto en código fuente como en documentación.

En el último tiempo el mercado de los *web browser* ha crecido bastante, principalmente debido a la robustez que éstos poseen y a la cantidad de años que llevan desarrollándose en la industria de software. Los navegadores más conocidos son: Google Chrome o su versión Open Source Chromium, Firefox, Internet Explorer, Opera y Safari; siendo los primeros 3 el enfoque de este trabajo, pues estos son los que poseen mayor cantidad de usuarios en la actualidad [statBrow].

La Web 2.0 se inició con el uso intensivo de tecnologías como AJAX, y esto ha permitido una nueva simbiosis entre el usuario, el *web browser* y el servidor o web server que se comunican entre sí. El navegador web es una herramienta indispensable para todo tipo de tareas computacionales como comunicacionales, su existencia ha penetrado completamente en las labores diarias de todos nosotros. En este mismo instante, la web a evolucionado nuevamente obteniendo un nuevo nombre: Web 3.0, donde se realiza el uso de inteligencia artificial y sistemas de recomendación para generar nuevos tipos de contenido para el usuario.

El Problema: Desarrollo de software y seguridad

Ningún desarrollo de software es igual al anterior. Por cada nuevo proyecto que surge es necesario ver qué tipo de proceso es el que se usará, qué personas serán parte del grupo de trabajo, qué condiciones económicas estará expuesto, qué *stakeholder* están pendientes de que el proyecto salga exitoso y un sin número de variables, no menos importantes a considerar. Por lo tanto, dependiendo de lo anterior, los sistemas podrían llegar a ser simples o muy complejos. En

consecuencia, se hace necesario tener ciertas metodologías que aseguren que se cumplan con todos los requerimientos funcionales como no-funcionales del sistema a construir. Sin embargo, un problema que existe recurrentemente, es que la mayoría del software construido contiene numerosos defectos y errores, generando así vulnerabilidades que son encontradas y explotadas por los atacantes, generando un compromiso parcial o total del sistema. Lo anterior sucede frecuentemente por que los sistemas no son desarrollados teniendo en cuenta la seguridad [Yoder98, Fer2004, Whyte2011]

Muchas veces al desarrollar sistemas, se prefiere utilizar API's (Application Programming Interface) 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. Lo expuesto anteriormente ejemplifica perfectamente con lo que tienen que lidiar los equipos de trabajo en proyectos de desarrollo de software, cuando dentro de sus preocupaciones la seguridad queda como un trabajo extra y no como parte del desarrollo completo.

Bien es sabido que un proyecto en producción que presente problemas que involucren a varias entidades, el costo asociado puede llegar a ser altísimo [cert], sin olvidar que podría llegar a afectar la confidencialidad, integridad y disponibilidad de los datos de los involucrados con el sistema. Por esto mismo, es imperante que sean entendidos, desde el comienzo, las preocupaciones de los Stakeholders y los Requerimientos de seguridad asociados, y que además todos los involucrados los entiendan perfectamente. Según [Whyte2011], la falta de preocupación en temas de seguridad en desarrollos de software no tiene una raíz principal, sino diversos factores como: la falta de estudios de seguridad en las mallas curriculares de las Universidades, pocos fondos para la investigación, la falta de iniciativa y preocupación desde la industria, el exceso de confianza de los desarrolladores, etc., son los causantes de las futuras vulneraciones en sistemas críticos.

La literatura que habla de la construcción de Secure software o software Seguro, indica que los practicantes de desarrollo de software deben entender, en gran medida, los problemas de seguridad que podrían llegar a ocurrir en sus sistemas. No basta con saber cómo unir las piezas, no basta con que cada pieza de por si sea segura, si los componentes del sistema no actúan de forma coordinada, probablemente éste no será seguro [Fer2013], dado que la seguridad es una Propiedad Sistémica que necesita ser vista de manera holística y al inicio del proceso.

Un sistema cualquiera conectado a Internet y accedido por un usuario a través de un navegador, estará en algún momento de su vida útil bajo amenazas que el desarrollador debe estar al tanto. Al conocer las amenazas es posible comunicar a los Stakeholders, de los posibles peligros a los que se enfrentan, aún cuando en el equipo de desarrolladores no exista un experto en seguridad. Bajo esta misma lupa, una Arquitectura de Referencia permitiría comunicar efectivamente los componentes, interacciones y comportamientos existentes entre el *browser* y el sistema con el que se comunica, de tal manera que sería posible entender mejor los posibles problemas de seguridad que el *browser* puede llegar a generar.

Motivación: ¿Por qué estudiar el Browser?

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 (desde cero) para el usuario del sistema.
- Actualmente la seguridad implementada que los *web browser* es bastante buena, dado que existen grandes compañias se aseguran de ello (Google, Microsft, Mozilla entre las más conocidas).
- El *browser* es una herramienta indispensable. La mayoría de los sistemas que lo usan en la vida cotidiana son de tipo: online banking, declaración de impuestos, promoción de empresas o tiendas, compras, y mucho más.

Sin embargo los sistemas que dependen del uso del browser, deben de tener en cuenta las posibles amenazas de seguridad a las que se enfrentarán por el solo hecho de usarlo. 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 tener el conocimiento de la seguridad del cliente web. El entendimiento de la estructura subyacente del web browser podría asegurar que las personas que trabajen en el desarrollo, comprendan los trade-off al momento de diseñar un software que necesite la colaboración del navegador web [Larrondo1996, Grosskurth2005, Grosskurth2006]; poniéndolo de otra forma, un atacante es capaz de explotar un ataque en el sistema, dado que tiene el conocimiento subyacente del browser, conoce sus vulnerabilidades y mecanismos de defensa.

En [Goertzel2007,Whyte2011] mencionan que la cantidad de tiempo dedicado en temas de seguridad, por los estudiantes en áreas de la Información/Computación/Software es casi nula. Si bien existen diversos factores [Whyte2011] que pueden ser causa de este comportamientos, principalmente las universidades y las industrias son un fuerte factor en la adopción de un cambio de mentalidad. Desarrollar un sistema crítico (o no) pero seguro, representa un gran desafío. No solo para los desarrolladores, si no también para los involucrados indirectamente, como los usuarios que navegan con un *browser* para hacer uso de éste sistema. Por lo mismo, es entendible que la seguridad sea un tema complicado de impartir, pero eso no significa que sea imposible. Nuestro trabajo tiene la intención de presentar un marco conceptual, que permita entender más este atributo de calidad y que pueda ayudar a otros desarrollos ser más seguros.

Este trabajo tiene una motivación principal. Ésta es ayudar a quien 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 software, obtengan el conocimiento necesario al momento de trabajar junto con el navegador web al realizar un desarrollo de software que dependa de éste.

De nuestro conocimiento hasta el momento, no existe ninguna propuesta de Arquitectura de Referencia del *browser* moderno, solo existe material desactualizado que no se adecua al paisaje actual [Grosskurth2005]. Creemos que una falta de conocimientos de seguridad con respecto al *browser*, podría afectar de forma directa el desarrollo de aplicaciones que lo utilizan. Por esto mismo, una guía o compilado de información, semi-formal, que una los conceptos y

componentes, como lo hace una Arquitectura de Referencia, podría enseñar a desarrolladores no expertos en seguridad, los peligros que existen. Nuestro trabajo considera a la seguridad como una propiedad sistémica que debe ser tomada en cuenta desde el inicio de su desarrollo [Fer2004, Fer2006, Braz2008, Fer2013, Fer2011].

Discusión Bibliográfica:
El primer paso para realizar el estado del arte con respecto a este punto fue realizar una búsqueda ordenada y a través de string de búsqueda dentro de web engines y bibliotecas digitales conocidas. Se utilizaron las siguientes plataformas para buscar documentación al respecto:

- Google, usando google dorks para filtrar resultados
- Scopus y Google Scholar, usando string de búsqueda y operadores booleanos para filtrar resultados.

Para aquellos buscadores donde era posible usar operadores booleanos también se trató de filtrar el contenido, para que pudiera mostrar resultados relacionados a: Browser y Reference Architecture. Sin embargo los resultados fueron bastante pobres. Desafortunadamente hasta la fecha no existen muchos trabajos relacionados a la construcción de una Arquitectura de Referencia para el *browser*. Dado que la búsqueda no entregó muchos resultados, se procedió a hacer un forward snowballing con el único paper que creemos entrega información similar a lo que se buscaba. Sin embargo, tampoco encontramos tanta información.

Lo encontrado:
En Larrondo-Petrie et. al [Larrondo1996] se realiza un análisis del *web browser*, con el fin de obtener un Modelo de Dominio, un Modelo de Objetos y un Feature Tree que describiera la estructura y funcionalidad entregada comúnmente por los *web browser*. El dominio, según explica el trabajo, es un set distintivo de objetos que se comportan de acuerdo a reglas y política que caracterizan el Dominio. El Análisis de Dominio es realizado para identificar dominios y cómo éstos interactuan con otros. La metodología usada para obtener los Dominios es el Object Oriented Analyis. Además de identificar, se clasifican estos dominios de acuerdo a su rol en el sistema terminado como: Dominio de Aplicación, Dominio de Servicio, Dominio de Arquitectura y Dominio de Implementación. El Modelo de Objetos sirve para entregar más detalles, un resumen general de las Entidades del *web browser* y sus relaciones. El Feature Tree pretende entregar detalles sobre los aspectos funcionales de la aplicación. El Modelo planteado, ségún el artículo, debería ser útil para los desarrolladores de software que construyen aplicaciones web basadas en el uso del *browser*. Este estudio se encuentra bastante lejos de lo que se quiere hacer en este trabajo, pero sirve para obtener un transfondo de lo que sucede en el *web browser*, aún cuando la información esté muy desactualizada.

En el trabajo de Grosskurth et al. [Grosskurth2005, Grosskurth2006] se utiliza una herramienta de ingeniería inversa, para obtener una arquitectura de referencia de muy alto nivel en base a dos navegadores open-source: Mozilla y Konqueror. Lo obtenido captura los subsistemas fundamentales comunes a los sistemas del mismo dominio, así como las relaciones entre estos subsistemas. En esta arquitectura se identifican los siguientes subcomponentes: Interfaz Usuaria, Persistencia de Datos, Browser Engine, Rendering Engine, Networking, Interprete de Javascript, XML Parser y Display Backend. Se menciona que estos componentes están estrechamente integrados (high coupling) con el Rendering Engine, lo cual tiene sentido en la arquitectura monoproceso que poseen Mozilla y Konqueror; es una decisión de diseño muy común en los

browser de la época. Al identificar estos componentes, se comenta que esto podría servir tanto en el diseño y durante la mantención de un sistema, pues mejora el entendimiento de ésta al ayudar a analizar los trade-off entre diferentes opciones de diseño; o también puede servir como un template para obtener nuevos diseños. Una vez obtenida la arquitectura conceptual, se inició una evaluación de ésta al comparar las arquitecturas concretas de cada *browser* open-source, extraídas desde el código fuente, para ver qué tanto el modelo conceptual era cercano a la realidad; la constante comparación permitió ademas refinar la Arquitectura de Referencia. Los browsers usados para validar fueron: Epiphany, Safari, Lynx, Mosaic y Firefox. Si bien la arquitectura presentada entrega bastante información a alto nivel, no desarrolla más que esa capa de abstracción, además parece ser que depende también de la implementación usada en la herramienta de ingeniería inversa.

En el documento [Godfrey2000] realizado en el año 2000, se describe la experiencia realizada al extender el trabajo del proyecto TAXFORM. Usando PBS, una herramienta de Ingeniería Inversa, se extrajo la arquitectura de software del navegador Mozilla, con el objetivo de entender la estructuración de sus componentes; además de crear vistas arquitecturales de alto nivel del sistema. El modelo arquitectónico obtenido contiene 11 subsistemas de alto nivel, de éstos los que más se destacaron fue el \textit{HTML Layout}, la implementación de herramientas y el código de la interfaz de usuario. En el año en que se lleva a cabo este estudio (2000), se menciona que la arquitectura ha decaído significativamente en muy poco tiempo, o su arquitectura no fue planificada cuidadosamente desde el comienzo; parte de lo anterior, el autor cree que es secuela de la Guerra de navegadores. Si bien el trabajo ayuda a entender un poco la estructura detrás del navegador, este trabajo es muy antiguo y la versión más actual del navegador ha cambiado bastante. Además, lamentablemente el enfoque de este estudio no es intentar entender lo que hace cada subsistema, si no que es la implementación de la herramienta misma para obtener la arquitectura de software del *browser* seleccionado.

En [Lwin2009] se propone un *browser* llamado Anfel SOFT, donde gracias al uso de Inteligencia Artificial, crea agentes que permiten mejorar la experiencia del usuario. El trabajo asegura que el *browser* será capaz de aprender el comportamiento de navegación del usuario, y guiará al usuario en su navegación para que ésta sea lo más efectiva posible. El paper obtiene los subsistemas que se pueden encontrar en un *browser* de la misma manera que lo realiza [Grosskurth2005]. Si bien la arquitectura que muestra refleja parte de lo visto en los 3 browsers escogidos en este estudio, no da detalles acerca de cada subsistema identificado. Además la Arquitectura de Referencia que entrega es la misma vista en [Grosskurth2005, Grosskurth2006], y a pesar que identifica otros posibles componentes, no agrega nada nuevo.

Podemos ver en los trabajos que algunos construyen una Arquitectura de Referencia basada en técnicas de Ingeniería Inversa. En cada uno de ellos el trabajo ha sido a muy alto nivel y la descripción de los subcomponetes del sistema es mínima. Si bien explican las relaciones entre éstos, no dan un mayor entendimiento en cómo se comportan en ciertas situaciones. En este trabajo se espera profundizar un poco más en la abstracción obtenida, incluyendo información de tanto los casos de uso del *browser* como las actividades que se realizan con otros usuarios. Desafortunadamente para esta memoria, no existe mucha literatura sobre el desarrollo de una Arquitectura de Referencia del *browser*, y de lo que hay, el trabajo más actual es el realizado por [Lwin2009] en el año 2009.

[statBrow]: StatCounter, Global Stats, Top 5 Desktop, Tablet and Console Browsers, 2015.
[Yoder98]: Yoder, J. and Yoder, J. and Barcalow, J. and Barcalow, J. Architectural patterns for enabling application security, 1998.
[Fer2004a]: Fernandez, Eduardo B. A Methodology for Secure Software Design, 2004.
[Fer2006]: Fernandez, Eduardo B and VanHilst, Michael and Petrie, Maria M Larrondo and Huang, Shihong, Defining security requirements through misuse actions, 2006.
[cert]: Computer Emergency Response Team, Carnegie Mellon University, Early Identification Reduces Total Cost (Segment from CERT's Podcasts for Bussiness Leaders), 2008.
[Whyte2011]: Whyte B. and Harrison J., State of Practice in Secure Software: Experts' Views on Best Ways Ahead, IGI Global, 2011.
[Fer2013]: Fernandez, Eduardo B., Security patterns in practice: designing secure architectures using software patterns, 2013.
[Larrondo1996]:Larrondo-Petrie, M.M. and Nair, K.R. and Raghavan, G.K., A domain analysis of Web browser architectures, languages and features, 1996.
[Grosskurth2005]: Alan Grosskurth and Michael W. Godfrey, A reference architecture for web browsers, 2005.
[Grosskurth2006]: Alan Grosskurth and Michael W. Godfrey, Architecture and evolution of the modern web browser, 2006.
[Godfrey2000]: Godfrey, Michael W and Lee, Eric H S, Secrets from the Monster: Extracting Mozilla's Software Architecture, 2000.
[Lwin2009]: Lwin, N.N.Z., Agent Based Web Browser, 2009.
[Goertzel2007]: Goertzel, Karen M and Winograd, Theodore and McKinley, Holly L and Oh, Lyndon J and Colon, Michael and McGibbon, Thomas and Fedchak, Elaine and Vienneau, Robert, Software Security Assurance: A State-of-Art Report (SAR), 2007.
[Braz2008]: Braz, Fabricio A and Fernandez, Eduardo B and VanHilst, Michael, Eliciting security requirements through misuse activities, 2008.
[Fer2011]: Fernandez, Eduardo B. and Yoshioka, Nobukazu and Washizaki, Hironori and Jurjens, Jan and VanHilst, Michael and Pernul, Guenther, Using Security Patterns to Develop Secure Systems, 2011.

II. HIPÓTESIS DE TRABAJO

Formule las hipótesis de trabajo señalando claramente su conjetura.

Hipótesis: La definición de una Arquitectura de Referencia de Seguridad para *web browsers* permite abstraer y capturar los principales aspectos estructurales y de comportamiento de éstos, facilitando la expresión de los patrones más conocidos de mal uso y seguridad relacionados con los navegadores.

La seguridad es una preocupación fundamental que muchos navegadores web utilizan de una u otra forma, una Arquitectura de Referencia de Seguridad (ARS) permite describir los elementos de seguridad disponibles en el *web browser*. Esto permite tener un elemento de guía para desarrolladores o profesionales del área de IT con poca experiencia en seguridad, que necesiten conocer los mecanismos de defensa disponibles en el navegador.

La metodología que se aplicará para desarrollar el trabajo:
Se mejorará una Arquitectura de Referencia ya obtenida en la memoria de pregrado y se desarrollará una Arquitectura de Referencia de Seguridad como la obtenida en [Fer2014], con la intención de abstraer lo mejor posible la infraestructura del web browser así como las defensas de seguridad.

Para poder validar la hipótesis:

- Se quiere medir la usabilidad del modelo con sujetos en el área de desarrollo de software. La usabilidad la queremos medir como: “cuán bien abstraen los conceptos de seguridad obtenidos con la Arquitectura de Referencia de Seguridad”. Para esto los sujetos deberían ser separados en: sujetos con conocimientos de seguridad y sujetos sin conocimientos en seguridad.
- Nuestros modelos obtenidos deben ser capaces de identificar y abstraer diferentes tipos de ataques basados en ingeniería social y defensas existentes en los browsers más actuales; o al menos los escogidos en este estudio: Google Chrome/Chromium, Firefox e Internet Explorer.

[Fer2014]: Fernandez, Eduardo B. and Monge, Raul and Hashizume, Keiko, Building a security reference architecture for cloud systems, 2014.

III. OBJETIVOS

III.1. Objetivos Generales

(Su extensión no debe exceder el espacio disponible)

Generar un cuerpo organizado de información sobre el *web browser* y su seguridad, de tal manera que en un documento se pueda sistematizar, organizar y clasificar el conocimiento adquirido, con una especificación semi-formal, dirigido a profesionales del área Informática que estén insertos en el área de desarrollo de software.

III.2. Objetivos Específicos

(Su extensión no debe exceder el espacio disponible)

- Comprender los conceptos relacionados al navegador web, sus componentes, interacciones o formas de comunicación, amenazas y ataques a los que puede estar sometido, como también los mecanismos de defensa.
- Identificar actores, componentes, funciones, relaciones, requerimientos y restricciones del navegador, para lograr abstraer una Arquitectura de Referencia (AR) a partir de la literatura técnica asociada y documentación disponible en Internet, para construir un 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.
- Construir un modelo conceptual de la seguridad del *web browser* a través de una Arquitectura de Referencia de Seguridad, identificando patrones de seguridad en el navegador.
- Profundizar el conocimiento en ataques relacionados con métodos de Ingeniería Social.
- Utilizar técnicas de Ingeniería de Software Experimental.

IV. METODOLOGÍA Y PLAN DE TRABAJO.

(Su extensión no debe exceder el espacio disponible)

1) Marco teórico y Estado del Arte: Revisión de conceptos relevantes del navegador, seguridad y Arquitecturas de Referencias existentes. Se examinarán artículos científicos, blogs o proyectos de investigación relacionados al tema.
2) Evaluación de Resultados Intermedios:
   a) Envío de papers a Asian PLoP
   b) Depurar el modelo obtenido en la memoria de pregrado.
3) Agregar nuevos patrones y adecuar modelo para especificar semi-formalmente estos patrones. Creación de la Arquitectura de Referencia de Seguridad.
4) Producir segunda versión de publicaciones: Se enviarán más trabajos a conferencias de tipo PLoP (Pattern Languages of Programs).
5) Documentar la tesis a partir de la compilación de papers obtenidos y el trabajo adelantado.

Actividad	Fecha
Marco teórico y Estado del Arte	Noviembre 2015 - Enero 2016
Evaluación de Resultados Intermedios	Noviembre 2015 - Febrero 2016
Agregar nuevos patrones y adecuar modelo	Enero - Abril 2016
Producir segunda versión de publicaciones	Enero - Abril 2016
Documentar la tesis	Enero - Abril 2016

V. TRABAJO ADELANTADO

- En la memoria de Pregrado se obtuvo un Estado del Arte sobre el *browser* y documentación sobre Arquitecturas de Referencias existentes. Falta averiguar sobre Arquitecturas de Referencia de Seguridad.
- Se han enviado 2 papers a la conferencia AsianPLoP con lo obtenido en la memoria de pregrado. El proceso llamado “shepherding” permite evaluar el paper enviado, al mismo tiempo que provee al autor la mejora del trabajo a través del intercambio continuo de sugerencias entre el autor/autores con el “shepherd”.

VI. RESULTADOS

VI.1. Aportes y Resultados Esperados

(Su extensión no debe exceder el espacio disponible)

- Se espera poder obtener una documentación semi-formal que permita comprender mejor la seguridad y estructura del *browser*. Para así entender que éste tiene una superficie de ataque muy usada por los atacantes de los sistemas con los que se conecta el usuario, utilizando frecuentemente ataques de ingeniería social.
- Crear un catálogo de patrones de mal uso que permitan instruir a los más novatos, sobre los tipos de ataques existentes.
- Arquitecturas de Referencia (AR) de su infraestructura como de los mecanismos de defensa existentes (ARS).
  - Incluyendo 2 nuevos patrones arquitecturales: el Browser Kernel y el Web Content Renderer.
  - Patrones de Seguridad existentes o nuevos y aún no documentados.
- Experimentación del modelo utilizando técnicas de ingeniería de software experimental.

VI.2. Formas de Validación

(Su extensión no debe exceder el espacio disponible)

Las Arquitecturas de Referencias (AR/RA) y Arquitecturas de Referencias de Seguridad (ARS/SRA) no son implementables. Éstos son modelos abstractos y no pueden ser evaluados con respecto a la seguridad o desempeño por medio de testing o exprimentación. Una RA o SRA es similar a un patrón y su uso es similar, permite ayudar en la implementación de nuevos sistemas, ser guía para novatos sobre el sistema en si y evaluar el sistema o compararlo con otros. Principalmente la evaluación se basará en cuán bien reflejan o representan los conceptos qu describen del sistema, qué tan completo y precisos son, cómo pueden ser aplicados a algún diseño or evaluación del sistema y que tan útiles son para ciertas funciones relevantes. La validación final se da cuando desarrolladores practicantes o expertos, en el área del tipo de sistema, los encuentran útiles y convenientes para construir o explicar una arquitectura concreta. La validación de los artefactos generados en este trabajo será a través de la revisión realizada por expertos en el área de computer science y patrones, por medio de la asistencia de conferencias. Al ser aceptados, presentados, mejorados y publicados en conferencias *PLoP, tendremos un respaldo de la utilidad de estos patrones. Los patrones antes de ser publicados, son refinados a través del proceso llamado “Shepherding”. Para esta tesis se espera ir a las conferencias: AsianPLoP y EuroPLoP.

Se utilizarán conocimientos en ingeniería de software experimental para probar la usabilidad de nuestra propuesta. Queremos medir qué tan bien el modelo abstrae los conceptos obtenidos y cuán bien son entendidos por los sujetos.

VII. RECURSOS

VII.1 RECURSOS DISPONIBLES

Señale medios y recursos con que cuenta el Departamento de Informática de la UTFSM, para realizar el proyecto de tesis (libros, software, laboratorios, etc.).

(Su extensión no debe exceder el espacio disponible)

Acceso a biblioteca electrónica para lectura de papers necesarios para la tesis.

VII.2 RECURSOS SOLICITADOS

Señale medios y recursos no disponibles en el Departamento de Informática de la UTFSM, necesarios para realizar el proyecto de tesis (libros, software, laboratorios, etc.). Su extensión no debe exceder el espacio disponible

Dado que la única manera de evaluar esta tesis consiste en la presentación de esta propuesta en conferencias, se es necesario el apoyo económico para presentar el/los trabajo/s y así cumplir con exigencias del programa.