Módulo de políticas de seguridad y privacidad para la realización de transacciones de comercio electrónico

Anexos

ERWIN ARNOLDO DAZA RENDÓN
FREDDY MINA GRUESO

Director: Ing. ROBERTO CARLOS NARANJO CUERVO

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información
Línea de Tecnologías Internet
Popayán, Junio de 2009

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

ANEXO A- DEFINICIONES GENERALES

1 Definiciones generales

Las definiciones mostradas a continuación representan los términos utilizados durante la realización del proyecto de grado y se exponen aquí para brindar mayor comprensión del proyecto.

1.1 Política de seguridad

Una política de seguridad permite declarar un conjunto de lineamientos obligatorios que deben seguirse dentro de una organización para garantizar la seguridad de los activos más importantes para esta. La efectiva de las políticas de seguridad radica en su diseño, implementación y mantenimiento, pues una política implementada sin tener en consideración estas etapas corre el peligro de no ser tenida en cuenta, parecer incompletas o redundante [1].

1.2 Estándar

Regla que especifica una acción o respuesta a seguir ante a una situación dada. Los estándares son orientaciones obligatorias que buscan hacer cumplir las políticas. Los estándares sirven como especificaciones para la implementación de las políticas [1].

1.3 Mejor Práctica

Es una regla de seguridad específica a una plataforma que es aceptada a través de la industria al proporcionar el enfoque más efectivo a una implementación de seguridad concreta. Las mejores prácticas son establecidas para asegurar que las características de seguridad de sistemas utilizados con regularidad estén configurados y administrados de manera uniforme, garantizando un nivel consistente de seguridad a través de la organización [1].

1.4 Recomendación

Las recomendaciones deben considerarse al implementar la seguridad. Aunque no son obligatorias, serán seguidas a menos que existan argumentos documentados y aprobados para no hacerlo [1].

1.5 Procedimiento

Los procedimientos definen específicamente cómo las políticas, estándares, mejores prácticas y recomendaciones serán implementados en una situación dada. Los procedimientos son dependientes de la tecnología o de los procesos y se refieren a plataformas, aplicaciones o procesos específicos. Son utilizados para delinear los pasos que deben ser seguidos para implementar la seguridad relacionada a dicho proceso o sistema específico [1].

1.6 XPCOM

Es el modelo de componentes de objetos multiplataforma de Mozilla y es similar al modelo de objetos de Microsoft (COM) [2].

1.7 Componente

Un componente es una pieza de código reutilizable o modular que implementa una o más interfaces definidas, en Mozilla pueden existir como un Singleton [3] (un objeto que es instanciado solo una vez y es utilizado por otro código) u objeto del cual se pueden crear varias instancias.

1.8 Interface

Es un conjunto de puntos de acceso a un componente.

1.9 Modulo

Conjunto de Componentes XPCOM agrupados dentro de una aplicación o DLL [4].

1.10 Objetos

Se refiere a objetos XPCOM o Java Script **¡Error! No se encuentra el origen de la eferencia.**, un objeto XPCOM es aquel que pertenece a una clase que implementa una interface dada.

1.11 Salvaguarda

Controles necesarios para ofrecer protección ante los riesgos de los activos de una organización o proceso dentro de esta.

1.12 ISO 17799 Y RFC 2196

La ISO 17799 es un código de buenas prácticas, útil cuando se desea implementar un Sistema de Gestión de Seguridad de la Información (SGSI), su objetivo es proteger uno de los activos más importantes, la información y los recursos que la sustentan, este objetivo se alcanza utilizando controles tales como: políticas de seguridad, practicas, procedimientos, estructuras organizacionales y funciones del software [6]. Recursos como la realización de un análisis de riesgos permiten encontrar los requerimientos de seguridad necesarios y ayudan a determinar las prioridades y acciones orientadas a gestionar el riesgo. Las revisiones periódicas de los riesgos y de los controles implementados para mitigarlos permiten:

- Reflejar cambios en los requerimientos.
- Tomar en cuenta nuevas amenazas y vulnerabilidades.
- Comprobar si los controles adoptados para disminuir el riesgo siguen siendo efectivos o no.

Gracias a esta ISO se pueden seleccionar los controles adecuados para mitigar los riesgos, estos controles se pueden adoptar de la misma guía, otros estándares o se pueden crear siguiendo sus recomendaciones, entre los controles implementados mas importantes tenemos el uso de políticas de seguridad, gracias a estos se puede decir que se tiene implementado en gran medida el SGSI.

Aunque esta ISO no es una guía detallada para la construcción de políticas si brinda un conjunto de pautas para desarrollarlas, estas pautas incluyen las partes que debe tener la documentación de la política: definición de la política y objetivos, consecuencias del no cumplimiento, procedimientos y normas de seguridad que deben seguir los usuarios de estas, etc.

Por otro lado el RFC 2196, que es un guía para desarrollar políticas de seguridad y procedimientos para los sitios que disponen de sistemas a través de Internet, nos enumera algunas características que debe cumplir una buena política de seguridad [7]:

- Deben existir mecanismos o procedimientos concretos que permitan su implementación o puesta en práctica, estos mecanismos deben ser los más apropiados y prácticos.
- Se deben utilizar herramientas de seguridad para obligar su cumplimiento.
- Deben poseer mecanismos de control para valorar la efectividad de las herramientas de seguridad utilizadas.

Esta guía también especifica algunos de los pasos necesarios para implementar un plan de seguridad: identificar lo que se esta tratando de proteger, contra que amenazas se protegerá, la probabilidad de estas amenazas, implementar medidas de protección rentables y realizar revisiones periódicas para encontrar puntos débiles.

Siguiendo las recomendaciones citadas en estas dos guías se puede lograr implementar y mantener las políticas necesarias de acuerdo a los requerimientos de seguridad de cualquier organización o proceso dentro de esta.

1.13 Componentes de una Política de Seguridad

Aunque la ISO 17799 y el RFC2196 son excelentes guías, se quedan cortas en algunos aspectos claves del diseño de las políticas que estén orientadas a brindar seguridad de los usuarios de transacciones B2C, por esta razón a los controles y recomendaciones incorporados en ellas se suma los detallados en el libro “Information Security Management Handbook” [8] para alcanzar un diseño mas robusto de las políticas de seguridad, este diseño debe incluir:

- ***La definición de la política de seguridad***: que es definir la política de seguridad, esta definición debe ser clara y redactada de forma sencilla, esto ayuda a que sea entendida y puesta en practica.

- ***Responsable***: Se refiere a la persona encargada de la política, esta será la persona encargada de la política mientras esta se mantenga en uso.

- ***La Descripción***: Es el que, lo que se pretende lograr al implementar la política.

- ***Definir con que amenazas se relaciona***: que amenazas tienen incidencia directa sobre la política.

- ***Reseñar la fecha de creación, fecha de la última revisión, estado de la política (en uso u obsoleta) e identificador:*** *Esto con el fin de hacerle seguimiento a la política de seguridad.*

- ***Anotaciones:*** Estas permiten crear un registro de los cambios que surgen en el tiempo y que están relacionados con las amenazas y vulnerabilidades que pueden afectar un activo.

- ***Excepciones:*** En este punto se deben gestionar las situaciones donde la implementación no es posible, la gestión incluye el nombre de la persona responsable de la documentación y vigilancia de la excepción, y el tiempo de duración de la misma.

- ***Estándares u orientaciones:*** No es más que los procedimientos adecuados que se deben seguir para que las políticas puedan ser cumplidas.

- ***Mejores Practicas:*** Tiene relación con la forma de configurar las herramientas o procedimientos que se deben seguir para garantizar el mejor funcionamiento o configuración de los activos software, hardware, etc.

- ***Guías:*** Son los pasos necesarios a seguir para lograr el cumplimiento de la política.

- ***Dimensiones que afecta:*** Entre estas características cual se ve afectada; autenticidad, integridad, confidencialidad, disponibilidad.

- **Consecuencias del no cumplimiento:** que ocurre en el caso de no cumplir con la política.

ANEXO B- CICLO DE VIDA DE LAS POLITICAS DE SEGURIDAD

2 Ciclo de vida de las políticas de seguridad

La Universidad Nacional dentro de su programa de Auditoria a Sistema Operativos, más concretamente en la parte de Administración e implementación de la seguridad, trata las políticas de seguridad y propone un ciclo de vida para estas, aunque se eliminaron algunos pasos del ciclo inicial el diseño presentado aquí (Figura 2.1) también ofrece la posibilidad de auditar el sistema en el futuro. El porqué de adoptar este sistema es debido a que fue creado teniendo en cuenta guías como el RFC2196 y escritos de algunos autores interesados en el tema de la seguridad [9].

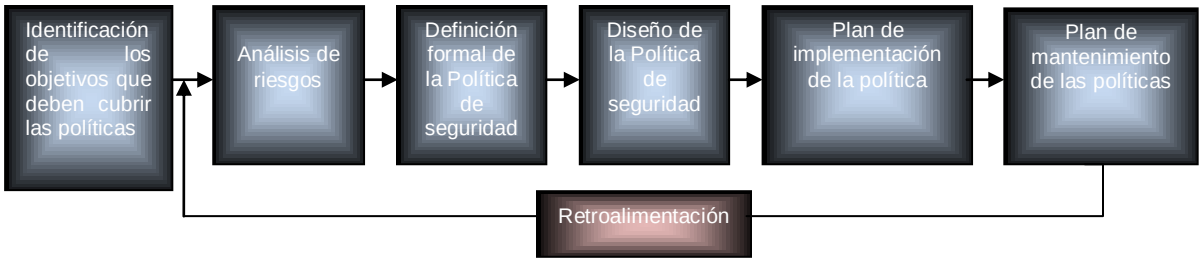


Figura 2.1. Ciclo de vida de una política de seguridad para transacciones B2C

Este ciclo de vida, Figura 2.1, consta de las siguientes etapas:

2.1 Identificación de los objetivos que deben cubrir las políticas

En esta etapa se debe identificar los objetivos que deben cubrir las políticas a diseñar, además se debe determinar el alcance de cada una. Los objetivos deben tener una estrecha relación con las dimensiones de valoración de los activos.

2.2 Análisis de riesgo

En todo el ciclo de vida el punto que requiere un mayor trabajo es el de análisis de riesgo, para este existen herramientas y metodologías muy variadas cada una con diferentes ventajas y desventajas **¡Error! No se encuentra el origen de la eferencia.**. Magerit [11], fue la metodología escogida para el desarrollo del análisis de riesgo, y los criterios que llevaron a esto fueron; su libre uso, documentación organiza y con ejemplos y adhesión de herramientas útiles para agilizar el análisis de datos, como es el caso de Pilar [12]. El trabajo con Magerit incluye:

2.2.1 La estructuración del proceso de análisis de riesgo

El cual implica la planificación del análisis de riesgo. Dentro de esta se establecen los objetivos a lograr en la gestión de riesgo y el ciclo de tareas para alcanzarlos.

2.2.2 Análisis de riesgos

El cual se incluye la definición, clasificación, estructuración y valoración de los activos (valoración global por uso, valoración por dimensiones, etc.), dependencias entre activos, identificación de amenazas y vulnerabilidades, cálculo de la probabilidad de ocurrencia, degradación de los activos, estimación y medición del estado del riesgo.

2.2.3 Gestión del riesgo

Selección de las salvaguardas necesarias para eliminar, mitigar o trasladar el riesgo, además de la verificación de la efectividad de las mismas.

2.3 Definición formal de la política de seguridad

Es importante que la definición de la política se haga de forma clara, sin ambigüedades y lo que se busca en esta etapa es definir formalmente la política.

2.4 Diseño de la política de seguridad

En esta etapa se debe adoptar una metodología que permita el desarrollo, implementación, mantenimiento y eliminación de las políticas. Desarrollar políticas sin tener en cuenta esto puntos hace que las políticas creadas no cumplan el rigor necesario para ser tenidas en cuenta. El libro de “The Security Policy Life Cycle: Functions and Responsibilities”, enumera algunos puntos necesarios para el diseño de una política de seguridad, este diseño permite la implementación de la política, su mantenimiento y eliminación. Algunos de los puntos tomados en este son: La declaración de la política o definición formal, estándares, mejores prácticas, guías y procedimientos, que complementados con puntos como la adición del responsable de la política, fecha de creación, anotaciones especiales, información referente a las amenazas contra la que se implementa, la descripción detallada de la políticas, excepciones, causas del no cumplimiento y dimensiones que afecta permiten obtener un diseño consistente de las políticas, en la Figura 2.2 se puede observar una plantilla tomada para el diseño de las políticas.

POLITICA: debe contener la definición formal de la política	
RESPONSABLE: identifica a la persona responsable de hacerle seguimiento a la política durante todo su ciclo de vida	
RELACION AMENAZA Es una lista con las amenazas contra las cuales se crea la política.	***DESCRIPCION*** Es esta se debe definir el objetivo de la política, contra que o para que se utiliza.
Excepciones En este punto se deben gestionar las situaciones donde la implementación no es posible, la gestión incluye el nombre de la persona responsable de la documentación y vigilancia de la excepción, y el tiempo de duración de la misma.	
Fecha de Creación: ***Fechas ultimas revisiones:*** ***Estado De la Política:*** *En uso* U obsoleta ***Identificador:***	
Anotaciones Estas anotaciones deben plasmar factores de riesgo encontradas durante las revisiones que se haga de la política y deben definir si se debe realizar un nuevo análisis de riesgo.	
Estándares u orientaciones	Orientaciones obligatorias para hacer cumplir.
Mejor Practica	Son establecidas para asegurar que las características de seguridad de los sistemas utilizados estén configurados y administrados de manera uniforme y consistente.
Guías	Declara la forma o formas como se pueden implementar las políticas, estándares o mejores practicas.
Dimensiones que afecta	Se debe nombrar aquí cuales son las dimensiones de valoración del activos (Privacidad o confidencialidad, integridad, autenticidad y no repudio) esto con el objetivo de saber en cual dimensión es valiosa la política).
Consecuencias del no cumplimiento	Son las consecuencias a las cuales nos enfrentamos al no cumplir la política.

Figura 2.2. Plantilla para el diseño de las políticas de seguridad

2.5 Plan de implementación de la política.

Terminado el diseño, la siguiente fase es la de implementación, en esta se debe definir las herramientas necesarias para tal fin, estudio de soluciones, etc.

2.6 Plan de mantenimiento de la política.

El asignar un responsable y un diseño con los elementos apropiados a cada política de seguridad permite que al ser implementada se le pueda hacer seguimiento para garantizar que se adecúe de cara a nuevos riesgos o condiciones cambiantes de la tecnología.

2.7 Retroalimentación.

La retroalimentación es necesaria debido a lo cambiante de las tecnologías de la información, esta retroalimentación debe realizarse al encontrar nuevas riesgos que afecten los activos, esto implica que se deba realizar un nuevo análisis de riesgos.

ANEXO C- METODOLOGIA DE ANALISIS Y GESTION DE RIESGO

3 Metodología de análisis y gestión de riesgos de los sistemas de información (Magerit)

Los usuarios de los sistemas de información, que frecuentemente no son técnicos, se preguntan si estos sistemas merecen su confianza, pero esta se ve mermada por cada fallo. Lo ideal es que los sistemas sean fiables; pero lo cierto es que se acepta convivir con sistemas que en ocasiones no lo son. El asunto no es tanto la ausencia de incidentes si no la confianza de que están bajo control: se sabe qué puede pasar y se sabe qué hacer cuando pasa. El temor a lo desconocido es el principal origen de la desconfianza y, en consecuencia, aquí se busca conocer para confiar: conocer los riesgos para poder afrontarlos y controlarlos.

Conocer las amenazas y vulnerabilidades a los que están sometidos los activos y elementos que los soportan se hace parte imprescindible para poder gestionar el nivel de riesgo al cual están expuestos los activos y por ello han aparecido multitud de guías informales, aproximaciones metódicas y herramientas de soporte, el gran reto de todas ellas es la complejidad del problema al que se enfrentan; complejidad en el sentido de que hay muchos elementos que considerar, y cuando no se es riguroso, se corre el peligro que las conclusiones sean de poco fiar. Es por ello que Magerit sigue una aproximación metódica que no deje lugar a la improvisación.

Gracias a esta aproximación metodológica de Magerit se tienen razones suficientes para su utilización, entre las cosas que se pueden hacer con la metodología tenemos:

- Permite encontrar los requerimientos de seguridad en una organización o proceso.
- Permite minimizar los riesgos y aumentar la confianza en los sistemas sujeto de análisis.
- Permite conocer y gestionar los riesgos al que están expuestos los elementos de trabajo, es imprescindible poder gestionarlos para garantizar un nivel de seguridad de mayor aceptación.
- Permite seleccionar un conjunto de salvaguardas o contramedidas para disminuir el riesgo.

3.1 Objetivos de Magerit

Entre sus principales objetivos están:

- Concientizar a los responsables de sistemas de información de la existencia de riesgo y de la necesidad de disminuirlos.
- Ofrecer un método sistemático para analizar y gestionar estos riesgos.
    - Uniformidad de los informes de análisis y gestión de riesgos.

- Preparar a las organizaciones para procesos de auditoría, acreditación y/o certificación.

3.2 Organización de las Guías

Magerit consta de un conjunto de guías las cuales permiten su adopción de forma ordenada, estas guías son:

- **La Metodología**: La cual incluye los pasos del análisis y de la gestión de riesgos, análisis de resultados, estructuración del proyecto de análisis de riesgos, consejos prácticos, etc.

- **El Catalogo de Elementos**: Este marca las pautas en cuanto a los tipos de activos, dimensiones de valoración, criterios de valoración, catálogo de amenazas por activos y salvaguardas. Cada sección expone un formato XML [13] para cada elemento, buscando su facilidad de publicación y de procesamiento por herramientas automáticas de análisis y gestión de riesgos [14]. El catálogo de elementos tiene como objetivos:

  - Facilitar la labor de las personas que realizar el análisis de riesgos, en el sentido de ofrecerles ítems estándar a los que puedan adherirse rápidamente, centrándose en lo específico del sistema (objeto del análisis).

  - Promover una terminología y unos criterios que permitan comparar, integrar y homogeneizar los resultados de los análisis realizados.

- ***El Libro de Técnicas***: El cual incluye técnicas específicas como: análisis por tablas, análisis algorítmico, arboles de ataque y otras técnicas generales. Gracias a estas técnicas se puede calcular el nivel de riesgos, o el costo que supondría un salvaguarda entre otros.

3.3 Introducción al Análisis y Gestión de Riesgos

La seguridad de la información consta de una serie de características ó atributos denominados dimensiones que hacen valiosos a los activos de una organización. La valoración que recibe un activo en una dimensión es la medida del perjuicio para la organización si el activo se ve dañado en dicha dimensión. Por ello el objetivo a proteger es la misión de la organización, teniendo en cuenta las siguientes dimensiones de seguridad y su significado [11].

Disponibilidad: Disposición de los servicios a ser usados cuando sea necesario. La carencia de disponibilidad supone una interrupción del servicio. La disponibilidad afecta directamente a la productividad de las organizaciones.

Integridad: Permite que la información mantenga las características de completitud y corrección. La integridad afecta directamente el correcto desempeño de las funciones de una organización.

Confidencialidad: Garantiza que la información solo sea accedida por las personas o sistemas con los permisos apropiados. Contra la confidencialidad pueden darse fugas y filtraciones de información, así como accesos no autorizados.

Autenticidad (de quién hace uso de los datos o servicios): De quien prestan los servicios o de los dueños de los datos, permite el no repudio.

3.3.1 Conceptos Principales

A continuación se definen los términos principales utilizados en el análisis de riesgos:

Riesgo: Es la medida o estimación de que una amenaza se materialice sobre un activo. El riesgo indica lo que le pueden pasar a los activos si no se protegen adecuadamente.

Análisis de riesgos: proceso metodológico para encontrar el nivel de riesgo al cual esta expuesta una organización, sistema, etc.

Gestión del riesgo: selección e implantación de salvaguardas para conocer, prevenir, reducir o controlar los riesgos identificados.

***Impacto económico*:** definido como el daño sobre el activo derivado de la materialización de la amenaza.

3.4 Realización del análisis y gestión de riesgos

El análisis y gestión de riesgos, utilizados por la metodología Magerit, involucra una serie de pasos, sin llegar a ser muy rigurosos, se describen los pasos tomados para implementar la metodología y alcanzar los objetivos buscados en este trabajo de grado.

3.4.1 Planificación

Como parte de la planificación, se identifican los objetivos concernientes al análisis de riesgo y su alcance, además de definir los criterios de evaluación tomados para medir los activos, amenazas, vulnerabilidades, probabilidades, impacto y riesgo asumido. El objetivo de esta etapa es crear el marco general de todo el análisis el cual incluye.

- Definir los objetivos del análisis de riesgos.

- Definir del dominio o ámbito que abarcará el análisis de riesgos.

- definir los criterios de evaluación
  Los objetivos en esta etapa son:

  - Determinar el catálogo de tipos de activos.
  - Determinar las dimensiones de valoración.
  - Determinar el nivel de valoración de los activos.
  - Determinar los niveles de valoración de las amenazas, frecuencia y degradación.

Estos objetivos permiten obtener como resultado los siguientes elementos:

- Un catalogo de tipos de activos.
- Una relación de las dimensiones de seguridad.
- Criterios de Valoración.

3.4.1.1 Objetivos del análisis de riesgo del proyecto

Los objetivos que busca cubrir el análisis de riesgo son los siguientes:

- Identificar las principales amenazas que afectan las transacciones hechas con tarjetas de crédito o débito en entornos de comercio electrónico B2C.

- Identificar los requerimientos principales de seguridad para garantizar la autenticidad de las partes, privacidad o confidencialidad de la información, el no repudio de las transacciones y la integridad de los datos en tránsito.

- Identificar el nivel de los riesgos asociados a los diferentes activos involucrados en transacciones B2C, con tarjeta crédito y débito.

- Hallar los mecanismos o controles necesarios para mitigar los riesgos encontrados.

3.4.1.2 Alcance del proceso de análisis de riesgos

El análisis de riesgo se realizará para aquellas amenazas presentes en transacciones B2C realizadas con tarjetas crédito y/o débito, a través de computadores de escritorio, y para aquellos riesgos que tienen incidencia en el lado cliente. Como referente se toman los datos encontrados en el capitulo dos de la monografía de grado que constituyen gran parte de los elementos involucrados en transacciones B2C y las amenazas a las que estos se enfrentan.

3.4.1.3 Criterios de Evaluación

Estos criterios deben permitir catalogar los activos por tipo y por dimensión además de mostrar los valores o rango de valores que se tomarán para su valoración. Para esto se definen dos catálogos (el de tipos de activos y el de dimensiones) y los criterios de valoración tomados para valorar cada activo.

- **Catalogo de tipos de activos**: El siguiente catalogo de activos se pudo obtener gracias al catalogo de elementos expuesto por la metodología y se utilizará para catalogar los diferentes activos. La Tabla 3.1 puede llegar a crecer dependiendo de la necesidad de adoptar nuevos activos.

Tipo de activo	Descripción
Datos	• Agrupan el conjunto de datos necesarios para la realización de la transacción y que son de vital importancia para la autenticación de los usuarios. Estos datos se clasifican de acuerdo a la metodología y por orden de importancia en: datos clasificados (secretos, y de carácter comercial) y personales.
servicios	• Servicios asociados a la gestión de certificados y cifrado de datos.
Aplicación	• Que permiten el acceso a sitios de comercio electrónico.

Tabla 3.1. Catalogo de activos

- **Dimensiones de seguridad**
  Las dimensiones se utilizan para valorar las consecuencias de la materialización de una amenaza dentro de una faceta o grupo de estas, para tal fin se hace necesario definir dentro del proceso que define cada una de estas dimensiones (Tabla 3.2). La valoración que se dé, tomará en consideración todas las dimensiones que afecten el activo.

Privacidad o confidencialidad
• La información financiera (numero de cuentas, claves de tarjetas, etc.) debe ser sólo de conocimiento del dueño de la tarjeta y del banco emisor de la misma, el emisor utiliza esta información para autenticar a los usuarios y verificar que existan fondos para la realización de la transacción. • La información debe mantener su carácter de confidencialidad y ser accesible solo por las personas o programas autorizados para tal fin.
Integridad
• Integridad de los datos en tránsito debido a las amenazas de la red, tal es caso de los snnifer. • Integridad de los programas o servicios requeridos en las transacciones.
Autenticidad
• Autenticidad de las partes involucradas en la transacciones • De quienes accedan los datos
No repudio
• No repudio de las transacciones realizadas

Tabla 3.2 Dimensiones de Valoración de las transacciones B2C

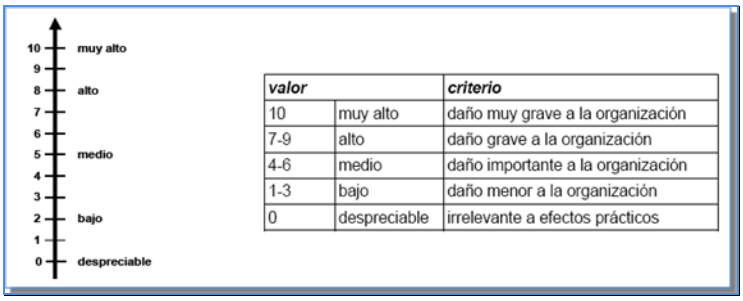
- **Criterios de Valoración**

  Para valorar los activos, teóricamente vale cualquier escala, sin embargo es importante que estas cumplan los siguientes puntos:

  - Una escala común que permita comparar riesgos.
  - Se use una escala centradas en diferencias relativas de valor [14].
  - Se use un criterio homogéneo que permita comparar análisis realizados por separado.

  Aunque es difícil hacer valoraciones cuantitativas, debido a datos escasos o desconocimiento del tema a tratar, etc. se intentará valorar los activos de esta forma teniendo como fuente los datos suministrados por el Instituto Nacional de tecnologías de la comunicación (INTECO[1]), para aquellos datos que no estén disponibles dentro de esta fuente se realizará una valoración cualitativa que responda a criterios subjetivos, discreción del usuario. La siguiente figura (Figura 3.1) muestra los criterios de valoración que junto con los datos obtenidos de las fuentes de INTECO permitirá valor los activos, de esta misma forma se pretende encontrar los valores para el impacto, degradación y frecuencias de ocurrencia,

[1] Centro de respuestas a incidentes de seguridad referentes a tecnologías de la información (TIC)



valor		*criterio*
10	muy alto	daño muy grave a la organización
7-9	alto	daño grave a la organización
4-6	medio	daño importante a la organización
1-3	bajo	daño menor a la organización
0	despreciable	irrelevante a efectos prácticos

Figura 3.1 Escala de valoración

3.4.2 Análisis de riesgos

Este proceso es el núcleo central de Magerit y su correcta aplicación condiciona la validez y utilidad de todo el proyecto. La identificación y estimación de los activos involucra los siguientes puntos.

1. Identificación de los activos y la valoración que merecen dentro de la organización (Modelo de valor).
2. Identificación de las amenazas que afectan a los activos (Mapa de riesgo).
3. Identificar las salvaguardas existentes y su eficacia
4. Estimación del impacto y del riesgo al que están expuestos los activos.
5. Interpretación de resultados.

Las entradas para esta fase son los objetivos y los criterios de valoración identificados en la etapa de planificación.

El análisis de riesgos implica realizar las actividades y tareas siguientes:

3.4.2.1 Actividades y tareas del análisis de riesgo

- **Caracterización de los activos**
  En esta actividad se identifican los activos relevantes caracterizándolos por el tipo de activo, identificando las relaciones entre los diferentes activos, determinando en qué dimensiones de seguridad son importantes y valorando esta importancia. Todo esto para encontrar el "modelo de valor". Las tareas necesarias en esta actividad son: identificación de los activos, encontrar las dependencias entre activos y valoración de los activos.

- **Caracterización de las amenazas**
  En esta se pretende encontrar las amenazas, su frecuencia de ocurrencia y la estimación de la degradación que causaría sobre los activos. El resultado de esta actividad es un mapa de riesgos. Las tareas necesarias son: identificación de las amenazas y valoración de las amenazas.

- **Caracterización de las salvaguardas**
  Para esto se identifican las salvaguardas implementadas para mitigar el riesgo, calificándolas por su eficacia frente a las amenazas que pretenden mitigar. El resultado de esta actividad es la evaluación de las salvaguardas. Las tareas necesarias son: la identificación y valoración de las salvaguardas existentes.

- **Estimación del estado de riesgo**
  En este punto se busca estimar el impacto y el riesgo, además de la interpretación de los datos obtenidos en el análisis de riesgo.

3.4.2.1.1 Caracterización de los activos o Modelo de Valor

Hay que hallar el valor que representan los activos para la organización así como de las dependencias entre los diferentes activos (clasificar y valorar los activos relevantes) para esto se siguen los siguientes pasos.

- ***Identificación y valoración por clasificación de los activos***.

La metodología de Magerit muestra una clasificación de los activos que se puede tomar para jerarquizarlos y clasificarlos, gracias a esto se pueden lograr una clasificación y descripción inicial de los activos utilizados en transacciones B2C, ver Tabla 3.3.

Clasificación de activos por tipo

dato		**Tipo de dato**	**Descripción**
identificador	**Activo**		
[s]	**Información de acceso y financiera**	Datos clasificados como secretos	Agrupan el conjunto de datos necesarios para la realización de la transacción y que son de vital importancia para la autenticación de los usuarios.
[com]	**Certificados digitales**	Datos de interés comercial	Permiten verificar la autenticad de las partes involucradas en las transacciones.
Servicio		***Tipo de Servicio***	***Descripción***
Identificador	**Activo**		
[sc]	**Sistemas criptográficos**	Servicios asociados a la criptografía de llave pública, ejemplo: SSL, SET, etc.	Servicios asociados a la gestión de certificados y cifrado de datos
Aplicación software		***Tipo de aplicación***	***Descripción***
Identificador	**Activo**		
[browser]	**Navegador**	Estándar	Permite el acceso a sitios

			de comercio electrónico.
[av]	**Antivirus**	Estándar	Da soporte a la seguridad del navegador web y el computador del usuario de forma general
[fw]	**firewall**	Estándar	Control de entradas y salida de datos
[com]	**Certificados digitales**	Estándar	Permiten verificar la autenticad de las partes involucradas en las transacciones.

Tabla 3.3. Clasificación de los activos

La clasificación obtenida en esta parte del análisis de riesgo se realizo siguiendo el catalogo de elementos, gracias a este se pueden clasificar los distintos activos dentro de una jerarquía, que no es excluyente, para la cual los activos mas valiosos son los datos, seguidos de los servicios y por ultimo aplicaciones software, esta clasificación inicial permiten tener un valoración inicial de los activos además de brindarles identificadores. Activos como información de acceso y certificados digitales se priorizan respectivamente por el tipo de dato(los clasificados como secretos tiene mayor prioridad que los de carácter comercial) esta clasificación puede depende de la legislación aplicable en cada lugar y circunstancia o del manejo que se le de dentro de la organización.

- **Dependencias entre activos**
  Además de la definición de las dimensiones sea hace necesario encontrar las dependencias entre activos, esto es porque es común encontrar que los activos más significativos para la organización dependan de otros activos menos significativos y estas dependencias hacen que su valoración cambie o se vea impactado de otra manera por lo cual se hace importante encontrar las dependencias entre todos los activos, a continuación se exponen estas dependencias.

	[s]	**[com]**	**[browser]**	**[pki]**	**[av]**	**[fw]**
[s] información de acceso y financiera				@		
[com] certificados				@		
[browser] navegador		@		@	@	@
[av] antivirus						
[pki] sistemas criptográficos						
[fw] firewall						

Tabla 3.4. Dependencias entre activos

Estas mismas dependencias se pueden dibujar en un árbol de dependencias el cual permite una visión más clara de las mismas (ver Figura 3.2).

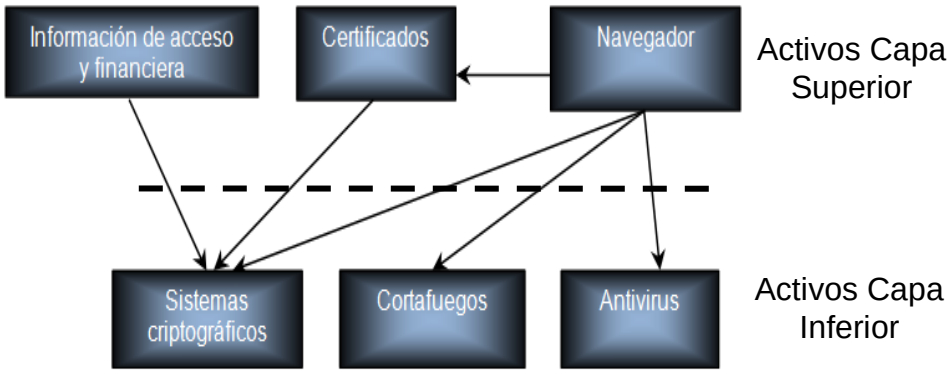


Figura 3.2. Árbol de dependencias entre activos

- **Valoración de los activos dentro de las dimensiones**
  Esta valoración es útil para identificar en cuales de las dimensiones se hacen importantes los activos o cuales de estas son impactadas por los diferentes activos, de la relación de la dimensiones con los activos también depende la valoración que se le dé al activo (Tabla 3.5).

ACTIVOS	DIMENSIONES				
	Privacidad o confidencialidad	Integridad	Autenticidad	No repudio	Valor
Información de acceso y financiera (Número cuenta, clave tarjeta, Contraseñas, etc.)	@		@		10
Certificados digitales		@	@		10
Navegador	@	@	@		8
Antivirus		@			7
Sistemas criptográficos	@	@	@	@	4
Firewall	@		@		7

Tabla 3.5. Valoración de los activos dentro de las dimensiones de valoración

- ***Valoración por frecuencia de uso***

Activos	*Uso*
Información de acceso y financiera (Número cuenta, clave tarjeta, Contraseñas, etc.)	***54.5%***
Certificados digitales	***2.3%***
Navegador	***100%***
Antivirus	***97.2%***
Sistemas criptográficos	***14.9%***
firewall	***80.3%***

Tabla 3.6. Clasificación de los activos por frecuencia de uso

La Tabla 3.6, muestra la clasificación por frecuencia de uso, y aunque no se hacen valoraciones de este tipo, esta clasificación puede ser algo concluyen puesto que esta información también permite tener una idea del porcentaje de exposición a las vulnerabilidades, los datos fueron obtenidos en un 66.66% de INTECO, y el 33.33% restante tomado de forma cualitativa.

3.4.2.1.2 Caracterización de las amenazas o Mapa de riesgo

Como ya se dijo el objetivo de esta etapa es realizar un mapa de riesgo el cual se puede representar por medio de la siguiente tabla (Tabla 3.7) en la cual se hace una relación entre los activos, amenazas y vulnerabilidades.

Activos	Amenazas	Vulnerabilidades
Antivirus	Nuevos virus	Necesidad de actualizaciones y parches
	Ataques del día cero[2]	Tipo de Protección reactiva
Navegador web	Virus en general	Configuración por defecto
	Cookies	Configuración por defecto del navegador
	Troyanos + Keyloggers	Teclados virtuales mas Falta de soporte ankeyloggers +teclados físicos +falta de soporte

[2] Ataques que se realizan desde el día de aparición de un nuevo código malicioso hasta la aparición de la firma para detectarlo.

	Phishing Pharming y similares Conexiones a sitios sin utilizar cifrado o cifrado deficiente Alojamiento de Scripts Troyanos	antivirus Navegadores sin soporte antiphishing. Autonomía del navegador para permitir conexiones sin cifrar. Recepción de scripts sin certificados De Autenticidad
Firewall	Virus en general	Configuración por defecto o muy permisivas
Certificados digitales	Suplantación de identidad.	Falta de verificación por parte de los usuarios
Sistemas criptográficos	ataques para descifrar la información	Longitud de las claves utilizadas para generar contraseñas, certificados, etc.
Información de acceso y financiera(Claves y contraseñas)	Virus en general Obtención fraudulenta Keyloggers + troyanos	Almacenamiento sin cifrar Falta de soporte antiphishing +falta de antivirus Falta de soporte antikeyloggers y antivirus

Tabla 3.7. Mapa de riesgos, amenazas y vulnerabilidades por activo

3.4.2.1.3 Caracterización de las salvaguardas

En esta etapa inicialmente se deben identificar las salvaguardas ya implementadas y su efectiva. Al ser este un primer análisis de riesgo no se tienen elementos necesarios para realizar esta etapa por lo cual se debe realizar mas adelante cuando se tenga implementadas las salvaguardas y se haya liberado el modulo software para comprobar su efectividad.

3.4.2.1.4 Caracterización del estado de riesgo

Una vez se determina que una amenaza puede afectar un activo hay que estimar el estado de riesgo, las tareas realizadas en esta etapa son: la estimación del impacto y la estimación del riesgo. Los productos de entrada son la caracterización de los activos, la caracterización de las amenazas y caracterización de las salvaguardas.
Objetivos: determinar el impacto potencial y el impacto residual al que está expuesto el sistema.

Formulas utilizadas para determinar el impacto y riesgo:

Impacto = Valor * Degradación.
Riesgo = Impacto * Frecuencia.

De donde frecuencia y degradación se definen como.

La frecuencia: cada cuanto se va a materializar la amenaza.
La degradación: cuanto se perjudica el activo con la materialización de la amenaza.

- **Calculo del impacto económico**:
  Realizar un análisis por tabla permite realizar una análisis cuantitativo del impacto económico y aunque esta misma aproximación pudo haberse tomar para calcular la probabilidad de ocurrencia se opto por realizar en esta parte un análisis cuantitativo gracias a algunos valores brindados por INTECO [15][16][17].

Activos	Amenazas	Degradación del activo	Valor	Impacto económico
Antivirus	Nuevos virus	10	7	70
	Ataques del día cero	10	7	70
Navegador Web	Virus en general	7	8	56
	Cookies	1	8	8
	Keyloggers + troyanos	10	8	80
	Phishing, pharming y similares.	10	8	80
	Conexiones a sitios sin utilizar cifrado o cifrado deficiente	10	8	80
	Código activo	3	8	24

Firewall	Virus en general	10	7	70
Certificados Digitales	Suplantación de identidad	10	10	10
Sistemas criptográficos	ataques para descifrar la información	3	4	12
Información de acceso y financiera(Claves y contraseñas)	Virus en general	10	10	20
	Obtención fraudulenta	10	10	20
	Keyloggers + troyanos	10	10	20

Tabla 3.8. Degradación de los activos y cálculo del impacto

- **Cálculo del riesgo**

  Para hallar el valor de la Medición del Riesgo, se debe multiplicar el impacto por la probabilidad de ocurrencia, con esto se tendrán valorados los activos (Tabla 3.9)

Activos	Amenazas	Vulnerabilidades	Impacto económico	Probabilidad de ocurrencia	Medición del Riesgo
Antivirus	Nuevos virus	Necesidad de actualizaciones y parches	70	0.359	25.13
	Ataques del día cero	Tipo de Protección reactiva	70	0.690	48.3
Navegador Web	Virus en general	Configuración por defecto	56	0.086	4.816
	Cookies	Configuración por defecto	8	1	8
	Keyloggers + troyanos	Teclados virtuales mas Falta de soporte antikeyloggers teclados físicos +falta de soporte antivirus	80	0.527	42.16

	Phishing, pharming y similares.	Navegadores sin soporte antiphishing.	80	0.299	23.92
	Conexiones a sitios sin utilizar cifrado o cifrado deficiente	Autonomía del navegador para permitir conexiones sin cifrar.	80	0.734	58.72
	Código activo	Configuración del navegador para aceptar controles activeX, plug-ins, java script, Visual Basic script, java applets.	24	0.3	7.2
Firewall	Virus en general	Configuración por defecto o muy permisivas	70	0.086	6.02
Certificados digitales	Suplantación de identidad	Falta de verificación por parte de los usuarios	10	0.760	7.60
Sistemas criptográficos	Ataques para descifrar la información	Longitud de las claves utilizadas para generar contraseñas y algoritmos de cifrado deficientes.	12	0.5	6
Información de acceso y financiera(Claves y contraseñas)	Virus en general	Almacenamiento sin cifrar	20	0.086	1.72
	Obtención fraudulenta	Falta de soporte antiphishing	20	0.146	2.92
	Keyloggers + troyanos	Falta de soporte antikeyloggers y antivirus	20	0.527	10.54

Tabla 3.9. Calculo del riesgo

- **Interpretación de los resultados**

En la interpretación de los resultados se busca establecer relaciones de prioridad por activos o grupos de activos, ya sea por orden de impacto o por orden de riesgo. En este caso se mide el riesgo por activos, teniendo en cuenta las dependencias entre cada uno de ellos, para obtener el valor acumulado del riesgo (Tabla 3.10).

Activo: Certificados Digitales		
Amenazas	**Priorización**	**Valor acumulado**
Suplantación de identidad.	1	7.6
Activo: Navegador Web		
Amenazas	**Priorización**	**Valor acumulado**
Conexiones a sitios sin utilizar cifrado o cifrado deficiente.	1	58.72
Keyloggers + troyanos	2	42.16
Phishing, pharming y similares	3	23.92
Cookies	4	8
Activo: Información de acceso y financiera (Claves y contraseñas)		
Amenazas	**Priorización**	**Valor acumulado**
Keyloggers + troyanos	1	10.54
Obtención fraudulenta	2	2.92
Virus en general	3	1.72
Activo: Sistemas Criptográficos		
Amenazas	**Priorización**	**Valor acumulado**
Ataques para descifrar la información	1	6
Activo: Antivirus		
Amenazas	**Priorización**	**Valor acumulado**
Ataques del día cero	1	48.3
Nuevos virus	2	25.13
Activo: Firewall		
Amenazas	**Priorización**	**Valor acumulado**
Virus en general	1	6.02

Tabla 3.10. Priorización de las amenazas por activos

Esta tabla representa en valor acumulado del riesgo y se a priorizado teniendo en cuenta las dependencias entre los activos.

3.5 Gestión del riesgo

El análisis de riesgos determina impactos y riesgos, el impacto refleja el daño posible, mientras que el riesgo se refleja en el daño probable. Si el impacto y el riesgo residual son altos, entonces hay que hacer algo, de lo contrario no.

Interpretación de los valores de impacto y riesgo residual.
Estos valores son una medida o métrica entre la inseguridad a la que estamos expuesto cuando no se aplica ningún tipo de salvaguarda y las medidas necesarias que reducirán estos valores a cantidades despreciables, son medidas o métricas de la carencia de seguridad. A continuación se listan algunas situaciones, a tener en cuenta, a la hora de interpretar los valores de impacto y riesgo residual.

- Si el impacto es igual al riesgo residual; la salvaguarda elegida no tiene ningún efecto, tal vez se deba a que no se han tenido elementos en cuenta o no es la apropiada.
- Si el valor del riesgo residual es despreciable se puede decir que la salvaguarda es efectiva y se puede afrontar el problema con cierta confianza.
- Si el valor del riesgo residual tiene un nivel superior a despreciable, entonces decimos que existe cierta exposición al riesgo.
- Para una correcta interpretación del riesgo residual, esta, debe tener una relación entre lo que se debería hacer y se ha hecho.

Selección de las salvaguardas.
Mientras la interpretación de los valores de impacto y riesgo residual vislumbre la necesidad de una salvaguarda, en la práctica, podemos desarrollar políticas y recomendaciones, todo esto con el fin de verificar que todas las amenazas hayan sido tratadas y estén siendo controladas [10].

Las salvaguardas permiten hacer frente a las amenazas. Son muchos los aspectos en los cuales puede actuar una salvaguarda para alcanzar sus objetivos de limitación y mitigación del riesgo. Esos aspectos, que a su vez se presentan en transacciones de comercio electrónico, son:

Procedimientos: que siempre son necesarios, a veces bastan procedimientos, pero otras veces los procedimientos son un componente de una salvaguarda más compleja. Se requieren procedimientos tanto para la operación de las salvaguardas preventivas como para la gestión de incidencias y la recuperación tras las mismas. Los procedimientos deben cubrir aspectos tan diversos como el desarrollo de sistemas, la configuración del equipamiento, etc.

Política de personal: que es necesaria cuando se consideran este tipo de sistemas para realizar transacciones. La política de personal debe crear conciencia de uso de mecanismos de seguridad, en las personas que utilizan este medio.

Soluciones técnicas, frecuentes en el entorno de las tecnologías de la información, que pueden ser:

[SW] aplicaciones (software)
[SE] Servicios (ejemplo: Servicios SSL, servicios criptográficos)

[HW] dispositivos físicos
[COM] protección de las comunicaciones
[FIS] seguridad física, de los locales y áreas de trabajo

La protección integral de un sistema de información requerirá una combinación de salvaguardas de los diferentes aspectos comentados, debiendo la solución final:

- Estar equilibrada en los diferentes aspectos
- Tener en cuenta las salvaguardas adecuadas a cada tipo de activos
- Tener en cuenta las salvaguardas adecuadas a la dimensión de valor del activo
- Tener en cuenta las salvaguardas adecuadas a la amenaza a conjurar

Las salvaguardas, especialmente las soluciones técnicas, varían con el avance tecnológico, ya que:

- Aparecen tecnologías nuevas
- Van desapareciendo tecnologías antiguas
- Cambian los tipos de activos a considerar
- Evolucionan las posibilidades de los atacantes
- Evoluciona el catálogo de salvaguardas disponibles.

ANEXO D- IMPLEMENTACION EN XUL

4 XUL

En la actualidad es muy común encontrar un sin número de extensiones o sobre posiciones para Mozilla Firefox, estas extensiones incluyen desde temas nuevos para el navegador hasta sobre posiciones de seguridad. No es común encontrar extensiones que exploten las ventajas de C++ mediante la creación de un Componente XPCOM a pesar que estas construcciones tienen la capacidad de alcanzar un mayor número de recursos del sistema sin limitarse solo a la funcionalidad que permite el Framework de Mozilla.

Gracias a el lenguaje de programación XUL[3], que es un lenguaje XML[4] [18] con todas sus características y ventajas y que sigue estándares del W3C.[5] como: XML, HTML[6] [19], CSS[7] [20], DOM[8] [21], etc. se pueden desarrollar aplicaciones ricas en todo lo que se refiere a definición de interfaces, personalizables, multiplataforma y multiplicación.

4.1 Extensiones o sobre posiciones

Una extensión en XUL permite, como su nombre lo dice, extender vía XUL y java Script el comportamiento del navegador Firefox o cualquier otra aplicación desarrollada con el Framework de Mozilla(Thunderbird, seamonkey, bugzilla, etc.), cuando se desarrollan extensiones para Firefox, por lo general, estas están orientadas hacia la construcción de barras de herramientas, skins[9] para modificar su apariencia, menús, etc. la mayoría de las extensiones se quedan cortas en cuanto al uso de componentes XPCOM que pueden mejor su funcionalidad, seguridad y más.

4.1.1 ¿Por qué utilizar una extensión?

Una extensión no solo cambia la apariencia del navegador, también permite adicionarle nuevos elementos, tales como: barras de herramientas, botones, paneles lateral, etc. Todo esto se puede lograr a través del nivel de interface de usuario (nivel de Scripting 1), localizando el identificador del elemento XUL y adicionando nuestra extensión a este elemento, además de permitir ampliar el navegador de forma gráfica se puede agregar funcionalidad extra a través de objetos Java Script, estos objetos incluyen desde objetos disponibles para servicios web como SOAP[10], navegación, manejo de eventos, etc. y la forma de acceder a ellos es por medio de constructores

[3] XUL (XML User-interface Language, Lenguaje de interface de usuario), es un lenguaje multiplataforma para describir interfaces de usuarios.

[4] XML (Extensible Markup Languaje), es un formato de texto.

[5] (W3C)World Wide Web Consortium, principal organización internacional que produce estándares para la Word Wide Web

[6] HTML (HiperText Markup Languaje)

[7] CSS (Cascading Style Sheet) Lenguaje para hojas de estilo que permite adicionar estilos a la estructura del documento.

[8] DOM (Document Object Model), es una API para acceder documentos HTML Y XML a través de Java Script.

[9] Conjunto de hojas de estilo(archivo con información de tamaño de los elementos, colores bordes, etc.), imágenes y comportamiento que se aplica a documentos XUL.

[10] Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse

java script u otro objeto, otra forma de agregar mas funcionalidad es a través de componentes e interfaces XPCOM accesibles desde Java Script gracias a la interfaces XPCONNECT[11], estos componentes e interfaces están organizados en paquetes tales como: desarrollo de aplicaciones, red, mail, DOM, seguridad y privacidad, etc. Todos estos objetos, componentes e interfaces hacen del desarrollo de extensiones una forma completa para desarrollar aplicaciones multiplataforma de acuerdo a la necesidad que se tenga [22].

4.1.2 ¿Cómo extender el navegador a través de una extensión o sobre posición?

Para explicar la forma de extender el navegador Mozilla Firefox por medio de una extensión, vamos a abordar la creación de la sobre posición utilizada en el proyecto: esto incluye la creación de la jerarquía de directorios, los documentos XUL necesarios, la forma de crear, compilar y registrar el componente XPCOM que luego será utilizado por la extensión y como empaquetar todo para que su instalación se pueda realizar de una forma rápida y segura desde el navegador.

4.1.3 Jerarquía de directorios.

La estructura básica que toda extensión debe contener y que es llamada proveedor del Chrome [23], es la siguiente.

Content: aquí se encuentran los documentos XUL, que definen el diseño de la interface y los archivos Java Script para proveer la funcionalidad y acceso a componentes XPCOM.

Skin: contiene las hojas de estilos (CSS) y las imágenes, las cuales definen la apariencia de la interfaz.

Locale: los documentos DTD [24] se encuentra aquí, estos contiene las cadenas de texto para la aplicación en un idioma determinado.

Para poder utilizar cualquier extensión hecha con XUL, esta se debe registrar mediante el servicio de registro del Chrome el cual se encarga de relacionar la dirección virtual de los paquetes Chrome con su localización física, esto es debido a que las aplicaciones en Mozilla no acceden a sus elementos mediante rutas de ficheros del sistema operativo, si no con directorios virtuales de Mozilla [25] (las rutas Chrome). Para informarle a este servicio de un nuevo Chrome, se utiliza el Chrome.manifiest, el cual es un fichero de texto que contiene una línea por cada proveedor que queremos registrar (content, local o skin), o por cada nueva sobre posición.

[11] XPCONNECT, capa intermedia entre Java Script y XPCOM que permite traducir los objetos XPCOM en objetos Java Script.

```
1) content modulob2c          jar:chrome/modulob2c.jar!/content/
2) locale  modulob2c  es-ES   jar:chrome/modulob2c.jar!/locale/es-ES/
3) skin    modulob2c  classic/1.0     jar:chrome/modulob2c.jar!/skin/
4) overlay chrome://browser/content/browser.xul     chrome://modulob2c/content/Overlay.xul
```

FIGURA 4.1. ARCHIVO CHROME.MANIFIEST DE LA EXTENSIÓN

Las líneas 1, 2 y 3 de la Figura 4.1 contienen tres elementos por línea.

- El tipo de proveedor (content, locale o skin) y el tipo de empaquetado, en nuestro caso es dentro de un archivo .jar [26].
- El nombre del paquete, modulob2c.
- La dirección física de los ficheros del proveedor.

La línea 4 se utiliza para registrar overlays, con esto se extiende una interface grafica preexistente, los elementos dentro de esta línea son; primero la interface grafica que se quiere extender, chrome://browser/content/browser.xul, la cual hace referencia a la interface grafica del navegador Firefox, y por último el nombre del paquete que contiene la interface que se va adicionar, Chrome://modulob2c/content/Overlay.xul.

Otro archivo importante es el install.rdf, cuya extensión RDF viene de Resource Description Manager [27], este archivo mostrado en la figura 4.2 sirve para describir la extensión. Dentro de este se define una etiqueta con la descripción de la extensión: el identificador de la extensión, su nombre, versión, quienes la crearon, y la aplicación objetivo (dentro de la suite Mozilla[12]) con su identificador, versiones mínima y máxima soportada.

```
<?xml version="1.0" encoding="UTF-8"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:em="http://www.mozilla.org/2004/em-rdf#">
  <Description about="urn:mozilla:install-manifest">
    <em:id>modulo@politicas.com</em:id>
    <em:name>SECURITY TRANSATION B2C</em:name>
    <em:version>1.0</em:version>
    <em:creator>Freddy Mina-Erwin Daza</em:creator>
    <em:description>Extension para transacciones B2C.</em:description>
    <em:targetApplication>
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}
         </em:id> <!-- firefox -->
        <em:minVersion>1.5</em:minVersion>
        <em:maxVersion>3.1.*</em:maxVersion>
      </Description>
    </em:targetApplication>
  </Description>
</RDF>
```

Figura 4.2. Archivo install.rdf con información de la extensión

[12] La suite Mozilla hace referencia a todas las aplicaciones creadas con el Frameword de Mozilla (Thunderbird, seamonkey, Firefox, etc.)

4.1.4 Documentos XUL necesarios

Para la extensión necesitamos dos documentos XUL; el overlay (Overlay.xul) que es el documento que contiene información de la sobre posición referente a que elementos se van cargar, y que documentos xul hacen parte de la sobre posición, y el documento con la interface de usuario que extenderá el navegador, (página lateral.xul).

4.1.4.1 Estructura del Overlay

Este contiene los siguientes elementos, la definición del menú (etiqueta menupopup [28]), atajos de teclado (etiqueta Keyset **¡Error! No se encuentra el origen de la eferencia.**) y el propagador de eventos (etiqueta broadcaster **¡Error! No se encuentra el origen de la referencia.**). Los cuales son los elementos clave que debe tener el overlay y que se muestran a continuación en la Figura 4.3.

```
<?xml version="1.0"?>
<!DOCTYPE overlay SYSTEM "chrome://modulob2c/locale/pagLateral.dtd">
<overlay id="pagLateralOverlay"
 xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

  <menupopup id="viewSidebarMenu">
    <menuitem key="key_openpagLateral" observes="viewEmptySidebar"/>
  </menupopup>

  <keyset id="mainKeyset">
    <key id="key_openpagLateral" command="viewEmptySidebar"
         key="&openpagLateral.commandkey;"
         modifiers="&openpagLateral.modifierskey;" />
  </keyset>

  <broadcasterset id="mainBroadcasterSet">
    <broadcaster id="viewEmptySidebar"
                 label="&pagLateral.titulo;"
                 autoCheck="false"
                 type="checkbox"
                 group="sidebar"
                 sidebarurl="chrome://modulob2c/content/pagLateral.xul"
                 sidebartitle="&pagLateral.titulo;"
                 oncommand="toggleSidebar('viewEmptySidebar')" />
  </broadcasterset>

</overlay>
```

Figura 4.3. Estructura del overlay

Por ser XUL un documento xml debe empezar con la declaración estándar para este tipo de documentos, la primera línea del código de la Figura 4.3 es utilizada para dicho objetivo, esta línea es obligatoria, la segunda hace referencia a la etiqueta overlay con el atributo xmlns referenciando el espacio de nombres[13] para trabajar con elementos xul y el id "pagLateralOverlay" con el cual se puede hacer referencia al overlay por medio del DOM para modificar sus atributos desde otros documentos XML.

En la etiqueta menupopup, se ha escogido para el atributo id el valor "viewSidebarMenu", este valor hace referencia en el navegador (Mozilla Firefox) al menú emergente del Panel Lateral, este valor podría haber sido cualquiera dependiendo de en que parte de la ventana del navegador se quiera ubicar nuestra sobre posición. Una de las herramientas útiles a la hora de decidir esta ubicación es DomExplorer , esta herramienta es una extensión que permite ver y navegar el árbol

[13] Indica los elementos que se reconocen como validos a la hora de interpretar el archivo xul.

de documentos de cualquier documento XUL la cual es una forma sencilla de identificar los elementos y atributos en este tipo de documentos.

4.1.4.2 Estructura de interface de usuario o Página Lateral.

Por medio de este documento o interface se podrá visualizar la información relacionada con la verificación de las políticas, recomendaciones por transacción, recomendaciones generales e información que el componente XPCOM nos pueda brindar del Firewall [31] y del software antivirus instalado. El documento incluye, ver Figura 4.4: una referencia al DTD (pagLateral.dtd) línea 2, utilizado para el idioma, la etiqueta Windows, línea 4, para la creación de la ventana que contendrá; las pestañas de la extensión (etiquetas tabs, líneas desde la 6 hasta la 9), la función IniciarModulo() para inicializar el modulo cuando la ventana sea descargada, la función unloadModulo para liberar el listener de la extensión cuando la pagina sea cerrada y los espacios de nombre para poder utilizar elementos html y elementos xul(xmlns:html=http://www.w3.org/1999/xhtml y xmlns=http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul respectivamente). La etiqueta tabpanels, línea 10, contiene en su interior dos etiquetas panels, líneas 11 y 13, en el cual se colocara respectivamente el código para la verificación de políticas y el despliegue de las recomendaciones generales y por transacción.

```
1) <?xml version="1.0"?>
2) <!DOCTYPE overlay SYSTEM
   "chrome://modulob2c/locale/pagLateral.dtd">
3)<?xml-stylesheet href="chrome://global/skin/" type"text/css" ?>

4) <window id="sbEmptySidebar"
   xmlns:html="http://www.w3.org/1999/xhtml"
   onload="IniciarModulo();" unload="unloadModulo();"
   xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only
   .xul" >

5)<tabbox orient="horinzontal" flex="1">
6)     <tabs orient="horizontal">
7)         <tab label="&pestanaPoliticas.titulo;"/>
8)         <tab label="&pestanaRecomendaciones.titulo;"/>
9)     </tabs>

10)     <tabpanels flex="1">
11)         <tabpanel id="pag_poli" orient="vertical"  flex="1">

12)         </tabpanel>

13)         <tabpanel id="pag_recom">

14)         </tabpanel>
15)     </tabpanels>

16)</tabbox>

17)</window>
```

Figura 4.4. Estructura de la página lateral

Figura 4.5. Sobre posición realizada gracias al archivo Overlay.xul

Gracias al archivo Overlay.xul y al identificador utilizado en esta, viewSidebarMenu, que hace referencia al menú emergente del panel lateral se puede agregar una entrada en este, Figura 4.5, para desplegar la interface de la extensión donde se verificara las políticas, ver Figura 4.6.

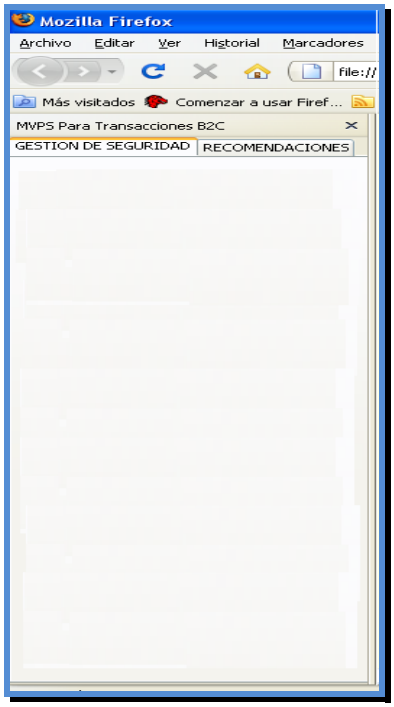


Figura 4.6. Interface inicial de la extensión, archivo pagLateral.xul

La Figura 4.6 contiene dos pestañas en las cuales se ubicara respectivamente la información de las políticas a verificar (gestión de seguridad) y las recomendaciones generales y por transacción (pestaña recomendaciones).

La siguiente figura, Figura 4.7, muestra los elementos necesarios en la interface del usuario para desplegar la información referente a la política encarga de verificar la autenticidad del sitio.

```
1) <groupbox flex="1" id="groupBoxActualNav">
2)    <caption label="&groupBoxActualNav.titulo;"/>

3)    <hbox orient="horizontal">
4)         <radio orient="horizontal" label="&radLabelBajo.label;"
            selected="false" id="rbt_actNavegbad"/>

5)         <radio orient="horizontal" label="&radLabelMedio.label;"
            selected="false" id="rbt_actNavegmid"/>

6)         <radio orient="horizontal" label="&radLabelAlto.label;"
            selected="false" id="rbt_actNaveghight"/>
7)    </hbox>

8)    <hbox orient="horizontal">
9)         <button label="Ver Detalles" id="btn-DetallesAS"
           popup="NavegadorVersion"/>
10)   </hbox>

11)   <popupset>
12)          <popup id="NavegadorVersion" position="after_start"
              onpopupshowing="CrearDetallesNivelActuNav();"
              onpopuphidden="BorrarDetallesNivelActuNav();">
13)          </popup>
14)       </popupset>

15)   </groupbox>
```

Figura 4.7. Código necesario para verificar la actualización del navegador

Los elementos utilizados para verificar cada política (Figura 4.7) son un conjunto de tres radio buttons y un button, estos tres radio buttons (líneas 4,5 y 6) representan los siguientes estados; bajo o inseguro, medio y alto, dependiendo del nivel de cumplimiento de la política, por otro lado el button será utilizado para desplegar información referente a la verificación de la política (línea 9). La información desplegada desde el button será diferente en todos los casos, por esto, se adiciona un menú desplegable, etiqueta popup(línea 12), que se encargar de adicionar los ítem al menú dependiendo del estado de cada política.

Para controlar la posición de los elementos o layout dentro de la pagina lateral se utilizan las etiquetas groupbox(elemento contenedor de las cajas), hbox(caja en posición horizontal) y vbox(caja en posición vertical), gracias a estos se puede dividir una ventana en cajas y posicionar los elementos dentro de estas de forma horizontal o vertical **¡Error! No se encuentra el origen de la referencia.**.

Adicionando este código a nuestra interface, archivo pagLateral.xul, esta se modifica de la siguiente manera, Figura 4.8.

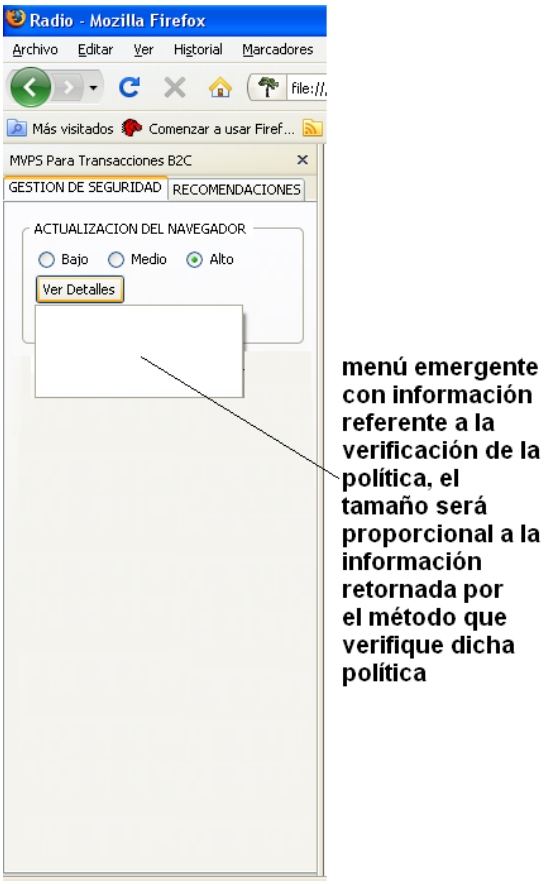


Figura 4.8. Interface con los elementos necesarios para la verificación de la política referente a la actualización del navegador

4.1.4.3 El componente principal de la extensión

El componente principal, de la herramienta MVPS, es nsIWebProgressListener, este componente puede ser implementado por los clientes desde código javaScript, nivel de aplicación **¡Error! No se encuentra el origen de la referencia.**. Ahora desde el ivel de aplicación y tomando las políticas escogidas para implementar, tres de ellas pueden ser desarrolladas con componentes del Framework de Mozilla, esto es gracias a que el Framework exponen más de 960 componentes y mas 1300 interfaces para acceder a ellos **¡Error! No se encuentra el origen de la referencia.**, el código ecesario para esto es el siguiente, ver Figura 4.9.

```
1)  var EventListenerChange_Url_Security = {
2)          stateIsRequest:false,
3)          QueryInterface : function(allD) {
4)          if(allD.equals(Components.interfaces.nsIWebProgress
5)          Listener) ||
6)          allD.equals(Components.interfaces.nsISupportsWeak
7)          Reference) ||
8)          allD.equals(Components.interfaces.nsISupports))
9)                  return this;
10)                 throw Components.results.NS_NOINTERFACE;
11)         },
12)  onLocationChange
13) :function(aProgress,aRequest,aLocation){
14)          VerificarCertificado(aLocation);
15)          NivelTransaccion();
16)          AbrirRecomTrans();
17)         },
18)  onSecurityChange : function(aProgres,aRequest,aStatus){
19)          VerificarCertificado(aLocation);
20)          VerificarNivelConexion(aStatus);
21)          NivelTransaccion();
22)          AbrirRecomTrans();
23)          return 0;
             }
      };
```

Figura 4.9. Componente principal de la extensión

Lo que se hizo en el código anterior, fue crear un objeto del tipo nsIWebProgressListener , en este objeto se redefinen las funciones:

- QueryInterface (línea 3 a la 10): que es un método de la interface base nsISupports y que se utiliza para descubrir interfaces y controlar el número de interfaces creadas.

- onLocationChange (línea 11): método de nsIWebProgressListener que se utiliza como observador de la barra de direcciones, sus parámetros; la interface aProgress, que es el objeto que envía la notificación de cambio, aRequest interface con información de la descarga) y aLocation la cual es la URI [35] de la pagina solicitada.

- onSecurityChange (línea 17): que es otro método de nsIWebProgressListener, el cual se dispara cuando hay transiciones de seguridad (http->https, https->http), sus parámetros; aProgress, aRequest (que en este caso guarda información del nuevo estado de seguridad) y aStatus(el cual es un valor compuesto de banderas del estado de seguridad y banderas de Seguridad Strength [36][36]).

Gracias a este objeto se pueden llamar funciones como; VerificarCertificado (utilizada para implementar la política referente a verificación de la autenticidad del sitio) y VerificarNivelConexion (utilizada para implementar la política referente a conexión con el sitio). Con los parámetros adecuados se puede recobrar el certificado digital de la URL solicitada (aLocation) y recobrar el valor de la bandera de estado de seguridad de la conexión o seguridad Strength (aStatus), los valores posibles para aStatus pueden ser:

- Valores del estado de seguridad: estado de conexión inseguro = 4, estado de conexión desconocido = 1, estado de conexión seguro = 2.
- Valores de Seguridad Strength: estado de conexión seguro alto = 262144, estado de conexión seguro medio = 65536, estado de conexión seguro bajo = 131072, certificados extended validation [37] = 1310722 o 1048576.

Funciones como NivelTransaccion y AbrirRecomTrans, son funciones del nivel uno de scripting, estas permiten en primera instancia mostrar el nivel de seguridad de la transacción dependiendo del cumplimiento de las políticas, y en segunda instancia desplegar un conjunto de recomendaciones de acuerdo al número de políticas no cumplidas y su nivel de cumplimiento.

La función que permite verificar el nivel de actualización del navegador es menos complicada, solo requiere de la utilización de algunos objetos del navegador para encontrar información referente a la versión del geecko [38], la cual es útil cuando se verifica con información de vulnerabilidades y cuyo resultado puede servir para valorar el nivel de actualización del navegador.

Gracias a los componentes nombrados hasta aquí se logra implementar la herramienta en un 70%, faltando por resolver la parte del soporte del equipo (uso de antivirus y firewall habilitado) a continuación se muestra la forma de implementar la funcionalidad restante.

ANEXO E- IMPLEMENTACION DEL COMPONENTE XPCOM

5 Cross-platform Component Object Model (XPCOM)

XPCOM es el modelo de objetos que Mozilla utiliza y junto a Java Script permiten el acceso a los componentes de Framework, estos componentes son accedidos a través de interfaces, en Mozilla el concepto de interface se define como el conjunto de funcionalidad que debe implementar el componente. Los componentes XPCOM son implementados generalmente en código nativo lo cual significa que pueden hacerse muchas mas cosas que las que se pueden hacer con código Java Script.

5.1.1 Como crear instancias de componentes desde código java Script

Los componentes dentro del Framework de Mozilla son creados utilizando los patrones de diseño Singleton (servicios) y factory [39] (factoría de clases) y esto determina la forma como son creados o utilizados los componentes, ejemplo:

var file = Components.classes["@mozilla.org/file/local;1"].createInstance();

En el ejemplo, Components es el objeto java script que controla la conexión a los componentes, classes es una array con todas las clases accesibles por medio de su Contract ID, ***@mozilla.org/file/local;1***, en Mozilla este identificador empieza con el signo @ y define el dominio del componente. file en este caso representa el modulo al cual pertenece y local;1 el tipo y la versión del componente. Si el componente se ha implementado con el patrón factory se utiliza la función createInstace de lo contrario se debe utilizar getService El resultado sería un objeto del tipo especificado en el contract ID desde el cual se pueden llamar todas las funciones definidas en la interface que implementa el componente.

5.2 Interface base

La interface principal en el framework XPCOM, es nsISupports, esta es la interface de la cual todos los componentes XPCOM deben heredan, gracias a esta interface se puede descubrir las interfaces que un determinado componente implementa y se gestiona el tiempo de vida del mismo. Los métodos de la interface son los siguientes:

5.2.1 Métodos de la interface nsISupports

- **QueryInterface:** Esta permite preguntar si un componente implementa una interface dada.

  Sintaxis: nsresult QueryInterface (const nsIID & uuid, void **result);

  Parámetros:
  - uuid o identificador de la interface solicitada.
  - result: es un parámetro de salida, este debe retornar la referencia de la interface solicitada.
- **AddRef:** Incrementar la variable refcount de la interface, esta variable es utilizada para controlar el numero de interfaces creadas, cuando esta variable llega a cero el componente se destruye.

- **Release:** Decrementa la variable refcount de la interface.

Existe un grupo de macros [40] que hacen que la implementación de esta interface se haga de forma transparente ahorrando tiempo de desarrollo y líneas de código. Estas macros son las siguientes.

5.2.2 Macros utilizadas para la implementación de nsISupports

- ***NS_DECL_ISUPPORTS:*** Esta macro es útil ya que implementa para nosotros funciones como AddRef y Release, útiles a la hora de gestionar el número de interfaces creadas, y QueryInterface necesaria para descubrir interfaces.

  Sintaxis:
  NS_DECL_ISUPPORTS

- ***NS_IMPL_ISUPPORTSn(clase, interface1,.., interfacen):*** Esta macro permite implementar la interfaces nsISupports, es útil ya que todo componente debe implementar esta interfaces y la macro hace esto por nosotros. La letra n al final hace referencia al número de interfaces que expone nuestro componente, el parámetro "clase" se refiere a la clase que debe implementar la interface nsISupport e interfacei a las interfaces que también deben implementarla.

  Sintaxis:
  NS_IMPL_ISUPPORTS1 (SoporteEquipo, ISoporteEquipo)

- ***NS_INIT_ISUPPORTS ():*** Esta macro implementa el constructor de la interface nsISupports.

  Sintaxis:
  NS_INIT_ISUPPORTS ();

5.3 NSGetModule

Otra cosa importante que hay que implementar es el código necesario para acceder el componente, importación del método NSGetModule, pero de la misma forma como existen macros para implementar la interface nsISupports lo hay para crear este punto de acceso. Para esto se requieren las interfaces nsIModule y nsIFactory, las macros utilizadas para esto son las siguientes:

- ***NS_IMPL_NSGETMODULE (clase, components):*** Esta Macro permite implementar la interface nsIModule, esta es utilizada para registrar de forma automática componentes, sus parámetros son el nombre del componente y una lista que incluye; el classname del componente, el identificador de la clase y el identificador del componente. Para poder utilizar la macro se debe incluir el archivo de cabecera nsImodulo.h, ligadas a esta macro están;

NS_IMPL_NSGETMODULE_WITH_CTOR (clase, components, ctor), *esta macro es similar a la anterior solo que permite el llamado de una función cuando el componente es creado,* ***NS_IMPL_NSGETMODULE_WITH_DTOR(clase, components, ctor),*** *igual que la primera función, solo que aquí, se permite el llamado de una función cuando el componente es destruido,* ***NS_IMPL_NSGETMODULE_WITH_CTOR_DTOR(clase, components, ctor, dtor),*** *este macro es una mezcla de las dos ultimas macros al permitir el llamado de una función cuando se crea y se destruye el componente.*

- ***NS_GENERIC_FACTORY_CONSTRUCTOR (clase):*** Esta macro implementar el patrón factoría, cuyo objetivo es ocultar la inicialización e implementación del componente a los clientes, gracias a esto se puede instanciar un componente u obtener una referencia a este solo con conocer su Contract ID o ClassID.

5.4 Creación de componentes XPCOM desde C++

La mayoría de los componentes XPCOM desarrollados se implementan en C++ y se compilan dentro de una librería compartida [41], DLL para sistemas Windows o DSO para sistemas UNIX, en la siguiente gráfica, Figura 5.1, podemos ver un componente desde el punto de vista del Framework.

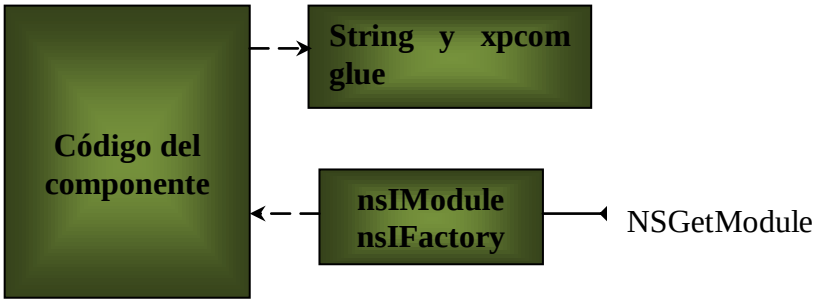


Figura 5.1. Un componente en el Framework XPCOM

Cuando se construye un componente y se compila dentro de una librería compartida este debe exportar un método, NSGetModule, este método es el punto de acceso a la interface: el método es llamado durante el proceso de registro y des registro, al igual que cuando se quiere averiguar que interfaces implementa el componente. Las interfaces nsIModule y nsIFactory son las encargadas de controlar el registro y creación del componente mientras que String y xpcom glue se utiliza para implementaciones de strings y punteros inteligentes los cuales son útiles para alcanzar interfaces y componentes que son de uso exclusivo del Framework o que no esta en estado congelado, unfrozen[14].

[14] Se refiere a aquellos componentes que no se han liberado o son de uso exclusivo del Framework y por tal motivo su utilización requiere de cuidado.

5.5 Herramientas necesarias para crear un componente XPCOM

- Un editor de texto para la creación de la Idl [42], se puede utilizar el bloc de notas.
- El ejecutable xpidl, que vine con cualquier versión del gecko, para la creación de los stubs y punto h.
- Un generador de identificadores únicos (uuidgen en Windows, disponible en Visual C++).
- El ejecutable regxpcom, opcional, también disponible con cualquier versión del gecko, útil para registrar el componente.
- Nmake [43], herramienta del Framework de punto Net para crear la DLL.

5.6 Clases necesarios para la creación del componente

Las clases necesarios para la creación del componente XPCOM se muestran en la figura 5.2.

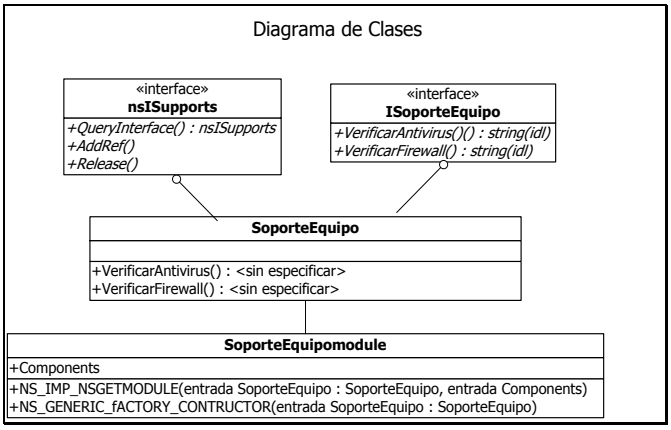


Figura 5.2. Diagrama de clases del componente

La interface nsISupports es necesaria para acceder las funciones: QueryInterface, que no es más que la encargada de preguntarle a un componente si implementa cierta interface, AddRef es llamada cuando se crea una nueva instancia del componente y Release útil cuando se destruye, esta interface es de implementación obligatoria pues es gracias a ella que se puede tener una referencia de los objetos creados. La interface IsoporteEquipo por otro lado es la que expone la funcionalidad básica para verificar el soporte del equipo, la función verificarAntivirus se encarga de buscar el antivirus instalado y retorna su nombre, y la función VerificarFirewall que devuelve el estado del firewall, activado o no. La clase SoporteEquipo es la encargada de implementar la interface IsoporteEquipo, primero buscando en el registro de Windows los antivirus instalados y posteriormente utilizando una librería de terceros para lograr la comunicación con el firewall. SoporteEquipoModule utiliza macros como NS_GENERIC_FACTORY_CONSTRUCTOR y *NS_IMPL_NSGETMODULE* que le permiten implementar las interfaces nsIFactory y nsIModule, las cuales son accesibles por medio del método nsGetModule, gracias a este método, se puede obtener

información acerca del componente(donde se encuentra ubicado, tipo de patrón implementado, factory o singleton).

La implementación de la interface nsISupports y de la clase SoporteEquipoModule se lograr gracias a la utilización de las macros ya mencionadas, gracias a estas, la reducción en el código es significativa, facilitando el desarrollo del componente, su registro y depuración.

5.7 Creación de la interface

La implementación de la interface se hace mediante el lenguaje de definición de interfaces XPIDL, que es una mezcla de lenguaje Java Script y C++. En la Figura 5.3 se muestra el código de la idl, el cual debe contener:

- El nsISupports.idl, la cual define la interface nsISupports y de la cual deben derivar todas las interfaces XPCOM (#include "nsISupports.idl").
- La declaración de la interface (interface ISoporteEquipo : nsISupports), especificando que hereda de nsISupport.
- Definir si la interface va hacer accesible desde JavaScript y el identificador de la interface, el cual debe ser único ([scriptable, uuid(e2dc8e44-6925-48b1-a886-525ad94a97d5)]).
- Por último los atributos y/o métodos de nuestra interface (string VerificarAntivirus();string VerificarFirewall();).

```
#include "C:\gecko-sdk\idl\nsISupports.idl"

[scriptable, uuid(e2dc8e44-6925-48b1-a886-525ad94a97d5)]

interface ISoporteEquipo : nsISupports{
  /*Esta primera función debe regresar el nombre del antivirus instalado

 */
   string VerificarAntivirus(); //retorna el antivirus instalado
   string VerificarFirewall();//retorna el estado del firewall
 };
```

Figura 5.3. Definición de la interface

5.8 Generando el stub y la librería de tipos xpt

Teniendo definida la interfaces idl se puede generar el esqueleto o stub (que servirá para crear el punto h de la aplicación y el archivo CPP con la lógica del negocio) y la librería de tipos que permite tener acceso al componente desde otros lenguaje como (Phyton, JavaScript, Ruby, etc). Para generar estos archivos desde la consola para símbolos del sistema de Windows se deben digitar las siguientes líneas.

- Digitar xpidl –m header IsoporteEquipo.idl, para crear el stub (IsoporteEquipo.h) el cual es útil a la hora de crear el punto h de la aplicación.

- Digitar xpidl – typelib IsoporteEquipo.idl, para crear la librería de tipos (ISoporteEquipo.xpt).

El stub creado (IsoporteEquipo.h), no es más que una serie de plantillas entre las que se pueden contar; una plantilla para la creación de la clase, una plantilla de los include y define, y plantillas de macros.

5.9 Creación del punto h a partir del stub.

El siguiente código es una parte del stub creado y en el se especifican las principales plantillas utilizadas (templates) para la creación del punto h de la aplicación y el archivo punto cpp.

```
#define ISOPORTEEQUIPO_IID \ /*Identificador de la clase o Classe
ID
                          Utilizado en el punto h de la
                     aplicación*/
  {0xe2dc8e44, 0x6925, 0x48b1, \
    { 0xa8, 0x86, 0x52, 0x5a, 0xd9, 0x4a, 0x97, 0xd5 }}

/*Use el código de la parte de abajo como plantilla para la
implementación de la clase, se debe reemplazar _MYCLASS_ por el
nombre de la clase
*/

/* Header file */
class _MYCLASS_ : public ISoporteEquipo
{
public:
  NS_DECL_ISUPPORTS
  NS_DECL_ISOPORTEEQUIPO

  _MYCLASS_();

private:
  ~_MYCLASS_();

protected:
  /* additional members */
};

/* Plantilla del método VerificarAntivirus (); */
NS_IMETHODIMP _MYCLASS_::VerificarAntivirus(char **_retval){

}

/* Plantilla del método VerificarFirewall (); */
NS_IMETHODIMP _MYCLASS_::VerificarFirewall(char **_retval){
```

```
}
```

Gracias a las plantillas incluidas en el código anterior se puede generar el punto h del componte y los identificadores del componente figura 5.4. Entre estos identificadores están: el contract id, con el cual se identifica el componente, el class name, para la identificación de la clase de una forma más entendible, y el classe id, que es el identificador de la clase en formato hexadecimal.

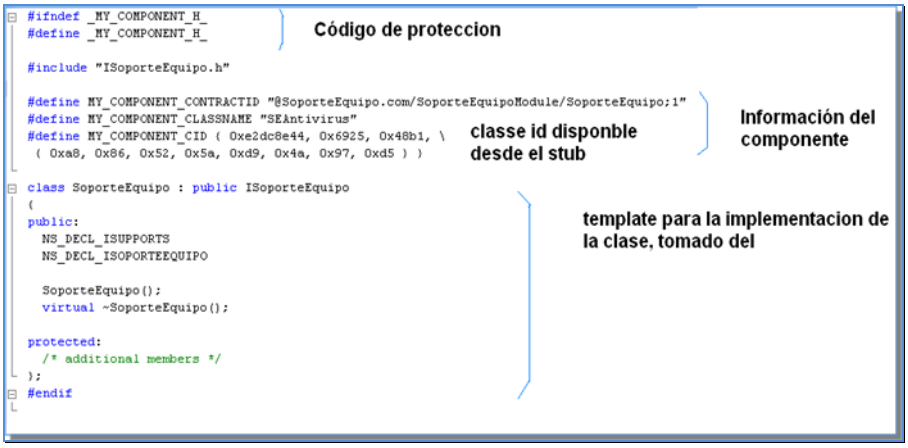


Figura 5.4. Definición de la clase del componente, archivo SoporteEquipo punto h

Para la implementación de la interface, archivo cpp Figura 5.5, se deben incluir los siguientes elementos; los archivos punto h requeridos para la implementación de las funciones verificar antivirus y verificar firewall, la macro necesaria para implementar la interface nsISupports, NS_IMPL_ISUPPORTS1 (SoporteEquipo, IsoporteEquipo), inicializar el constructor de nsISupports desde el constructor del componente y el espacio para implementar las funciones. NS_IMETHODIMP es una de las tantas macros utilizadas para facilitar la escritura del código y en este caso se utiliza para comprobar que los parámetros pasados y retornados por las funciones sean los adecuados. Hasta esta parte no sea introducido el código de las funciones este se explicara a continuación.

```
#include "SoporteEquipo.h"
#include "string.h"
#include "nsMemory.h"
#include "stdio.h"
#include "stdlib.h"
#include "WinXPSP2FireWall.h"
#include "Windows"

NS_IMPL_ISUPPORTS1 (SoporteEquipo, ISoporteEquipo)

SoporteEquipo::SoporteEquipo () { /* inicialización de miembros desde el contructor */
        NS_INIT_ISUPPORTS();
}
SoporteEquipo:: ~SoporteEquipo (){  /* destructor */
}

NS_IMETHODIMP SoporteEquipo::VerificarAntivirus (char **_retval){
return NS_ERROR_NOT_IMPLEMENTED;
}

NS_IMETHODIMP SoporteEquipo::VerificarFirewall(char **_retval){
return NS_ERROR_NOT_IMPLEMENTED;
}
```

Figura 5.5. Implementación del componente SoporteEquipo.ccp

El código necesario para implementar las funciones de verificación del antivirus y firewall se muestran a continuación, este se explicará entre líneas tomando primero el código de la verificación del antivirus.

Código para verificación del antivirus:

```
char myResult[2000]="";                 /*variable utilizada para el retorno del nombre del
                                        antivirus*/
FILE *archivo;          //declaración de un puntero de tipo File
const char *ErrorCargaArchivo="Error de archivo";//constante para
                                        //retorno de error al cargar el archivo
char lineaReg[2000]="";  //Variable para almacenar las cadenas  del registro
char aux[2000];         //Variable auxiliar para el manejo de cadenas del registro
char *tokens;           //para recuperar cadenas del registro
char rutaReg[2000]="C:\\ListProgram.txt";/*ruta con el archivo
                                        //temporal creado por el comando regedit. */

system("regedit                 /a                      C:\\ListProgram.txt
\"HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall")
;                       /*este comando de regedit permite crear un archivo con
información del software instalado y que se encuentra dentro del registro, la ruta
HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\  Windows\\CurrentVersion\\Uninstall*/


if((archivo=fopen(rutaReg,"rb+"))==NULL){  /*abrir el archivo con el registro de los
programas instalados*/
strcat(myResult,ErrorCargaArchivo); //control del error al abrir el archivo
     }

   else{
        while(!feof(archivo)){//recuperar línea por línea datos desde el archivo
```

```
            fscanf(archivo,"%s",lineaReg);
            strncpy(aux,((const char *)lineaReg),13);
            aux[13]='\0';/*solo se necesita tomar los 13 primeros
                        caracteres del registro*/
           if(strstr(aux,"DisplayName")){/*Buscamos la etiqueta
                                        DisplayName en registro*/
                  tokens=strtok(lineaReg,"=");/*al encontrar la etiqueta sacamos en nombre
del software instalado*/
                  tokens=strtok(NULL,"=");

                  if(strstr(tokens,"\"ESET")){
                        if (!strstr(myResult,"Nod 32")){
                        strcat(myResult,"Nod 32");
                        }
                   }

                  if(strstr(tokens,"\"Kaspersky")){
                        if (!strstr(myResult,"Kaspersky")){
                        strcat(myResult,"kaspersky");
                        }
                   }

                  if(strstr(tokens,"\"BitDefender")){
                        if (!strstr(myResult,"BitDefender")){
                        strcat(myResult,"BitDefender");
                        }
                   }

                  if(strstr(tokens,"\"BitDefender9")){
                        if (!strstr(myResult,"BitDefender9")){
                        strcat(myResult,"BitDefender9");
                        }
                   }

                  if(strstr(tokens,"\"Norton")){
                        if (!strstr(myResult,"Norton")){
                        strcat(myResult,"Norton");
                        }
                   }

                  if(strstr(tokens,"\"Symantec")){
                        if (!strstr(myResult,"Symantec")){
                        strcat(myResult,"Symantec");
                        }
                   }

                  if(strstr(tokens,"\"AVPersonal")){
                        if (!strstr(myResult,"AVPersonal")){
```

```
        strcat(myResult,"AVPersonal");
        }
    }

if(strstr(tokens,"\"Spy")){
        if (!strstr(myResult,"Spy")){
        strcat(myResult,"Spy");
        }
    }

if(strstr(tokens,"\"Spybot")){
        if (!strstr(myResult,"Spybot")){
        strcat(myResult,"Spybot");
        }
    }

if(strstr(tokens,"\"Zone")){
        if (!strstr(myResult,"Zone")){
        strcat(myResult,"Zone");
        }
    }

if(strstr(tokens,"\"Trend")){
        if (!strstr(myResult,"Trend")){
        strcat(myResult,"Trend");
        }
    }

if(strstr(tokens,"\"SinEspias")){
        if (!strstr(myResult,"SinEspias")){
        strcat(myResult,"SinEspias");
        }
    }

if(strstr(tokens,"\"ewido")){
        if (!strstr(myResult,"ewido")){
        strcat(myResult,"ewido");
        }
    }

if(strstr(tokens,"\"mcafee.com")){
        if (!strstr(myResult,"MCafee")){
        strcat(myResult,"MCafee");
        }
    }

if(strstr(tokens,"\"AVG")){
        if (!strstr(myResult,"AVG")){
```

```
                    strcat(myResult,"AVG");
                    }
                }

                if(strstr(tokens,"\"Panda")){
                    if (!strstr(myResult,"Panda")){
                    strcat(myResult,"Panda");
                    }
                }

                if(strstr(tokens,"\"Avast4")){
                    if (!strstr(myResult,"Avast4")){
                    strcat(myResult,"Avast4");
                    }
                }

            }


          }
        fclose(archivo);
        }
```

/*El siguiente código controlar el mapea al tipo dato que se desea retornar y en caso de error lo gestiona*/

```
if(!_retval)
    return NS_ERROR_NULL_POINTER;

  *_retval = (char*) nsMemory::Clone(myResult,
                                     sizeof(char)*(strlen(myResult)+1));
    return *_retval ? NS_OK : NS_ERROR_OUT_OF_MEMORY;
```

Código verificación del firewall:

```
char myResult[6]=""; //variable a retornar true or false(firewall activado o
                     //desactivado)
  const char *FirewallEnable="true";
  const char *FirewallDisable="false";
  ///////////////inicio codigo inicializacion/////////////////
  INetFwProfile* m_pFireWallProfile=NULL;
  HRESULT hr = S_FALSE;
      INetFwMgr* fwMgr = NULL;
      INetFwPolicy* fwPolicy = NULL;
      VARIANT_BOOL bFWEnabled;

      FW_ERROR_CODE ret = FW_NOERROR;
      try
        {
```

```
                if( m_pFireWallProfile )
                        throw FW_ERR_INITIALIZED;

                // Crea una instancia del firewall settings manager.
                hr = CoCreateInstance( __uuidof(NetFwMgr), NULL,
CLSCTX_INPROC_SERVER, __uuidof( INetFwMgr), (void**)&fwMgr );

                if( FAILED( hr ))
                        throw FW_ERR_CREATE_SETTING_MANAGER;

                // retorna el local firewall policy.
                hr = fwMgr->get_LocalPolicy( &fwPolicy );
                if( FAILED( hr ))
                        throw FW_ERR_LOCAL_POLICY;

                // retorna el firewall profile currently in effect
                hr = fwPolicy->get_CurrentProfile( &m_pFireWallProfile );
                if( FAILED( hr ))
                        throw FW_ERR_PROFILE;

        }
        catch( FW_ERROR_CODE nError)
        {
                ret = nError;
        }
                if( fwPolicy )
                fwPolicy->Release();
        if( fwMgr )
                fwMgr->Release();

  ////////////////////Fin Codigo de inializacion        ////////////////////////

  try
        {

                hr = m_pFireWallProfile->get_FirewallEnabled( &bFWEnabled );
                if( FAILED(hr))
                        throw FW_ERR_FIREWALL_IS_ENABLED;

                if( bFWEnabled != VARIANT_FALSE )
                        strcat(myResult,FirewallEnable);
           else{
        strcat(myResult,FirewallDisable);
      }
        }
        catch( FW_ERROR_CODE nError )
        {
                return nError;
```

```
        }
    //////////fin codigo verificacion firewall enable
  //El siguiente código controlar  el mapea  al tipo dato  que se desea retornar y //encaso de error lo gestiona.
  if(!_retval)
      return NS_ERROR_NULL_POINTER;

   *_retval = (char*) nsMemory::Clone(myResult,
                              sizeof(char)*(strlen(myResult)+1));
   return *_retval ? NS_OK : NS_ERROR_OUT_OF_MEMORY;
```

5.10 Implementación de las interfaces nsImodule y nsIFactory

Como ya se digo, estas interfaces son el punto de acceso al modulo (figura 5.6) y permiten además gestionar el registro del componente de forma transparente. Las macros utilizadas son:

- NS_IMP_NSGETMODULE(SoporteEquipo,components), cuyos parámetros son el nombre de la clase y una constante del tipo nsModuleComponentInfo, esta constante contiene; el class name, class id, contract id y el constructor del componente.
- NS_GENERIC_FACTORY_CONSTRUCTOR (SoporteEquipo): con esta macro se implementa el patrón factory para la clase pasada como parámetro.

```
#include "SoporteEquipo.h"
#include "nsIGenericFactory.h"
#include "nsIModule.h"
#include "nsIFactory.h"

 NS_GENERIC_FACTORY_CONSTRUCTOR(SoporteEquipo)

static const nsModuleComponentInfo components[] =
 {
     {
        MY_COMPONENT_CLASSNAME,
        MY_COMPONENT_CID,
        MY_COMPONENT_CONTRACTID,
        SoporteEquipoConstructor
     },
 };

 NS_IMPL_NSGETMODULE("SoporteEquipo",components);
```

Figura 5.6. Implementación de nsImodule y nsIFactory

5.11 Creación de la DLL

Cuando se crean componentes XPCOM, en C++ y que van a ser utilizados por el navegador Firefox en sistemas Windows, estos deben ser compilados dentro de una DLL, los pasos seguidos para crearla incluyen:

- Tener en un directorio los archivos; SoporteEquipo.ccp (con el código de nuestro componente), los archivos de cabecera, soporteEquipoModule.ccp (utilizado para auto registrar el componente), y el archivo soporteEquipo.mak(utilizado para compilar y crear la DLL)

- Por medio del interprete de comandos utilizar la herramienta nmake (nmake /f soporteEquipo.mak), con lo cual se genera una DLL en la cual se encapsula nuestro componente, y el cual puede ser utilizado en sistemas operativos Windows.

5.12 Empaquetado del componente

Hay que saber inicialmente que para el empaquetado se debe suministrar los archivos que controlan como se instalan y donde se encuentran los recursos de la extensión, estos archivos son: chrome.manifiest e install.rdf de los cuales ya se hablo en los capítulos iníciales. El chrome.manifiest e install.rdf deben ir dentro de la carpeta principal ModuloB2C y dentro de la carpeta componentes se deben colocar la DLL y/o DSO (dependiendo del sistema operativo en el cual se vaya a instalar el componente) además de la librería de tipos, ver figura 5.7.

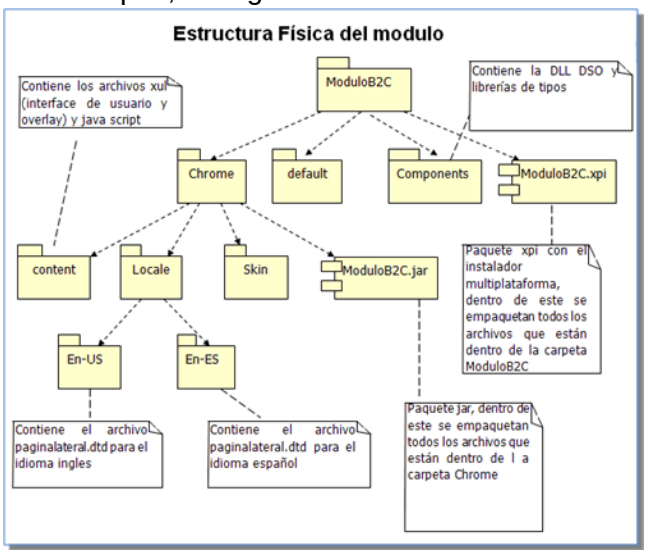


Figura 5.7. Estructura física del modulo

El paquete xpi, el cual es un archivo Zip con la extensión renombrada a xpi es un instalador multiplataforma y para su consecución primeramente se deben empaquetar los archivos dentro de la carpeta Chrome en una punto jar, que al igual que el xpi es un Zip renombrado a jar, finalmente se empaqueta todos los archivos que contiene la carpeta moduob2c con la extensión xpi, el paquete obtenido se muestra en la Figura 5.8.

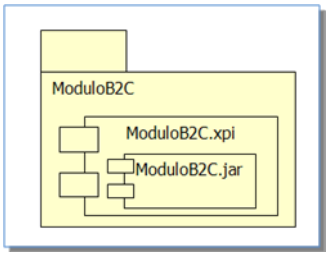


Figura 5.8. Estructura del instalador XPI

5.13 Instalación del componente

La instalación de la extensión se realiza abriendo el archivo xpi con el navegador Firefox o desde este, pero antes de esto se debe colocar el conjunto de recomendaciones generales, que vienen dentro de la carpeta recomendaciones, dentro de la carpeta de instalación del navegador (C:\Archivos de programa\Mozilla Firefox\chrome) para que la extensión los pueda localizar. Los archivos dentro de la carpeta recomendaciones son archivos txt con recomendaciones del nivel de conexión, de seguridad del sitio, acerca de phishing y de las transacciones en curso. A continuación se enumeran los pasos necesarios para su instalación.

- Copiar la carpeta con el instalador xpi dentro de cualquier parte del disco duro y hacer doble clic sobre el archivo modulob2c.xpi, Figura 5.9.

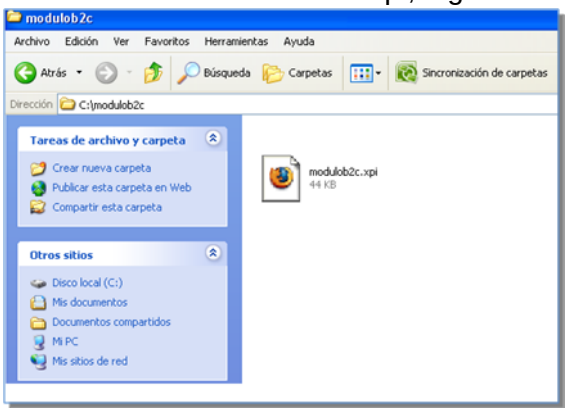


Figura 5.9. Instalador XPI

- Dar clic en el botón, instalar ahora , del wizard de Firefox para la instalación de nuevo software y que se despliega después de dar clic en el instalador xpi.
-

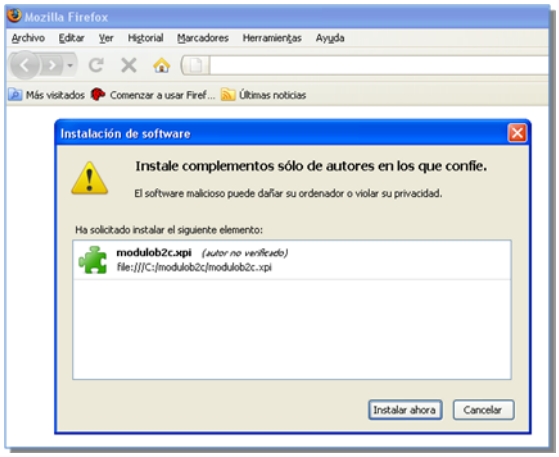


Figura 5.10. Wizard de la instalación de nuevo software en Firefox

- Reiniciar Firefox desde la ventana de complementos

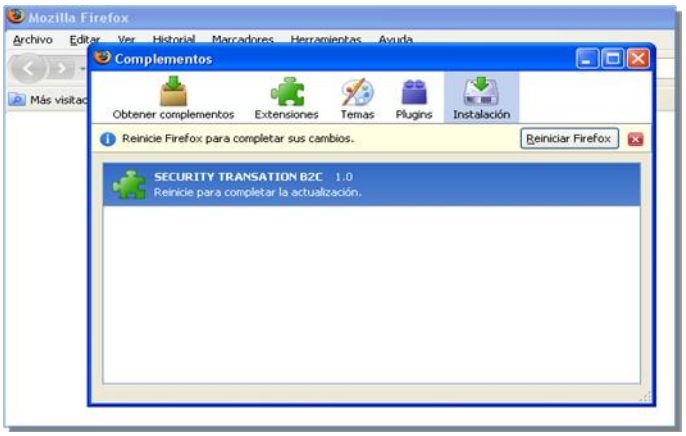


Figura 5.11. Ventana de complementos

Los tres pasos anteriores son los únicos requeridos para instalar el componente, ahora, para la verificación de la instalación al reiniciar Firefox se puede abrir la ventana de complementos, Figura 5.12, (ruta herramientas, complementos) para verificar que la extensión aparezca en esta.

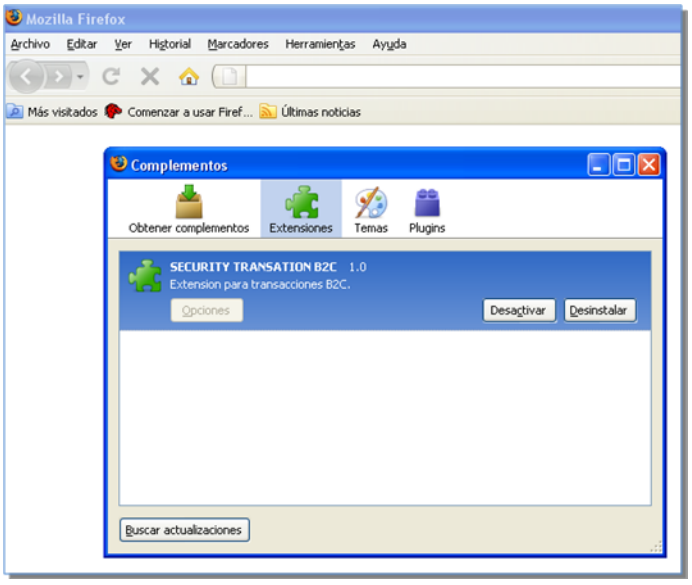


Figura 5.12. Ventana de complementos con información de la extensión instalada

En la Figura 5.13, se puede ver la interface de la extensión, la cual está ubicada como un panel lateral en Firefox, este panel contiene dos pestañas, la principal, gestión de seguridad en donde se muestra la información referente a las políticas verificadas (botón ver detalles), su nivel de cumplimiento (bajo, medio o alto) y el nivel de la transacción y la pestaña recomendaciones en donde se pueden ver las

recomendaciones generales y de la transacción en curso. Este panel es accesible desde la barra de navegación, ver, panel lateral, MVPS para transacciones B2C.

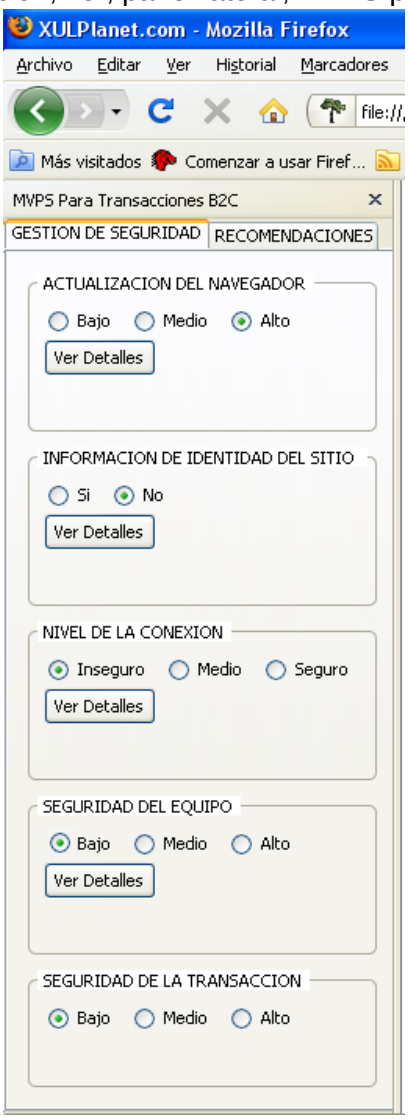


Figura 5.13. Interface de la extensión

ANEXO F- PRUEBAS

6 Pruebas modulo de verificación de políticas de seguridad para transacciones B2C (MVPS)

6.1 Introducción

Después de realizar el desarrollo del Módulo de Verificación de Políticas de Seguridad para transacciones electrónicas B2C (MVPS), se procede a verificar el resultado de la implementación con el fin de determinar la calidad de la aplicación.
Una estrategia de prueba del software debe incluir pruebas de bajo nivel que verifiquen que todos los pequeños segmentos de código fuente se han implementado correctamente, así como pruebas de alto nivel que validen las principales funciones del sistema frente a los requisitos del negocio. A partir de esto, el software se probará desde dos perspectivas diferentes:

1. La lógica interna del programa que se comprobará utilizando técnicas de diseño de casos de prueba de caja blanca.
2. Los requisitos del software que se comprobarán utilizando técnicas de diseño de casos de prueba de caja negra.

Las pruebas realizadas, deben tener las siguientes características:

- Una buena prueba tiene una alta probabilidad de encontrar un error.
- Una buena prueba no debe ser redundante. Es decir, no debe existir un motivo por el cual se deba realizar una prueba que tenga el mismo propósito que otra.
- Una buena prueba no debería ser ni demasiado sencilla ni demasiado compleja.

6.2 Pruebas de caja blanca

Este tipo de pruebas se basa en la minuciosa evaluación de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado.

6.2.1 Prueba del camino básico

Esta técnica de prueba de caja blanca permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar una medida como guía para la definición de un *conjunto básico* de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

6.2.2 Complejidad ciclomática

La *complejidad* ciclomatica V(G) es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se utiliza en el contexto de prueba del camino básico, el valor calculado como complejidad

ciclomática define el número de *caminos independientes* del c*onjunto básico* de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.
Un *camino independiente* es cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias de proceso o una nueva condición.

6.2.3 Construcción de grafos de flujo de código

Se construye un grafo de flujo de código en el cual cada sentencia de código representará un nodo del grafo. Luego de construido el grado se obtiene su respectivo valor de la complejidad ciclomática, con el fin de cubrir al menos una vez cada camino del grafo.

Serán utilizadas las formas establecidas en [44]para calcular la complejidad ciclomática de cada grafo:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

2. La complejidad ciclomática, V(G), de un grafo de flujo G se define como:
   V(G) = A – N + 2
   Donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

3. La complejidad ciclomática, V(G), de un grafo de flujo G también se define como:
   V(G) = P + 1

Donde P es el número de nodos predicados contenidos en el grafo de flujo G.
Nodo Predicado: Es aquel nodo que contiene una condición y se caracteriza porque dos o más aristas emergen de él.

En primera instancia se realizarán pruebas para el proceso de carga e inicialización del módulo. En segundo lugar las pruebas estarán enfocadas en encontrar errores en la cada una de las implementaciones de las políticas establecidas en el capítulo 3 de la monografía de grado.

Inicialización del Módulo MVPS

Código fuente del archivo LoadModulo.js que se ejecuta cuando el usuario inicializa el Módulo MVPS:

```
try(//INICIAR EL MODULO

 1 top.getBrowser().addProgressListener(EventListenerChange_Url_Security);      //adicionar listener nsiweprogresslistener
 2 NivelActualNav();                                                            //verificar el actual nivel de actualizacion del navegador

 3 var browserSecurityI=top.getBrowser().securityUI;
 4 VerificarNivelConexion(browserSecurityI.state);
 5 VerificarCertificado(top.getBrowser().currentURI);
 6 NivelTransaccion();                                                          //seleccionar el nivel de seguridad de la transaccion
 7 VerificarSoporteEquipo();                                                    //verificar el soporte de seguridad del equipo (firewall y antivirus)

 8 AbrirRecomTrans();                                                           //Abrir recomendaciones de la transaccion
 9 AbrirRecomGeneral();
   }

catch(e){//SI OCURRE ALGUN ERROR AL ARRANCAR EL MODULO LANZAR EXCEPTION

      alert("ERROR AL ARRANCAR EL MODULO");

    }
```

Figura 6.1. Inicialización modulo MVPS-Archivo LoadModulo.js

A partir del código fuente en la Figura 6.1 se puede construir el respectivo grafo de flujo para el proceso de Inicialización del módulo:

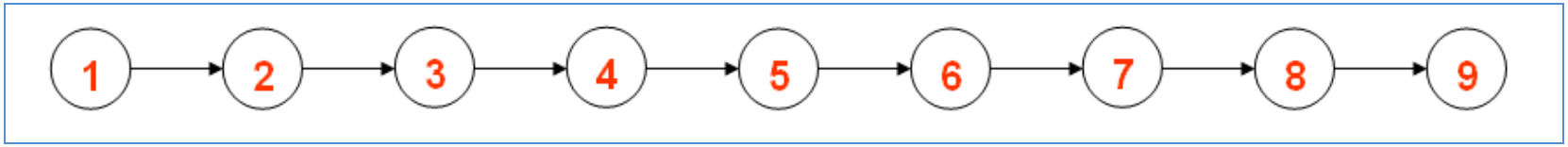


Figura 6.2. Grafo inicialización modulo MVPS-Archivo loadModulo.js

Se utiliza el segundo método para calcular la complejidad ciclomática del grafo de flujo de la Figura 6.2.

V(G) = 8 aristas – 9 nodos + 2 = 1

Caminos independientes: 1
Camino 1: 1 – 2 – 3 – 4 – 5 - 6 – 7 – 8 – 9

Se puede visualizar que cada nodo del grafo está relacionado con cada sentencia del código fuente de la Figura 9.1. Cabe destacar que los nodos 2, 4, 5, 6 y 7 son nodos compuestos que tienen más código ya que cada uno de ellos representa un método que tiene desarrollada una funcionalidad específica; a causa de ello, los nodos compuestos están compuestos de más nodos y por ende cada uno de ellos forman un grafo independiente, como se puede ver a continuación:

Nodo 2:

```
function NivelActualNav(){              //funcion para verificar la actualizacion del navegador + gecko
   1  var navegadorVersion=navigator.userAgent;  //Cadena que contiene la version del navegador y la version del gecko

   2  agente = navegadorVersion.split("Firefox/");
   3  agentegecko = navegadorVersion.split("rv:");
   4  versionNavegador= agente[(agente.length)-1];
   5  version=agente[(agente.length)-1];
   6  versiongecko = agentegecko[1];
   7  agentegecko = versiongecko.split(")");
   8  versiongecko=agentegecko[0];
   9  agente =version.split(".");
  10  version=agente[0];
  11  var estado="true";


  12  switch (version){    //dependiendo de la version del gecko se obtiene el objeto radio button del grupo groupBoxActualNav en la pagina lateral
          case '1' :
      13          document.getElementById("rbt_actNavegbad").setAttribute("selected", estado);
                  break;
          case '2' :
      14          document.getElementById("rbt_actNavegmid").setAttribute("selected", estado);
                  break;
          case '3' :
      15          document.getElementById("rbt_actNaveghight").setAttribute("selected", estado);
                  break;
          }
    }
```

Figura 6.3.Nivel Actual Navegador-Archivo UtilsPolitica.js

De la Figura 6.3 se obtiene el siguiente grafo flujo:

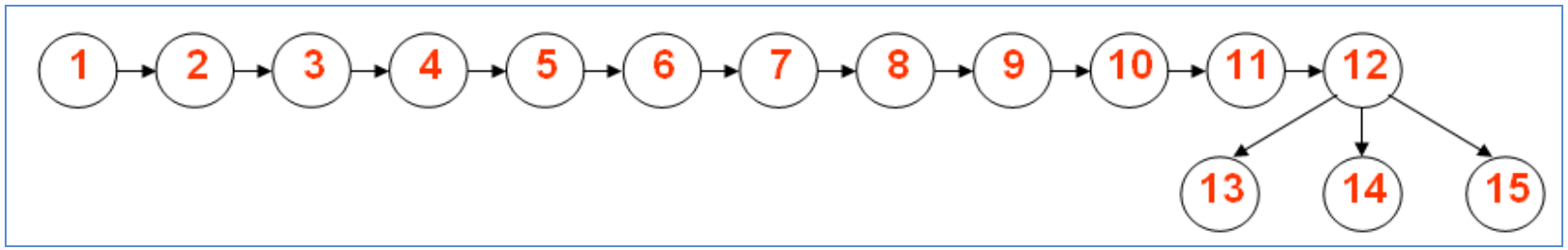


Figura 6.4.Grafo Nivel Actual Navegador-Archivo UtilPoliticas.js

Para el cálculo de la complejidad ciclomática en el grafo de flujo de la Figura 6.4 se utiliza el segundo método:
V(G) = 14 aristas – 15 Nodos + 2 = 1

Caminos independientes: 1, aunque se puede ver que el nodo 12 es un select case el cual presenta tres casos, por lo cual se concluye que el total de caminos independientes son 3

Camino 1: 1 – 2 – 3 – 4 - 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13
Camino 2: 12 – 14
Camino 3: 12 – 15

El grafo de la Figura 6.4 muestra los nodos de los cuales está compuesto el nodo 2 de la Figura 6.2.

Nodo 4:

```
function VerificarNivelConexion(aStatus){                    //funcion para verificar el nivel de la conexion, los valores posible son
                                                              //262144 y 262146 estado de la conexion STATE_SECURE_HIGH
1 if(aStatus==1048576){//verificar extended validation        //2 estado seguro
2   extendedValid="Si";                                       //1048576 estado de la conexion STATE_IDENTITY_EV_TOPLEVEL
                                                              //esto es cuando se usa un certificado Extended Validation
  }                                                           //65356 estado de la conexion STATE_SECURE_MED
                                                              //131072 estado de la conexion STATE_SECURE_LOW
3 else{                                                       //4 estado de la conexion STATE_IS_INSECURE
4   extendedValid="No";                                       //1 estado de la conexion STATE_IS_BROKEN (estado desconocido)
  }

    //verificar tipo de conexion, valores en application development - web browser
5 if(aStatus==262144 || aStatus==262146 || aStatus==1048576 ||aStatus==2){          //estado de la conexion alto
6    rbtSeg=document.getElementById("rbt_conexSiteInSeg");
7    rbtSeg.setAttribute("selected",f);
8    rbtSeg=document.getElementById("rbt_conexSiteMed");
9    rbtSeg.setAttribute("selected",f);
10   rbtSeg=document.getElementById("rbt_conexSiteSeg");
11   rbtSeg.setAttribute("selected",t);
  }
12 if(aStatus==65536){                                                            //estado de la conexion medio
13   rbtSeg=document.getElementById("rbt_conexSiteInSeg");
14   rbtSeg.setAttribute("selected",f);
15   rbtSeg=document.getElementById("rbt_conexSiteMed");
16   rbtSeg.setAttribute("selected",t);
17   rbtSeg=document.getElementById("rbt_conexSiteSeg");
18   rbtSeg.setAttribute("selected",f);
  }
19 if(aStatus==131072 || aStatus==1 || aStatus==4){                               //estado de la conexion inseguro
20   rbtSeg=document.getElementById("rbt_conexSiteInSeg");
21   rbtSeg.setAttribute("selected",t);
22   rbtSeg=document.getElementById("rbt_conexSiteMed");
23   rbtSeg.setAttribute("selected",f);
24   rbtSeg=document.getElementById("rbt_conexSiteSeg");
25   rbtSeg.setAttribute("selected",f);
  }
}
```

Figura 6.5. Verificar Nivel Conexión -Archivo UtilsPoliticas.js

De la Figura 6.5 se puede obtener el siguiente grafo de flujo:

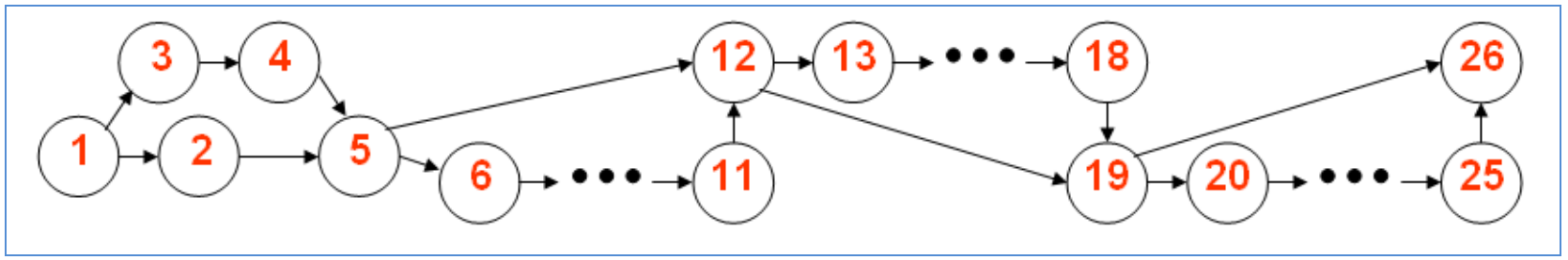


Figura 6.6. Grafo Verificar Nivel Conexión- Archivo UtilPoliticas.js

El cálculo de la complejidad ciclomatica en el grafo de flujo de la Figura 6.6 se calcula de la siguiente manera:
V(G) = 1 nodo predicado + 1 = 2

Caminos independientes: 2
Camino 1: 1 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26
Camino 2: 1 – 2 – 5 – 6
Hay que aclarar que los nodos 5, 12 y 19 no se consideran nodos predicados ya que solamente tienen un solo camino. Esto se puede constatar en la Figura 6.5.

El grafo de la Figura 6.6 muestra los nodos de los cuales está compuesto el nodo 4 de la Figura 6.2.

Nodo 5:

```
function VerificarCertificado(url){       //esta funcion obtiene el certificado de un sitio y lo verifica

1 var browserSecurity=top.getBrowser().securityUI;
2 document.getElementById("rbt_identyNo").setAttribute("selected",t);
3 document.getElementById("rbt_identySi").setAttribute("selected",f);

4 if(url !="undefined")
  {
5   lastpagevisit=url.prePath;//la nueva pagina
6   getProtocolo();
  }
7 if (browserSecurity)
  {
8   var prov = browserSecurity.QueryInterface(Components.interfaces.nsISSLStatusProvider);
9   if (prov)
    {        //utilizar excepcion para este punto
10     var status = prov.SSLStatus;  //obtener la propiedad sslstatus
11     if (status)
       {
12        status = status.QueryInterface(Components.interfaces.nsISSLStatus);
13        if (status)
          {//si la pagina tiene un certificado
             CheckUseCert();   //seleccionar radio buton uso certificado
14           var nsIX509Cert=status.serverCert;
15           logFirma=status.cipherName;            //obtener el algoritmo utilizado para la firma del certificado
16           verific=nsIX509Cert.VERIFIED_OK ;
17           revoc= nsIX509Cert.CERT_REVOKED;
18           Emisor=nsIX509Cert.issuer;
19           commoname=nsIX509Cert.commonName ;
20           issuerCommonName=nsIX509Cert.issuerCommonName;
21           domainmismatch=status.isDomainMismatch;
22           untrusted=status.isUntrusted;
23           isNotValidAtThisTime=status.isNotValidAtThisTime;
24           issuerOrganitation=nsIX509Cert.issuerOrganization;
25
26           verificationResult = nsIX509Cert.verifyForUsage(Components.interfaces.nsIX509Cert.CERT_USAGE_SSLServer);
```

```
27          switch (verificationResult)
            {    //establecer el uso del certificado en la variable verifyCert
               case 0:
28                    VerifyCert="OK";
                      break;
               case 1:
29                    VerifyCert="SIN VERIFICAR";
                      break;
               case 2:
30                    VerifyCert="REVOCADO";
                      break;
               case 4:
31                    VerifyCert="HA EXPIRADO";
                      break;
               case 8:
32                    VerifyCert="NO CONFIABLE";
                      break;
               case 16:
33                    VerifyCert="ISSUER NO CONFIABLE";
                      break;
               case 32:
34                    VerifyCert="ISSUER DESCONOCIDO";
                      break;
               case 64:
35                    VerifyCert="CA INVALIDA";
                      break;
               case 128:
36                    VerifyCert="USO NO PERMITIDO";
                      break;
            }
         }
37       else
         {
38          logFirma=status.cipherName;       //obtener el algoritmo utilizado para la firma del certificado
         }
      }
    }
  }
39}
```

Figura 6.7. Verificar Certificado-Archivo UtilsPoliticas.js

A partir de la Figura 6.7 se puede construir el grafo:

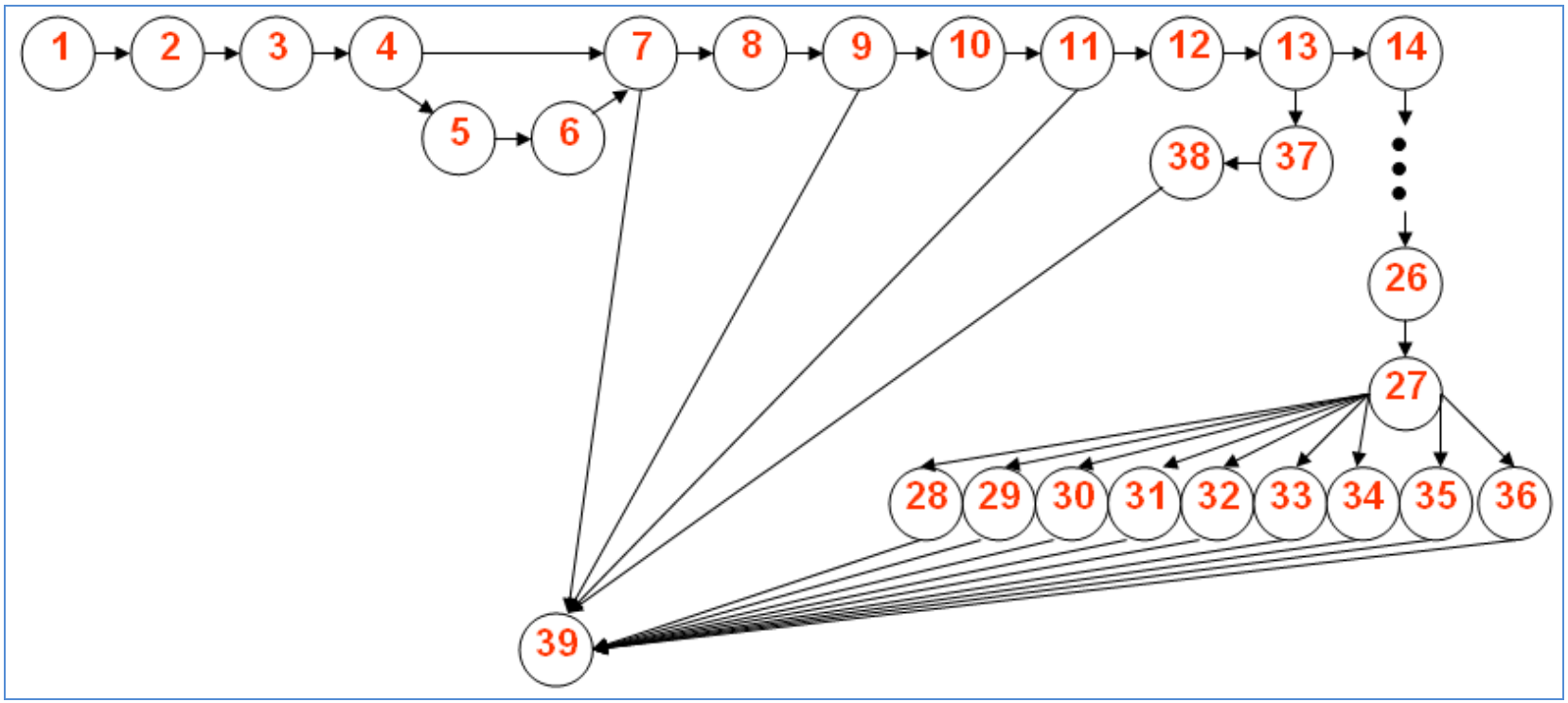


Figura 6.8.Grafo Verificar Certificado-Archivo Utilspoliticas.js

El cálculo de la complejidad ciclomática en el grafo de flujo de la Figura 6.8 se calcula de la siguiente manera:
V(G) = 1 Nodo predicado + 1 = 2
Se aclara que los nodos 4, 7, 9 y 11 no son nodos predicados. Esto se puede verificar con la Figura 6.7.

Caminos independientes: Se puede observar que el nodo 27 es un select case, el cual presenta 9 casos; por lo cual, el total de caminos independientes son 10.

Camino 1: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27 – 28 – 39

Camino 2: 27 – 29 – 39
Camino 3: 27 – 30 – 39
Camino 4: 27 – 31 – 39
Camino 5: 27 – 32 – 39
Camino 6: 27 – 33 – 39
Camino 7: 27 – 34 – 39
Camino 8: 27 – 35 – 39
Camino 9: 27 – 36 – 39
Camino 10: 13 – 37 – 38 – 39

El grado de la Figura 6.8 muestra los nodos de los cuales se compone el nodo 5 de la Figura 6.2.

Nodo 6:

```
function NivelTransaccion(){                                  //esta funcion es utiliza para establecer el nivel de seguridad de la transaccion
1  rbtSeg=document.getElementById("rbt_identySi");            //verificar si esta seleccionado el boton de uso de certificado en la pagina
   var segTrans;                                              //variable para guardar el nivel de seguridad de la transaccion
2  if(rbtSeg.getAttribute("selected")=="true")
   {   //nivel de la transaccion si la pagina no tiene certificado
3      rbtSeg=document.getElementById("rbt_segTransBaj");
4      rbtSeg.setAttribute("selected",t);
5      rbtSeg=document.getElementById("rbt_segTransMed");
6      rbtSeg.setAttribute("selected",f);
7      rbtSeg=document.getElementById("rbt_segTransAlt");
8      rbtSeg.setAttribute("selected",f);

   }
   else
   {   //nivel de la transaccion si la pagina tiene certificado
9      switch (verificationResult)
       {  //verificacion del certificado
          case 0 :
10                segTrans=3;           //verificacion del certificado es okey
                  break;
          case 1 :
11                segTrans=1;           //sin verificar el certificado
                  break;
          case 2 :
12                segTrans=1;           //certificado revocado
                  break;
          case 4 :
13                segTrans=1;           //certificado expiro
                  break;
          case 8 :
14                segTrans=1;           //certificado no confiable
                  break;
          case 16:
15                segTrans=1;           //issuer no fiable
                  break;
          case 32:
16                segTrans=1;           //issuer desconocido
                  break;
```

```
          case 64:
17                segTrans=1;           //CA es invalida
                  break;
          case 128:
18                segTrans=1;           //Uso no permitido
                  break;
        }
19      if((logFirma=="md2WithRSAEncryption") ||(logFirma=="md5WithRSAEncryption")) {  //especificacion algoritmo firma para nivel confiable
20         segTrans=2;
        }
21      switch (segTrans)
        { //verificacion del certificado
          case 1 ://Seleccion nivel de la transaccion bajo
22              rbtSeg=document.getElementById("rbt_segTransBaj");
23              rbtSeg.setAttribute("selected",f);
24              rbtSeg=document.getElementById("rbt_segTransMed");
25              rbtSeg.setAttribute("selected",f);
26              rbtSeg=document.getElementById("rbt_segTransAlt");
27              rbtSeg.setAttribute("selected",t);
                break;
          case 2 ://Seleccion nivel de la transaccion medio
28              rbtSeg=document.getElementById("rbt_segTransBaj");
29              rbtSeg.setAttribute("selected",f);
30              rbtSeg=document.getElementById("rbt_segTransMed");
31              rbtSeg.setAttribute("selected",t);
32              rbtSeg=document.getElementById("rbt_segTransAlt");
33              rbtSeg.setAttribute("selected",f);
                break;
          case 3 ://Seleccion nivel de la transaccion alto
34              rbtSeg=document.getElementById("rbt_segTransBaj");
35              rbtSeg.setAttribute("selected",t);
36              rbtSeg=document.getElementById("rbt_segTransMed");
37              rbtSeg.setAttribute("selected",f);
38              rbtSeg=document.getElementById("rbt_segTransAlt");
39              rbtSeg.setAttribute("selected",f);
                break;
        }
    }
  }
```

Figura 6.9. Nivel Transacción -Archivo UtilsPoliticas.js

A partir de la Figura 6.9 se crea el grafo expuesto a continuación:

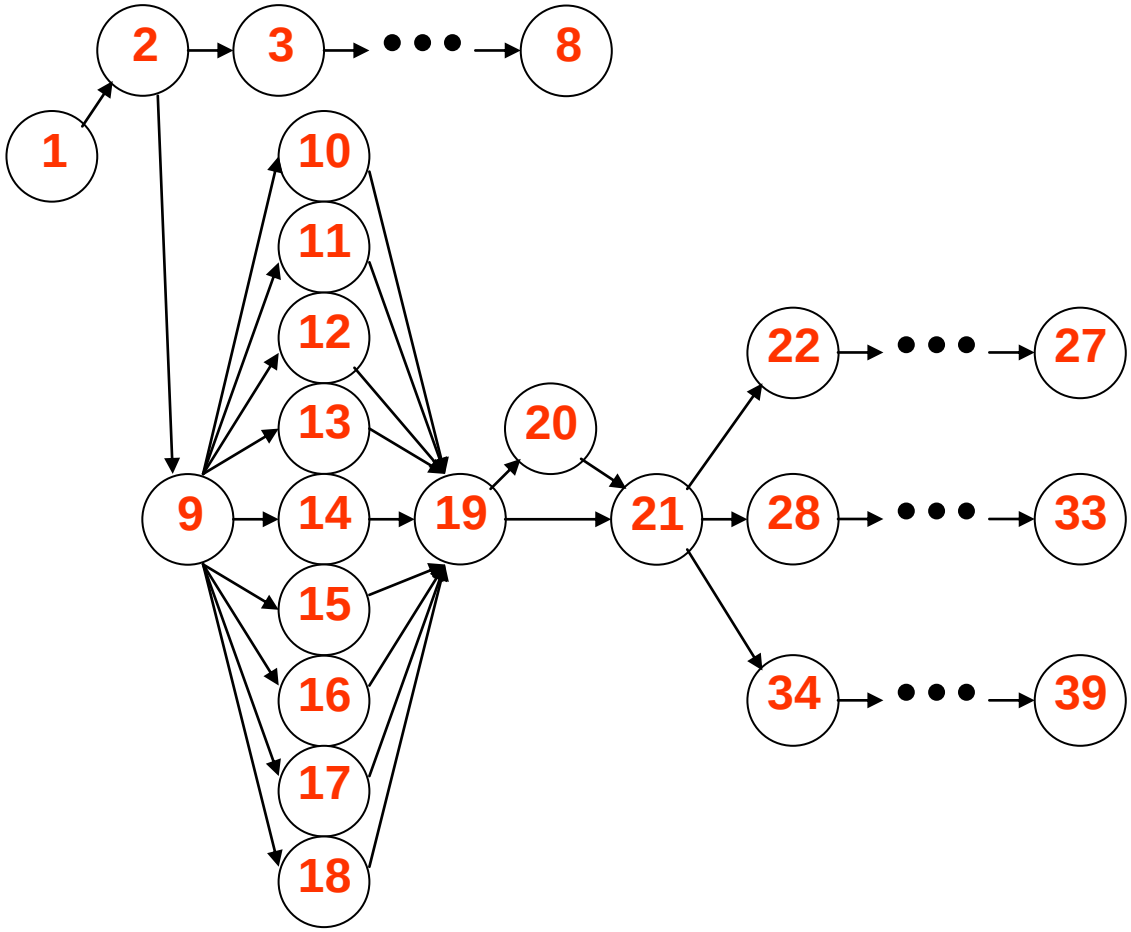


Figura 6.10.Grafo Nivel Transacción-Archivo UtilsPoliticas.js

El cálculo de la complejidad ciclomática en el grafo de flujo de la Figura 6.10 se calcula de la siguiente manera:
V(G) = 1 nodos predicados + 1 = 2
Se aclara que el nodo 19 no se considera un nodo predicado. Esto se puede verificar en la Figura 6.9.

Caminos independientes: Se puede observar que los nodos 9 y 21 son nodos especiales de tipo select case. En donde el nodo 9 presenta 9 casos y el nodo 21 presenta 3 casos. Por lo cual, el total de caminos independientes es de 10.

Camino 1: 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8
Camino 2: 1 – 2 – 9 – 10 – 19 – 20 – 21 – 22 – 23 – 24 – 25 – 26 – 27
Camino 3: 9 – 11 – 19 – 20 – 21 – 28 – 29 – 30 – 31 – 32 – 33
Camino 4: 9 – 12 – 19 – 20 – 21 – 34 – 35 – 36 – 37 – 38 – 39
Camino 5: 9 – 13 – 19 – 20 – 21 – …
Camino 6: 9 – 14 – 19 – 20 – 21 – …
Camino 7: 9 – 15 – 19 – 20 – 21 – …
Camino 8: 9 – 16 – 19 – 20 – 21 – …
Camino 9: 9 – 17 – 19 – 20 – 21 – …
Camino 10: 9 – 18 – 19 – 20 – 21 – …

Los tres puntos suspensivos quieren decir, que del nodo 21 en adelante, se puede tomar cualquiera de los casos especiales de dicho nodo.

El grado de la Figura 6.10 representa los nodos que componen el nodo 6 de la Figura 6.2.

Nodo 7:

```
function VerificarSoporteEquipo(){     //Obtiene el componente para verificar el soporte de seguridad del equipo
    try{
    1 var SoporteEquipo=Components.classes["@SoporteEquipo.com/SoporteEquipoModule/SoporteEquipo;1"].createInstance();
    2 SoporteEquipo=SoporteEquipo.QueryInterface(Components.interfaces.ISoporteEquipo);

    3 AntiVirusNom=SoporteEquipo.VerificarAntivirus();//Esta funcion retorna informacion del antivirus
    4 EstadoFirewall=SoporteEquipo.VerificarFirewall();//Esta funcion retorna el estado del firewall; ausencia, activado o no activado
    5 SetNivelSopEquipo();
        }

     catch(e){
           alert("error en modulo doporte equipo");

         }
   }
```

Figura 6.11. Verificar Soporte Equipo-Archivo UtilsPoliticas.js

De la Figura 6.11 se obtiene el siguiente grafo de flujo:

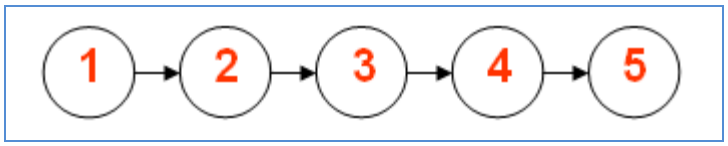


Figura 6.12. Grafo Soporte Equipo-Archivo Utilspoliticas.js

La complejidad ciclomática del grafo de flujo de la Figura 6.12 es:
V(G) = 0 nodos predicados + 1 = 1

Caminos independientes: 1
Camino 1: 1 – 2 – 3 – 4 – 5

El grafo de la Figura 6.12 representa los nodos que componen el nodo 7 de la Figura 6.2.

Cambios en la URL

Los cambios en la URL hacen referencia al evento lanzado por el navegador en el instante en que el usuario cambia de dirección electrónica en la url del browser.
Los tipos de cambios tratados por el módulo MVPS son los presentados cuando se sufre un cambio en el protocolo de conexión o cuando se pasa a otro sitio Web.

Código fuente del archivo LoadModulo.js que se ejecuta cuando se activa el evento que escucha cambios en la URL del navegador:

```
1 var EventListenerChange_Url_Security = {//objecto del tipo nsIWebProgressListener para actualizar el modulo cuando cambie la seguridad o url
      stateIsRequest:false,       //solo recibe el event onStateChange si el parametro stateFlags incluye STATE_IS_REQUEST
      2 QueryInterface : function(aIID) {
            if (aIID.equals(Components.interfaces.nsIWebProgressListener) ||    //application development - web browser
                aIID.equals(Components.interfaces.nsISupportsWeakReference) ||
                aIID.equals(Components.interfaces.nsISupports))
                    return this;
            throw Components.results.NS_NOINTERFACE;
        },

        onLocationChange : function(aProgress,aRequest,aLocation) {
            3 VerificarCertificado(aLocation);
            4 NivelTransaccion();
            5 AbrirRecomTrans();

        },


        onSecurityChange : function(aWebprogres,aRequest,aStatus){//funcion utilizada para fijar el nivel de seguridad de la conexion
          6 VerificarNivelConexion(aStatus);
          7   NivelTransaccion();
          8   AbrirRecomTrans();
              return 0;

      }

    };
```

Figura 6.13 Cambio en la URL-Archivo LoadModulo.js

A continuación se visualiza el grafo del anterior código fuente:

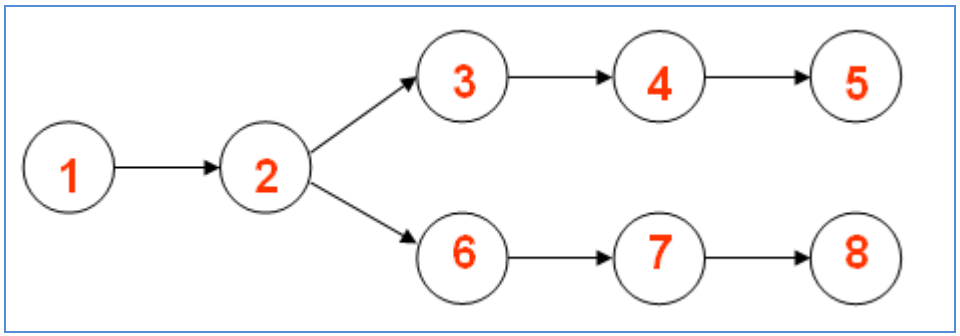


Figura 6.14 Grafo cambio en la URL-Archivo LoadModulo.js

La complejidad ciclomatica del grafo de flujo de la Figura 6.14 es:
V(G) = 1 nodo predicado + 1 = 2

Caminos independientes: 2
Camino 1: 1 – 2 – 3 – 4 – 5
Camino 2: 2 – 6 – 7 – 8

De la Figura 6.13 podemos ver que los nodos 3, 4 (que es el mismo 7) y 6 son compuestos. En donde el nodo 3 está representado por la Figura 6.7 y el grafo de la Figura 6.8; el nodo 4 esta representado por la Figura 6.9 y por el grafo de la Figura 6.10; y el nodo 6 por la Figura 6.5 y por el grafo de la Figura 6.6.

6.2.4 Casos de prueba que forzarán la ejecución de cada camino del conjunto básico

Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino. Los casos de prueba que satisfacen el conjunto básico previamente descrito son:

6.3 Pruebas de caja negra

Las pruebas de caja negra son aquellas que se llevan a cabo sobre la interfaz del software. Es decir, con este tipo de casos de prueba se pretende demostrar que las funciones del software son operativas (ya que se centran en los requisitos funcionales del software), que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como la integridad de la información externa se mantiene.

Con las pruebas de caja negra se examina algunos de los aspectos del modelo fundamental de sistema sin tener mucho en cuenta la estructura lógica interna del software.

La prueba de caja negra no es una alternativa a la prueba de caja blanca, más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca no alcanzan a verificar.

Según **¡Error! No se encuentra el origen de la referencia.**, las pruebas de caja negra intentan encontrar errores de las iguientes categorías:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o accesos a bases de datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y terminación.

ANEXO G- MANUAL DE USUARIO

En este manual se explicarán todos los aspectos a tener en cuenta sobre el Módulo de Verificación de Políticas de Seguridad (MVPS).

7 Características

MVPS presenta las siguientes características:

- Está construido como una extensión para el navegador de código libre Mozilla Firefox, por lo cual su funcionamiento está supeditado a dicho navegador.
- Necesita de los servicios brindados por Framework 1 o superior de .NET y el gecko 1.9 o superior, para su correcto funcionamiento.
- Capaz de detectar si el sitio Web visitado presenta certificado digital y si este es emitido por una Entidad Certificadora reconocida.
- Detecta si la conexión del navegador con el sitio Web es segura.
- Verifica el protocolo utilizado para la conexión.
- Comprueba si la máquina en la cual está instalado tiene el firewall activado y si tiene instalado alguno de los antivirus más reconocidos.
- Presenta un conjunto de recomendaciones de seguridad que se deben tener en cuenta en el sitio Web visitado.

8 Componentes de MVPS

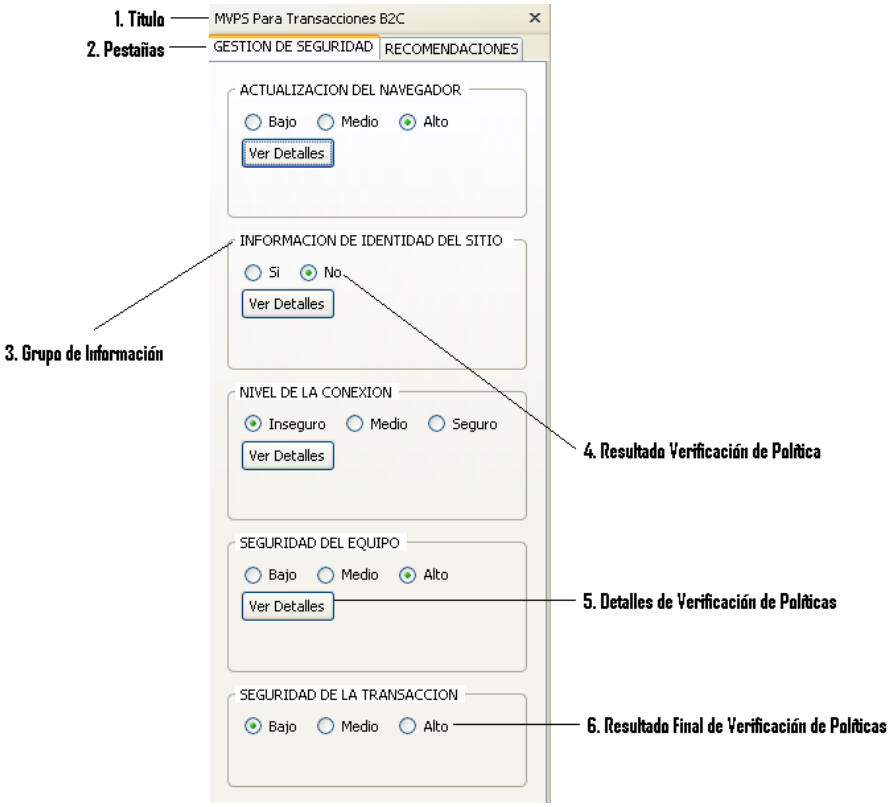


Figura 8.1 Componentes Pestaña Gestión de Seguridad

Titulo: Presenta el título del módulo “Módulo de Verificación de Políticas de Seguridad para Transacciones B2C (MVPS)”.

Pestañas: La primera llamada “GESTION DE SEGURIDAD” muestra los resultados obtenidos después de validar las políticas de seguridad en el sitio Web visitado.
La segunda llamada “RECOMENDACIONES” presenta una serie de recomendaciones de seguridad que el usuario debe tener en cuenta para el sitio Web que está visitando.

Grupo de Información: Agrupa las verificaciones en grupos de información.

Resultado Verificación de Políticas: Presenta el resultado obtenido después de verificar cada una de las políticas de seguridad.

Detalles de Verificación de Políticas: Botón que muestra la información más detallada de los resultados obtenidos en el verificación de políticas.

Resultado Final de Verificación de Políticas: Presenta el resultado final ponderado de todos los resultados obtenidos en los otros grupos.

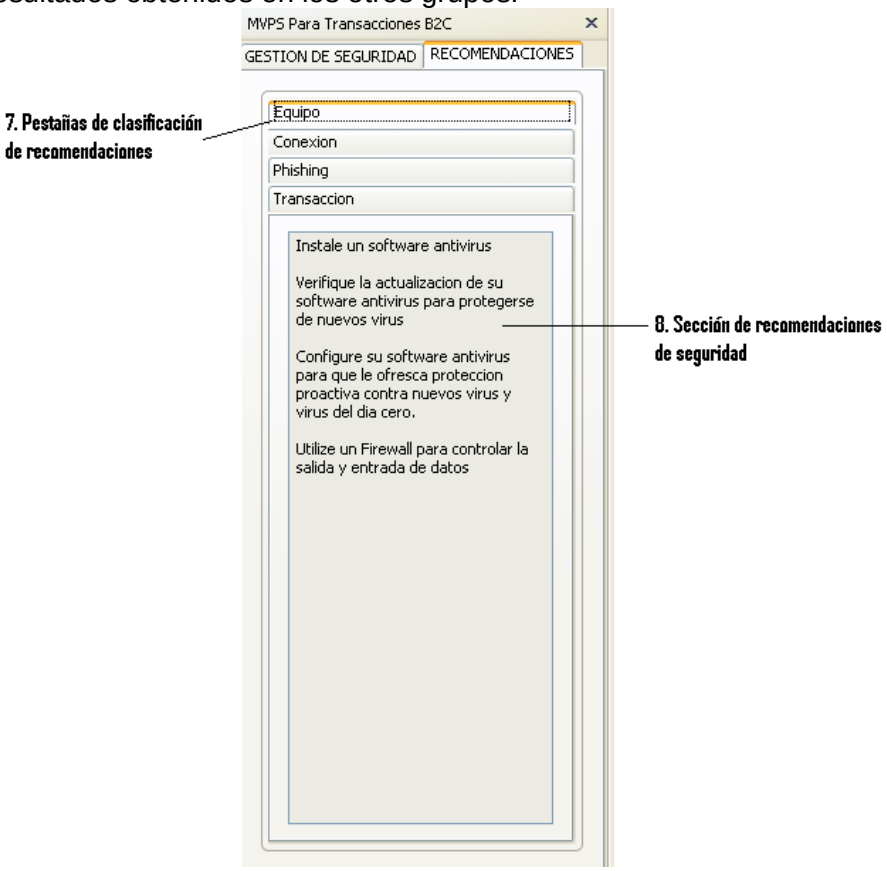


Figura 8.2 Componentes Pestaña Recomendaciones

Pestañas de clasificación de recomendaciones: Agrupan y clasifican las recomendaciones de seguridad según el tipo de información.

Sección de recomendaciones de seguridad: En esta sección se muestran cada una de las recomendaciones de seguridad que brinda el módulo MVPS.

9 Guía de uso de MVPS

A continuación se presentan los pasos a seguir para utilizar lo servicios brindados por el módulo:

Paso 1

Se ejecuta Mozilla Firefox para navegar por Internet.
Cuando se tenga cargado el navegador Web, se procede a activar el módulo de la siguiente manera: Ver – Panel lateral – MVPS Para Transacciones B2C. Ver Figura 9.1:

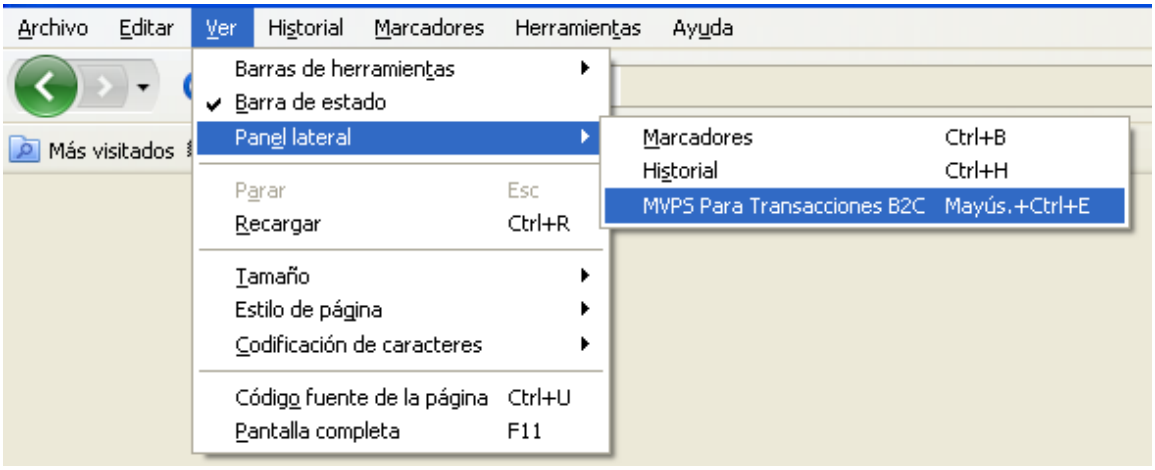


Figura 9.1 Activación del módulo MVPS

Paso 2

Se procede a navegar por Internet, digitando la dirección electrónica del sitio Web que se desee visitar en la URL del navegador.

Como ejemplo particular se ha visitado el Sitio Web del Banco de Bogotá para mostrar la funcionalidad del módulo.

En primera instancia se puede observar que el módulo MVPS en el Grupo de Información llamado Actualización del Navegador (seleccionado en rojo), específicamente en su respectivo botón Detalles de Verificación de Políticas, despliega información referente a la versión del navegador y la versión del Gecko utilizada ver Figura 9.2:

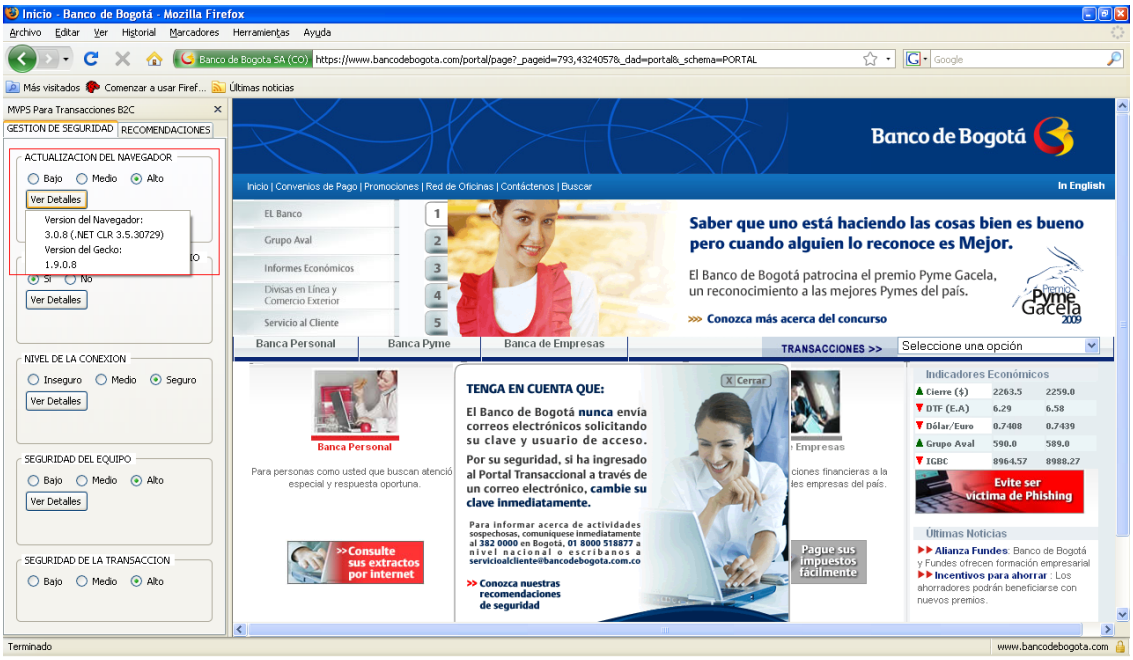


Figura 9.2 Detalles Actualización del Navegador

En el Grupo llamado Información de Identidad del Sitio (seleccionado en rojo), en el botón Detalles de Verificación de Políticas se despliega información referente a Dirección real del sitio Web visitado, validación de existencia de Certificado Extended Validation, Verificación del estado del Certificado, Nombre real del sitio, Nombre de la Organización que expide el Certificado Digital. Ver Figura 9.3:

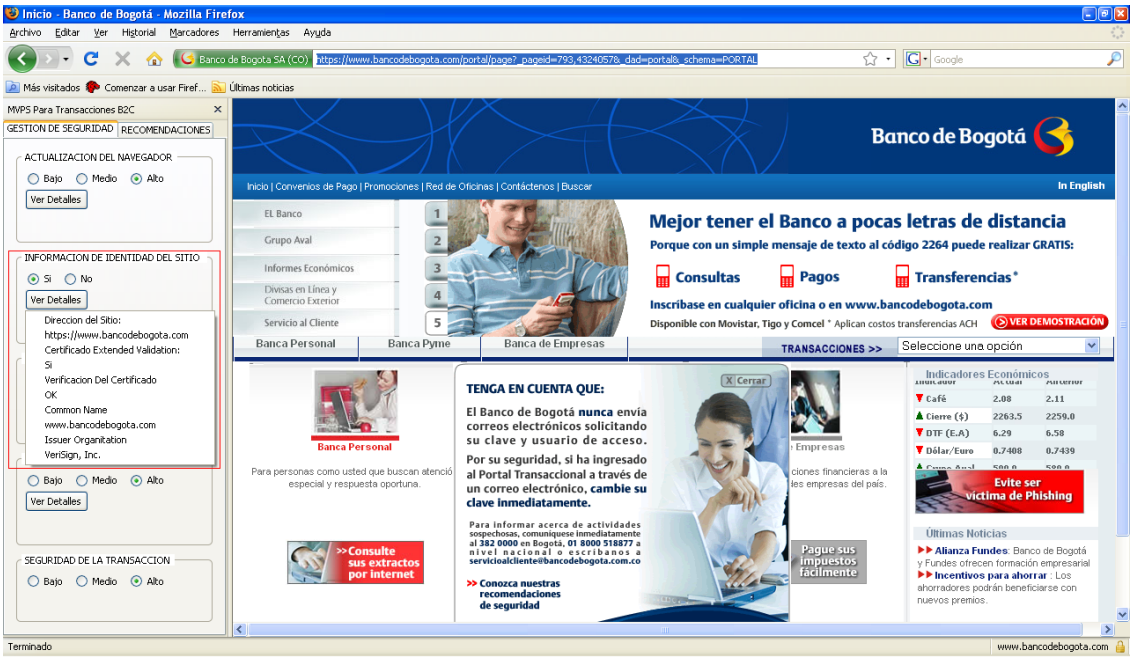


Figura 9.3 Detalles Información de Identidad del Sitio

En el Grupo llamado Nivel de la Conexión (seleccionado en rojo), en el botón de Detalles se despliega información sobre el Nivel de Cifrado o Algoritmo de Cifrado utilizado para la conexión, se valida el Tipo de Conexión (segura o insegura), se verifica el Protocolo establecido para la conexión. Ver Figura 9.4:

Figura 9.4 Detalles Nivel de la Conexión

En el Grupo llamado Seguridad del Equipo (seleccionado en rojo), en el botón de Detalles se despliega información sobre la Seguridad del Equipo local. Se valida el antivirus instalado y si el Firewall del Sistema Operativo se encuentra Activado. Ver Figura 9.5:

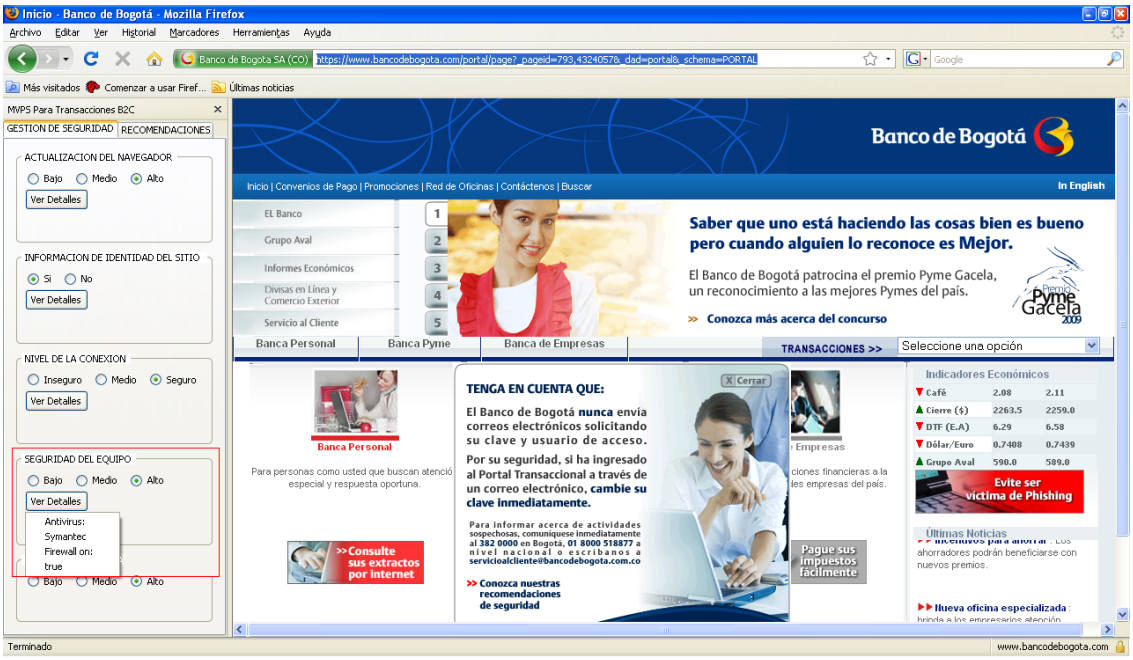


Figura 9.5 Detalles Seguridad del Equipo

Finalmente, en el último Grupo llamado Seguridad de la Transacción (seleccionado en rojo) se da el resultado final de la validación de las políticas de seguridad verificadas anteriormente. Ver Figura 9.6:

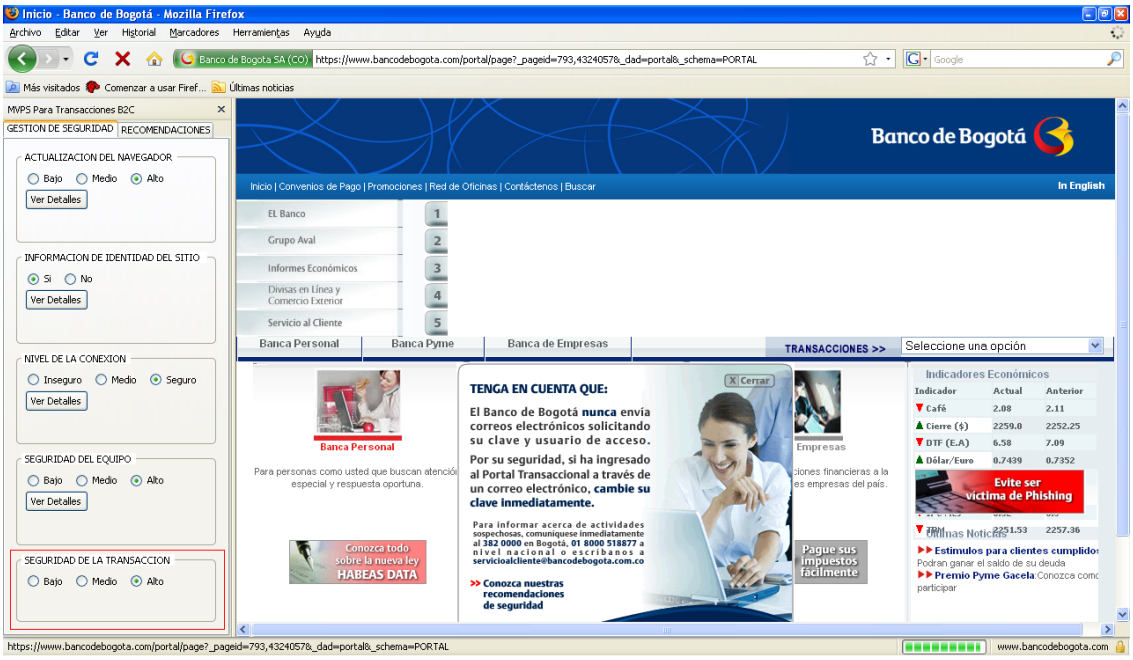


Figura 9.6 Detalles Seguridad de la Transacción

Además de la Validación de las políticas de seguridad vistas anteriormente, se presenta en la pestaña de Recomendaciones un conjunto de Recomendaciones de seguridad a tener en cuenta en las transacciones de comercio electrónico B2C:

Figura 9.7 Conjunto de Recomendaciones de Seguridad

BIBLIOGRAFÍA

[1] Universidad Nacional de Colombia, Vicerrectoría General, dirección Nacional de Informática y comunicaciones. *Guía para la elaboración de políticas de seguridad*. Colombia: 2003 [ref. mayo 20 de 2008]. http://www.unal.edu.co/seguridad/documentos/guia_para_elaborar_politicas_v1_0.pdf

[2] Nigel McFarlane. *Rapid Application Development with Mozilla*. Prentice Hall: 2004, pp 2-3.

[3] Doug, Turner; Ian, Oeschger. *Creating XPCOM Components*. 2003, pp 26-27**.** http://www.mozilla.org/projects/xpcom/book/cxc/pdf/cxc.pdf

[4] Microsoft. *Archivos DLL* (online). Noviembre de 2007. http://msdn.microsoft.com/es-es/library/1ez7dh12.aspx

[5] Web estilo. Java Script (en línea). España: Agosto de 2006. http://www.webestilo.com/javascript/

[6] Instituto argentino de normalización (IRAM). *Information technology, code of practice for information security management.* ISO 17799:2005. Segunda edición. Buenos Aires (Argentina): IRAM, 2002. 82p.

[7] The Internet Engineering Task Force [IETF]. *RFC2196 Site Security Handbook* [Categoría informativo]. [s.l]: B. Fraser 1997 [ref. junio 20 de 2008].Disponible en la web: http://translate.google.com.co/translate?hl=es&sl=en&u=http://asg.web.cmu.edu/rfc/rfc2196.html&sa=X&oi=translate&resnum=10&ct=result&prev=/search%3Fq%3Drfc%2B2196%26hl%3Des

[8] Patrick D. Howard. *Information Security Management Handbook*. [The Security Police Life cycle:Functions and Responsabilities]. [s.l]: Tipton & Krause, CRC Press LLC, 2003.

[9] Universidad Nacional de Colombia. *Políticas de seguridad informática* (en línea).2005. http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060035/lecciones/Capitulo2.html

[10] Guevara Campo Carolina, Mera Fabián Andrés. *Criterios para establecer políticas de seguridad de al información y plan de contingencia, caso de estudio el centro de datos de la Universidad del Cauca*. Universidad del Cauca, Facultad de Ingeniería Electrónica y Telecomunicaciones, Departamento de Sistemas, Popayán cauca: 2008.

[11] Ministerio de Administraciones Públicas. *Metodología de Análisis y Gestión de riesgos de los Sistemas de Información*. I-Método, NIPO 326-05-047-X.

Magerit versión 2(en línea). Madrid: junio 2006. 154p. http://www.csi.map.es/csi/pdf/magerit_v2/metodo_v11_final.pdf

[12] Mañas José A. *Herramientas para el análisis y gestión de riesgos* (en línea). Septiembre 2004. p 10. http://www.csi.map.es/csi/tecnimap/tecnimap_2004/comunicaciones/tema_06/6_022.pdf

[13] W3C. *Extensible Markup Language (XML)* (en línea). Abril de 2009. http://www.w3.org/XML/

[14] Ministerio de Administraciones Públicas. *Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información*. II – Catálogo de Elementos, NIPO 326 – 05 – 047 – X. Magerit versión 2 (en línea). Madrid: junio de 2006, pp5-14. http://www.csi.map.es/csi/pdf/magerit_v2/metodo_v11_final.pdf

[15] Instituto Nacional de Tecnologías de la Comunicación INTECO. *Estudio sobre Usuarios y Entidades Públicas y Privadas afectadas por la práctica fraudulenta conocida como Phishing* (en línea). 2007 [ref. de 5 de julio de 2008]. pp. 36, 64, 90, 104. http://www.inteco.es/search/Resultados_de_la_Busqueda/?allSearchField=Estudio+sobre+Usuarios+y+Entidades+P%C3%BAblicas+y+Privadas+afectadas+por+la+pr%C3%A1ctica+fraudulenta+conocida+como+Phishing

[16] Instituto Nacional de Tecnologías de la Comunicación INTECO. *Estudio Confianza E-Commerce*. pp. 22, 29, 30, 46

[17] Instituto Nacional de Tecnologías de la Comunicación INTECO. *Ataques del día cero* (en línea). http://www.av-comparatives.org/seiten/ergebnisse_2008_05.php

[18] W3C. *Extensible Markup Language (en línea)*. http://www.w3.org/XML/

[19] W3C. *HTML 5 (*en línea*)*. http://dev.w3.org/html5/spec/Overview.html

[20] W3C. *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification (*en línea*)*. http://www.w3.org/TR/CSS21/

[21] W3C. *DOM (en línea)*. http://www.w3.org/DOM

[22] Mozilla Developer Center, XPCOM API reference. https://developer.mozilla.org/en/XPCOM_API_Reference.

[23] Mozilla Developer Center. The Chrome URL (en línea). Junio de 2009. https://developer.mozilla.org/en/XUL_Tutorial/The_Chrome_URL

[24] W3C. *DTDS (*online). http://www.w3.org/TR/xhtml1/dtds.html

[25] Carmona Cejudo José María. Extensión para clasificación de correo electrónico en Mozilla Thunderbird. Escuela técnica superior de ingeniería informática. Málaga; febrero de 2008.pag 131. 46-47 p

[26] Osmosis Latina. Archivos jar (en línea). 2008 http://javabasico.osmosislatina.com/curso/progbasico/jars.htm

[27] W3C. Resource Description Framework (RDF): Concepts and Abstract Syntax (en línea). Febrero de 2004. http://www.w3.org/TR/rdf-concepts/

[28] Mozilla developer Center. Menupopup (en línea). Enero de 2009. https://developer.mozilla.org/en/XUL/menupopup

[29] Mozilla developer Center. keyset (en línea). Febrero de 2009. https://developer.mozilla.org/en/XUL/keyset

[30] Mozilla developer Center. Broadcaster (en línea). Febrero de 2009. https://developer.mozilla.org/en/XUL/broadcaster

[31] Argarate, S. *Seguridad en las Transacciones online de Comercio Electrónico* (en línea). Tesis de Grado, México. 127 p. http://www.monografia.com/trabajos-pdf/seguridad-e-comerce/seguridad-e-comerce.pdf

[32] Mozilla developer Center. Layout (en línea). Diciembre de 2006. https://developer.mozilla.org/en/CSS/Getting_Started/Layout

[33] Mozilla Developer Center. nsIWebProgressListener (en lìnea). Octubre de 2006. https://developer.mozilla.org/en/nsIWebProgressListener

[34] Doug, Turner; Ian, Oeschger. *Creating XPCOM Components*. 2003. 176-183 pp. http://www.mozilla.org/projects/xpcom/book/cxc/pdf/cxc.pdf

[35] The Internet Engineering Task Force [IETF]. *RFC2196 Uniform* Resource Identifiers (URI): Generic Syntax (en línea). The internet society 1998. http://www.ietf.org/rfc/rfc2396.txt

[36] Brianary. *Firefox: Using nsIWebProgressListener* (en línea). Agosto 2006. http://brianary.blogspot.com/2006/08/firefox-using-nsiwebprogresslistener.html

[37] Verising. *Preguntas frecuentes: SSL con Extended Validation* (en línea). 2009. http://www.verisign.es/ssl/ssl-information-center/extended-validation-ssl-certificates/index.html?sl=t49740343840000018&set=b073449

[38] Mozilla Developer Center. *Gecko* (en lìnea). Marzo de 2009. https://developer.mozilla.org/en/Gecko

[39] Doug, Turner; Ian, Oeschger. *Creating XPCOM Components*. 2003. 20-21 pp. http://www.mozilla.org/projects/xpcom/book/cxc/pdf/cxc.pdf

[40] Doug, Turner; Ian, Oeschger. *Creating XPCOM Components*. 2003. 75-89 pp. http://www.mozilla.org/projects/xpcom/book/cxc/pdf/cxc.pdf

[41] Microsoft. *Archivos DLL* (online). Noviembre de 2007. http://msdn.microsoft.com/es-es/library/1ez7dh12.aspx

[42] Mozilla Developer Center. *XPIDL* (en línea). Mayo de 2009. https://developer.mozilla.org/en/XPIDL

[43] Lucent Technologies. *Nmake User´s Guide* (online). Septiembre 1998. 295 p. http://www1.bell-labs.com/project/nmake/download/manual/3.1.2/nmake_user_1098.pdf

[44] Pressman Roger S. *Ingeniería del software un enfoque practico (quinta edición).* Mc Graw Hill. España: 2002. 640p.