Master en Ciberseguridad y Privacidad
2018-2019

DESARROLLO DE UNA HERRAMIENTA DE INTERCEPTACIÓN Y MODIFICACIÓN DE CONTENIDOS PARA EL NAVEGADOR GOOGLE CROME: MITB CHROME EXTENSION

Antonio Bartolomé García Murillo
Universidad Rey Juan Carlos
Correo-e: ab.garciam@alumnos.urjc.es

Abstract. *To improve the user experience and functionality in your web browsing sessions, there are tools that are installed in the browser itself, these tools are known as extensions. The extensions allow us from view of specific content, such as files, videos, to blocking advertising or pop-up windows. But sometimes we underestimate the ability of these tools, which could have not so beneficial ends. In this document, the research work is exposed, in the field of offensive security, to demonstrate the potential of browser extensions to steal and share data transparently to the user, raise awareness about the problem of uncontrolled use, an unknown origin of these tools or use of browsers on computers and public places for connection to websites or private portals. The attack patterns that use the browser as an attack vector, are generally known as Man in the Browser (MitB), in this work a Google Chrome extension has been created to make MitB.*

Resumen*. Para mejorar la experiencia y funcionabilidad del usuario en sus sesiones de navegación web, existen herramientas que se instalan en el propio navegador, estas herramientas son conocidas como extensiones. Las extensiones nos permiten desde la visualización de contenidos específicos, como ficheros, videos, hasta el bloqueo de publicidad o ventanas emergentes. Pero en ocasiones subestimamos la capacidad de estas herramientas, que podrían tener fines no tan beneficiosos. En este documento, se refleja el trabajo de investigación, en el ámbito de la seguridad ofensiva, que pretende demostrar el potencial de las extensiones del navegador para robar y compartir datos de manera transparente al usuario, concienciar sobre la problemática de un uso no controlado, un origen desconocido de estas herramientas o el uso de navegadores en equipos y lugares públicos para conexión a sitios web o portales privados. Los patrones de ataque que usan el navegador como vector de ataque, son conocidos en términos generales como Man in the Browser (MitB), en este trabajo se ha creado una extensión de Google Chrome para hacer MitB.*

1 Introducción y contexto

Los navegadores son aplicaciones que permiten acceso a internet y navegar por la web. La aparición de los primeros navegadores web data de 1990 WorldWideWeb [1]. Desde entonces los navegadores han ido evolucionando y avanzando a medida que avanzaba también la web. Uno de los hitos más importantes es la aparición de herramientas o módulos, habitualmente de terceros, para personalizar los navegadores, llamados extensiones.

En el año 2005 aparecen las primeras referencias a los ataques que infectan el navegador a través, en muchos casos, de las extensiones, y que permiten escuchar las comunicaciones entre usuario y host. A estos ataques se les denominó Man in the Browser (Mitb) [2].

Por centrarnos en el navegador de Chrome, se han detectado diferentes extensiones que permiten el robo de datos, casi siempre bancarios. En junio del 2018 se detectó una extensión en la web store destinada a usuarios brasileños de banca on-line [3]. El 15 de enero del 2019 se detectó otra extensión

también en la propia web store, que permitía el robo de tarjetas de crédito [4]. Más recientemente un Proxy que, ayudándose de una extensión, redireccionaba de webs confiables de bancos de Latinoamérica hacia sitios fraudulentos con el fin del robo de credenciales [5].

Volviendo a los navegadores, Chrome es un navegador relativamente joven, creado en el año 2008 por Google, pero que sin embargo, en poco tiempo ha conseguido situarse entre los navegadores más utilizados, desbancando al propio Internet Explorer (navegador nativo de Microsoft Windows). La figura 1 muestra la evolución del uso de los navegadores en el tiempo.

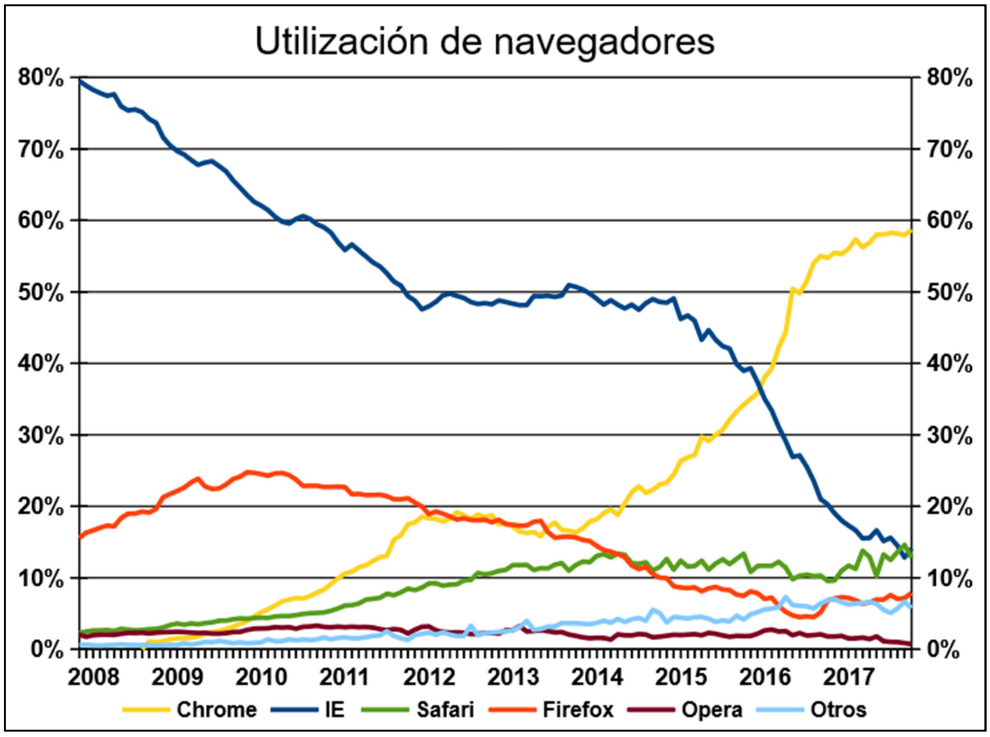

Figura 1: Evolución del uso de los navegadores en el tiempo [6]

Este trabajo se realizará para el navegador Google Chrome. El contexto de desarrollo es el de las extensiones para Chrome [7], en concreto se usará la versión 2 del manifiesto de Google. El manifiesto es la guía de referencia que marca las directrices e instrucciones que puede contener el fichero *manifest.json*, donde se describen el contenido de la extensión, sus permisos y configuración.

No se ha tenido en cuenta el borrador de la próxima actualización del manifiesto de Google Chrome para la versión 3 [8], que Google tiene intención de lanzar antes de final de año. Los motivos son, que esta versión fue anunciada por Google a mediados del 2019 con el trabajo ya empezado, los cambios técnicos serían significativos y se requeriría una migración integral del proyecto, ya que afecta a la API webRequest, básica para el trabajo . Implicaría prácticamente realizar el trabajo desde 0.

Más adelante, en el apartado de Conclusiones, se comentará, con más detalle, el impacto que tendrá esta actualización en el presente proyecto.

El resto de este documento se estructura de la manera que sigue. 2 Objetivos, describe con más profundidad que es un Man in the Middle (MitM), un Man in the Browser (MitB) y se aborda el problema a resolver y su finalidad. En el aparados 2.1 Planificación, se exponen los pasos seguidos, tiempo, plazos para el desarrollo del trabajo y las versiones disponibles. 2.2 Metodología se resumen de manera breve las tecnologías utilizadas.

En el apartado 3 Estado del arte, se presentan algunos de los malware bancarios más importantes basados en MitB, analizados para el presente trabajo. Obtenidos de publicaciones OWASP.

En el apartado 4 Desarrollo, se aborda la implementación, más a fondo en dos sub-apartados, 4.1 Implementación, que métodos se han usado, de que herramientas. Y 4.2 Arquitectura, se entra en más detalle dentro de cada uno de los módulos explicando que operaciones y funciones realizan.

En el apartado 5 Validación y evaluación, se explica cómo se prueba la herramienta, algunos problemas encontrados en las pruebas durante el desarrollo y su solución.

El apartado 6 Conclusiones, se abordan las ideas y cuestiones surgidas tras el desarrollo, el sub-apartado 6.1 Trabajo Futuro, se expone como se podría continuar el trabajo y sus implicaciones.

2 Objetivos

En este trabajo se pretende desarrollar una herramienta para desplegar un ataque Man in the Browser (MitB) (Como ya se habría puesto en el resumen, puede directamente llamarse MitB). El objetivo es puramente educativo y de investigación. Se rechaza el uso para otros fines.

Derivada del patrón de ataque popular Man in the Middle (MitM), donde el atacante se sitúa en medio de la comunicación entre dos nodos, normalmente por envenenamiento de la cache ARP, entre la capa de enlace y la de red de la Pila OSI. El diagrama de un ataque MitM tradicional puede verse en la Figura 2.

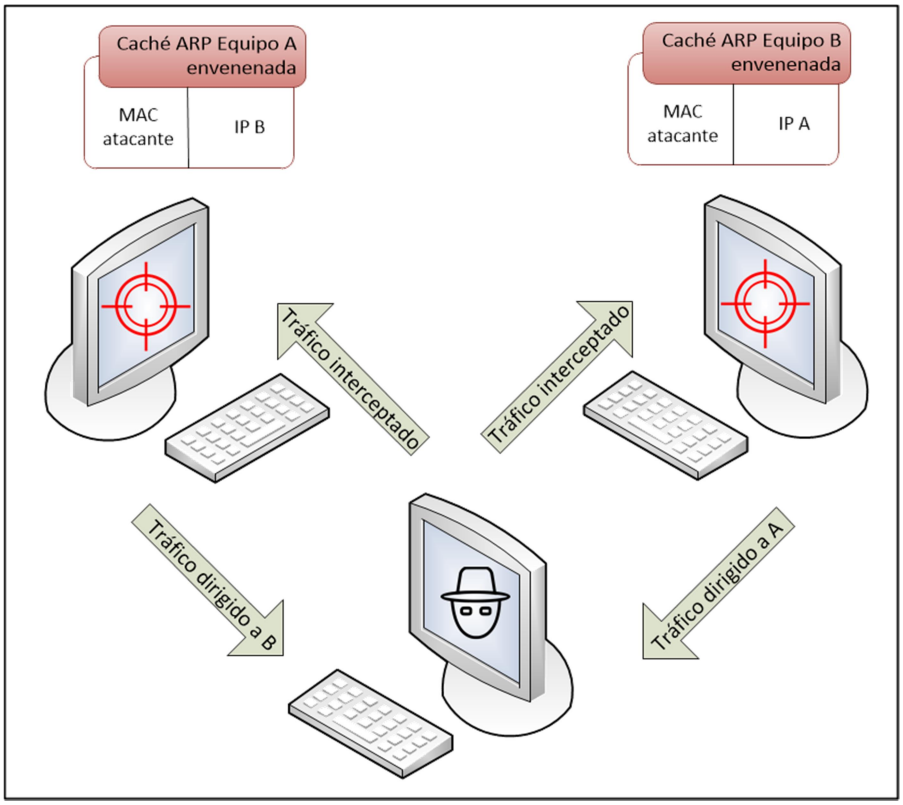

Figura 2: Diagrama de un tradicional ataque MitM

Esto puede permitir la captura de los paquetes de red que viajan o se envían de la placa de red. Pero dichos paquetes, en la mayoría de los casos, viajan cifrados y encriptados con protocolos robustos.

Sin embargo, MitB ocurre a más alto nivel, concretamente en la capa de aplicación. Con lo que el ataque podría ser más crítico, cuantas más capas de segmentación y encapsulación sea capaz de escalar, acercándose a las capas más relacionados con las personas. Robando y manipulando información y operaciones directamente de la víctima. El diagrama de un ataque MitB puede verse en la Figura 4.

Los complementos o extensiones del navegador son herramientas muy potentes, que permiten mejorar y facilitar la experiencia de los usuarios durante sus sesiones de navegación, pero, ¿a qué precio?, ¿realizan lo que se espera de ellas?, o dicho de otro modo ¿qué certeza tenemos de que una extensión instalada durante la navegación para, por ejemplo, visualizar un contenido, no realiza otras operaciones más allá de aquello para lo que fue instalada? La respuesta es ninguna.

```
"permissions": [
  "activeTab",
  "contextMenus",
  "storage"
],
```

Como ocurre en otras plataformas, la instalación de una extensión en un navegador, lleva aparejada la aceptación de una configuración (manifest.json), donde se reflejan, entre otros, los permisos de host (sitios web donde la extensión puede trabajar) y los permisos de API (sub-herramientas de software muy potentes). La figura 3 muestra la lista de permisos contenida en una archivo manifest.json.

Figura 3: Permissions [9]

Un usuario convencional, sin conocimientos específicos, desconoce en muchas ocasiones, que extensiones tiene instaladas en su navegador, como se instalaron, que permisos tienen, para hacer que cosas. Además, las herramientas anti-malware, por lo general, no están preparadas para analizar en profundidad el navegador y sus extensiones.

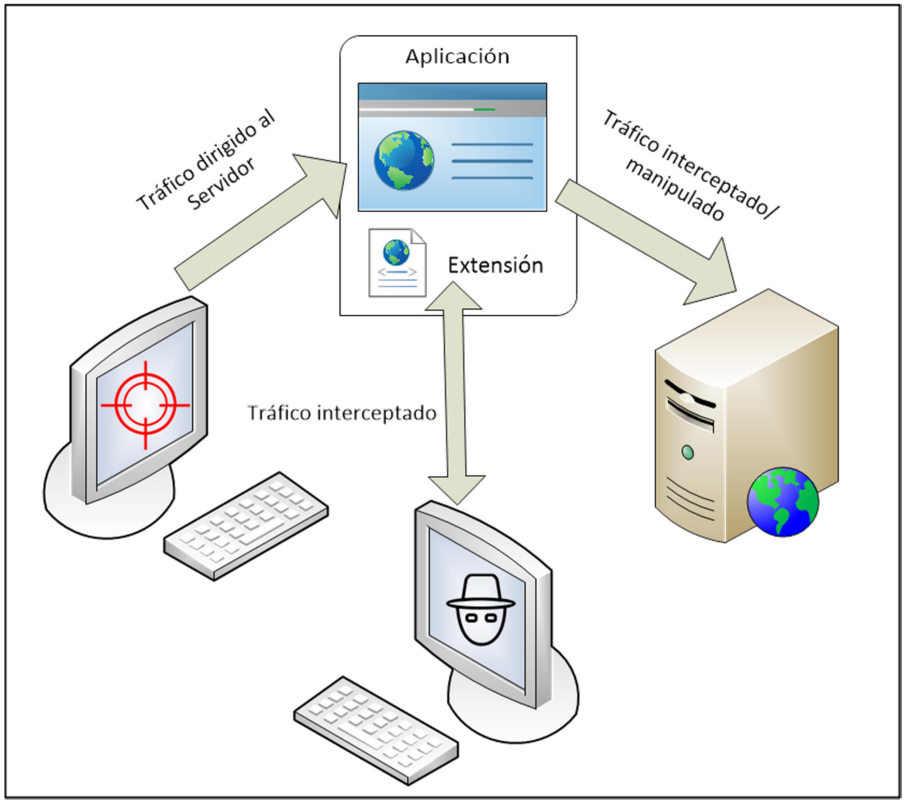

Figura 4: Diagrama de ataque MitB

Todos estos factores, crean el caldo de cultivo ideal para que, cualquier atacante o ladrón de información, pueda desplegar un ataque del tipo MitB empleando las extensiones del navegador.

Pues bien, en el desarrollo de este trabajo, se pretende utilizar el potencial que tienen dichas extensiones para desplegar el ataque MitB, con el fin de demostrar la vulnerabilidad y grado de exposición de los usuarios en la red.

2.1 Planificación

La idea principal es que las capturas o esnifes obtenidos durante una sesión de navegación de un equipo objetivo, sean enviados a un servidor controlado por un adversario. En líneas generales los pasos seguidos han sido:

1. Estudiar, recabar información y comprender este patrón de ataque.
2. Valoración y estudio de los diferentes navegadores del mercado, para el desarrollo de la extensión.
3. Elección del servidor del adversario y tecnologías del mismo (base de datos, entorno web).
4. Desarrollo de una extensión que pueda capturar las URL que visita el objetivo y su envío al servidor, para ser guardadas en una base de datos.
5. Más adelante se dota a la extensión de más funcionalidad como capturar las pulsaciones de teclas, clicks de ratón, la ip del equipo objetivo.
6. Finalmente, se incluye funcionalidad para interactuar con el servidor adversario, a través de un panel de control alojado en dicho servidor, pudiendo interferir, bloquear y redirigir la navegación.
7. Se probará la extensión y se evaluarán pruebas de funcionamiento.

Las etapas del proyecto y su duración se observan en el siguiente gráfico. Figura 5.

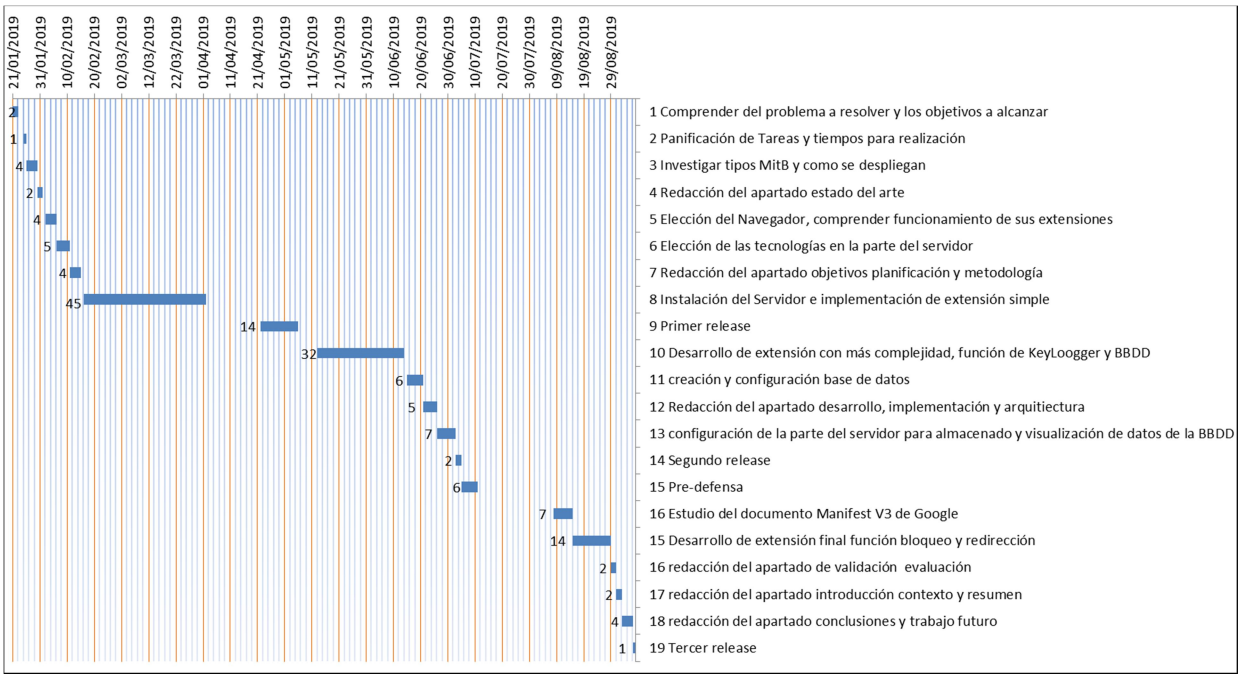

Figura 5: Evolución del proyecto en tiempo

El proyecto se encuentra disponible en el repositorio de github, del Master de Ciberseguridad y Privacidad de la Universidad Rey Juan Carlos [10]. Tres versiones.

2.1.1 Primera versión [Release 1]

Extensión sencilla que captura URL del navegador donde se instala, la envía a un servidor donde es almacenada en un fichero *fichero1.txt*. Paralelamente, a través de una página *panel.php* dentro del mismo servidor, se puede monitorizar en tiempo real la URL que contiene el fichero. Los ficheros son:

1. LastUrl (directorio con la extensión descomprimida)
   1.1. manifest.json (fichero de manifiesto)
   1.2. LastUrls.js (extensión de fondo)
2. panel.php (fichero de servidor con panel de control)
3. gate.php (fichero de servidor, receptor de datos de la extensión)

2.1.2 Segunda versión [Release 2]

Extensión con algo más de complejidad, se incluye un script de contenido para capturar teclas y clicks de ratón *KeyLogger.js*, un módulo JavaScript con funciones de envío al servidor *SendModule.js*. Además en la parte del servidor, funcionalidad para almacenamiento en base de datos, obtención de la IP del origen. Y en la página panel.html monitorización de teclas y clicks de ratón y registros de la base de datos. Los ficheros son:

1. LastUrl (directorio con la extensión descomprimida)
   1.1. manifest.json (fichero de manifiesto)
   1.2. LastUrls.js (extensión de fondo)
   1.3. SendModule.js (módulo de envío)
   1.4. KeyLogger.js (script de contenido que captura teclas y clicks de ratón)
   1.5. background.html (redirección a la extensión de fondo)
2. panel.html (fichero de servidor con panel de control)
3. gate.php (fichero de servidor, receptor de datos de la extensión)
4. tablaUrl.php (fichero de servidor, consultas en base)
5. tablaKey.php (fichero de servidor, consultas en la base)

2.1.3 Tercera versión [Release 3]

Extensión con funcionalidad de redirección y bloqueo de sitios webs. A través del panel de control del lado del servidor se pueden mandar sitios para bloquear o redireccionar. Los ficheros son:

1. LastUrl (directorio con la extensión descomprimida)
   1.1. manifest.json (fichero de manifiesto)
   1.2. LastUrls.js (extensión de fondo)
   1.3. SendModule.js (módulo de envío)
   1.4. KeyLogger.js (script de contenido que captura teclas y clicks de ratón)
   1.5. background.html (redirección a la extensión de fondo)
2. panel.html (fichero de servidor con panel de control)
3. gate.php (fichero de servidor, receptor de datos de la extensión)
4. tablaUrl.php (fichero de servidor, consultas en base)
5. tablaKey.php (fichero de servidor, consultas en la base)
6. data.php (fichero de servidor, recibe datos de dominio y URL, para bloqueo /redirección)

2.2 Metodología

En esta sección se describen de manera general las tecnologías empleadas para el desarrollo del trabajo.

2.2.1 Google Chrome

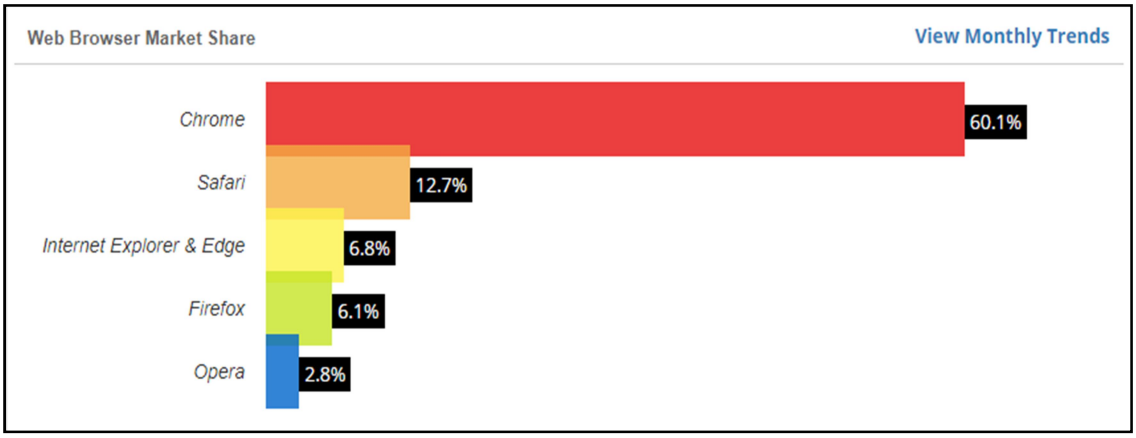

Figura 6: Cuota de mercado de navegadores, datos de Abril 2019 [11]

Google Chrome es el navegador elegido, por los siguientes motivos.

- Es uno de los navegadores de escritorio más usados y populares. La figura 6 muestra la cuota de mercado de navegadores.
- Posible migración de la extensión de Chrome a Firefox.

Por tanto, la idea es hacer la extensión lo más general y utilizable posible, con la posibilidad de migrarla a Firefox en un futuro.

2.2.2 JavaScript

Los lenguajes de programación empleados para el desarrollo de extensiones suelen ser CSS y JavaScript. Los navegadores más populares como Mozilla Firefox y Google Chrome utilizan JavaScript para sus extensiones. JavaScript es un lenguaje interpretado orientado a la programación web, es decir, no requiere de previa compilación y es interpretado por el navegador web.

- Orientado a Objetos.
- Tipado Dinámico.

2.2.3 Ubuntu Server

En el otro extremo se ha optado por usar para el servidor Ubuntu Server 18.04. Distribución de Linux basada en Debian, especialmente orientada al uso de servidores, sobretodo servidores web. Concretamente este trabajo se ha realizado con una Máquina Virtual de dicho sistema operativo.

- Sin entorno gráfico.
- Económico y ligero a nivel de recursos.

2.2.4 Apache

Potente Servidor web HTTP de código abierto multiplataforma, implementa el protocolo HTTP/1.1.

- Soporte de seguridad SSL y TLS.
- Puede realizar autentificación de datos utilizando SGDB.
- Puede dar soporte a diferentes lenguajes, como PHP, Python….

2.2.5 PHP

Acrónimo recursivo de PHP, Hypertext Preprocessor, es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web.

- Puede ser incrustado en HTML.
- Se ejecuta en el lado del servidor.

2.2.6 HTML

HyperText Markup Language, lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software para elaboración de páginas web en sus diferentes versiones.

- Define el contenido de una página web.
- Creación de marcos, campos, contenedores y botones.

2.2.7 MySQL

Sistema de gestión de bases de datos relacional, considerado como la base de datos de código abierto más popular del mundo.

- Amplio subconjunto del lenguaje SQL.
- Orientada a entornos de desarrollo web.

2.2.8 phpMyAdmin

Herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando un navegador.

- Puede crear y eliminar Bases de Datos, tablas, campos y ejecutar cualquier sentencia SQL.
- Interfaz Web para la gestión gráfica.

La siguiente figura muestra la estadística del código usado en el presente trabajo

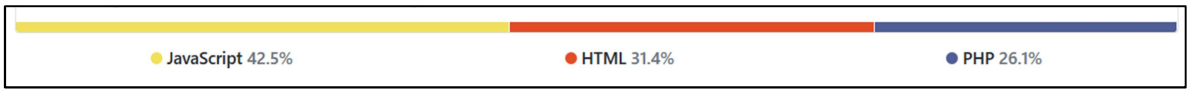

Figura 7: Estadísticas de código del proyecto en github

3 Estado del arte

Para este trabajo, se han analizado los malware bancarios más importantes basados en MitB que nos aportan una idea de que funciones y características específicas pueden desplegar, como se comportan y cuáles son sus vectores de ataque, como se infectan y propagan. La presente información se ha obtenido del siguiente informe de OWASP. [12]. En la Figura 8 podemos observar un resumen de los vectores de ataque de Malware MitB más relevantes.

Spyeye es considerado el sucesor de Zeus y considerado globalmente como el kit de Banking Malware más avanzado que se utiliza realmente. Este kit fue concebido como botnet fácil de administrar a través de un panel de control basado en la web. Se basa en los ataques MiTB (Man in The Browser) para realizar su tarea, proporciona un archivo de configuración cifrado personalizado con complementos, código de inyección web, lista de coleccionistas (envían los datos robados). SpyEye es capaz de inyectar código HTML.

Zeus es un troyano bancario identificado por primera vez en 2007, diseñado como Botnet basado en HTTP específicamente diseñado para robar credenciales de banca en línea. A pesar del hecho de que ZeuS Kit ya no se desarrolla, tiene una difusión notable. La funcionalidad del kit ZeuS se basa en ataques MiTB, un archivo de configuración encriptado contiene disparadores de URL y código HTML para ser inyectado.

Tatanga apareció en el primer semestre de 2011 como un troyano basado en MiTB, diseñado para robar credenciales de banca en línea y falsificación (Post Transaction Attack) del saldo real de la víctima. Al igual que los troyanos vistos anteriormente, también Tatanga utiliza archivos de configuración cifrados (3-DES) para almacenar complementos y código de inyección web. Además, Tatanga es capaz de tomar direcciones de correo electrónico, eliminar las botnets de los competidores, infectar archivos para aumentar la propagación de malware y eliminar software antivirus.

Urlzone es un troyano bancario que apareció en 2009, su característica principal es la capacidad de ocultar la evidencia del fraude cambiando sobre la marcha el saldo mostrado a la Víctima. Para lograr el robo de dinero. Urlzone utiliza un enfoque clásico de MiTB.

Sunspot apareció por primera vez a finales de 2011 como un troyano basado en MiTB diseñado para robar credenciales de banca en línea. Inyección de código de navegador, keyStroke Logger, capacidades de captura de pantalla, roba información personal confidencial necesaria para llevar a cabo ataques de suplantación de identidad del usuario, buenas capacidades de anulación de antivirus.

Oddjob se descubrió públicamente por primera vez el 22 de febrero de 2011, la característica peculiar de Oddjob es la capacidad de mantener abierta la sesión de la víctima incluso después de cerrar sesión, esto implica que los delincuentes podrán robar dinero al suplantar a la víctima tocando el ID de sesión. Oddjob funciona mediante la inyección de código malicioso en los navegadores Internet Explorer y Firefox, el código está contenido en archivos de configuración personalizados que intercepta las solicitudes GET y POST, inyección de código HTML a través del enfoque MiTB y secuestro de sesión.

Trojan	Attack Vectors										Type	
	Http Injection	Browse Redirect	Form Grabbing	Stored Passwords Theft	KeyStroke Logging	Bypass MFA	Video/Stream Capture	Certificate Theft	Install Backdoor	Instant Messaging	Automatic	Manual
MB -> MiTB MM -> MiTM B -> Both O -> Other	MB	MM	B	B	B	B	O	O	O	O		
ZeuS	*	*	*	*	*	*	*	*	*	*	*	*
SpyEye	*	*	*	*	*	*	*	*	*	*	*	*
Carberp	*		*	*		*	*				*	*
Tatanga	*					*			*			*
Urlzone	*		*		*	*	*		*		*	*
Gozi	*				*							*
Shylock	*	*				*					*	*
Sunspot	*		*		*		*					*
Haxdoor	*		*		*				*			*
SilentBanker	*	*	*		*	*	*	*	*			*
Limbo	*		*	*					*			*

	Actual Threat
	Less Spreaded
	Not Active (2012)*

Above scheme is an updated schema inspired by Marco Morana's Threat Modelling of Malware-Based Attacks Presentation

*still a threat but unseen for a while

Figura 8: Vectores de ataque de Malware MitB [12]

Ramnit es un prolífico malware que muestra una amplia gama de transformaciones durante su arco de existencia, entre estas variaciones también está el robo financiero. Ramnit es esencialmente un troyano de puerta trasera con la capacidad de realizar también ataques MiTB. Capacidades de puerta trasera, infectar archivos Office, ejecutables de Windows, comunicación segura de C&C con SSL, capacidades de anulación de antivirus y capturador de cookies.

Cridex se convirtió en una amenaza muy extendida para las actividades de Home Banking en el primer período de 2012. El malware generalmente se entrega a través de correos electrónicos maliciosos que contienen enlaces abreviados a los sitios web de BlackHole Exploit Kit. Permite descargar y ejecutar archivos, subir y buscar archivos, robar certificados locales, capacidades de MiTB.

Tinba significa TinyBanker, la definición se deriva de las dimensiones reducidas, aproximadamente 20 KB, del binario. Tinba confía en el ataque MiTB (Man in The Browser) y tiene un importante nivel de invisibilidad a la tecnología AntiVirus. El malware se inyecta en los siguientes procesos del sistema: explorer.exe, svhost.exe.

4 Desarrollo

En este apartado se describe con más detalle y en profundidad el trabajo realizado, explicando las técnicas métodos y eventos utilizados y desarrollando que operaciones y funciones realiza cada fichero del proyecto.

4.1 Implementación

A continuación se describen los métodos, eventos utilizados y cuál es la función de cada uno de ellos para la parte de la extensión. En la parte del servidor se explican los ficheros que realizan las operaciones de captura, almacenamiento y monitorización, además de la estructura de la base de datos MySQL.

4.1.1 Extensión

En su fichero manifest.json, la extensión tiene la siguiente tabla de permisos, uso en todos los host <all_urls> y diferentes permisos de APIS. Obsérvese la figura 9.

```
"permissions": [
  "tabs",
  "webRequest",
  "webRequestBlocking",
  "<all_urls>",
  "storage"
],
```

Figura 9: Permisos de la extensión

De todos, quizás el más importante es, la API Chrome. webRequest que permite observar y analizar el tráfico e interceptar, bloquear o modificar solicitudes en vuelo. Es una API muy potente y utilizada por bloqueadores de publicidad o contenidos para adultos, entre otros. La figura 10 muestra el ciclo de vida de una solicitud web y los eventos utilizables. Además con webRequestBlocking damos permiso explicito para bloqueo.

Chrome.tabs nos permite controlar el comportamiento de las pestañas, podremos activar, por ejemplo, eventos para detectar su cierre actualización, cambio de pestaña activa.

Chrome.storage nos permite acceder al almacenamiento local para pasar datos, desde KeyLogger.js hasta el script de fondo.

Como parte interesante a mencionar en el fichero manifest.json es la inclusión de un script de contenido *KeyLogger.js*. Un script de contenido será inyectado en todas las páginas que se visitan desde el navegador donde se encuentra la extensión.

4.1.1.1 chrome.WebRequest

Concretamente se usa el evento *onBeforeRequest* que se dispara cuando una solicitud está a punto de ocurrir y antes de cualquier tipo de conexión TCP. Lo que permite, entre otras, cancelar o redirigir solicitudes.

Este evento es utilizado en el script de fondo *LastUrls.js* dos veces.

- Una tiene con oyente todas las <all_urls> intercepta los sitios web visitados por el objetivo.
- Dos el oyente es modificado en tiempo de ejecución en función del valor obtenido del servidor, sirve para recibir el parámetro con el dominio que se debe bloquear o redireccionar.

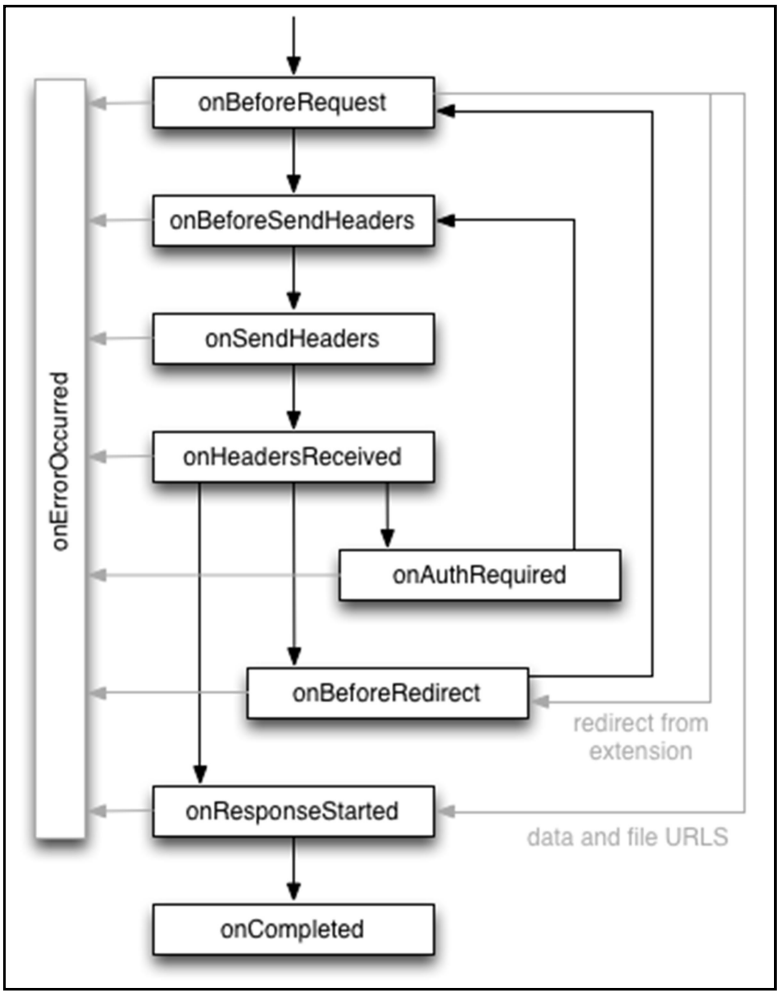

Figura 10: Ciclo de vida de una solicitud web [13]

4.1.1.2 chrome.Tabs

Utilizado también en el script de fondo *LastUrls.js*. Se utilizan los eventos *onRemoved* para detectar el cierre de una pestaña y *onActivated* que detectará el cambio de pestaña activa, en esos dos casos se mandará al servidor los datos contenidos en el local storage (recibidos por *KeyLogger.js*). También se

usa el método *getSelected* del que nos valemos para obtener la URL visitada por el objetivo. Además del método *sendMessage* que envía un mensaje al script de fondo para que borre su variable local.

4.1.1.3 chrome.storage

- Se usa en el script de fondo con los métodos *get* para obtener los datos del *KeyLogger*, *clear y remove* que limpiaran y borraran el almacenamiento con cada toma de datos.
- Desde el script de contenido *KeyLogger.js* se usa con el método *set* que irá introduciendo los datos de teclas y clicks.

4.1.1.4 window.addEventListener

Se usa para detectar eventos en las ventanas, no requieren de un permiso explícito en el *manifest.json.*

- En el script de fondo emplea los parámetros *beforeunload* y *load* que detecta el cierre y la creación de una ventana respectivamente, entonces comprueban el almacenamiento local y vacía los datos.
- En el script de contenido emplea los parámetros *keydown* y *click* que detecta las teclas pulsadas y clicks de ratón en la ventana, introduciéndolos en el almacenamiento local.

4.1.1.5 chrome.runtime.onMessage

Utilizado por el script de contenido *KeyLogger.js* para recibir la orden de borrado de su variable local *cadena[].*

4.1.1.6 XMLHttpRequest

Objeto AJAX que se encargará de enviar los datos al servidor adversario y recibir del mismo los datos para la función de bloqueo y redirección. Se declara en dos partes de la extensión.

- En el módulo de envío *SendModule.js* utilizado para el envío de los datos al servidor.
- En el script de fondo para recibir los parámetros de bloqueo y redirección, en este caso serán actualizados los oyentes y parámetros de la función para el evento *onBeforeRequest* encargado de esta funcionalidad.
-

4.1.2 Servidor

Bajo el entorno web proporcionado por apache a través de la ip se accede a los ficheros que complementan el proyecto, en resumen.

- Cuatro ficheros .php para recibir los datos y guardarlos en la base de datos y otros ficheros, además de realizar consultas en la base.
- Otros cuatro ficheros .txt para los datos de las capturas inmediatas y los parámetros de bloqueo y redirección.
- Un fichero .html que es el panel de monitorización y control del servidor, para interactuar con la extensión.

Además una base de datos MySQL denominada *LastUrls* con dos tablas que almacenan los snifes del objetivo.

- *Tecla* almacena en formato string las cadenas con teclas pulsadas y clicks de ratón, además de la ip origen, tabid origen y la hora del evento.
- *Url* almacena en formato string las urls visitadas, además de la ip origen, tabid origen y la hora del evento.

Para creación y configuración de la base de datos y sus estructuras se ha utilizado la herramienta phpMyAdmin, en el repositorio del proyecto se encuentra un pequeño tutorial con los pasos a seguir.

4.2 Arquitectura

En esta sección se detalla con más profundidad las operaciones y funciones que realiza cada fichero del proyecto. Las dividiremos en dos tipos en función de la funcionalidad.

4.2.1 Captura de contenido

La figura 11 muestra un diagrama que representa de modo gráfico la función de Captura de contenido.

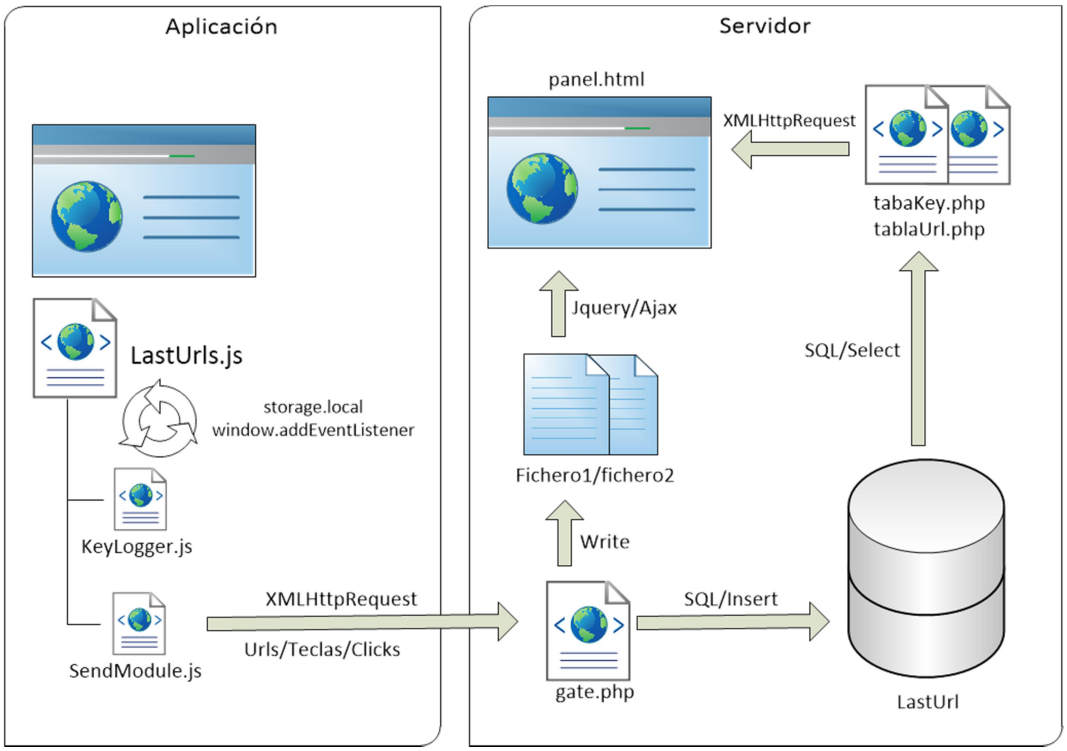

Figura 11: Diagrama del proyecto para captura del contenido

4.2.1.1 El servidor adversario

gate.php: recibe los datos que envía la extensión y guarda las Urls en *fichero1.txt* y en la tabla *Urls* de la base *LastUrl*, y guarda los datos del *KeyLogger* en *fichero2.txt* y en la tabla *Tecla* de las base *LastUrl.*

panel.html: monitoriza la última URL, IP, identidad de la Pestaña (IdTab) y la hora, almacenadas en *fichero1.txt* y las últimas teclas/clicks del *KeyLogger*, IP, identidad de la Pestaña (IdTab) y la hora, almacenadas en *fichero2.txt* . Más abajo se pueden realizar consultas a la base de datos, mostrando las últimas X capturas o todo el conjunto de datos de las tablas *Tecla* y *Urls*.

tablaKey.php: Consulta la tabla *Tecla* de la base de datos y muestra los datos solicitados en formato tabla, los parámetros del número de registros los obtiene de *panel.html.*

tablaUrl.php: Consulta la tabla Urls de la base de datos y muestra los datos solicitados en formato tabla, los parámetros del número de registros los obtiene de panel.html.

4.2.1.2 El equipo objetivo

LastUrls.js: actúa como módulo principal, dispara un evento justo antes de producirse una solicitud web, capturando la URL. Además recibe los datos que el *KeyLogger* ha dejado en el almacenamiento local. Pasa los datos anteriores al módulo de envío.

KeyLogger.js: es un script de contenido que se inyecta en todas las páginas, dispara eventos que capturan cada tecla y cada click de ratón y los almacena en storage.local.

SendModule.js: tiene la función sendData que se encarga de recibir los datos del módulo principal y a través del objeto AJAX XMLHttpRequest, enviarlos por método POST al servidor. La url del servidor adversario, donde se envían los datos se encuentra codificada en base64, para evitar cualquier referencia en claro dentro del código.

4.2.2 Bloqueo y Redirección

La figura 12 representa un diagrama de modo gráfico para la función de Bloqueo y Redirección.

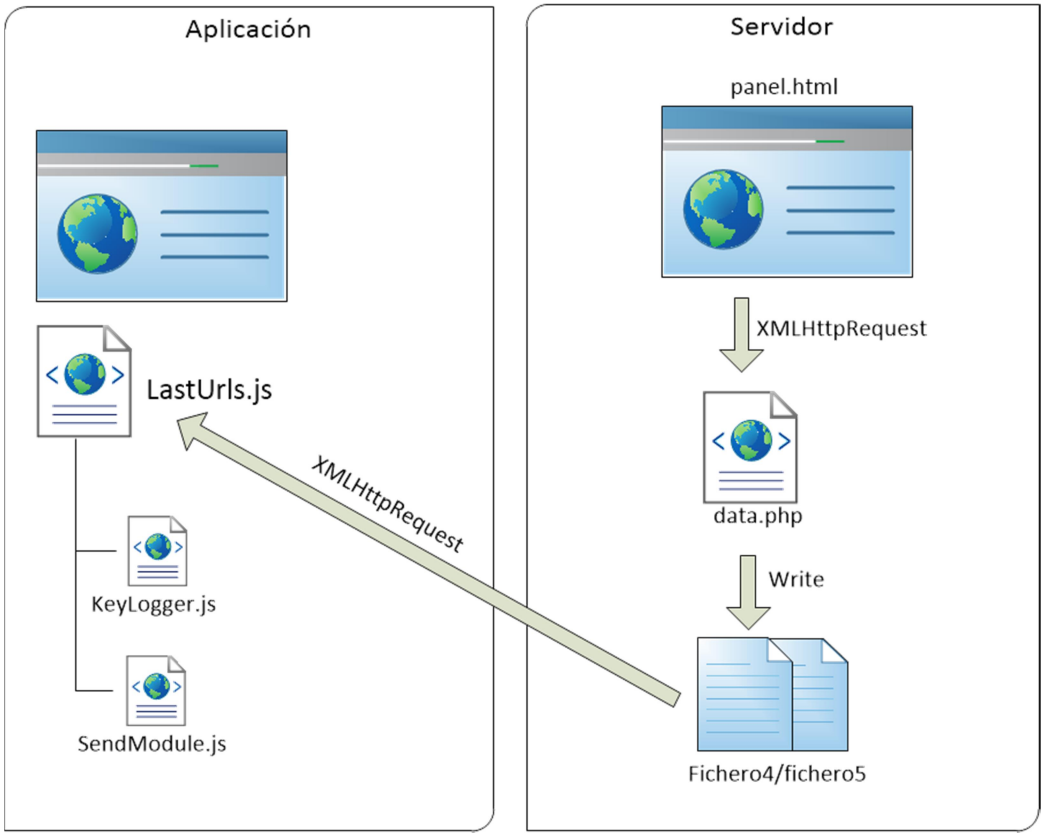

Figura 12: Diagrama del proyecto para Bloqueo y Redirección

4.2.2.1 El servidor adversario

panel.html: Tiene dos campos para introducir el dominio que desea ser bloqueado o redireccionado, y la url hacia la que se redirecciona.

data.php: recibe los datos que envía panel.html sobre el dominio y url para bloquear y redireccionar y los guarda en *fichero3.txt* y *fichero4.txt.*

4.2.2.2 El equipo objetivo

LastUrls.js: Actualiza cada segundo si ha cambiado el contenido de los *fichero3.txt* y *fichero4.txt,* actualizando los datos para las solicitudes web de estos sitios, procediendo a su bloqueo o redirección, según el caso.

SendModulo.js: También contiene las url de los ficheros 3 y 4 codificadas en Base64. Para evitar cualquier referencia en claro dentro del código.

5 Validación y evaluación

Al realizar navegación en Google Chrome desde el equipo objetivo (donde se haya instalado la extensión) se comprobará desde otro navegador en la dirección del servidor del adversario *http://IP/panel.html* como se registran los datos en tiempo real (sitios visitados y teclas y clicks pulsados). Con cada transición y cambio de URL en el navegador del objetivo, deberían mostrarse los datos en *http://IP/panel.html*. Además a través de la parte inferior de esa página se pueden consultar los datos almacenados. Obsérvense las figuras 13 y 14.

149	192.168.1.38	{"IdTab":155,"Keys":"["CLICK"]","Time":"3/9/2019 23:13:22"}
148	192.168.1.38	{"IdTab":155,"Keys":"["CLICK","f","a","c","b","o","o","Backspace","Backspace",
147	192.168.1.38	{"IdTab":174,"Keys":"["CLICK","a","f","a","f","Tab","a","d","f","a","f","Enter","C
146	192.168.1.38	{"IdTab":166,"Keys":"["n","v","e","n","t","o"," ","d","e"," ","b","o","a","d","i","l",
145	192.168.1.38	{"IdTab":166,"Keys":"["CLICK","c","o"]","Time":"3/9/2019 23:09:58"}
144	192.168.1.38	{"IdTab":166,"Keys":"["CLICK"]","Time":"3/9/2019 23:09:53"}

Figura 13: datos mostrados en panel.html tabla *Tecla*

Adicionalmente se puede comprobar el contenido de las últimas capturas directamente en los ficheros *fichero1.txt, fichero2.txt.* Y las capturas almacenadas a través de consultas en el panel de phpMyAdmin.

Para la función de bloqueo se debe rellenar el campo *sitio* con el dominio a bloquear (no es necesario completar un pattern de URL valido, lo creará el servidor, es decir bastaría con meter *youtube.com*, *amazon.es*....) el campo de *redirección* se quedará en blanco. Todas las solicitudes realizadas desde el navegador objetivo a esos dominios serán bloqueadas.

980	192.168.1.38	{"IdTab":150,"Url":"https://www.elcorteingles.es/bio/coldplay/musica/","Time":
979	192.168.1.38	{"IdTab":155,"Url":"https://www.facebook.com/campaign/landing.php?&campa extra_2=campaignid%3D1555704227%26adgroupid%3D57976163406%26mat %3D%26param1%3D%26param2%3D&gclid=EAIaIQobChMI6Iy89si15AIVyI
978	192.168.1.38	{"IdTab":155,"Url":"https://www.facebook.com/campaign/landing.php?&campa extra_2=campaignid%3D1555704227%26adgroupid%3D57976163406%26mat %3D%26param1%3D%26param2%3D&gclid=EAIaIQobChMI6Iy89si15AIVyI

Figura 14: datos mostrados en panel.html tabla *Urls*

Para la función de redirección además de completar el campo *sitio* de la manera anterior, se debe completar el campo *redirección*. En este caso de debe introducir un URL completa, que debe esta estar alojada en el servidor adversario u otro servidor. Todas las solicitudes realizadas desde el navegador objetivo a el dominio del campo *sitio* serán redireccionadas a la URL del campo *redirección*.

De la versión uno a la versión dos se pensó en separar las opciones de envío y comunicación con el servidor en un módulo independiente llamado *SendModule.js*. Para poder usar un módulo fuera del script de fondo con el Estándar ES6 se debía crear un script de fondo en .html y redireccionarlo al script *LastUrls.js*.[14]

Se incluyó entonces una función que utilizaba XMLHttpRequest y que contaba con todo lo necesario para el envío al servidor, incluyendo la URL de destino codificada en base64. Esta función *sendData* se exportaba fuera del módulo con la idea de importarla tanto en el script de fondo *LastUrls.js*, como en el de contenido *KeyLogger*.js y, de esta manera cada módulo fuera capaz de enviar sus snifes. Sin embargo, desde este último no se podía acceder a dicha función, solo podía comunicarse con el script de fondo. Entonces se optó por el uso de la misma función desde el propio script de contenido, pegándola directamente en su código. Eso genera un bloqueo de contenido en el propio navegador (Figura 15).

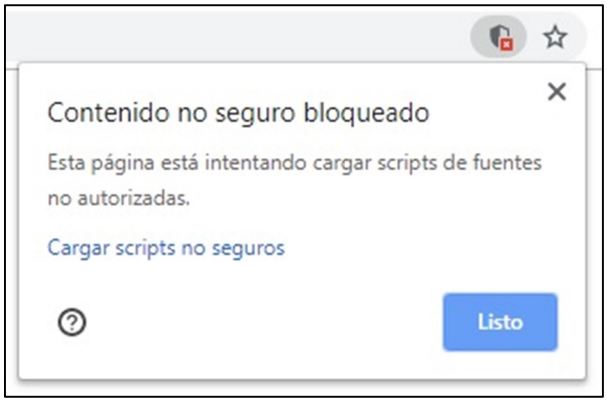

Figura 15: bloqueo de contenido al mandar datos desde un script de contenido

Con algo más de detalle en la consola web se observa el siguiente mensaje, donde se informa que un sitio cargado en HTTPS está haciendo una solicitud a un sito HTTP (Figura16).

```
▸Mixed Content: The page at 'https://www.google.es/' was loaded over HTTPS, but   VM55:1
requested an insecure XMLHttpRequest endpoint 'http://192.168.1.42/panel.php'. This
request has been blocked; the content must be served over HTTPS.
```

Figura 16: bloqueo de contenido en consola web.

En ese punto se optó por la utilización del almacenamiento local para mandar los datos del *KeyLogger.js* hacia el script de fondo y que este último se encargara tanto de sus envíos de URL como de los envíos de teclas y clicks provenientes del *KeyLogger*.

6 Conclusiones

El objetivo del proyecto queda alcanzado, se ha podido comprobar cómo, con una simple herramienta, se pueden realizar operaciones tan sensibles y críticas, pasando desapercibidas a un usuario convencional y herramientas Anti-Malware.

En el presente trabajo no se tiene en cuenta la aplicación o técnica que se debería desplegar para infectar el navegador objetivo. Obviamente mediante el engaño un adversario podría crear un sitio web fraudulento que informe sobre la falta o necesidad de un plugin o complemento, tipo FlasPlayer, Java….descargando e instalando la extensión. Pero, como digo, se escapa del objeto del trabajo.

En cuanto a la Versión 3 del Manifest de google, ¿Qué impacto tendría en el presente trabajo? Pues según anuncian en su documento, restringirán las capacidades de bloqueo de la API de webRequest. Dando como alternativa, para realizar bloqueos, la API declarativeNetRequest, basada en listas declarativas dinámicas, pero es una versión Beta (no definitiva). Sin embargo, en el apartado API Changes/ Summary del documento dice:

"In Manifest V3, we will strive to limit the blocking version of webRequest, potentially removing blocking options from most events (making them observational only). Content blockers should instead use declarativeNetRequest (see below). It is unlikely this will account for 100% of use cases (e.g., onAuthRequired), so we will likely need to retain webRequest functionality in some form."

Por lo que no deja claro que lo restrinjan del todo, al no cubrir el 100 de los casos de uso. ¿Por lo menos lo mantendrán de manera observacional? Quizá eso si permite el envío de la Url y las teclas pero no el de bloqueo y redirección.

Un método que se utiliza es tabs.getSelected (obsoleto desde Chrome 33) y que parece que desaparecerá en esta nueva versión. Sin embargo existen alternativas como tabs.getCurrent o tabs.Query.

En cuanto a la parte de permisos de manifiesto, se deben diferenciar en dos partes, los permisos de host y los de API, lo que no tendría un impacto significativo en el trabajo.

Sin embargo, aparte de los permisos que se aceptan en el momento de la instalación de la extensión (figura 17), se pretende que los permisos de host los dé el usuario en tiempo de ejecución pudiendo permitir la extensión en algunos sitios, en todos o ninguno, y potenciando el uso de permisos de activeTab, limitando por tanto, el de <all_urls>. Todo ello viene derivado por la poca penalización que hay para el uso de permisos para todas las urls, los desarrolladores no quieren que el usuario esté aceptado permisos con cada web que visitan.

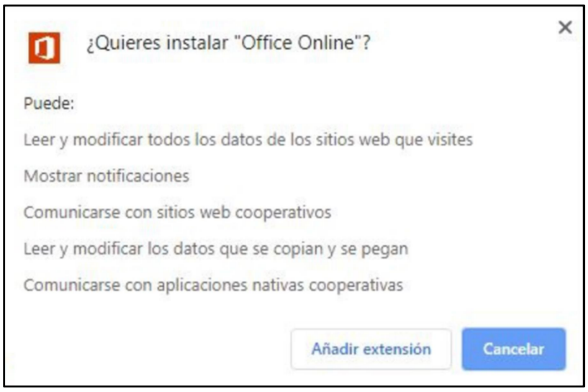

Figura 17: Solicitud de permisos en la instalación de una extensión

Según google el 80 % de las 1000 extensiones principales solicitan el acceso a todos los dominios.

Esto sí tendría un impacto significativo en este trabajo, ya que alertaría al usuario constantemente de que la extensión quiere acceder a un host. Pienso que se dificulta, de manera considerable, el MitB planteado en este proyecto con la nueva versión de manifiesto de google.

6.1 Trabajo futuro

Adaptación o migración del trabajo a la versión 3 del manifiesto de Google, cuando este sea definitivo y se aclaren los conceptos, que parece que aún no quedan claros.

El desarrollo de una herramienta que detecte estos comportamientos extraños en el navegador objetivo, partiendo de la base del uso de las funciones de AJAX para la comunicación fuera del contexto local. Podría consistir en una misma extensión anti MitB.

Migración de la extensión al navegador de Mozilla Firefox, existe un asistente dentro del sitio para desarrolladores de Firefox [15].

Referencias

[1] ¿Cómo era navegar por la web en 1990? [Último acceso Marzo 2019]

[2] Concepts against Man-in-the-Browser Attacks [Último acceso 26 Enero 2019]

[3] Una extensión "Man-in-the-middle" para Chrome [Ultimo acceso 28 Enero 2019]

[4] Detectamos una extensión activa desde febrero en Chrome Web Store que roba tarjetas de crédito [Último acceso Febrero 2019]

[5] Nuevo troyano bancario basado en una extensión de Chrome afecta a entidades Latinoaméricanas [Último acceso 30 Julio 2019]

[6] Historia de la Web: los navegadores [Último acceso 3 Febrero 2019]

[7] Página desarrolladores extensiones de Chrome [Último acceso 3 Mayo 2019]

[8] https://docs.google.com/document/d/1nPu6Wy4LWR66EFLeYInl3NzzhHzc-qnk4w4PX-0XMw8/preview#heading=h.xgjl2srtytjt [Último acceso agosto 2019]

[9] Permisos Chrome [Último acceso septiembre 2019]

[10] GitHub del proyecto completo [Último acceso 07 septiembre 2019]

[11] Cuota de uso de Navegadores por meses [Último acceso Mayo 2019]

[12] OWASP_Anti-Malware_-_Knowledge_Base [Último acceso 02 Febrero 2019]

[13] Chrome.webRequest [Último acceso Septiembre 2019]

[14] ES6 modules in chrome extensions [Último acceso 1 Junio 2019]

[15] Porting a Google Chrome extension [Último acceso 6 Septiembre 2019]