

UNIVERSIDAD NACIONAL AUTÓNOMA DE
MÉXICO

FACULTAD DE CIENCIAS



Tarea Semana 4:
User Agents

Miguel Concha Vázquez
416062401

Tarea presentada en cumplimiento con la asignatura de Análisis de
Software Malicioso impartida por el profesor
JONATHAN BANFI VÁZQUEZ
24 de febrero de 2019

Índice

1 Desarrollo y replicación de resultados	2
1.1 Inicio de las máquinas virtuales	2
1.2 Inicio del servidor web	2
1.3 Configuración para los <i>User Agents</i>	3
1.4 Habilitado del <i>plugin</i>	6
1.5 Ejecución del <i>script</i> y de Wireshark	7
1.6 Consulta del cliente	8
1.7 Creación de nuevos registros de <i>user agents</i>	9
1.8 Peticiones con distintos valores de <i>User Agents</i>	11
1.8.1 <i>IE</i>	11
1.8.2 <i>Google Chrome</i>	13
1.8.3 <i>Mozilla Firefox</i>	14
1.8.4 <i>Safari</i>	15
1.8.5 <i>Opera</i>	16
2 Cuestionario	17
3 Conclusiones	18
4 Referencias	19

1. Desarrollo y replicación de resultados

1.1. Inicio de las máquinas virtuales

Comenzamos abriendo **VMware** e iniciando las máquinas virtuales a partir del *snapshot* del primer estado. En este caso se usaron:

- **Debian 7** como el servidor web en donde se configuraron el */var/www/UserAgents/index.html* y se crearon las páginas para los distintos navegadores. Desde aquí se monitoreó el tráfico de red con **Wireshark** y se ejecutó el *script fakedns.py*.
- **Windows 7** para representar al cliente que consulta el sitio web. Dado que tiene que realizar la consulta DNS, esta es resuelta por el *script* al estar en la misma red local; el script le devuelve la dirección lógica IP de la máquina con Debian.

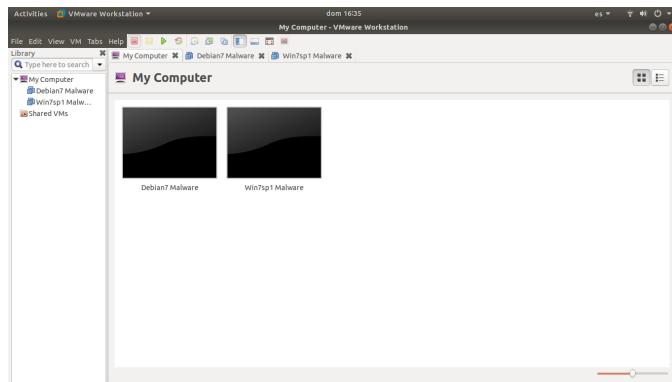


Figura 1: Iniciando VMWare.

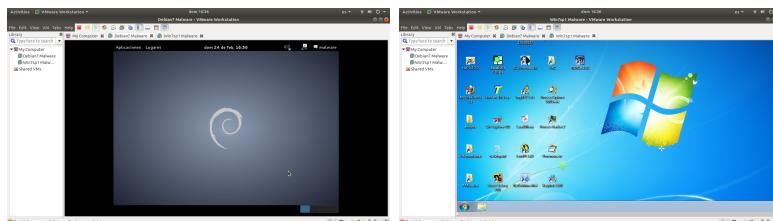


Figura 2: Máquinas virtuales: **Debian 7** y **Windows 7**.

1.2. Inicio del servidor web

Primero nos convertimos en **root** con el comando **sudo su** y escribiendo la contraseña *malware*.

En la máquina con **Debian 7** se procedió a iniciar el servidor web **Apache2**. Primero se usó el comando **netstat** con opciones **-nat** para poder ver la lista de todas las conexiones activas en el equipo en cuestión, cerciorándose de que el servicio no estaba aún activo. Posteriormente, se inició con el comando **/etc/init.d/apache2 start** y se comprobó que se hubieran reflejado los cambios al usar nuevamente **netsat -nat**.

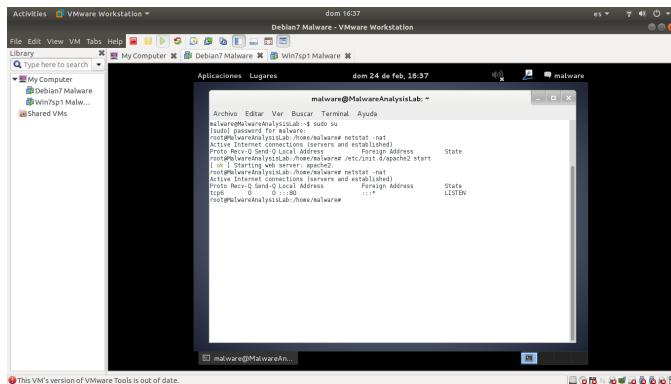


Figura 3: Iniciando VMWare.

1.3. Configuración para los *User Agents*

Luego, se hizo un listado del contenido del directorio web **/var/www** y nos movimos al directorio de los navegadores **/var/www/UserAgents**, listando luego su contenido. Pudimos ver en consola el contenido actual de los tres archivos para los navegadores *Internet Explorer*, *Google Chrome* y *Mozilla Firefox*:

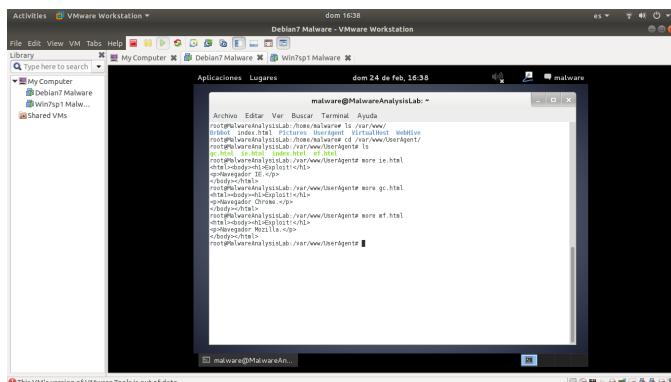


Figura 4: Revisando el contenido de los archivo en `/var/www/UserAgents`.

Después procedí a crear dos nuevos archivos para los *user agents* de *Safari* (`sf.html`) y *Opera* (`op.html`), cambiando luego los permisos de todos los archivos con la extensión `html`:

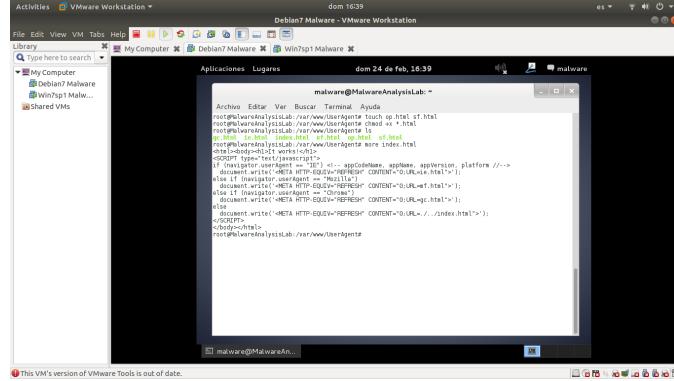


Figura 5: Creando los nuevos archivos, cambiando los permisos y revisando el contenido actual de `index.html`.

Hecho esto, continué modificando el archivo de `index.html` con tal de poder validar ahora las nuevas cadenas de los *user agents*:

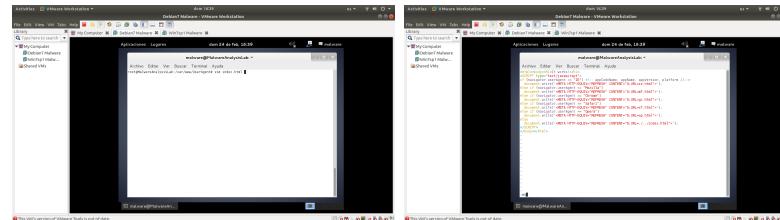


Figura 6: Agregando validaciones para *Safari* y *Opera*.

Finalmente, uno por uno fui editando los archivos:

- `ie.html`
- `gc.html`
- `mf.html`
- `sf.html`
- `op.html`

Lo cual se muestra a continuación:



Figura 7: Editando el *html* de los archivos para los distintos *user agents*. Los archivos fueron incluidos en el directorio `htmls/` del comprimido proporcionado.

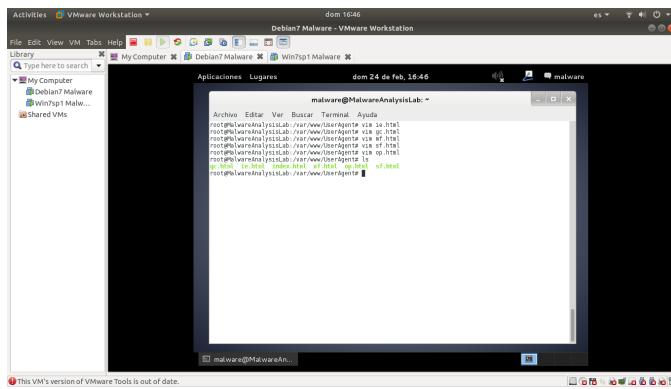


Figura 8: Viendo que todo estaba en orden una vez terminado el proceso anterior.

1.4. Habilitado del *plugin*

En la otra máquina virtual se procedió a abrir el navegador *Mozilla Firefox*:

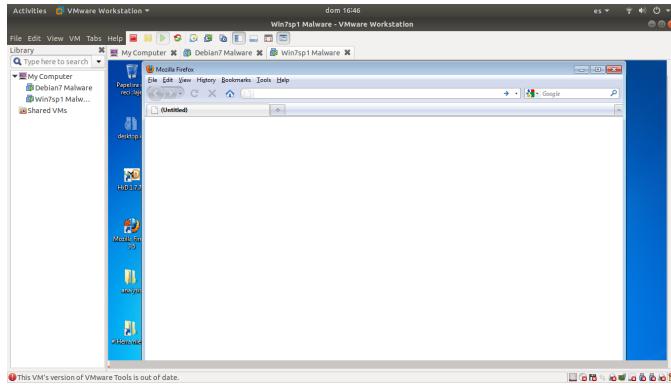


Figura 9: Ejecutando el navegador web.

Luego, dando *click* en **Tools->Add-ons** pudimos ver la ventana de *plugins*:

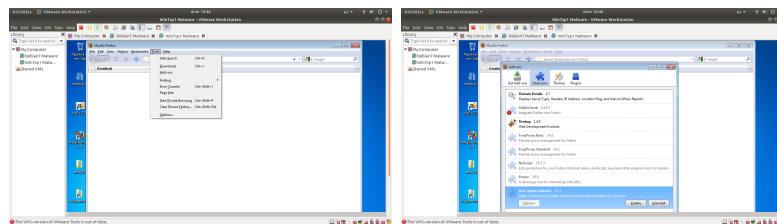
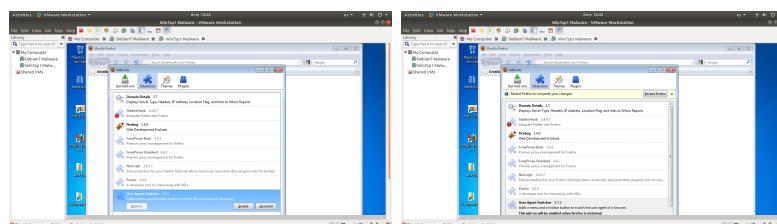


Figura 10: Accediendo a las extensiones del navegador.

Una vez ahí, habilité la extensión de *User Agent Switcher* dando *click* en **Enable**; para reflejar los cambios se tuvo que reiniciar el navegador web:



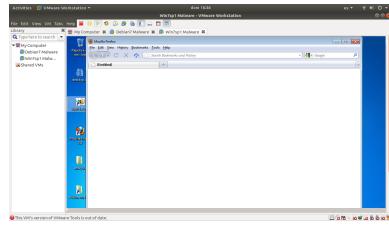


Figura 11: Habilitando la extensión y reiniciando *Firefox*.

1.5. Ejecución del *script* y de Wireshark

En la máquina con Debian 7 luego se ejecutó el *script* de Python, *fakedns.py*:

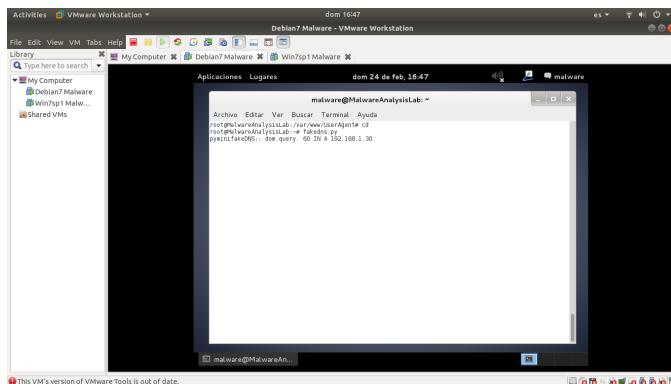


Figura 12: Ejecución del *script*.

Después, se comenzó un proceso para Wireshark, el *sniffer* de red para poder analizar el tráfico:

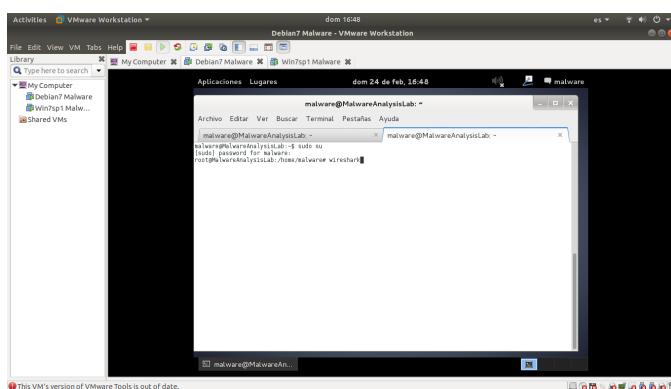


Figura 13: Abriendo Wireshark.

Para comenzar a analizar el tráfico de red fue importante ejecutarlo con credenciales administrativas, de modo tal que nos permitiera ver y elegir las interfaces de red. Elegí la interfaz de red de área local (LAN) de Ethernet `eth0`, después de lo cual solamente fue necesario dar *click* en **Captura->Start**:

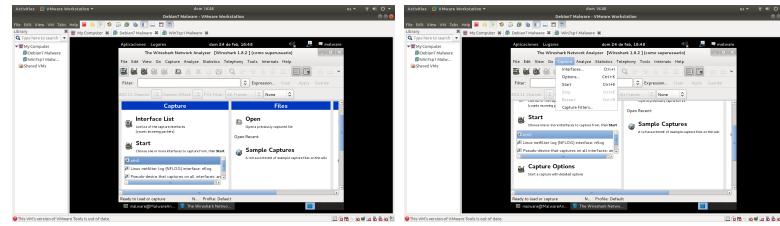


Figura 14: Seleccionando la interfaz de red e iniciando la captura de tráfico.

1.6. Consulta del cliente

Con la máquina del cliente, usando el navegador *Mozilla Firefox* procedimos a realizar una consulta a un sitio web:

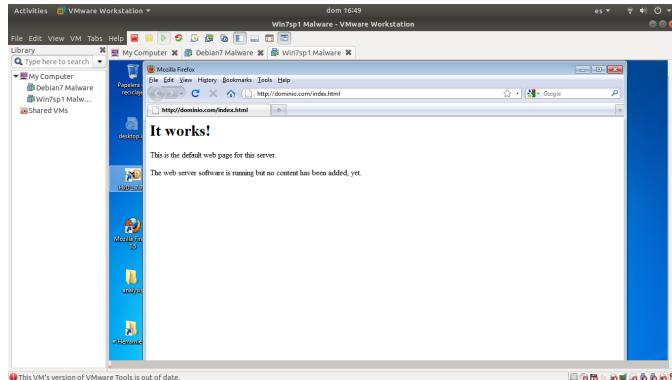


Figura 15: En este caso consultamos `http://dominio.com/index.html`.

Por otro lado, una vez hecho lo anterior, en el otro equipo pudimos dar un seguimiento a la petición desde *Wireshark* al hacer un seguimiento del flujo de TCP (*TCP stream*), dando *click* derecho y seleccionando **follow TCP stream**:

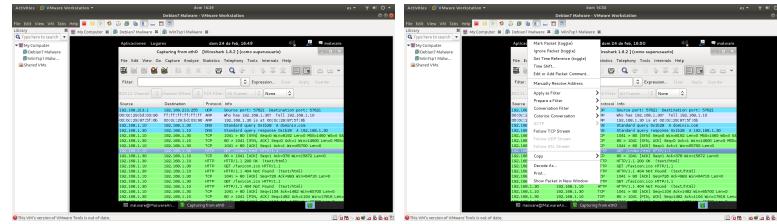


Figura 16: Siguiendo el flujo para analizarlo más a detalle.

Pude observar que en efecto el *user agent* de la petición correspondía al navegador web que había usado, *Mozilla Firefox*:

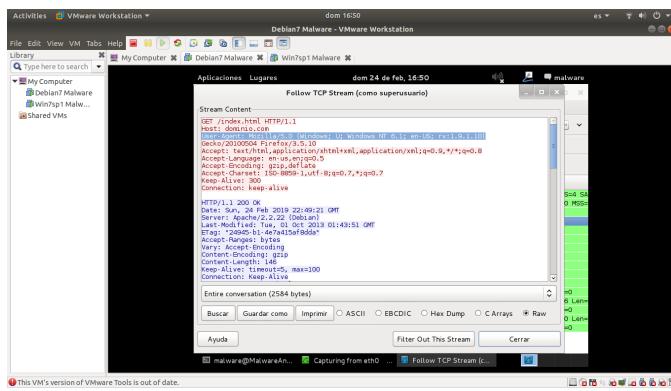


Figura 17: El *user agent* de la consulta era el adecuado.

1.7. Creación de nuevos registros de *user agents*

En el cliente, aprovechando el *plugin* habilitado antes, procedí a crear nuevos registros para otros navegadores web. Primeramente creé una carpeta PoC para agruparlos a todos, dando *click* inicialmente en **Tools->Default User Agent->Edit User Agents...**:

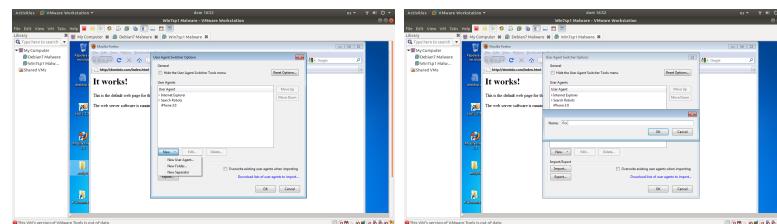


Figura 18: Creación de la carpeta (New->NewFolder)

Después creé nuevas entradas dentro de la carpeta, una para cada *user agent*, con New->New User Agent...:

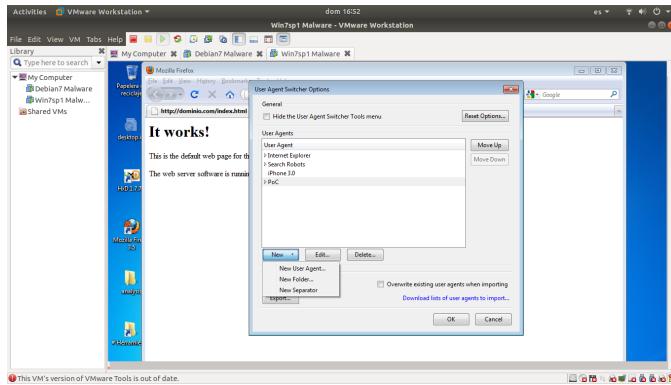


Figura 19: Creación de cada entrada.

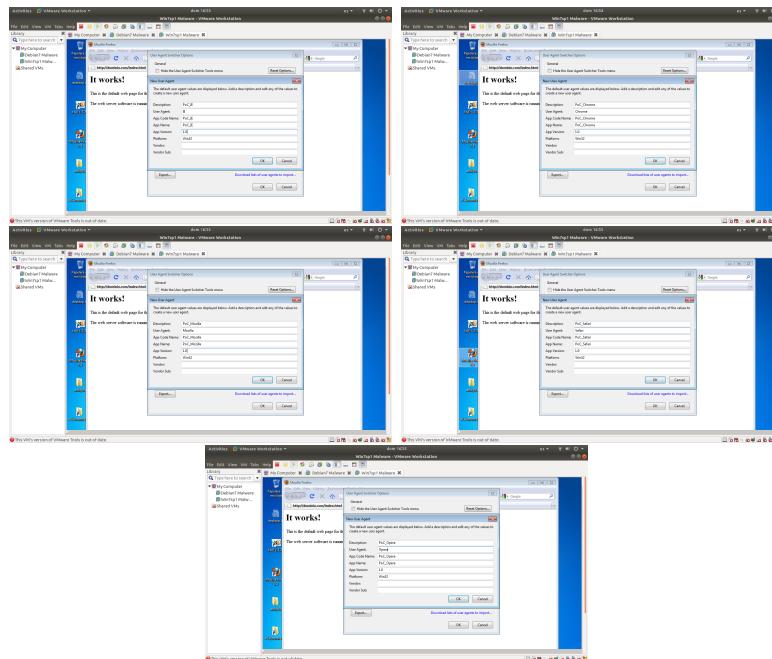


Figura 20: Creación cada una de las entradas dentro de la carpeta PoC.

Dando como resultado al final:

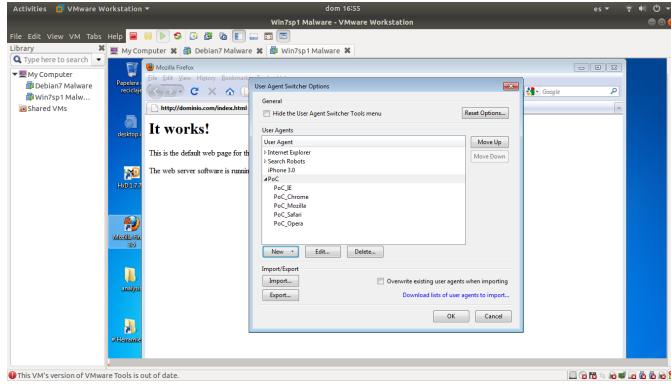


Figura 21: Todas las entradas para los *user agents* creadas.

1.8. Peticiones con distintos valores de *User Agents*

A partir de este punto comencé a elegir distintos valores del *user agent* para el cliente con ayuda del *plugin* y los registros recién creados, viendo que al acceder a <https://www.dominio.com/UserAgent/index.html> aparecía el contenido `html` correspondiente al valor de la cadena que coincidía tal como habíamos configururado desde un inicio.

1.8.1. IE

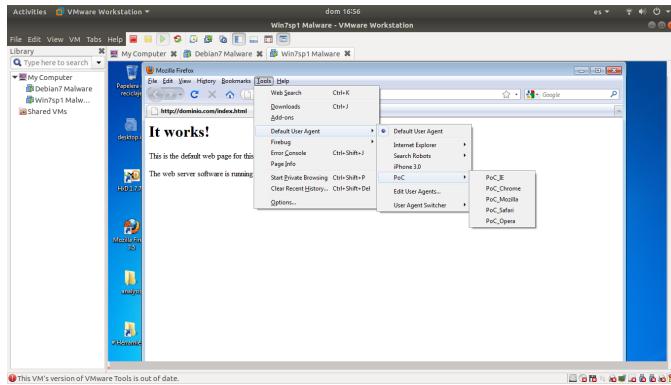


Figura 22: Cambiando el valor del *User Agent*.

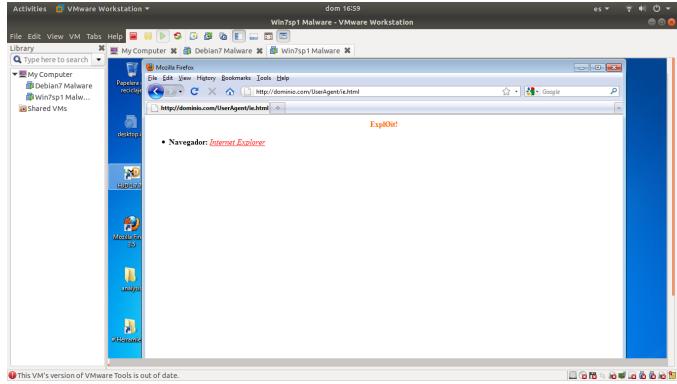


Figura 23: Consultando <https://www.dominio.com/UserAgent/index.html>. Vemos que nos redirige al *Index* personalizado.

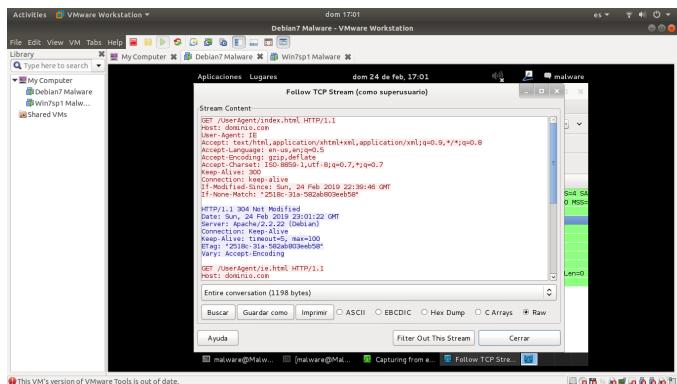


Figura 24: Verificando el tráfico de red capturado con Wireshark el valor de la cadena para el *User Agent*.

1.8.2. Google Chrome

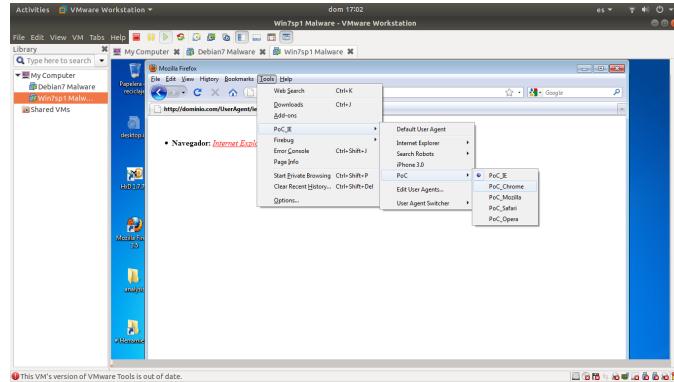


Figura 25: Cambiando el valor del *User Agent*.

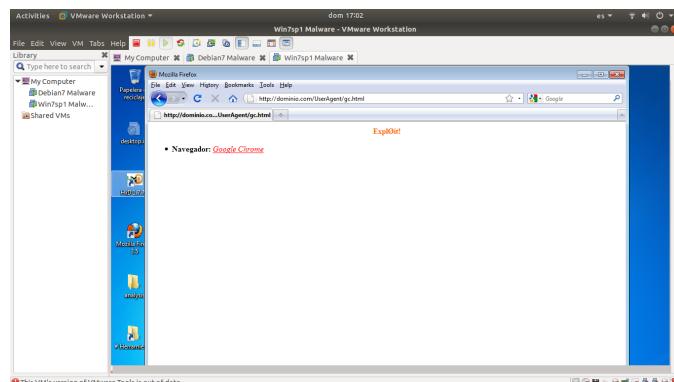


Figura 26: Consultando <https://www.dominio.com/UserAgent/index.html>. Vemos que nos redirige al *Index* personalizado.

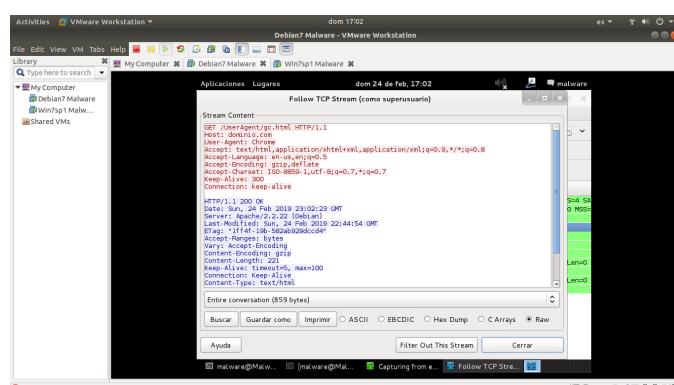


Figura 27: Verificando el tráfico de red capturado con **Wireshark** el valor de la cadena para el *User Agent*.

1.8.3. Mozilla Firefox

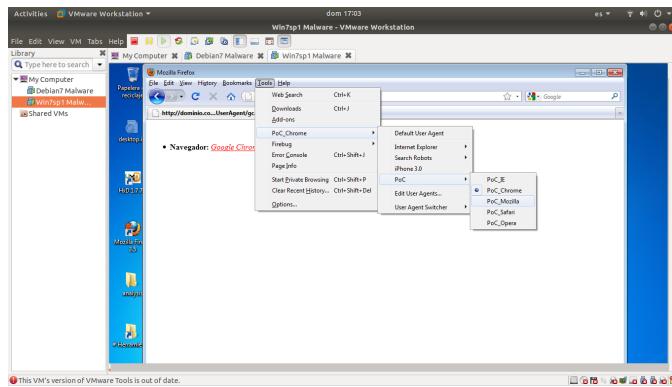


Figura 28: Cambiando el valor del *User Agent*.

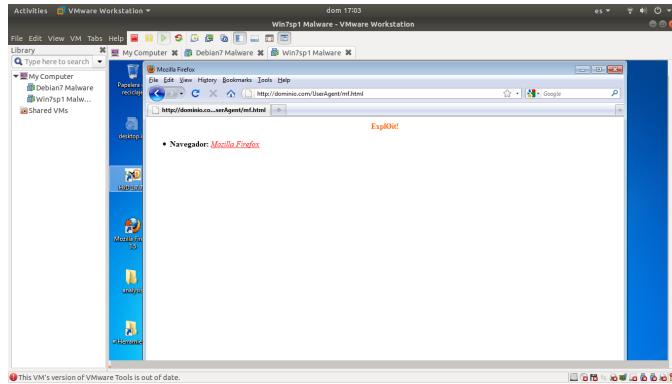


Figura 29: Consultando <https://www.dominio.com/UserAgent/index.html>. Vemos que nos redirige al *Index* personalizado.

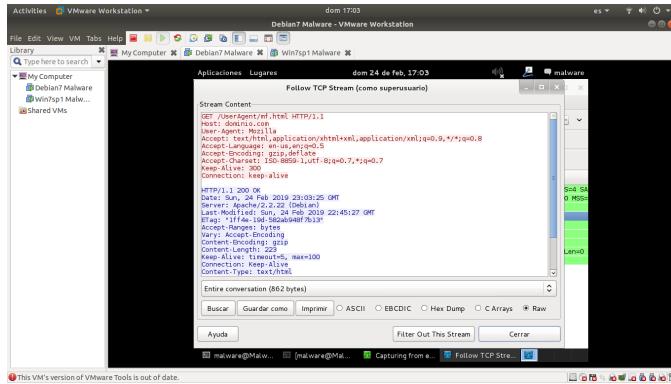


Figura 30: Verificando el tráfico de red capturado con Wireshark el valor de la cadena para el *User Agent*.

1.8.4. *Safari*

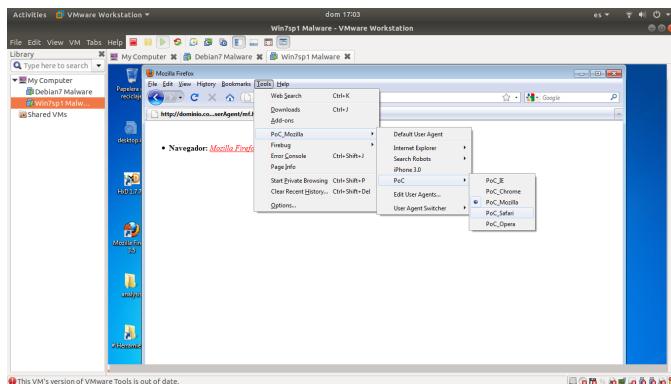


Figura 31: Cambiando el valor del *User Agent*.

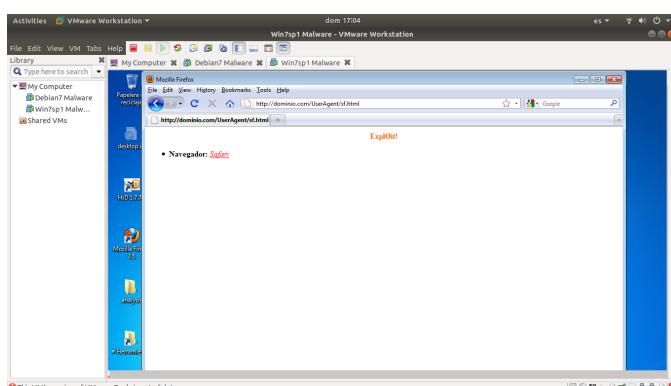


Figura 32: Consultando <https://www.dominio.com/UserAgent/index.html>. Vemos que nos redirige al *Index* personalizado.

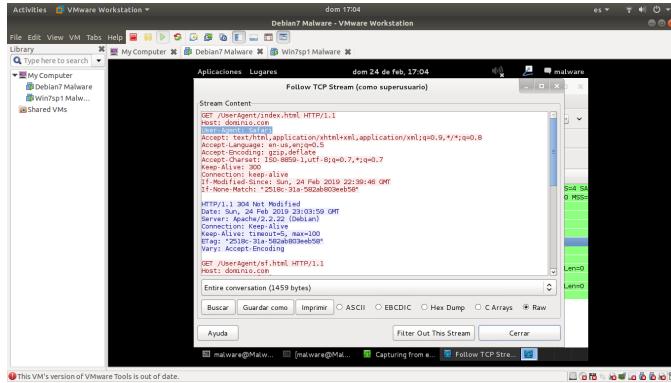


Figura 33: Verificando el tráfico de red capturado con Wireshark el valor de la cadena para el *User Agent*.

1.8.5. *Opera*

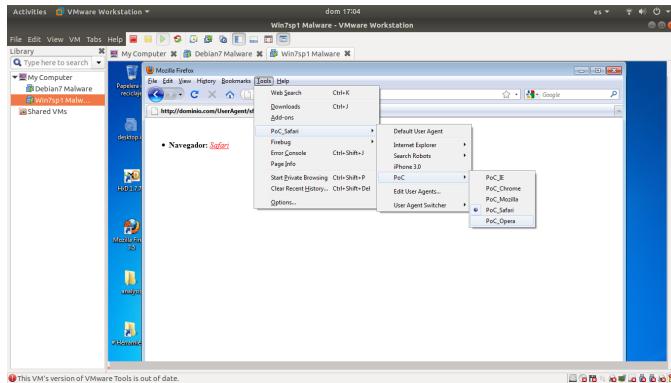


Figura 34: Cambiando el valor del *User Agent*.

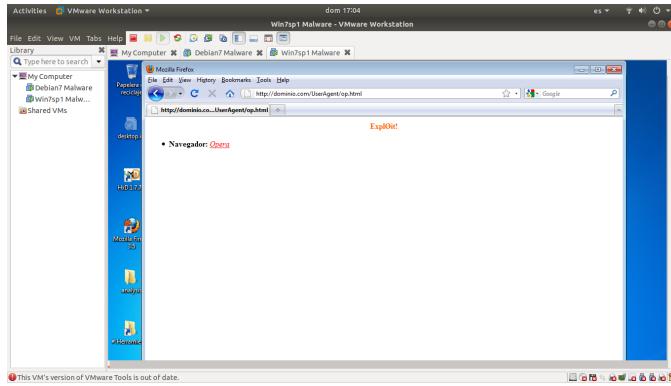


Figura 35: Consultando <https://www.dominio.com/UserAgent/index.html>. Vemos que nos redirige al *Index* personalizado.

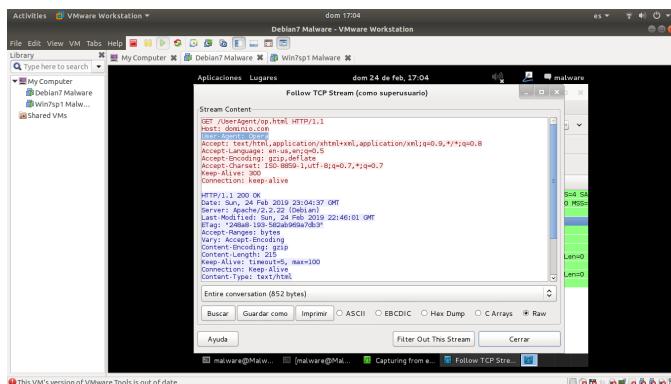


Figura 36: Verificando el tráfico de red capturado con Wireshark el valor de la cadena para el *User Agent*.

2. Cuestionario

Ejercicio 5

1. ¿Cómo se puede saber si se está siendo víctima de un programa como `fakedns.py`?

Respuesta:

El *script* en cuestión debe ser ejecutado con permisos administrativos, tiene su uso dentro del campo del análisis de malware y se utiliza para poder controlar las respuestas del pequeño servidor de nombres de dominio (*DNS*)

Server) que constituye. En este caso cualquier petición por un nombre de dominio que tiene que resolverse a una dirección IP, se logra regresando una dirección IP prefijada —en nuestro caso la dirección de la máquina con **Debian 7** en el laboratorio, el servidor web—, que sería el valor del registro de tipo A que se devuelve ante cualquier petición.

En un contexto real, se pueden tener ataques de DNS **spoofing** que sirven para poder alterar los valores de los registros de un servidor DNS y así poder en consecuencia controlar lo que se devuelve ante las consultas de nombres de dominio de una víctima, similar a lo que ocurre con el *script* al preguntar por cualquier nombre de dominio. Esto podría hacerse con un *script* como en nuestro caso o alternado por ejemplo el archivo `/etc/hosts`, el primer punto de consulta de una máquina para evitar así que tenga que recurrir a un servidor DNS para resolver nombres de otras máquinas.

Estos ataques pueden representar riesgos grandes para empresas e individuos, pero por fortuna pueden ser detectados de diversas formas:

- Al revisar los procesos activos, podría notarse por ejemplo que algo del tipo `fakedns.py` está trabajando.
- En caso de que un atacante nos esté redirigiendo a un sitio falso para continuar así con un ataque (por ejemplo, para hacer *phishing*), podrían checarse los certificados del sitio y por qué organismo fueron expedidos.
- Se pueden usar herramientas como *Catchpoint DNS Experience Test*, *dnstraceroute* y *DNS Nameserver Spoofability Test*, mismas que miden el tiempo que se tarda en resolver nombres de dominios emulando al DNS o verifican que las consultas en efecto sean respondidas por el servidor de nombres adecuado. Esto también es útil porque permite ver si uno es vulnerable a estos ataques incluso antes de que se susciten.

3. Conclusiones

El uso de la cadena o *tag* para el *user agent* dentro de las peticiones HTTP (*Hypertext Transfer Protocol*) son realmente útiles en la práctica para que los servidores puedan identificar desde qué navegador web se piden los recursos y así lograr un acuerdo de entrega del contenido, esto es, seleccionar apropiadamente los parámetros de entrega de los recursos solicitados.

También pueden ser utilizados para evitar que algunos metabuscadores (*web crawlers*) accedan a ciertos recursos de un sitio. Gracias a ciertas extensiones es posible cambiarlos desde nuestros navegadores favoritos, además de que el administrador del sitio puede fácilmente configurar las páginas a las que de deberá redireccionar al cliente con base en los valores de estas etiquetas.

Es importante tener en cuenta que existen vulnerabilidades en torno a estos conceptos. En caso de que el sitio web se vea comprometido, el atacante podría aprovechar para realizar ciertos ataques basándose justamente en la información proporcionada por los encabezados de HTTP, como el del navegador web, por ejemplo. A su vez, existen vectores vulnerables de servidores web para poder lanzar *bugs* como **shellshock**, que podrían sustituir esta cadena por un *script* de *bash* que podría pasar desapercibido y ser ejecutado.

Finalmente, en esta práctica usamos el *script* de `fakedns.py` (`/Documents/Tools/Others`), lo cual nos llevó a ahonda en ataques que se pueden hacer con el servidor de nombres en mente. Es importante estar atentos a los ataques de *MiTM*, *man-in-the-middle* en todo momento y ver que no se falsifiquen los valores de los registros de los DNSs, además de comprobar que somos redirigidos a sitios confiables.

4. Referencias

Referencias

- [1] Contreras Flores Paulo. “Notas de Clase: Capa de Aplicación”, Facultad de Ciencias, UNAM.
- [2] W3. <https://www.w3.org/TR/cuap>. Consultado el día: 23 de febrero de 2019.
- [3] Beta News, Inbar Raz. <https://betanews.com/2017/03/22/user-agent-based-attacks-are-a-low-key-risk-that-shouldnt-be-overlooked/>. Consultado el día: 23 de febrero de 2019.
- [4] Catchpoint, Nithyanand Mehta . <http://blog.catchpoint.com/2016/03/02/dns->. Consultado el día: 23 de febrero de 2019.
- [5] BugTraq, Ilya Medvedovsky . <https://bugtraq.ru/library/security/dns.html>. Consultado el día: 23 de febrero de 2019.

- [6] Tranquilidad Tecnológica. <http://www.tranquilidadtecnologica.com/2006/04/servidor-fake-dns-en-python.html>. Consultado el día: 23 de febrero de 2019.