**UNIVERSIDAD AUTONOMA DE MADRID**

**ESCUELA POLITECNICA SUPERIOR**



# PROYECTO FIN DE CARRERA
# Ingeniería de Telecomunicación

## Diseño, implementación y análisis de un sistema de detección de fraude online

**José Carlos Corrales Casas**

**Junio 2016**

# Diseño, implementación y análisis de un sistema de detección de fraude online

**AUTOR: José Carlos Corrales Casas**
**TUTOR: Jorge Bueno Moreno**
**PONENTE: Jorge E. López de Vergara Méndez**

**Dpto. Tecnología Electrónica y Comunicaciones**
**Escuela Politécnica Superior**
**Universidad Autónoma de Madrid**
**Junio 2016**

## *Resumen*

Los mecanismos de seguridad convencionales centran sus esfuerzos en crear un perímetro de seguridad que evite que sus sistemas sean atacados. Sin embargo, en la sociedad 2.0 actual se hace imprescindible preocuparse por identificar los ataques que se llevan a cabo a las personas fuera del control tradicional.

Entre estas amenazas a usuarios finales, las herramientas más utilizadas por los atacantes son los ataques de suplantación de identidad, envenenamiento de DNS o *software* malicioso. En todos ellos los ciberdelincuentes utilizan diversas técnicas de planificación y distribución, lo que hace más complicado poder tener visibilidad de los ataques en fase temprana. Este falta de visibilidad a tiempo, implica que numerosos usuarios sean víctimas de los ataques mucho antes de que estos puedan ser identificados por los equipos de seguridad, poniendo de manifiesto la necesidad de algún sistema que permita minimizar estos retrasos y dote de soluciones a los administradores de seguridad para poder mitigar los ataques antes de que varios usuarios hayan sido afectados por los mismos.

En este Proyecto Final de Carrera se pretende cubrir parte de esta carencia diseñando un sistema de detección y análisis de fraude online. Este sistema permitirá detectar los ataques definidos en el párrafo anterior y los filtrará para lograr optimizar el tiempo de gestión de los administradores de seguridad. Para ello se utilizarán diversos mecanismos de detección en paralelo que permitan recolectar información en todos aquellos puntos donde los ataques son más visibles. De este modo se podrá minimizar el tiempo de vida de los ataques y se podrá mitigar el impacto de los mismos.



## *Palabras Clave*

Detección, *phishing, Pharming, malware,* consola de gestión, fraude online.

## *Abstract*

Current security solutions are focused on build a strong perimeter in order to protect their systems from the attackers. However, 2.0 society requires additional solutions which allow security researcher to detect and mitigate attacks against users out of the perimeter.

The most common weapons that are being used by the attackers are the Phishing scams, the pharming scams and the malware scams. In all of them the cibercriminals use different distribution methods and even different planning techniques, because of that is harder to identify the attacks in real time.

This lack of visibility, implies that a lot of users can be compromised before than the attacks are detected by the security administrators. In this scenario, is needed a new mechanism that minimize the delay between the attack distribution and the attack detection.

During this *Proyecto Final de Carrera (PFC)* is going to be designed and implemented a tool to cover this lack, an online fraud detection tool. This system will detect most common fraud attacks, and will process and filter them in order to optimize the threat time management. In order to achieve the project's goals, the tool will include different mechanisms for detecting data. Thereby the security administrators will reduce the lifetime of the attacks and will mitigate their impact.



## *Keywords*

Detection, *phishing, Pharming, malware,* dashboard, online fraud.

# *Agradecimientos*

Me gustaría agradecer en primer lugar a los "Jorges" (Bueno y López de Vergara) por toda la dedicación y esfuerzo empleado en guiarme hacia la consecución de este proyecto. Su experiencia académica y conocimientos técnicos han logrado ayudarme de manera rápida y eficaz en la realización del mismo.

También me parece imprescindible agradecer este logro a mis compañeros y amigos en el trabajo: Ruso, un genio cuya simple presencia me ha ayudado a motivarme en los momentos difíciles y que ha logrado que mi cerebro "*procastinador*" no hiciera de las suyas, y Pablo, mentor y amigo que a base de paciencia ha logrado pulirme hasta hacer de mí la persona que hoy soy.

En el plano más personal se me hace imposible no mencionar en primer lugar a mis familiares, los cuáles me han aguantado, ayudado y empujado a no abandonar cuando parecía lo más fácil.

En este grupo, sin duda, mención especial merece María: la persona que ha tenido a bien sacrificar buena parte de sus vacaciones y de su vida conmigo mientras yo me dedicaba a compatibilizar los estudios, con el trabajo y con la vida personal. Gracias por entenderme y ayudarme a desconectar cuando me hacía falta, pero sobre todo gracias por ser quien me ha dado fuerzas cada día para continuar adelante en esta batalla.

Por último, pero no menos importante por ello, me veo obligado a agradecer este proyecto a mis amigos de toda la vida, todos sin distinción han aportado algo en este difícil camino: risas, viajes, horas de estudio…muchos buenos momentos. Sin embargo hay uno que ha sido especialmente importante para que esté hoy escribiendo este proyecto. Roro, eres un ejemplo y una referencia que admiro pero sobre todo un amigo que siempre ha estado para abrirme una ventana cuando veía las puertas cerradas, GRACIAS POR TODO.

# INDICE DE CONTENIDOS

# INDICE DE ILUSTRACIONES

# INDICE DE TABLAS

# Glosario

| | |
|---|---|
| API | Application Programming Interface |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| IP | Internet Protocol |
| HTTP | Hypertext Transfer Protocol |
| HTML | Hypertext Markup Language |
| DNS | Domain Name System |
| URL | Uniform Resource Locator |
| URI | Uniform Resource Identifier |
| P2P | Peer to Peer |
| PIN | Personal Identification number |
| MIME | Multipurpose Internet Mail Extensions |
| RFC | Request for comments |
| FTP | File Transfer Protocol |
| SMTP | Simple Mail Transfer Protocol |
| ASCII | American Standard Code for Information Interchange |
| ARQ | Automatic Repeat Request |
| OSI | Open System Interconnection |
| BBDD | Bases de datos |
| JSON | JavaScript Object Notation |
| ISP | Internet Service Provider |
| AV | Antivirus Vendor |
| SSL | Secure Socket Layer |

# 1 Introducción

## *1.1 Motivación*

La ciberseguridad tradicional se basa en la creación de un perímetro de seguridad robusto en torno a la compañía, interconectando distintos dispositivos de seguridad activa y de seguridad pasiva. Un muro a priori infranqueable que bien diseñado y administrado supone el escudo perfecto contra los atacantes.

Sin embargo, incluso aceptando como válido el sistema de protección perfecto que realmente no existe, estos sistemas no tienen en cuenta que en la sociedad de la información actual la robustez de todo sistema de seguridad depende de la fortaleza del eslabón más débil, que en la mayoría de los casos son elementos que se encuentran fuera del control total de la compañía: usuarios, empleados, empresas de terceros, etc. Se ha comprobado a su vez que a pesar de una mayor concienciación social en lo referente a fraude en la red, estos ataques cada vez son más exitosos y es debido tanto a la sofisticación en las técnicas de ataque como al incremento en el componente de ingeniería social asociado a los mismos.

Este vector de riesgo no controlado es perfectamente conocido por los ciberdelincuentes, que están centrando sus esfuerzos en atacar a las grandes empresas a través de elementos no controlados directamente por las mismas, lo que pone de manifiesto la necesidad de ampliar también el punto de mira en el lado de la ciberseguridad extraperimetral. Es necesario identificar lo que nos está atacando en el exterior.

Por otro lado es también importante tener en cuenta que la efectividad de estos ataques también se encuentra en la viralidad de los mismos. Existen grandes sistemas de difusión que permiten hacer llegar estos vectores de infección a miles de personas y es por ello que incluso con tasas altas de mitigación a través de los sistemas tradicionales (anti SPAM, antivirus, Firewall, etc.) un pequeño índice de afectados es altamente lucrativo para los atacantes.

Actualmente, según las métricas presentadas por el reputado y reciente informe de Verizon, [1] el 23% de los usuarios pincha en los enlaces de phishing y un 11% ejecuta los adjuntos contenidos en ellos. Lo que unido a los datos facilitados por APWG [2] en el que se observa un incremento sostenido en las muestras de malware distribuidas, 14 millones de nuevos binarios, y los más de 150.000 ataques de phishing detectados durante el último trimestre de 2015, permite tener una idea del alcance de este tipo de riesgos para una compañía.

Los atacantes tienen como método habitual de dispersión de sus ataques el envío de SPAM con enlaces de phishing, malware, etc. Normalmente estos correos de SPAM contienen algún documento adjunto en el que se pretende persuadir al cliente para realizar la descarga de "recibos", "facturas", que realmente son binarios maliciosos. Otro escenario habitual es la notificación de anomalías en la cuenta del cliente y la necesidad del registro de este en su servicio de banca accediendo a través de un enlace contenido en el propio correo.

En este proyecto se pretende diseñar, implementar y analizar un sistema completo que permita detectar en tiempo real ataques de suplantación de identidad y a su vez añadir

análisis de contenido malicioso en los binarios adjuntos en los correos, pudiendo de este modo identificar campañas activas de fraude contra los usuarios.

## 1.2 Objetivos

En el proyecto de fin de carrera planteado se pretende demostrar la mejora significativa introducida en la detección de fraude online utilizado para atacar a los usuarios domésticos. Se pretende ofrecer una identificación rápida y precisa para permitir minimizar el tiempo de vida de los ataques más comunes distribuidos en la red.

Para ello se desplegará una arquitectura basada en varios niveles de detección y procesamiento que permitan realizar un análisis automatizado con el fin de limitar los falsos positivos y reducir la necesidad de análisis manual. La estructura estará formada en la parte de detección por servidores de backend y frontend que simulen una página web corporativa, su correspondiente sistema homólogo replicado por el atacante para realizar la suplantación de identidad y un sistema que permita recibir correos maliciosos.

La estructura de procesamiento se encargará de analizar todo el tráfico recibido en los servidores legítimos y en el sistema de recepción de correos para identificar potenciales ataques y amenazas de fraude online.

La tecnología utilizada estará basada en software libre y estándares conocidos, con el fin de plantear un sistema de bajo coste, replicable y asumible en cualquier infraestructura.

Además, se realizará un procesado en varios niveles para evitar la duplicidad de alertas y realizar filtrados previos en base a listas de reputación, que eviten identificar como ataques websites legítimos.

Finalmente se realizará una prueba de funcionamiento en la que se desarrollaran varios ataques de phishing, y malware, y se analizará como la plataforma es capaz de identificarlos y procesarlos para su posterior gestión por parte de los servicios de respuesta ante incidentes.

Con todo ello se pretende lograr con unos costes contenidos en infraestructura y software una arquitectura de detección temprana de fraude que permita minimizar los tiempos de vida de los ataques y por lo tanto reducir la ventana de exposición de los usuarios.

## 1.3 Organización de la memoria

El resto de la memoria consta de los siguientes capítulos:

- *Estado del Arte.* Se realizará una introducción al contexto del cibercrimen en nuestra sociedad, profundizando en las técnicas más comúnmente utilizadas para lograr comprometer la información de los usuarios. Mediante la comprensión del comportamiento de estos atacantes se plantearan las soluciones a llevar a cabo en el proyecto. También se realizará una labor de documentación preliminar que permita obtener los conocimientos necesarios para el desarrollo de la solución de detección antifraude.

- *Análisis de requisitos.* En este capítulo se analizarán los requisitos necesarios para garantizar unos ratios aceptables de detección y procesamiento en la herramienta. También se evaluaran las distintas alternativas que satisfacen dichos requisitos con el fin de seleccionar aquellos parámetros y tecnologías que demuestren una mayor eficiencia para el proyecto desarrollado.
- *Diseño e Implementación de la herramienta.* A lo largo de este capítulo se mostrará en detalle el diseño e implementación de la solución elegida. Se mostrarán cada uno de los pasos seguidos, estructuras de datos utilizadas, esquemas de funcionamiento y las configuraciones de los dispositivos existentes en el laboratorio.
- *Integración, pruebas y resultados.* En este capítulo se exponen los resultados de las pruebas realizadas con diversos tipos de ataques y se evalúa la eficacia en ratios de detección y procesamiento de la solución.
- *Conclusiones y trabajo futuro.* Este capítulo final resumirá las conclusiones del trabajo realizado y se ofrecerá un planteamiento sobre las líneas de desarrollo para continuar y ampliar la utilidad de la solución.
- *Anexos técnicos.* Estos capítulos ofrecen información complementaria del PFC, tales como los manuales de configuración o del programador.
- *Anexos.* Estos capítulos ofrecen el detalle del pliego de condiciones y el presupuesto de ejecución del Proyecto Final de Carrera.

# 2 Estado del arte

En este capítulo se va a proceder a estudiar el comportamiento de los atacantes y las distintas técnicas utilizadas para comprometer información de los usuarios finales.

También se estudiarán los protocolos de comunicación y sistemas de servicios y seguridad involucrados en la realización de este proyecto: Sistema de log, servidores web, HTTP, análisis estático y dinámico de binarios, Cuckoo.

## *2.1 eCrime - Fraude Online*

La sociedad 2.0 en la que vivimos ha puesto al alcance de un solo clic una gran cantidad de ventajas y funcionalidades de las que disfrutamos sin percatarnos del riesgo derivado de ellas. Hoy en día son millones los usuarios que utilizan la banca electrónica para sus movimientos diarios, la e-administración con el fin de evitar largas esperas e Internet ha sustituido a las bibliotecas como fuente de consulta del conocimiento. En resumen, utilizamos los medios digitales y electrónicos de manera natural y cotidiana y somos muy conscientes de sus beneficios, pero no tanto de sus riesgos.

Estos privilegios y ventajas que ofrece Internet han sido también utilizados por los ciberdelincuentes para obtener un mayor alcance e impacto en sus estafas, hoy en día es posible lanzar ataques que lleguen a millones de personas por un precio casi irrisorio, apenas unos cientos de euros. Este riesgo se ve agravado por la falta de concienciación del usuario medio en lo que a seguridad en la red se refiere creando el caldo de cultivo idóneo para las aparición de bandas organizadas de cibercriminales que han visto en el fraude online un negocio tan lucrativo como las drogas, o la prostitución pero con un "significativo" menor riesgo, como se muestra en la **¡Error! No se encuentra el origen de la referencia.**.

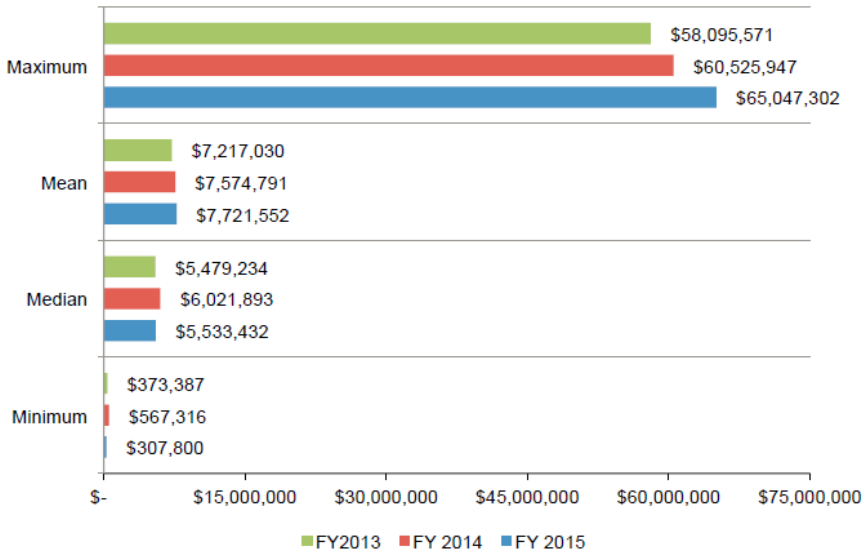



**Ilustración 2-1: Coste del cibercrimen por empresa [3]**

Estas mafias han evolucionado desde los cibercriminales aislados que trabajaban por intereses propios desde sus casas, hasta auténticas empresas del fraude online. Grupos organizados que cuentan con grandes recursos económicos, acceso a técnicos muy cualificados, comunidades propias en las que intercambiar conocimientos y servicios…incluso oficinas en las que sus trabajadores llevan a cabo sus jornadas laborales de Lunes a Viernes como el resto de la sociedad.

En este contexto en el que existen multitud de campañas masivas coexistiendo en el tiempo y el espacio se hace necesario analizar y entender los ataques contra usuarios finales más comunes llevados a cabo en la red: qué son, cómo se propagan y cómo se pueden detectar.

### 2.1.1 *Phishing* o suplantación de Identidad

*Phishing* [4] es un término que denomina un tipo de delito encuadrado dentro del ámbito de las estafas *online*, y que se caracteriza por intentar adquirir información confidencial de forma fraudulenta (como pueden ser contraseñas o información detallada sobre tarjetas de crédito u otra información bancaria). El *phisher* suplanta la identidad de una persona o empresa de confianza en una aparente comunicación oficial electrónica, logrando de este modo los datos del usuario.

Los ataques de *phishing* pueden ser planificados y llevados a cabo de distintas maneras aunque las más común es la detallada a continuación:

1. El atacante recopila información de aquellas cuentas de correo a las que quiere enviar sus ataques. En este punto existen dos opciones: que se trate de ataques dirigidos (*spear phishing*) o por el contrario que se trate de ataques masivos. En ambos casos existen numerosos sitios en internet que permiten disponer de listados completos de emails filtrados y agrupados por sectores, países, compañías, etc, como se muestra en la Figura 2-2 .


**Ilustración 2-2: Ejemplo página de venta de listas de correos**

2. El atacante monta la infraestructura del *phishing*. Para ello puede comprometer infraestructuras vulnerables de terceros, puede montar su propia infraestructura con servidores y dominios dedicados, o puede incluso comprar en el mercado negro lotes de servidores comprometidos. A continuación aloja el contenido fraudulento (habitualmente una página web suplantando a la original) y realiza el envío masivo de los correos de *phishing* a sus víctimas.

3. El usuario recibe un *email* del atacante que simula provenir de una entidad conocida. Para lograr un mayor impacto, el atacante utiliza distintos elementos que dotan de mayor credibilidad a los correos logrando de este modo una mayor tasa de éxito en el engaño a las víctimas. Algunas de estas técnicas pueden ser: Modificar las cabeceras del correo de manera que el remitente simule ser una entidad legítima o el uso de logotipos, colores y firmas de la propia entidad suplantada.

4. En dicho correo se le insta al usuario a clicar en un link contenido en el cuerpo del correo con el pretexto de renovaciones de contraseña pendientes, cambios en las políticas de seguridad de la compañía o solicitudes varias de información adicional.

5. Una vez el usuario ha seguido el enlace contenido en el correo le aparecerá en el navegador la web con el contenido fraudulento diseñado por el atacante. Un *website* con una apariencia prácticamente idéntica al legítimo de la entidad atacada, de manera que el usuario cree estar accediendo a la web lícita y se siente confortable y seguro al introducir sus datos en el navegador.

6. Tras el compromiso de la información, el atacante envía la misma a paneles de control externos gestionados por él, donde centraliza todos los datos comprometidos. Estos paneles pueden ser simples cuentas de correo en servidores de correo públicos como Hotmail, YahooMail, etc. Además para no levantar sospechas en la víctima, en multitud de ocasiones realiza una redirección a la web real de la entidad suplantada. De este modo se logra confundir a la víctima haciéndola creer que no ha pasado nada.

La Figura 2-3 muestra las fases presentadas más arriba:

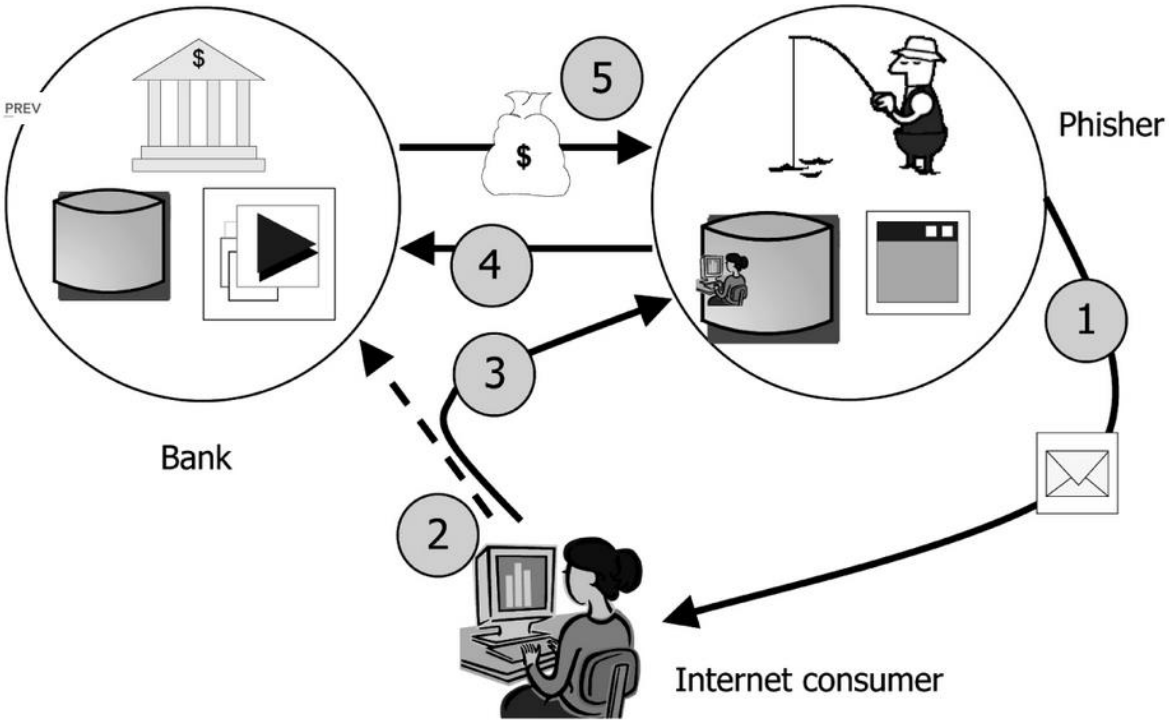



**Ilustración 2-3: Flujo de un ataque de phishing contra entidad bancaria [20]**

### 2.1.2 *Pharming* o envenenamiento de DNS

*Pharming* [6], o envenenamiento de DNS, es la explotación de una vulnerabilidad en el software de los servidores de DNS o en el de los equipos de los propios usuarios, que permite a un atacante redirigir un nombre de dominio a otra máquina distinta. De este modo, el dominio accedido desde el navegador de la víctima muestra la URL legítima de la compañía a pesar de estar navegando a una página fraudulenta.

Cuando un usuario teclea una dirección Web en su navegador en forma de texto ésta debe ser convertida a una dirección IP, que tiene forma numérica (por ejemplo 150.244.56.51). Este proceso es lo que se llama resolución de nombres, y de ello se encargan los servidores de DNS. En cada ordenador hay un archivo en el que se almacena una pequeña tabla con nombres de servidores y direcciones IP, de manera que no haga falta acceder a los servidores de DNS para determinados nombres de servidor. Los *malware* menos sofisticados simplemente modifican dicho archivo local para llevar a cabo la resolución del dominio de una entidad víctima de manera fraudulenta, asociándolo a un servidor con una página que suplanta a la original.

Sin embargo, existen otros métodos más avanzados como atacar al propio enrutador de la víctima o redirigir todo su tráfico web hacia un servidor *proxy* que realice una redirección hacia un servidor fraudulento al solicitar ciertas páginas bancarias legítimas.

### 2.1.3 *Malware* (*Malicious Software*, software malicioso)

*Malware* [6] (del inglés *malicious software*), es un tipo de software que tiene como objetivo infiltrarse o dañar una computadora o sistema de información sin el consentimiento de su propietario. Los *malware* se pueden categorizar principalmente en función de sus métodos de propagación y de su funcionamiento. A continuación se muestra una categorización de la mayor parte de los malware existentes (ver Figura 2-4):

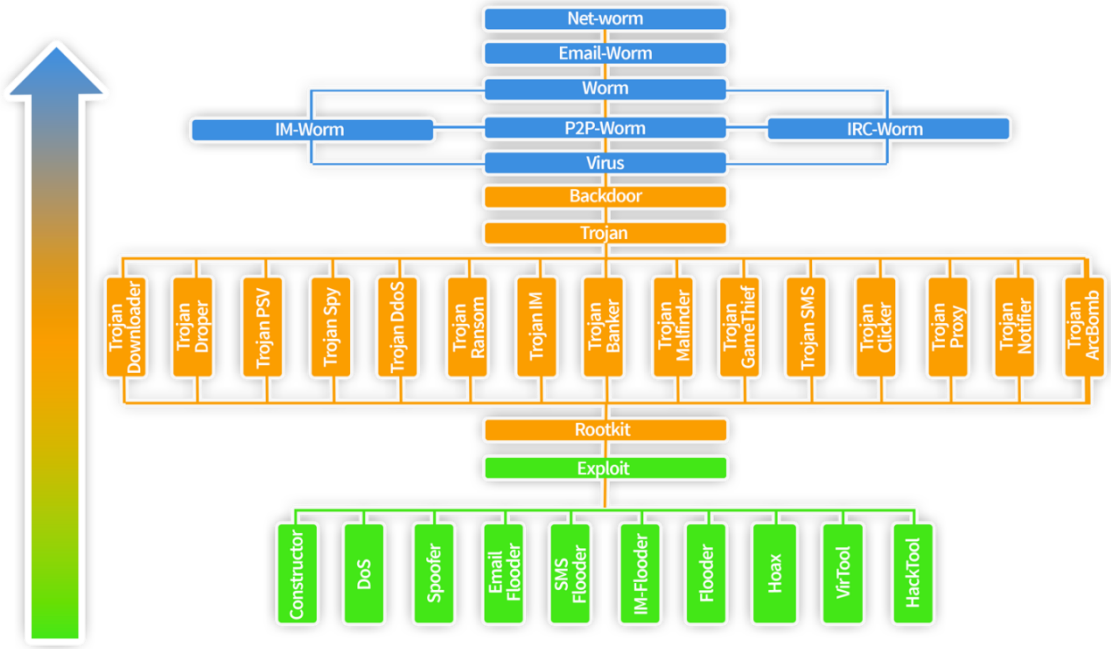



**Ilustración 2-4: Tipos de Malware**

La tendencia de estos *malware* también ha sufrido variación en los últimos años, pasando de los dañinos y molestos virus/gusanos, a un *malware* más silencioso y apenas perceptible pero mucho más peligroso, los troyanos.

Se denomina troyano [8] o caballo de Troya a un software malicioso que se presenta al usuario como un programa aparentemente legítimo e inofensivo pero que al ejecutarlo ocasiona daños.
Los troyanos están compuestos principalmente por dos programas: un cliente, que envía las órdenes que se deben ejecutar en el equipo infectado y un servidor situado en el lado de la víctima que recibe las órdenes del cliente las ejecuta y devuelve una respuesta. Para evitar sufrir bloqueos en el tráfico, la estructura de conexión cliente-servidor en muchos casos se realiza de manera inversa (el servidor se conecta al cliente), de este modo se logra evadir las medidas de seguridad tradicionales ya que la mayoría de *firewalls* o controladores de tráfico no analizan los paquetes de salida por suponer como válido el tráfico de subida. Además, este tipo de comunicación pone solución a otro de los problemas más recurrentes sufridos por los atacantes, la necesidad de conocer constantemente la IP de las víctimas donde se aloja el servidor. De este modo, son las propias víctimas las que buscan en el archivo de configuración del malware donde deben conectarse e inician la comunicación.

Los mecanismos de propagación de los *malware* tienen similitudes con las campañas de *phishing* mencionadas en el apartado 2.1.1, suelen ir contenidos en archivos adjuntos de correos electrónicos, como se muestra a continuación:

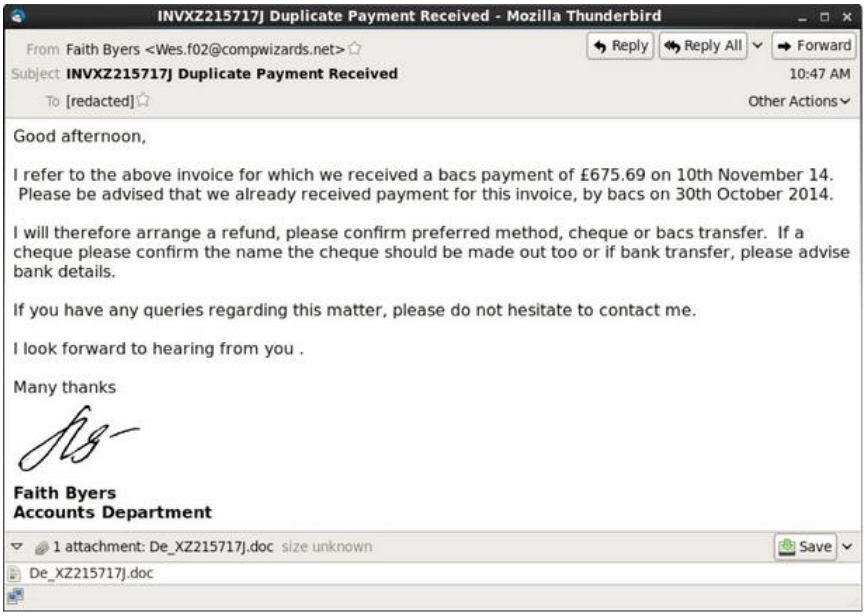



**Ilustración 2-5 Ejemplo campaña *malware* (Dridex)**

Sin embargo existen otros vectores de infección habituales derivados de una navegación insegura en la red:

- Descarga de programas de redes P2P.
- Páginas web atacadas que contienen *exploits* que se ejecutan en nuestros navegadores vulnerables.
- Ingeniería social, mensajes como "haz clic y gane un premio"

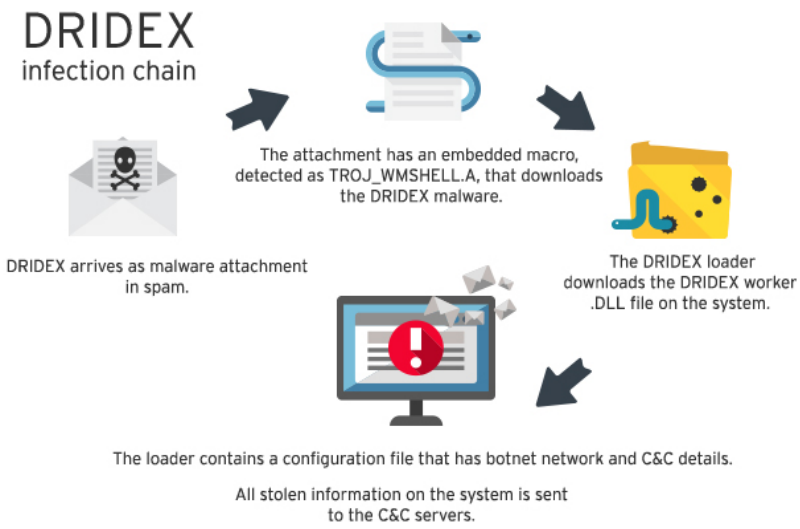



**Ilustración 2-6 Ejemplo cadena de infección de un malware (Dridex) [9]**

En lo relativo a la funcionalidad, los troyanos incluyen, entre otros, mecanismos de capturas de credenciales mediante la utilización de técnicas de *keylogging*, inyección de formularios, capturas de pantalla o incluso grabaciones de vídeo. A continuación se detallan el funcionamiento de algunas de estas técnicas:

- Inyección y captura de formularios (*Form grabber*).

  Actualmente existen métodos de doble factor de autenticación en la mayoría de portales de login, algunos de los más habituales son las tarjetas de coordenadas, PINs o preguntas adicionales como puedan ser fechas de nacimiento, etc. Por ello las bandas de crimen electrónico realizan inyecciones adicionales de campos en los formularios legítimos de las entidades con el fin de capturar la mayor cantidad posible de información que permita posteriormente suplantar a la víctima de manera satisfactoria (ver Figura 2-7). Es importante tener en cuenta que este tipo de ataques se producen en el nivel de aplicación, previo cifrado de la conexión y los datos por lo que el atacante dispondrá de la información en claro.


**Ilustración 2-7 Inserción de formularios en página de login**

- Registro de pulsaciones de teclado (*keylogging*).

  Un *keylogger* [10] (derivado del inglés: *key* (tecla) y *logger* (registrador); registrador de teclas) es un tipo de software o un dispositivo hardware específico que se encarga de registrar las pulsaciones que se realizan en el teclado, para posteriormente memorizarlas en un archivo o enviarlas a través de internet.

- Captura de pantalla y grabación de vídeo

  En muchas ocasiones las páginas de inicio de sesión de usuarios incluyen teclados virtuales con el fin de evitar que se registren las teclas pulsadas o que se capturen las credenciales del formulario. Para lograr detectar estas pulsaciones virtuales los atacantes han implementado soluciones que permiten capturar la pantalla cada vez que el usuario ha realizado clic con el ratón. Otra evolución de esta técnica es la grabación de vídeos donde se puede observar todo el proceso de inicio de sesión.

## 2.2 Protocolo HTTP

HTTP o *Hyper-Text Transfer Protocol*, es un protocolo de comunicación definido dentro de la capa de aplicación del modelo OSI que permite la transmisión de información en Internet entre los servidores web, los navegadores y las aplicaciones. Este protocolo se trata de un modelo sin estado, por lo que la información de las sesiones anteriores no es guardada y es necesario utilizar otros mecanismos para garantizar los conceptos de sesión en el cliente.

En lo relativo a versiones, a pesar de existir versiones más modernas (1.2, o 2.0) la versión aceptada por convención es la 1.1, la cual se basa en un modelo cliente/servidor (petición/respuesta) desarrollado sobre los protocolos IP de red y TCP de transporte descritos brevemente al inicio de esta sección. (ver Figura 2.8)

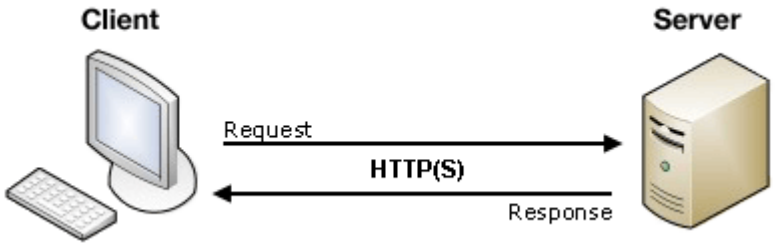



**Ilustración 2-8: Comunicación HTTP cliente/servidor**

Este modelo se basa principalmente en que el contenido se encuentra almacenado en los servidores web, y los usuarios realizan peticiones a través del protocolo HTTP para consumir dicha información a través de las respuestas de los servidores.

Es importante destacar que dentro de esta comunicación no se alojan únicamente las respuestas en texto plano, sino que cualquier contenido audiovisual como imágenes, videos o documentos pueden ser transmitidos utilizando las etiquetas de tipo MIME (*Multipurpose Internet Mail Extensions*) habilitadas para tal fin dentro del protocolo HTTP. En esta memoria no profundizaremos en el estudio de las distintas etiquetas MIME *type* ya que no es relevante para el proyecto en sí.

Sin embargo sí que es importante profundizar en la aparición de un elemento indispensable para la globalización de internet como lo conocemos hoy en día, la URI o Uniform Resource Identifier que será descrita más adelante en la sección 1.1 de esta memoria.

### 2.2.1 Formato y sintaxis del mensaje HTTP

Los mensajes HTTP siguen un formato dividido en tres partes que permiten tener visibilidad sobre el estado de la petición, los atributos incluidos en la misma y los datos contenidos (ver Figura 2-9).

*Start line* → Esta línea es obligatoria y permite indicar qué/cómo es lo que hay que hacer (*method*) y evaluar el estado de esa petición mediante distintos códigos de respuesta (*status*).
*Headers* → Son los atributos de "configuración" de la petición, se especifican los parámetros de la comunicación solicitada/recibida.
*Body* → Incluye el mensaje entregado/solicitado. Pueden ser datos en texto plano, en binario o incluso podrá ir vacío.

En cuanto a la sintaxis empleada se pueden observar diferencias entre los mensajes de petición (*Request*) y de respuesta (*Response*). Es importante entender perfectamente el significado y las variables contenidas en esta sintaxis, haciendo especial hincapié en los campos <*method*> <*status*> <*headers*>:

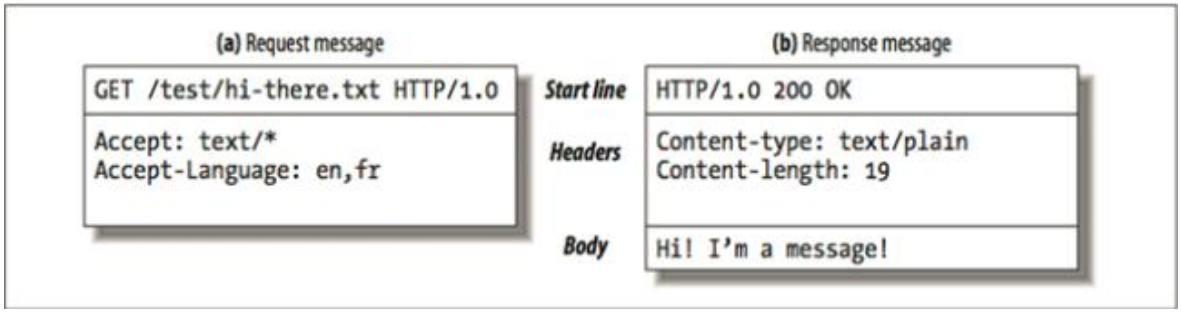



**Ilustración 2-9: Mensaje de petición y respuesta HTTP**

### 2.2.2 Métodos HTTP

Los métodos indican al servidor cuáles son las acciones que el cliente quiere llevar a cabo. Estos métodos se corresponden con la primera palabra enviada en la *start line* y permiten numerosas opciones aunque únicamente son obligatorios para cumplir el estándar HTTP aquellos métodos definidos como seguros, aquellos cuya ejecución no supone ningún tipo de acción en los servidores (GET, HEAD). Sin embargo es importante conocer también el resto de opciones posibles en toda comunicación HTTP.

**Tabla** 2-1**:Métodos HTTP**

| ***Method*** | **Definición** |
|---|---|
| *GET* | Es el método más habitual. Es utilizado para demandar el envío de recursos al cliente. Las peticiones habituales de un usuario al navegar son de este tipo. |
| *POST* | Permite enviar información al servidor para que sea ejecutada/almacenada en el mismo. Normalmente es común su uso para la cumplimentación de formularios con los que el usuario necesita interactuar. |

| | |
|---|---|
| *HEAD* | Su uso es igual al de GET con la diferencia de solicitar únicamente la información relativa a las cabeceras de un recurso. |
| *PUT* | Permite escribir/alojar documentos en el servidor desde el cliente. Es similar a GET pero a la inversa no se "pide" información sino que se "ofrece".La principal diferencia con POST es que PUT es idempotente, es decir, produce el mismo resultado sin importar cuantas veces se realice la operación y que debe utilizarse cuando conocemos con exactitud la URL del recurso a escribir/alojar. |
| *TRACE* | Es utilizado para diagnosticar el camino llevado por los paquetes durante la comunicación HTTP y de este modo poder detectar si alguna de las partes involucradas en la comunicación ha modificado el mensaje original. |
| *OPTIONS* | Permite listar los métodos soportados por un determinado servidor web. |
| *DELETE* | Este método permite eliminar contenido en el servidor a través de una instrucción de los clientes. |

### 2.2.3 Códigos de estado HTTP

Los códigos de estado o *status code son* una cadena de tres dígitos que se encarga de anunciar cuál ha sido el estado de las peticiones realizadas. El primero de estos dígitos se encarga de asociar el mensaje a uno de los 5 tipos de respuesta clasificados:

- 1: Información.
- 2: Conexión satisfactoria.
- 3:Redirección.
- 4:Error de cliente.
- 5:Error de servidor.

Todos los códigos definidos se pueden encontrar descritos en la RFC [10]del protocolo y se listan a continuación para tener visibilidad sobre los mismos. Se han separado en un código de colores que permita identificar cuando una petición ha ido bien o cuando ha habido problemas. El azul corresponde a mensajes informativos, el verde se corresponde con peticiones satisfactorias, el amarillo implica algún contratiempo pero no necesariamente tiene por qué ser negativo, y el rojo que ha habido algún problema en la comunicación.

**Tabla 2-2: Códigos de estado HTTP**

| ***Status Code*** | ***Reason Phrase*** | **Definición** |
|---|---|---|
| 100 | *Continue* | Sirve para indicar que la comunicación inicial se ha recibido y que el servidor está a la espera de nuevas peticiones por parte del cliente para completar la comunicación. |
| 101 | *Switching protocols* | Informe del cambio de protocolos llevado a cabo por el servidor a petición del cliente. |
| 200 | *OK* | La petición ha sido correcta y contiene el recurso solicitado. Es la que habitualmente deberíamos esperar al navegar en sitios públicos. |
| 201 | *Created* | Confirma la creación de algún objeto en el servidor (normalmente solicitado a través del método PUT). |
| 202 | *Accepted* | La petición ha sido aceptada aunque todavía no ha sido atendida. Este mensaje no garantiza que la comunicación vaya a ser exitosa, pero sí que el formato de la petición es a priori el aceptado y que se ha recibido sin problema. |
| 203 | *Non-authoritative information* | Este código se devuelve cuando la información se encuentra cacheada o disponible en dispositivos intermedios que no son el servidor original y que no disponen de los mecanismos para validar las cabeceras de la petición original. |
| 204 | *No Content* | Mensaje de respuesta incluido cuando el *start line y las cabeceras* están cumplimentadas pero no existe contenido (*body*) incluido en la petición. Normalmente es utilizado en los casos en los que se refresca la petición de los navegadores. |
| 205 | *Reset Content* | Utilizar por los navegadores para resetear el contenido HTML de una página. |

| | | |
|---|---|---|
| 206 | *Partial Content* | Indica que la comunicación ha sido exitosa de manera parcial. Es decir, se ha podido acceder a un fragmento de la información. |
| 300 | *Multiple Choices* | Respuesta entregada cuando el recurso solicitado puede ser accedido incluyendo varias opciones. El usuario debe seleccionar cuál de las opciones es la deseada para acceder al recurso contenido en la URL. |
| 301 | *Moved Permanently* | Informa del cambio de alojamiento del recurso solicitado. En las cabeceras del mensaje debe detallar el nuevo emplazamiento del recurso solicitado. |
| 302 | *Found* | Modificación temporal del emplazamiento del recurso solicitado. Durante un tiempo será accesible desde la URL incluida en la cabecera de la respuesta. |
| 303 | *See Other* | Indica al navegador que el contenido está accesible en otra URL. |
| 304 | *Not Modified* | Indica que el recurso solicitado no ha sido modificado y que por consiguiente no es necesario enviar de nuevo la información en el *body*. |
| 305 | *Use Proxy* | Indica la necesidad de acceder al recurso utilizando un *proxy* valido autorizado. |
| 307 | *Temporary Redirect* | Similar al código 302. |
| 400 | *Bad Request* | Se ha detectado una petición malformada en el lado del cliente. |
| 401 | *Unauthorized* | El recurso solicitado requiere autenticación y el cliente no la ha provisto. |
| 403 | *Forbidden* | La petición es rechazada por el servidor al considerar que el cliente no tiene los privilegios necesarios para acceder al recurso solicitado. |
| 404 | *Not Found* | El recurso solicitado no se encuentra en la ruta indicada ni se conocen trazas para considerar que se haya movido a una nueva ubicación. |
| 405 | *Method not allowed* | El método utilizado en la petición no se encuentra implementado en el servidor. Normalmente los métodos no aceptados se limitan a configuraciones de seguridad. |

| | | |
|---|---|---|
| 406 | *Not aceptable* | El cliente puede especificar en la petición qué tipo de recursos pueden aceptar en la respuesta. En caso de que el servidor no pueda satisfacer estos requisitos incluye el código para indicar al cliente los motivos. |
| 407 | *Proxy Authentication Required* | Indica que es necesario acceder a través de un proxy autenticado y válido. |
| 408 | *Request Timeout* | Cuando el servidor está a la espera de la respuesta del cliente para culminar la comunicación, si este último se retrasa se implementa un mecanismo para liberar los sockets y evitar que la máquina se colapse por peticiones que "secuestren" los slots del servidor disponibles. Este mecanismo consiste en la solicitud del servidor de un timeout que una vez cumplido cierra la conexión sin necesidad de validación del cliente. |
| 409 | *Conflict* | Indica que la petición está originando algún tipo de conflicto en el recurso. |
| 410 | *Gone* | Similar al 404 a diferencia que en este caso se garantiza que el contenido un día existió aunque ha podido ser eliminado. |
| 411 | *Length Required* | El servidor necesita la cabecera que indique la longitud del contenido requerida. |
| 412 | *Precondition Failed* | Usado cuando se realizan peticiones condicionales y estas fallan. |
| 413 | *Request Entity Too Large* | Mensaje notificado por el servidor cuando recibe un *body* mucho mayor del soportado. |
| 414 | *Request URL Too Long* | Mensaje notificado por el servidor cuando recibe una URL mucho mayor de lo soportado. |
| 415 | *Unsupported Media Type* | El servidor no soporta el content type enviado por el cliente. |
| 416 | *Requested Range Not Satisfiable* | El servidor no puede devolver un rango del recurso especificado por el cliente. |
| 417 | *Expectation failed* | Notificación del incumplimiento de las expectativas de la petición del cliente. |
| 500 | *Internal Server Error* | Se ha detectado algún error que imposibilita la comunicación con el servidor. |
| 501 | *Not Implemented* | El cliente solicita peticiones no soportadas por el servidor. |

| | | |
|---|---|---|
| 502 | *Bad Gateway* | El servidor realiza funciones de proxy y no logra comunicarse con su puerta de enlace. |
| 503 | *Service Unavailable* | Notifica una indisponibilidad temporal del servicio solicitado. Es el mensaje habitual cuando hay caídas puntuales del servidor web (Apache, Nginx,etc). |
| 504 | *Gateway Timeout* | Mensaje que informe de la intención del servidor de cerrar la conexión a través de su puerta de enlace por no haber recibido respuesta en un tiempo predefinido. |
| 505 | *HTTP versión not supported* | El servidor notifica que ha recibido una versión del protocolo no soportada. |

### 2.2.4 Cabeceras HTTP (*Headers*)

Las cabeceras HTTP definen los parámetros de la comunicación entre el cliente y el servidor. Estas cabeceras se engloban en diferentes subconjuntos en función de la utilidad y el origen de las mismas:

- Cabeceras de propósito general → Son utilizadas tanto por los clientes como por los servidores y ofrecen información común de la comunicación.
- Cabeceras de petición → Son propias de los mensajes de petición. Ofrecen información extra a los servidores acerca del tipo de información que el cliente espera recibir.
- Cabeceras de respuesta → Son propias de los mensajes de respuesta originados en los servidores y ofrecen información del mismo.
- Cabeceras de entidad → Estas cabeceras definen el *payload* del mensaje HTTP y ofrecen información sobre su contenido.
- Cabeceras Adicionales/Extendidas →Cabeceras no estandarizadas que ofrecen la posibilidad de mostrar información adicional en la comunicación.

#### *2.2.4.1 Cabeceras para identificar la petición y el contenido accedido.*

A lo largo de esta memoria se va a mostrar únicamente la información específica de las cabeceras de petición (*Request Header*) y las cabeceras de entidad (*Entity header*) ya que son las que potencialmente serán recibidas en el servidor web y aquellas que permitirán identificar quienes son los atacantes y cuál es el tipo de información solicitada. En particular además nos centraremos únicamente en dos subconjuntos de las mismas: *Content-header* y *Request-informational Header*.

**Tabla 2-3: Cabeceras HTTP**

| ***header*** | ***Type*** | **Definición** |
|---|---|---|
| *Client-IP* | *Request* | Muestra la dirección IP desde la que el cliente se ha conectado al servidor. |
| *From* | *Request* | Muestra la dirección de correo desde la cual se ha recibido el mensaje del cliente. |

| | | |
|---|---|---|
| *Host* | *Request* | Muestra el hostname y el puerto desde el que se originó la petición a la cual el servidor responderá- |
| *Referer* | *Request* | Muestra la información de la URL que ha originado la petición sobre el recurso del servidor. Sirve para tener visibilidad de aquellos sites que referencian al servidor. El análisis de esta cabecera será útil en los mecanismos de detección implementados posteriormente por permitir ver de dónde vienen las peticiones de los clientes |
| *UA-Color* | *Request* | Ofrece información de las configuraciones de color que acepta el dispositivo utilizado por el cliente. Permite adaptar la respuesta del servidor para mejorar la experiencia del usuario. |
| *UA-CPU* | *Request* | Ofrece información del fabricante de CPU del cliente. |
| *UA-Disp* | *Request* | Ofrece información relativa a la pantalla del cliente (tipo de dispositivo, resolución, etc.). |
| *UA-OS* | *Request* | Informe del nombre y versión del Sistema Operativo instalado en la máquina del cliente. |
| *UA-Pixels* | *Request* | Ofrece información relativa a los pixels de la pantalla del cliente. |
| *User Agent* | *Request* | Detalla información que permite identificar el nombre y datos de la aplicación que está realizando la petición al servidor. |
| *Content-Base* | *Entity* | La URL base accedida y que permite resolver URLs relativas en el servidor. |
| *Content-Encoding* | *Entity* | El tipo de codificación utilizada en el cuerpo del mensaje |
| *Content-Language* | *Entity* | El lenguaje en el que fue escrito el mensaje. |
| *Content-length* | *Entity* | La longitud del tamaño del mensaje. |
| *Content_Location* | *Entity* | Ubicación del recurso accedido. |
| *Content –Md5* | *Entity* | Un checksum-Md5 que permite comprobar la integridad del cuerpo del mensaje. |
| *Content- Range* | *Entity* | Define el rango de bytes que la información supone sobre el total del recurso |
| *Conten-type* | *Entity* | El tipo de objeto que es incluido en el mensaje. |

## *2.3 Uniform Resource Identifier (URI)*

Las URI o Identificadores Uniformes de Recursos se corresponden con la nomenclatura estandarizada para los recursos alojados en Internet. Esta estandarización permite una navegación mucho más sencilla y una identificación única y unívoca de los recursos accesibles desde Internet. En este proyecto se hace indispensable conocer la nomenclatura y formatos utilizados para una correcta identificación y procesado de las mismas.

Dentro de estas URI se pueden distinguir dos tipos, las URL (*Uniform Resource Locator*) y las URN (*Uniform Resource Name*). Las URL describen perfectamente cuál es la localización de un determinado recurso dentro de un servidor de manera que a través de las mismas se puede acceder al mismo. Por otro lado las URN definen la localización de un

determinado recurso en base a su nombre independientemente de en qué servidor se encuentre alojado, de este modo el recurso puede ser movido entre varios servidores y ser siempre accesible utilizando la misma URN.

Actualmente son las URLs los identificadores estandarizados y utilizados ya que, a pesar de que las URNs ofrecen un mayor potencial actualmente continúan en fase experimental.

#### *2.3.1.1 Uniform Resource Locator (URL)*

Las URLs son los identificadores que un navegador necesita para poder encontrar la información que se está buscando.
Normalmente las URLs contienen tres partes muy indispensables:

1. El esquema (*scheme*): en él se define **cómo** se debe acceder al recurso buscado. En particular el protocolo utilizado (HTTP, FTP, SMTP, etc.).
2. El *host*: en él se describe **dónde** se encuentra alojada la información que estamos buscando
3. La ruta del archivo (*path*): En él se define **cuál** es la dirección exacta dentro del servidor que nos va a permitir encontrar el recurso solicitado.

Sin embargo, existen otros 6 componentes que pueden servir para construir una URL siguiendo el siguiente formato:

| <scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>#<frag> |
|---|

#### *2.3.1.2 Esquema de una URL*

A continuación se describen todos los componentes de la sintaxis URL:

**Tabla 2-4: Componentes de una URL**

| Componente | Definición |
|---|---|
| *Scheme* | Qué protocolo será utilizado para la comunicación con el servidor. |
| *User* | En el caso de ser un acceso privado que requiera autenticación este componente definirá el usuario que solicita el acceso al recurso. |
| *Password* | La contraseña asociada al usuario anterior. Debe estar separada del mismo por dos puntos (:) |
| *Host* | El nombre del servidor o la dirección IP que permita alcanzar el equipo que está alojando el contenido consultado. |
| *Port* | El puerto de la máquina en el cual el host está escuchando y esperando recibir peticiones. Algunos protocolos tienen puertos estándar por defecto (HTTP:80, FTP:20/21,SMTP 25,etc) |

| | |
|---|---|
| *Path* | La ruta local dentro del servidor donde se encuentra el contenido consultado. Será separada de la URL anterior mediante una barra(/) |
| *Parameters* | Algunos esquemas permiten introducir parámetros en la consulta. Estos parámetros siempre irán en una pareja clave: valor y serán separados por punto y coma (;) |
| *Query* | En algunas ocasiones los esquemas permiten interactuar con las aplicaciones que se encuentran en los servidores. Estas consultas dependerán exclusivamente del tipo de lenguaje utilizado por lo que no tienen un estándar global. |
| *Fragments* | En ocasiones se puede desear acceder únicamente a un fragmento de un contenido por lo que se podrá especificar el mismo utilizando su nombre precedido del símbolo almohadilla (#) |

#### *2.3.1.3 Caracteres soportados en URL*

En lo relativo al tipo de caracteres que se pueden utilizar en la construcción de las URLs cabe destacar que son todos aquellos que sean representables mediante código ASCII. Sin embargo existen algunas excepciones importantes a tener en cuenta ya que existen ciertas restricciones impuestas para una correcta y segura representación de dichos caracteres.

**Este listado de restricciones o reservas se recogen en la siguiente tabla:Tabla 2-5: Caracteres soportados en URL**

| Carácter | Motivo de la limitación |
|---|---|
| *%* | Reservado como código de escape para los caracteres codificados |
| */* | Reservado para delimitar la ruta de los recursos |
| . | Reservado para la definición de la ruta |
| *#* | Reservado para delimitar el acceso a fragmentos. |
| *?* | Reservado para delimitar las consultas. |
| *;* | Reservado para delimitar los parámetros |
| *:* | Reservado para separar el scheme del usuario y el puerto. |
| *$,+* | Reservado. |
| *@&=* | Reservado por la particularidad de su uso en determinados esquemas. |
| *{}\|\\^~[]* | Restringido su uso por seguridad ya que los dispositivos de red en ocasiones pueden modificarlos. Necesario codificarlos. |
| *<>”* | Restringido por ser inseguro ya que son utilizados para varias usos fuera de las URLs. |
| *0x00-0x1F, 0x7F* | Restringido al rango listado. Fuera del mismo los caracteres no son imprimibles según el código ASCII |

#### *2.3.1.4 Ejemplos de comunicación con URL sobre HTTP*

Un ejemplo para mostrar el tipo de comunicación esperado utilizando los identificadores (URL) sobre el protocolo HTTP sería como el siguiente:

- Se solicita el recurso a través de su URL:http://example.com/index.html
- Se abre una conexión en el puerto 80 (predefinido para http) del host example.com.
- Se envía el mensaje de petición:

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: nombre-cliente
Referer: www.google.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Connection: keep-alive
```

**Ilustración 2-10: Ejemplo petición HTTP**

- Se recibe la respuesta del servidor con el contenido solicitado:

```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 2003 23:59:59 GMT
Content-Type: text/html
Content-Length: 1221

<html lang="eo">
<head>
<meta charset="utf-8">
<title>Titulo del sitio</title>
</head>
<body>
<h1>Página principal de tuHost</h1>
(Contenido)
  .
  .
  .
</body>
</html>
```

**Ilustración 2-11: Ejemplo respuesta HTTP**

## *2.4 Conclusiones*

Es necesario conocer en profundidad cómo funcionan los protocolos de comunicación sobre los que se alojan los contenidos en los servidores para poder diseñar e interpretar los mecanismos que permitan monitorizar todos los log de estos servidores en búsqueda de patrones sospechosos que permitan identificar fraude online. Además el formato de los mensajes HTTP y de las URLs también es necesario conocerlos ya que será el tipo de información que se necesite procesar y por ello se ha profundizado en este apartado en el conocimiento de todos ellos.

En los siguientes capítulos utilizaremos el análisis de las URLs y de las cabeceras HTTP para poder identificar los ataques de suplantación de identidad distribuidos por los ciberdelincuentes.

.

# 3 Análisis de requisitos

En este capítulo se va a proceder a analizar los requisitos de funcionamiento y de rendimiento necesarios para desarrollar el mecanismo automatizado de detección de fraude online descrito en este proyecto.

El mecanismo de detección implementado necesitará monitorizar todos aquellos canales desde los que puede identificar comportamientos fraudulentos contra su compañía o sus usuarios. Estos canales son principalmente dos: los correos electrónicos recibidos, y las peticiones registradas en los servidores web de la compañía. Debido precisamente a que los escenarios de detección son tan heterogéneos y teniendo en cuenta que cada uno de ellos requiere de una implantación y procesado específico, el diseño de la plataforma se realizará de manera modular para garantizar que el proyecto es versátil y flexible ante futuros cambios o mejoras.

Sin embargo, la finalidad primera de la solución es proveer de una herramienta capaz de dar visibilidad de manera sencilla a un administrador para poder identificar en fase temprana los posibles ataques de fraude que están siendo perpetrados contra su compañía. Por ello los distintos módulos deben ser interconectados en una consola de gestión central que permita tener toda la información procesada y filtrada en un único punto.

A lo largo del capítulo se detallarán los requisitos para cada uno de los módulos involucrados en el desarrollo de la herramienta: las distintas monitorizaciones, los análisis de los distintos tipos de datos y la representación de la información.

## *3.1 Monitorización del servidor*

Para poder identificar patrones sospechosos asociados a las conexiones sobre el servidor es necesario disponer de sistemas que se encarguen de logear todas las peticiones recibidas en el servidor web legítimo. De este modo se podrán identificar intentos de clonado que busquen la suplantación del website original.

Los requisitos principales asociados a esta monitorización son:

- Detección en Real-time. Según el informe de Verizón de 2015[2] la mayor parte de las víctimas de fraude online son afectadasdurante las primeras horas de vida del ataque. Este hecho pone de manifiesto la necesidad de visibilidad en fase temprana y penalizará las alternativas que introduzcan retardo.
- Parseado de los *logs* para su posterior procesado y filtrado. Un servidor web debería recibir miles de peticiones por segundo y la solución de detección debería abstraer al administrador de la necesidad de buscar entre el ruido los potenciales ataques.

## *3.2 Monitorización de buzones de correo*

En muchas ocasiones las peticiones del atacante no tienen por qué llegar directamente al servidor HTTP legítimo. Por ello, es importante disponer de buzones de correo que se encarguen de recibir los ataques "*on the wild*" que están circulando por la red. El cometido principal sería poder identificar los mismos correos que los usuarios o trabajadores recibirían

con el contenido malicioso (URL, adjunto) y lograr así ampliar la visibilidad sobre los ataques activos.

Los requisitos principales de esta monitorización son:

- Detección en tiempo real.
- Procesado del contenido del correo y parseado para extraer URLs y adjuntos.
- Procesado de las URLs para evitar falsos positivos. Ver Análisis de URLs
- Análisis de los adjuntos. Ver Análisis de adjuntos

### 3.2.1 Análisis de URLs

Las URLs contenidas en un correo o en las trazas de un *log* no tienen por qué ser maliciosas a priori. Todo el mundo intercambia correos con hipervínculos y en la mayoría de los casos son inofensivos.

Por ello es necesario que exista un mecanismo capaz de analizar y filtrar las URLs descartando aquellas que provienen de fuentes reputadas, fijando de este modo la atención únicamente en aquellos orígenes desconocidos o poco confiables.

### 3.2.2 Análisis de adjuntos

Al igual que sucede con las URLs, no todos los archivos adjuntos tienen que ser considerados como peligrosos. Sin embargo, teniendo en cuenta lo comentado anteriormente en la sección 2 de esta misma memoria acerca del hecho de que los atacantes buscan comprometer los dispositivos de los usuarios de manera "silenciosa", es mucho más complicado identificar cuando algún adjunto puede contener *malware* y puede suponer un riesgo latente para los usuarios/empleados de una compañía.

Por ello es necesario disponer de un mecanismo que permita realizar análisis de archivos en profundidad. Estos análisis deberán ser realizados de manera estática y dinámica.

#### *3.2.2.1 Análisis estático de archivos*

Este análisis es llevado a cabo analizando la información del archivo adjunto pero sin la necesidad de ejecutarlo.

Algunas de las técnicas utilizadas para discernir si un archivo binario es malicioso o no serían las siguientes:

- *Obtención del tipo de archivo*. Cuando un adjunto es de tipo .exe es un indicador fiable de que puede ser malicioso. Aunque cada vez son menos frecuentes por existir ataques más sofisticados que utilizan, por ejemplo, macros en documentos ofimáticos para la infección.
- *Cálculo del MD5*. Este dato puede ser utilizado para comprobar si el archivo recibido está identificado como malicioso por alguna de las BBDD de firmas existentes en la red.
- *Desensamblado y depuración del código*: este tipo de análisis son llevados principalmente a mano ya que requieren de la lectura y comprensión del código contenido en el archivo.

#### *3.2.2.2 Análisis dinámico de archivos*

Algunos archivos binarios maliciosos únicamente pueden ser detectados si son ejecutados y forzados a mostrar el cometido para el que fueron diseñados. Por ello, el análisis dinámico se lleva a cabo mediante la ejecución controlada de los archivos y la monitorización de los comportamientos asociados a los mismos.

Algunas de las técnicas utilizadas son las siguientes:

- *Análisis de tráfico de red*: para identificar peticiones sospechosas.
- *Cambios en el registro*: permiten por ejemplo identificar intentos del archivo binario por garantizar su ejecución tras un reinicio.
- *Monitorización de procesos*. Permite detectar si el archivo binario se ha inyectado en algún proceso legítimo del sistema.
- *Análisis de memoria*.
- Etc.

## *3.3 Mecanismo de detección proactiva*

Los cibercriminales pueden planificar y ejecutar sus ataques de varias maneras por lo que es necesario diversificar los mecanismos de detección para poder tener el mayor alcance y visibilidad posible. Uno de los mecanismos utilizados por los atacantes para realizar suplantaciones de identidad es clonar la página legítima de la compañía objetivo, por eso es necesario tener un mecanismo que alerte de cuando una página ha sido clonada.

Los requisitos principales de esta detección son:

- Mecanismos alternativos para identificar los ataques de suplantación de identidad cuando no se han recibido trazas en el servidor ni correos en el buzón.
- Detección en tiempo real.

## *3.4 Consola de gestión central*

Toda la información de alertas debe ser accesible desde un único punto, logrando de este modo evitar la necesidad de tener que revisar manualmente distintos dispositivos y optimizando los tiempos de respuesta y gestión sobre los incidentes de fraude online.

Los requisitos principales de esta consola son:

- Integrar todos los eventos de las monitorizaciones.
- Identificar los eventos de manera única y unívoca.
- Ofrecer información que permita consultar qué pasa, cuando ha pasado, y dónde está pasando.
- Ofrecer información específica de cada tipo de ataque para poder realizar acciones correctivas más precisas.

## *3.5 Conclusiones*

En este capítulo se han definido los requisitos necesarios para el correcto funcionamiento de los distintos módulos involucrados en el desarrollo de la herramienta. Para ello se ha puesto

especial hincapié en el hecho de proveer resultados en tiempo real y en ofrecer distintos entornos para la detección de ataques.

# 4 Diseño e Implementación de la herramienta

En este capítulo se realizará el diseño e implementación del proyecto: una herramienta de detección de fraude online. Todo el proceso se llevará a cabo tras el estudio de los requisitos necesarios para completar el mismo de manera exitosa. Posteriormente se realizará un diseño de infraestructura sobre aquellas alternativas analizadas más óptimas y se procederá a la integración y configuración de todos los dispositivos intermedios involucrados en el montaje del laboratorio. Este laboratorio permitirá realizar todas las pruebas necesarias para garantizar el correcto funcionamiento de los mecanismos de detección de fraude implementados.

Cabe destacar que no se desglosará con especial detalle la configuración de aquellos servicios o dispositivos intermedios necesarios para las pruebas pero que se encuentran fuera del alcance de esta memoria. Algunos ejemplos de los mismos son: servicios para el envío y gestión del correo en el servidor (Postfix). Código HTML de la página alojada en los servidores (tanto legítimos como fraudulentos), contratación y configuración de servidores, alojamiento, etc.

## *4.1 Diseño general*

La estructura diseñada permite el aislamiento de cada uno de los módulos otorgándoles de este modo una total flexibilidad de diseño y configuración para que se puedan ajustar a requisitos cambiantes durante el transcurso del tiempo.

El diseño general de la herramienta se ha planteado agrupando a su vez los distintos módulos en capas funcionales. Cada módulo tendrá tareas pequeñas pero perfectamente definidas e imprescindibles en el conjunto de la herramienta.

Para ello se ha planteado un flujo secuencial basado en los siguientes esquemas:


**Ilustración 4-1: Lógica de la herramienta**

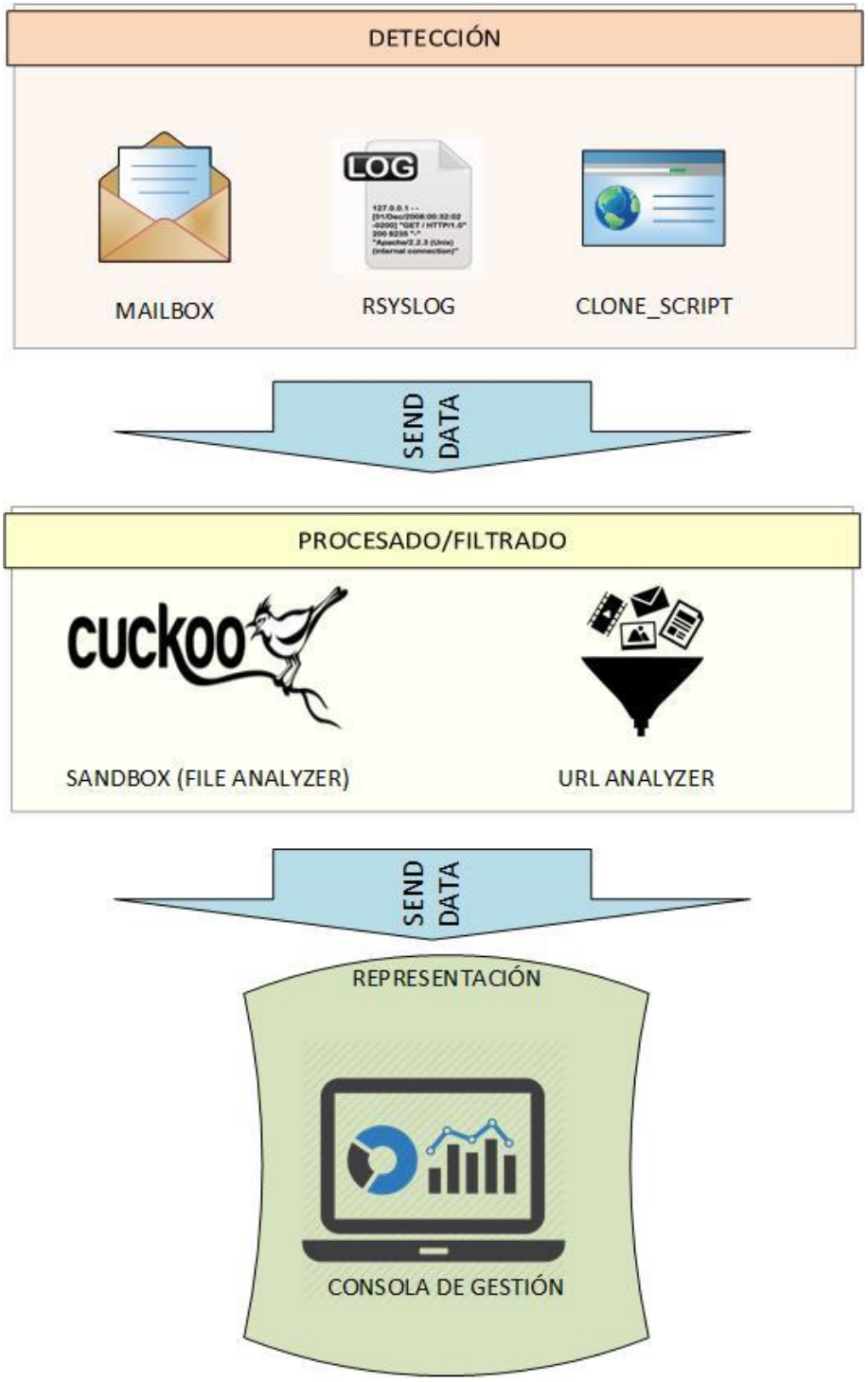



**Ilustración 4-2: Esquema general de la herramienta**

A continuación se explica el esquema general de la herramienta y cuáles son las funcionalidades principales de cada una de las capas en las que se ha delimitado el mismo.

1. *MECANISMOS DE DETECCIÓN*: La primera capa de la herramienta se encargará de montar distintos mecanismos independientes entre sí para lograr la mayor

cobertura posible a la detección de fraude online. Una vez detectados los posibles incidentes serán reportados a la capa de procesamiento para que los analice.

2. *MECANISMOS DE PROCESAMIENTO/FILTRADO*: Esta segunda capa se encarga de agrupar aquellos módulos encargados del análisis de adjuntos y del análisis de URLs. Una vez realizado el procesado y filtrado de la información detectada se encarga de reportar las alertas a la consola de gestión.

3. *CUADRO DE MANDO PARA LA REPRESENTACIÓN DE ALERTAS*: Este nivel se corresponde con la explotación y visualización de las alertas generadas. El usuario accederá al mismo para comprobar el estado de la detección de una manera sencilla y muy visual pudiendo además disponer de los detalles específicos de las distintas amenazas (mecanismo de detección que lo originó, fecha, recursos asociados, etc.).

## *4.2 Fase 1: Mecanismos de detección*

Para la consecución de unos ratios satisfactorios en la detección de incidentes de fraude online se ha tenido en cuenta que los atacantes pueden utilizar distintos mecanismos para la realización de los ataques de suplantación de identidad (*phishing* o *pharming*) y distintos canales para la distribución de los ataques de *malware*.

Por ello se han buscado soluciones complementarias que se encarguen de cubrir el mayor espectro posible para que los puntos ciegos de monitorización de fraude sean limitados al máximo.

Entre estas soluciones se encuentran:

- El envío del *log* del tráfico web recibido en el servidor legítimo para su procesado y la identificación de URLs sospechosas, aquellas que no corresponden con origines legítimos o conocidos por la compañía.
- Un *script* oculto en la página web legítima que permite detectar cuando la página ha sido clonada y está corriendo en un servidor fraudulento.
- La monitorización de un buzón de alertas habilitado para recibir correo malicioso contra la compañía.

### 4.2.1 Análisis de *Referers*

Los atacantes pueden dejar trazas en los servidores web al realizar la referenciación del sitio web legítimo desde sus páginas fraudulentas (como se explicó en la sección 2de esta memoria). Por ello es necesario poder recoger los *logs* de los servidores web y es necesario procesarlos de manera rápida para lograr una detección temprana.

Con este objetivo se han planteado varias alternativas durante la fase de diseño. Alternativas tales como el almacenado y procesado de los *logs* en el mismo servidor o el envío de los *logs* a un FTP. Sin embargo, todas estas opciones han sido descartadas por considerar que carecían de alguno de los requisitos indispensables del proyecto y finalmente se ha optado por realizar la recogida de *logs* mediante las configuraciones por defecto de los *logs* de acceso de Apache, la centralización de los mismos mediante *rsyslog* y el procesado de su contenido mediante la solución en *cloud* de Loggly.

#### *4.2.1.1 Logs de Apache*

Apache es el servidor HTTP más extendido de la red y ha sido elegido para alojar el contenido de nuestra página web por tratarse de un servidor muy estable, altamente configurable, multiplataforma, modular y, sobre todo, de código abierto, lo que facilita enormemente el soporte debido a que tiene una gran comunidad respaldándolo.

Para el desarrollo de este proyecto se ha utilizado la última versión estable en la red que es Apache 2.4.20. En esta versión (anteriormente también) aparece nativa la posibilidad de registrar los mensajes de error del servicio y las peticiones de acceso al servidor. El primero de estos *logs* se almacena por defecto en la ruta:

| /var/log/apache2/error.log. |
|---|

Mientras que el segundo de ellos se almacena por defecto bajo la misma ruta en el archivo:

| /var/log/apache2/access.log. |
|---|

Las especificaciones y ubicaciones de ambos archivos pueden modificarse al configurar la directiva *ErrorLog* (en el caso del *log* de errores) y *CustomLog* (en el caso del *log* de acceso) en el archivo de configuración de apache alojado en la siguiente ruta:

| /etc/apache2/apache.conf |
|---|

Sin embargo, y dado que el propósito de este proyecto es centrarse en la detección de fraude online, se detallaran únicamente la configuración y procesado de los log de acceso. En estos *logs* se puede obtener información sobre todas las peticiones que recibe el servidor, y por lo tanto, entre ellas, mediante el análisis de la cabecera *HTTP-REFERER*, se pueden identificar posibles ataques que estén enviando paquetes desde *hosts* fraudulentos a la web suplantada.

Para poder obtener visibilidad sobre los campos relativos a los *Referer* es importante editar y configurar la directiva *LogFormat*. Esta directiva permite definir los campos y el grado de detalle que se desea almacenar en el registro de acceso, y en nuestro caso utilizaremos la opción Combinada que permite incluir todos los datos de las cabeceras que necesitamos:

```
LogFormat "%h %l %u %t \"%r\" %>s %O \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %O" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

**Ilustración 4-3: LogFormat combined.**

Los parámetros incluidos en esta directiva son los siguientes:

- h: host remoto del cliente que realiza la petición.
- l: Es la identidad del usuario determinada por identd
- u: es el usuario determinado por la autenticación HTTP si la hubiera.
- t: es el timestamp del momento en el que la petición fue recibida.
- >s: es el status code enviado por el servidor al cliente.
- 0: Indica el número de bytes enviados, incluida la cabecera.
- {Referer}i\: Se corresponde con la cabecera Referer y muestra el servidor del que proviene el cliente.

- {User-agent}i\: La cabecera de petición HTTP-User-Agent que incluye la información del navegador del cliente.

Una vez están configuradas las peticiones que serán registradas en el *log*, serán como las siguientes obtenidas del servido legítimo montado para las pruebas del laboratorio:

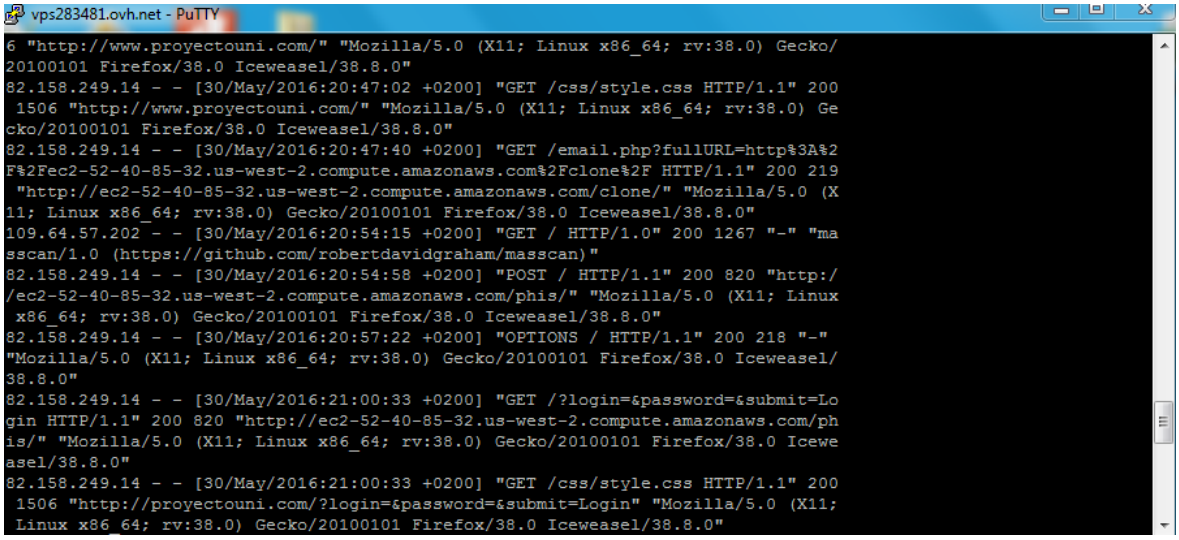



**Ilustración 4-4: Ejemplo access.log APACHE**

### *4.2.1.2 Envío de logs: RSyslog*

Una vez se han registrado los *logs* en la máquina local del servidor HTTP, es recomendable su envío a otro sistema que permita almacenarlos y procesarlos en tiempo real de manera que la detección de anomalías puede ser llevada a cabo con el menor retraso posible, minimizando de este modo el impacto de los posibles ataques.

Para ello se ha considerado utilizar Rsyslog, un software de código abierto que permite enviar desde los sistemas UNIX mensajes de *log* a través de una infraestructura IP basándose en el protocolo Syslog. Además añade el envío utilizando TCP lo que permite tener la certeza de que no se pierden paquetes en la comunicación, cuestión de vital importancia en la fase de registro de cualquier herramienta.

El procotolo Syslog define el estándar para generar los mensajes de *log* con identificadores que permitan diseñar un sistema modulado que separe los mensajes del sistema que los ha generado del sistema que los almacena y en última instancia del sistema que los procesa. De este modo se logra una gran modularidad que permita depurar y aislar mejor cualquier problema en la infraestructura de registro de actividad.

El archivo de configuración que se necesita modificar para que los *logs* de Apache sean enviados a través de Rsyslog se encuentra en la siguiente ruta:

| /etc/rsyslog.d/21-apache.conf |
|---|

En este archivo será necesario incluir la ruta donde está alojado el archivo de *log* de Apache tras la variable: *$InputFileName*

### *4.2.1.3 Almacenamiento y parseado de los logs: Loggly*

La modularidad y posibilidad de distribuir las funciones en diversas infraestructuras era uno de los requisitos de la solución para permitir versatilidad y una futura evolución más cómoda y menos traumática.

Con esta finalidad se ha considerado almacenar los *logs* en una solución en *cloud* que permita acceder a los mismos para su procesado desde cualquier parte del mundo y en cualquier instante, y entre las alternativas contempladas se ha considerado que Loggly era la opción más ventajosa por los siguientes motivos:

- Dispone de conectores para integrar los *logs* de Rsyslog.
- Dispone de parseador automático para los *logs* estándar de Apache. Permitiendo de este modo disponer de la información procesada, clasificada y consultable.
- Dispone de acceso mediante *token* a una API de consulta que permite integrar los resultados con terceras herramientas o *scripts*.
- Dispone de un potente motor de búsqueda que permite realizar el filtrado vía interfaz o vía API del contenido registrado.

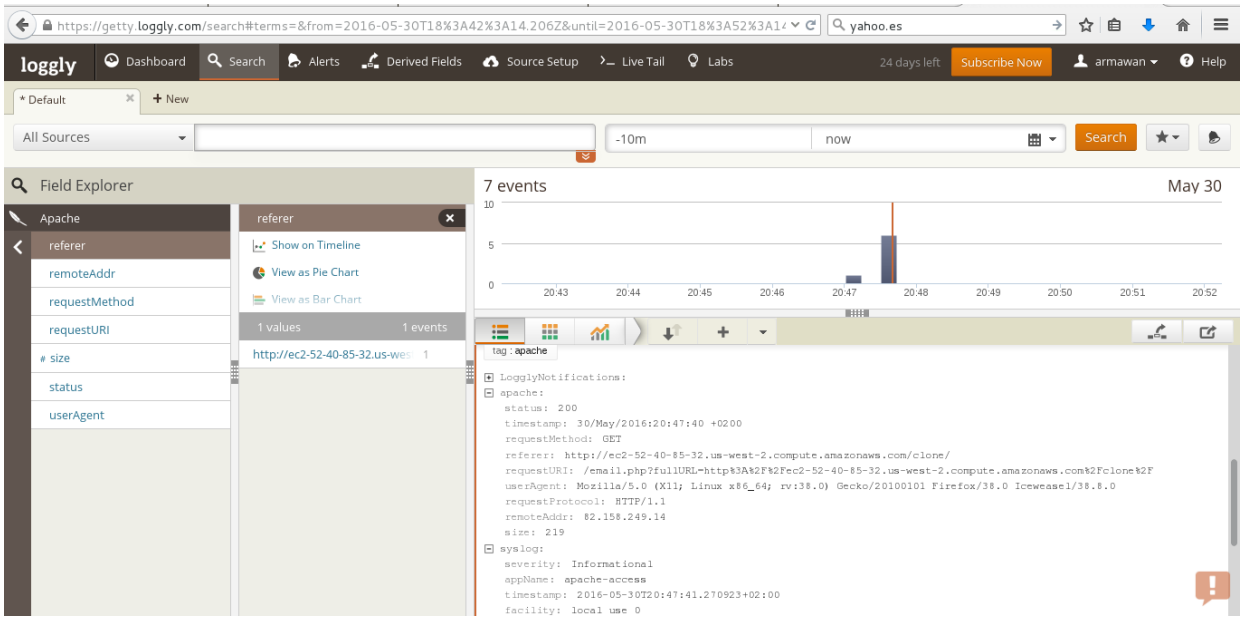



**Ilustración 4-5: Interfaz Loggly (almacenamiento de Logs en cloud)**

### 4.2.2 Script de detección de clonado

Los cibercriminales buscan lograr conseguir el mayor beneficio con la menor inversión posible a la hora de llevar a cabo sus ataques. Por ello en muchas ocasiones para realizar los ataques de suplantación de identidad realizan un clonado de la página legítima de la compañía a atacar descargando todos los archivos en ella contenida. De este modo se garantizan que los estilos, colores y diseños se asemejan a la página objetivo y las potenciales víctimas tienen más complicado percatarse de que están siendo víctimas de un intento de estafa.

Este escenario pone de manifiesto la necesidad de incluir en el proyecto de detección de fraude online mecanismos que nos alerten de cuando la página de la compañía está siendo clonada y que nos informen a su vez de en qué *host* fraudulento está la misma siendo replicada para poder mitigar el riesgo del ataque actuando contra estos host.

Para lograr este propósito se ha considerado incluir un "código trampa" dentro de la página para que el mismo sea copiado cada vez que el atacante realiza el clon de la página original. Este código se encargará de validar si está siendo ejecutada en el servidor legítimo y en caso contrario invocará de manera remota a una función que se encargará de enviar una notificación de correo al buzón de alertas del mecanismo de detección.

Para evitar que el código sea fácilmente detectable por un atacante se ha optado por realizar la monitorización a través de un JavaScript, que además será ofuscado y comprimido con el fin de hacer su contenido ilegible e indescifrable, como se muestra en la figura 4-6.

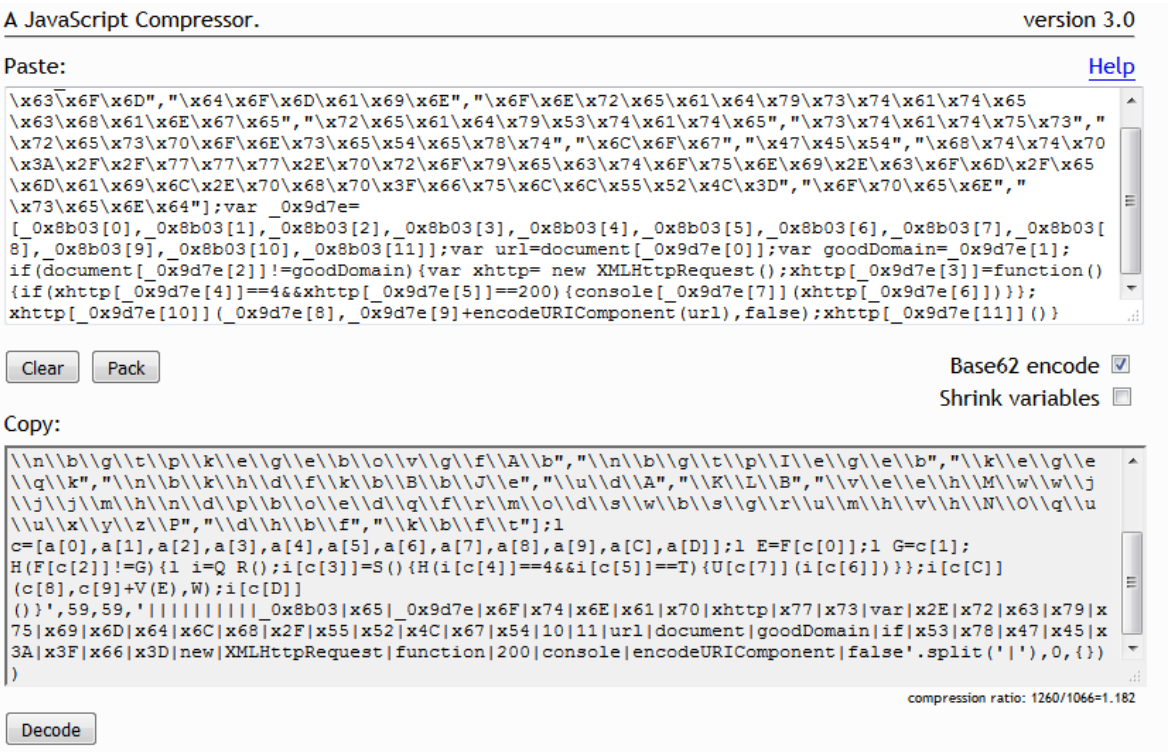



**Ilustración 4-6: Cifrado y compresión de JavaScript. [21][22]**

### 4.2.3 Buzón de correos

Es sabido que habitualmente el vector utilizado por los atacantes para distribuir sus ataques de *phishing* o *malware* es el SPAM, la distribución de cantidades ingentes de correos cuyo éxito se basa en la viralidad de los mismos y en el hecho de que apenas unas cuentas víctimas, en ocasiones una única víctima, son suficientes para considerar la campaña exitosa.

Por esto es importante tener una herramienta que nos permita identificar ataques contra nuestra compañía a través de un simple buzón de correos. Exponiendo premeditadamente el buzón de correos por distintas listas de distribución se puede lograr ser "objetivo" de los cibercriminales y por lo tanto lograr recibir los ataques en el buzón como si de una potencial víctima nos tratáramos.

Además otra de las alternativas para identificar ataques mediante el correo es ofrecer el buzón a los clientes o empleados de la compañía para que estos reporten cualquier tipo de actividad sospechosa que hayan podido recibir en sus distintos correos personales.

Procesando en tiempo real toda esta información y descartando los falsos positivos es posible obtener una interesante y económica medida de detección de fraude, como se muestra en la figura 4-7.

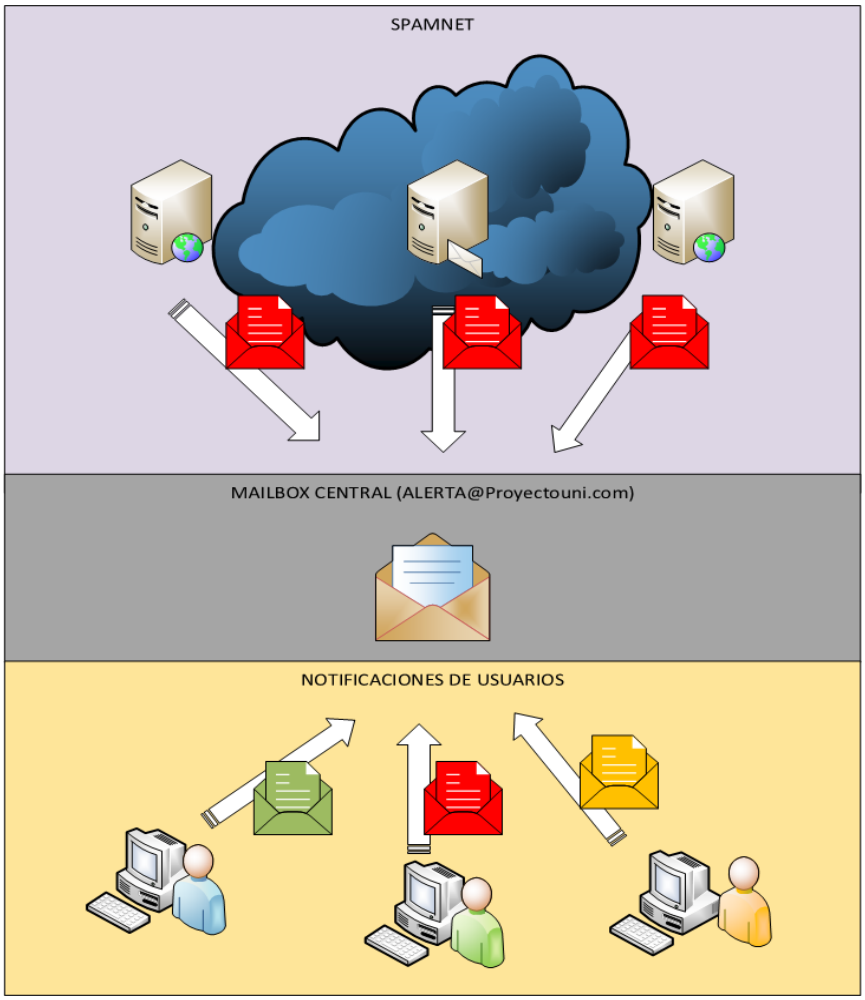



**Ilustración 4-7: Diagrama de funcionamiento de mailbox**

## *4.3 Fase 2: Procesado de la información*

Detectar potenciales ataques, almacenando cantidades ingentes de información sin filtrar no puede ser el único objetivo de una herramienta que pretenda trabajar en tiempo real. Para que un administrador de seguridad sea eficiente identificando amenazas de fraude online es necesario limitar el número de alertas a aquellas que realmente sean susceptibles de ser priorizadas.

Precisamente por ello y con el fin de cumplir con uno de los requisitos definidos en el capítulo 3, es necesario realizar un procesado de toda la información para poder filtrar los resultados y arrojar sobre la consola únicamente aquellos resultados susceptibles de ser considerados sospechosos. Este requisito ha sido impuesto principalmente porque los recursos de seguridad suelen ser limitados y necesitan poder centrar su atención solamente en aquellos incidentes que previamente han sufrido un procesado suficiente. De otro modo, la ventaja adquirida al detectar los ataques de fraude online en fase temprana quedaría anulada por la cantidad de alertas que los equipos de seguridad tendrían que procesar.

### 4.3.1 Definición de funciones

A continuación se describirán las funciones incluidas en cada uno de los subsistemas. Las mismas se dividen en dos programas realizados para recuperar la información de las distintas fuentes de detección: *fetchmail* y *fetchsyslog*.

#### *4.3.1.1 Fetchmail.py*

Fetchmail.py es un programa cuyo objetivo es conectarse a los distintos buzones de correo incluidos en la red de detección, clasificar su contenido, procesarlo y enviarlo a la consola de gestión. Para ello utiliza varias funciones de manera secuencial, según se comenta en los siguientes subapartados.

*Argumentos*: host del buzón de correo, puerto del buzón de correo, usuario del buzón de correo, contraseña del buzón de correo, host donde se aloja la *sandbox* de análisis, puerto de la *sandbox* de análisis, opción de descarga, lista blanca.

##### *4.3.1.1.1 Getmails*

Conecta con el servidor de correo y recupera la información de los correos en él contenida. Por defecto recupera únicamente aquellos correos que no hayan sido leídos, aunque se puede forzar que se recuperen todos los correos contenidos en el buzón independientemente de que hayan sido procesados o no.

*Argumentos:* host del buzón de correo, puerto del buzón de correo, usuario del buzón de correo, contraseña del buzón de correo, opción de descarga.

##### *4.3.1.1.2 Clasiffymail*

Clasifica el correo entrante descargado en base a los flujos de detección configurados. Para ello realiza distintos niveles de clasificación:

- Comprueba si el asunto del correo contiene la etiqueta [ALERT]. De este modo se puede considerar que el correo ha venido de una alerta detectada por el sistema de detección de clonado.
- Comprueba si el correo contiene URLs. Para poder realizar esta validación llama a la función *urls*:

    - En caso negativo muestra por pantalla un mensaje en el que indica que el correo procesado no contiene ninguna URL.
    - En caso afirmativo guarda las URLs y las filtra con la lista blanca de URLs legítimas.

- Comprueba si el correo contiene adjuntos utilizando la función *attachments* y los envía a la *sandbox* utilizando la función *sendfiletosandbox*.

*Argumentos:* Recibe un mail, una lista blanca y una URL con los parámetros para el envío a la *sandbox*.

#### *4.3.1.1.3 Urls*

Recibe los emails recuperados del buzón y devuelve un listado con todas las URLs contenidas en los mismos. Para ello utiliza la siguiente expresión regular que valida los caracteres contenidos en la URL:

```
URL_REGEX = (r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|'
             r'(?:%[0-9a-fA-F][0-9a-fA-F]))+')
```

**Ilustración 4-8: Expresión regular parseo URLs**

#### *4.3.1.1.4 Attachments*

Recibe los *emails* recuperados del buzón y devuelve un contenedor que incluye el nombre del archivo y el *payload* para su análisis.

#### *4.3.1.1.5 Sendfiletosandbox*

Realiza el envío de los archivos obtenidos del procesado de correos a la *sandbox* para realizar su análisis en profundidad. De este modo se pueden obtener informes detallados del comportamiento de los archivos con el fin de poder determinar su riesgo.

Espera la respuesta de la *sandbox* para garantizar que el archivo ha sido enviado y de este modo dar por finalizada su ejecución.

*Argumentos*: el adjunto a enviar, la URL de la *sandbox*.

### *4.3.1.2 Fetchsyslog.py*

El programa fetchsyslog.py se encarga de conectarse a través de la API de Loggly al sistema de almacenamiento de *logs* remoto que hemos configurado. De este modo logra tener acceso en tiempo real a todas las peticiones de acceso recibidas por el servidor web y podemos

identificar de manera automática y sin esfuerzo las posibles peticiones relacionadas con atacantes que intentan hacer ataques de *Pharming* o *Phishing*.

Al igual que fetchmail.py, fetchsyslog.py se sirve de un subconjunto de funciones para llevar a cabo la ejecución completa.
*Argumentos*: Recibe el usuario de loggly, la password de loggly y la whitelist sobre la que tenemos que filtrar los resultados.

#### *4.3.1.2.1 fetch_new_referers*

Se encarga de descargar todos los nuevos HTP-Referer aparecidos desde la última ejecución de la función.

Para poder realizar el filtrado del total de información contenida en el *log*, se sirve de distintas funciones en la API de Loggly, lo que permite recorrer toda la información en búsqueda únicamente de los campos requeridos.

```
for row in res['events']:
    url = row.get('event', {}).get('apache', {}).get('referer')
    timestamp = row.get('event', {}).get('syslog', {}).get(
        'timestamp')
```

**Ilustración 4-9: Bucle de programación para recuperar datos de referers**

Devuelve una lista de objetos que contienen la URL de los *referers* y el *timestamp* de cuando fueron detectados.
*Argumentos:* Recibe tanto la URL *cloud* del acceso a Loggly como el usuario y la contraseña para lograr acceder.

### 4.3.2 Cuckoo Sandbox- Análisis de malware

Como su propia página define, Cuckoo es un Sistema de Análisis de *malware*. Este sistema permite realizar los análisis de manera automatizada simplemente "subiendo" los mismos al entorno de simulación, logrando de este modo un notable incremento en la capacidad de procesamiento de binarios en los equipos de detección de fraude y respuesta ante incidentes. Esto ofrece una gran ventaja, que será explotada en este proyecto ya que, de otro modo, plantear el análisis en profundidad de los archivos adjuntos a un buzón de correos manualmente habría sido una utopía.

En cuanto a las características principales de Cuckoo están dos que son especialmente significativas y cumplen con los requisitos definidos en el proyecto:

- Es 100% modular. Esto permite crear e interconectar módulos para mejorar el funcionamiento del sistema.
- Es 100% de código abierto. Permite realizar numerosas modificaciones en la configuración y tiene el respaldo de una gran comunidad detrás. De este modo Cuckoo permite configurar tanto la fase de procesamiento como la de informes del sistema.

Además, los distintos módulos (ver Figura 4-10) permiten llevar a cabo los análisis estáticos y dinámicos definidos en el apartado 3.2.2 de Análisis de adjuntos.

- Análisis de archivos ejecutables y ofimáticos en distintos entornos virtualizados.
- Detección de comportamiento sospechoso y reporte del mismo para su posterior análisis.
- Volcado de memoria y tráfico para identificar recursos de red fraudulentos. (Paneles de control del *malware*, URLs de descarga de binarios maliciosos, etc.)
- Comprobación de ratios de detección de antivirus. Permite identificar si las muestras analizadas son consideradas maliciosas por los distintos motores antivirus del mercado. Para ello se conecta a la API de Virustotal.

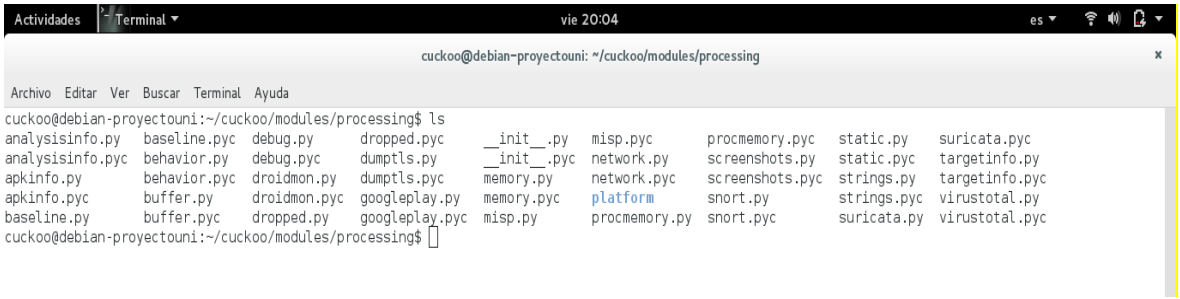



**Ilustración 4-10: Listado de modulos de procesamiento disponibles en Cuckoo**

Para realizar el análisis de binarios es necesario integrar Cuckoo en el flujo de procesamiento que se ha definido en el proyecto. Por ello, una vez se ha instalado y configurado el sistema de análisis de malware será necesario realizar el envío de muestras desde los mecanismos de detección como se ha explicado en los apartados anteriores de esta memoria. Sin embargo, con realizar el procesado y envío de muestras de manera automatizada no es suficiente, es necesario recoger los resultados de los informes de análisis y ponerlos al alcance de los usuarios en la consola de gestión.

Para ello se ha creado un módulo de procesamiento adicional a los que incluye Cuckoo por defecto denominado proyectouni.py.

#### *4.3.2.1 Proyectouni.py*

Este módulo inicialmente se va a conectar a través de la API de Cuckoo a la máquina de análisis de *malware* y va a quedar a la espera de que los nuevos análisis finalizados le sean anunciados.

Posteriormente procesará todos los campos del informe y se quedará con aquellos que hemos definido como relevantes en el modelo de datos del proyecto devolviéndolos en formato JSON para su representación en la consola de detección, como se muestra a continuación:

```
requests.post(API_ENDPOINT,
    json={"md5": md5,
          "filename": filename,
          "yara_matches": str(signature_text) + " critical",
          "av_results": av_results,
          "report_url": report_url})
cuckoo@debian-proyectouni:~/cuckoo/modules/reporting$
```

**Ilustración 4-11: Información devuelta por el módulo de reporting creado**

Para todo esto se sirve de dos funciones principalmente :

#### *4.3.2.1.1 Splitfields*

Esta función se encarga de obtener todos los campos contenidos en el reporte de Cuckoo. Esto lo realizar recorriendo las direcciones devueltas y accediendo a los distintos campos que se encuentran separados por puntos.

*Argumentos*: recibe el nombre del reporte, y devuelve el nombre del campo y su valor.

#### *4.3.2.1.1 Run*

Ejecución principal del módulo que se encarga llamar a la función *splitfields* e ir almacenando todos los valores en las variables que serán devueltas en el JSON y posteriormente serán representados con exactitud desde la consola.

### 4.3.3 Esquema de funcionamiento

A continuación se describen los flujos de funcionamiento de la herramienta:

*Fecthmail.py* se conecta al buzón de correo y, en función del argumento enviado, comprueba si existen correos nuevos o se descarga todos los existentes. Posteriormente invoca a la función *ClasiffyMail* para que se encargue de procesar cada uno de los correos y clasificarlos. La Figura 4-12 muestra esquema de funcionamiento.

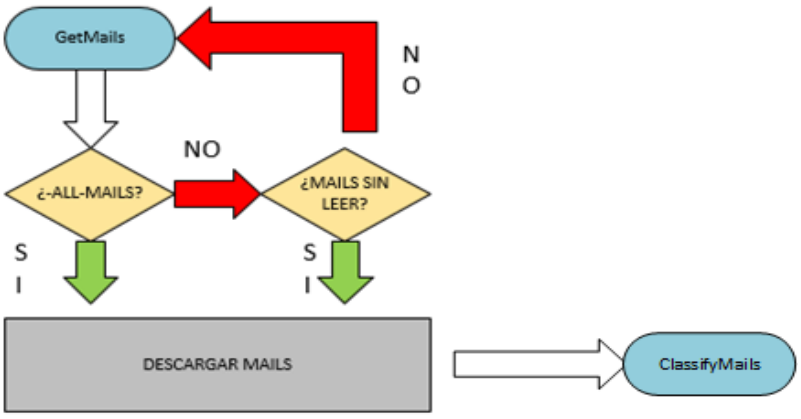



**Ilustración 4-12: Diagrama funcionamiento obtención de correos**

Una vez se han enviado a clasificar los correos se recorren uno a uno para ver si cumplen algunos de los requisitos de parseo establecidos. En caso de coincidencias crea las alertas correspondientes. Para ello se sirve de las funciones auxiliares *urls y attachments* que se encargan de sacar los listados de URLs y adjuntos contenidos en cada correo. La herramienta realiza un bucle hasta que la lista de URLs y adjuntos está vacía y vuelve a buscar otro correo que clasificar. La Figura 4-14 muestra este proceso.

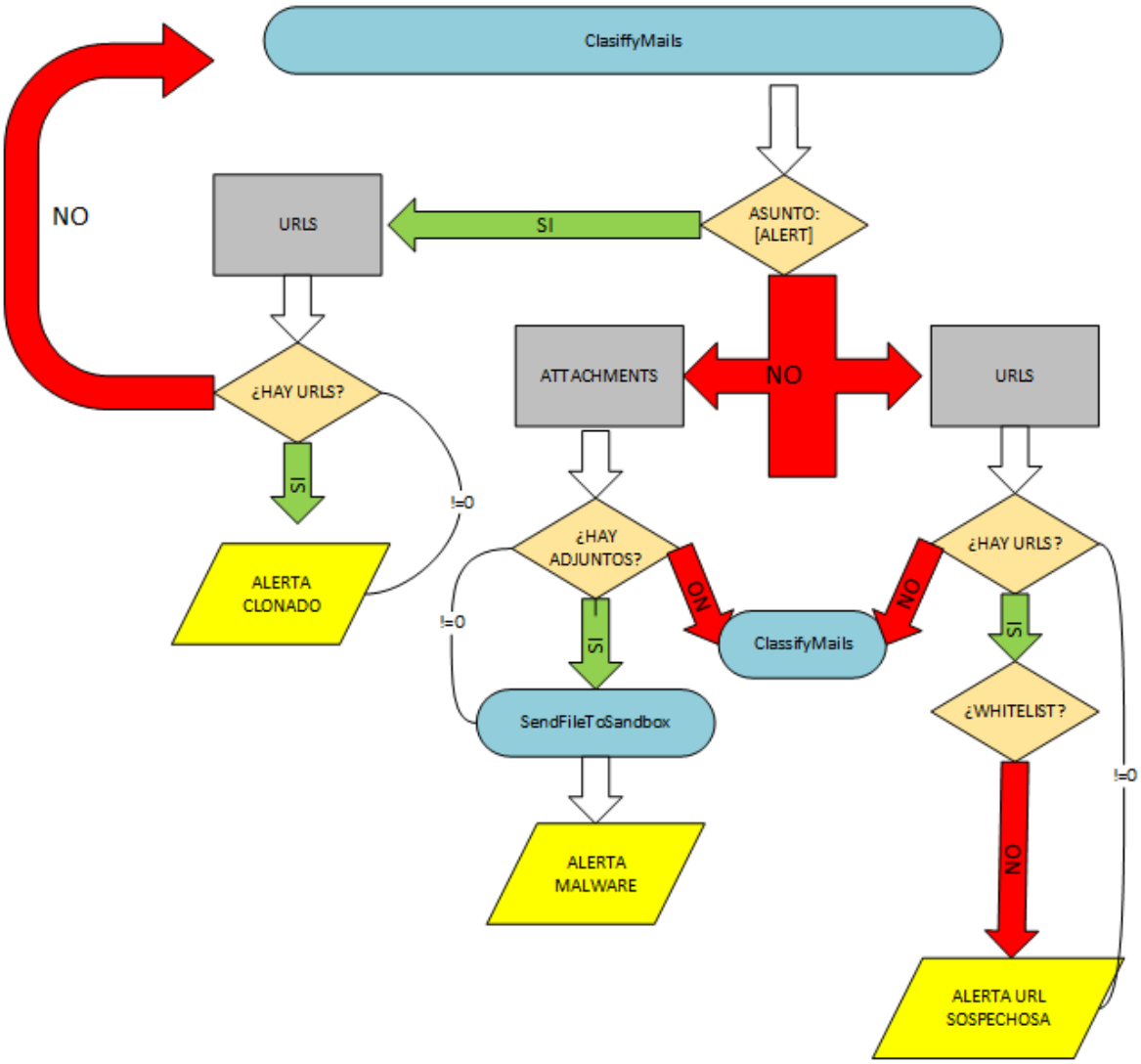



**Ilustración 4-13: Diagrama funcionamiento clasificación de correos**

*Fetchsyslog.py* se conecta a la API de almacenamiento y procesado de *logs* de *Loggly* y se encarga de descargarse todos los *logs* de la última hora (periodo de tiempo configurable), extrae las cabeceras HTTP-Referer y parsea la URL contenida en las mismas.

Posteriormente compara toda la información con las listas blancas y si no existen coincidencias, por no tratarse de URLs legítimas o verificadas, levanta alertas en la consola de gestión. La Figura 4-13 muestra este esquema de funcionamiento.

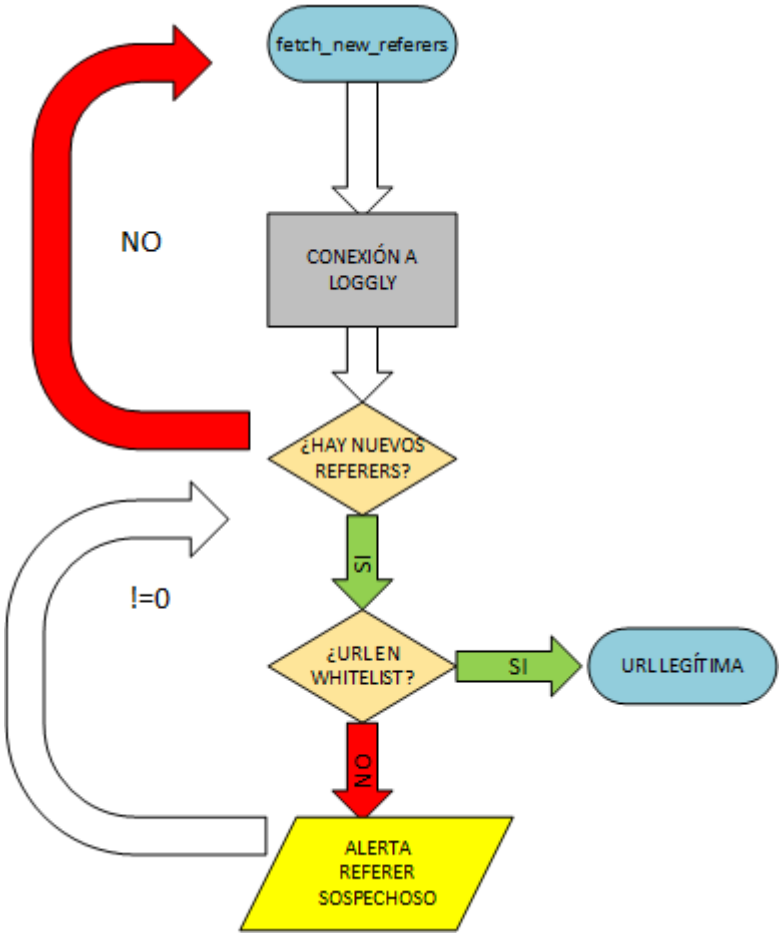



**Ilustración 4-14: Diagrama funcionamiento obtención datos syslog**

## *4.4 Fase 3: Representación de la información.*

La representación de la información será llevada a cabo en un servidor externo que recibirá toda la información detectada y procesada y tendrá un panel de consulta que será accesible desde la web para cualquiera de los operadores que requieren el acceso para la mitigación o investigación de los ataques.

Con el fin de poder organizar la información en la fase de procesamiento, se definirán varios modelos de datos en función de las tipologías de ataque.

### 4.4.1 Modelo de datos de *Phishing*

Los ataques de *phishing* serán categorizadas en función de su origen y tipología de ataque y se añadirán campos de información de contexto. Con ello se pretende ampliar la visibilidad sobre los ataques para poder realizar posteriores investigaciones y análisis:

- *Origen:* Se distinguen tres posibles orígenes relacionados con los mecanismos de detección.
    - *Mailbox (MAI):* Relacionado con aquellas alertas que han sido detectadas tras el parseo y procesado de las URLs contenidas en un correo recibido en el buzón de *alerta@proyectouni.com*.
    - *Syslog (SYS):* Relacionado con aquellas alertas que han sido detectadas tras el registro de peticiones en el *log* de Apache y el análisis de las cabeceras HTTP-REFERER contenidas en las trazas almacenadas en el Rsyslog.
    - *Clone Script (CLO):* Relacionado con aquellas URLs que son consideradas orígenes de posibles ataques por clonado del website legítimo.

- *Tipología de Ataque:* Se distinguen tres tipos de ataques distintos asociados a los intentos de ataque a través de incidentes de suplantación de identidad o *Phishing*.
    - *Clonado.*
    - *URL sospechosa.*
    - *Referer Sospechoso.*
- *Timestamp:* Será recogida y mostrada la información del evento para tener identificado cuando el mismo fue detectado y poder realizar filtrados que permitan tener trazabilidad sobre los ataques sufridos en determinados periodos de tiempo.
- *URL:* El detalle explícito de la URL que está atacando a la compañía y que está llevando a cabo un ataque de *Phishing* o *Pharming*.

### 4.4.2 Modelo de datos de *Malware*

En el caso de los posibles ataques de *malware* no se hace diferenciación de Origen o tipología ya que de los mecanismos implementados en este proyecto solamente podremos recibir muestras analizables de los adjuntos contenidos en los correos de SPAM recibidos en el buzón de correo.

Sin embargo, y dado que existe mucha información asociada al análisis de los binarios y que puede ser muy útil para mitigar el impacto de los ataques de *malware*, se ha considerado ampliar el tipo de información a ofrecer en esta pestaña:

- *Filename:* Nombre del archivo analizado.
- *MD5:* Hash que permite identificar a un archivo de manera única y unívoca independientemente de su nombre de archivo.
- *Firmas críticas:* Muestra el número de firmas que se asocian con comportamientos maliciosos dentro del archivo analizado.
- *AV Results:* Resultado del análisis estático realizado al consultar a los distintos motores antivirus contenidos en la Cuckoo. Permite nombrar e identificar el ratio de detección del binario en los motores antivirus más extendidos del mercado.
- *Timestamp: Fecha* de detección del ataque.
- *Report_URL (detalles):* Contiene la ruta donde es almacenado el informe completo del análisis del archivo para poder ser consultado desde la consola.

### 4.4.3 Consola Web de gestión

Para una correcta clasificación y visualización de los ataques se va a dividir la información en dos pestañas en función del contenido detectado: *Phishing* o *Malware*.

El frontal web (ver Figura 4-13) será desarrollado usando la plantilla por defecto de Django Admin que contiene una interfaz de gestión que se ajusta a las necesidades básicas del proyecto. En la mencionada plantilla se harán modificaciones para eliminar módulos y que los campos e informaciones requeridas sean los definidos en los distintos modelos de datos.


**Ilustración 4-15: Consola de gestión "Detector de fraude on-line"**

## *4.5 Conclusiones*

Durante el presente capítulo se han detallado las configuraciones y desarrollos llevados a cabo para la creación de la herramienta de detección de fraude online.

Para ello se ha seguido una distribución modular interconectando distintas soluciones de código abierto y distintos desarrollos realizados por el ingeniero para su correcto engranaje. Además se ha seguido una metodología dividida en fases también para el procesamiento de la información de manera que el proyecto sea suficientemente versátil y flexible para futuras mejoras o cambios.

En este punto es relevante hacer mención a la necesidad de utilizar la potencia de las distintas API existentes en proyectos como Loggly o Cuckoo, que permiten una integración programática muy potente para proyectos de este tipo.

Una vez finalizado todo el diseño y elaboración del detector de fraude online, dispondremos de una solución a muy bajo coste, escalable y eficaz para la identificación de fraude on-line en tiempo real.

# 5 Integración, pruebas y resultados

Las pruebas de funcionamiento del detector de fraude on-line necesitan, además de la integración del propio mecanismo, replicar una infraestructura completa que simule tanto el lado del servidor web atacado como el lado del atacante que ha realizado la suplantación de identidad.

Para lograr implantar estos entornos se han seguido los siguientes pasos (ver Figura 5-1):

1. Registrar un dominio: proyectouni.com que será utilizado para el sitio web legítimo.
2. Contratar un servidor virtual en cloud (OVH) que permita alojar la página web en el servidor de Apache.
3. Contratar un servidor virtual EC2 en Amazon Web Services (AWS). En él se alojarán los intentos de ataque de *phishing*.
4. Configurar un servidor mail asociado a la cuenta alerta@proyectouni.com que servirá para el envío de los distintos correos y para simular el buzón de correo trampa que será procesado en búsqueda de las amenazas.
5. Contratar el servicio en la nube de Loggly al que serán enviados los *logs* vía rsyslog del Apache.
6. Instalar el mecanismo de procesado y la consola del detector en una máquina local que se integre con todos los anteriores.

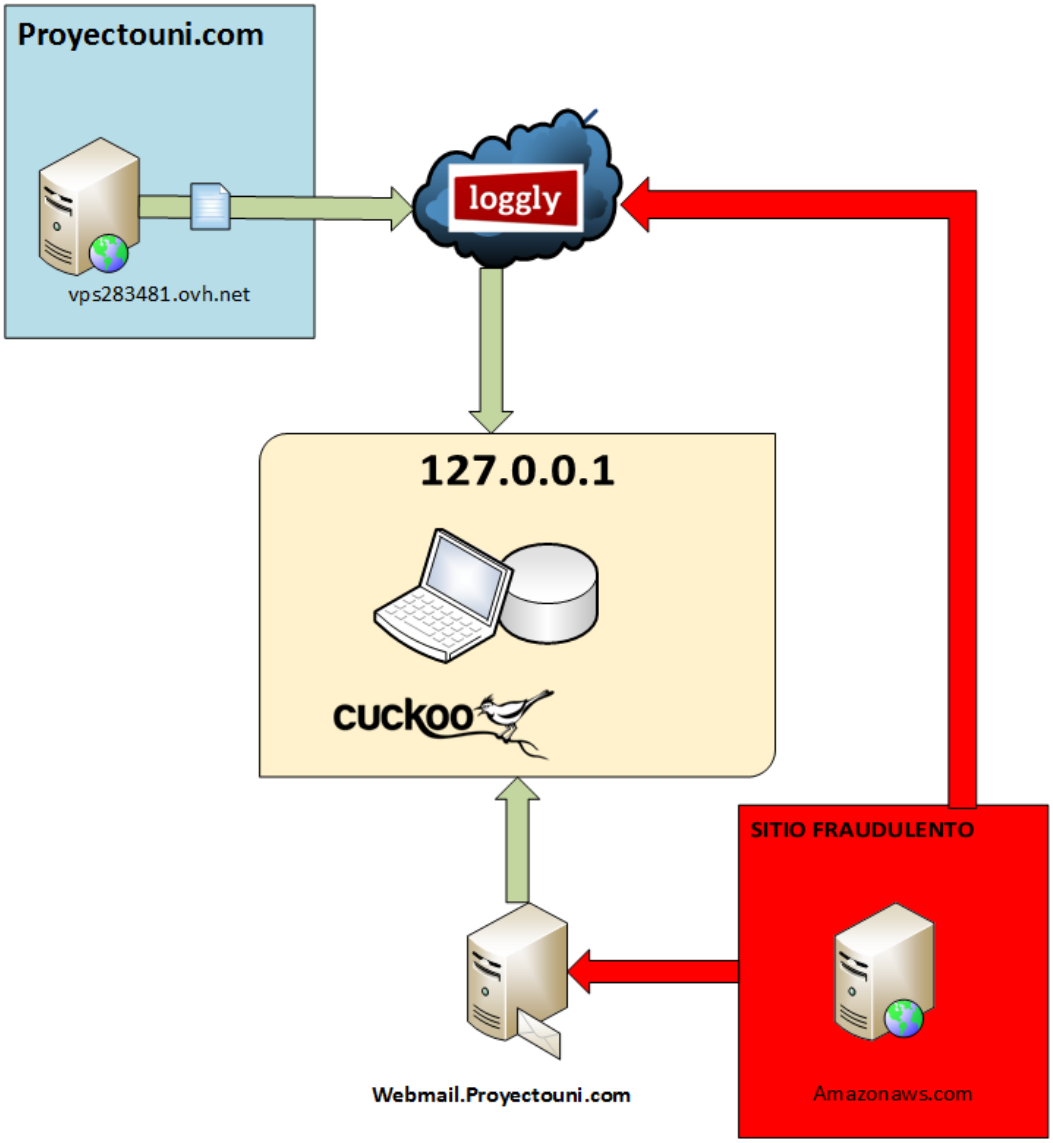



**Ilustración 5-1: Infraestructura de pruebas del detector**

## *5.1 Ataques de Phishing*

Los ataques más comunes y sencillos, desde el punto de vista técnico, de llevar a cabo por un ciberdelincuente son aquellos que buscan suplantar la identidad de una compañía creando websites muy similares a los legítimos pero que se encuentran alojados en infraestructuras fraudulentas bajo el control de los atacantes.

El principal problema es que los atacantes pueden optar por diseñar y llevar a cabo su intento de fraude de distintas maneras y es por ello que las pruebas deben realizarse para intentar cubrir los distintos escenarios que se plantean en una situación real.

Con esta finalidad se han tenido en cuenta tres posibles mecanismos de detección en el proyecto, cada uno de los cuales irá enfocado a detectar cada una de las metodologías seguidas por los cibercriminales.

### 5.1.1 *Phishing* por clonado

Los atacantes buscan la relación coste-beneficio más favorable para lograr sus objetivos. Un ataque de inversión mínima y que no requiere de grandes conocimientos técnicos es realizar el clonado de la web legítima utilizando herramientas como *HTTrack [23]*, como se muestra en la Figura 5-2.

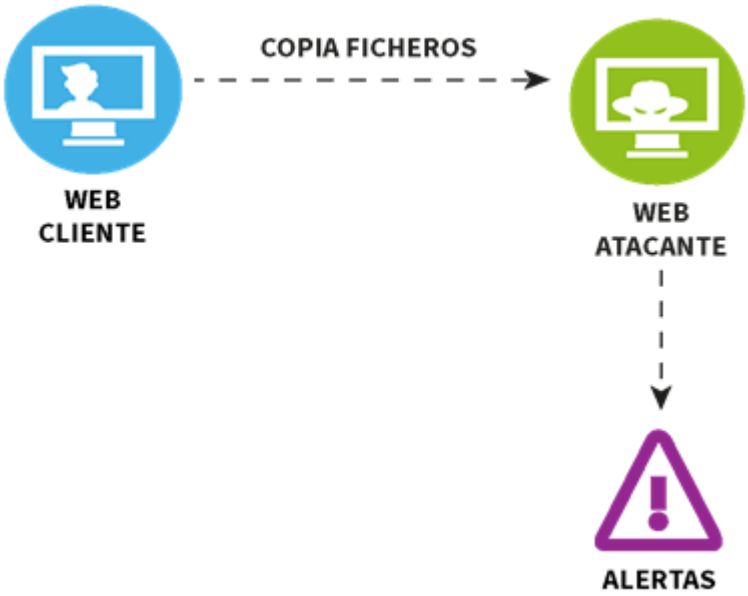



**Ilustración 5-2: Esquema de detección por clonado**

Para este tipo de ataques el mecanismo de detección implementado es el *CloneScript* introducido como Javascript dentro de la página web. De este modo, cuando el atacante se descarga los archivos para realizar la copia se descarga también el mecanismo de detección que, al ser ejecutado con un servidor no legítimo, llamará al servidor legítimo para que inicialice el mecanismo de alerta. La Figura 5-3 muestra este caso.

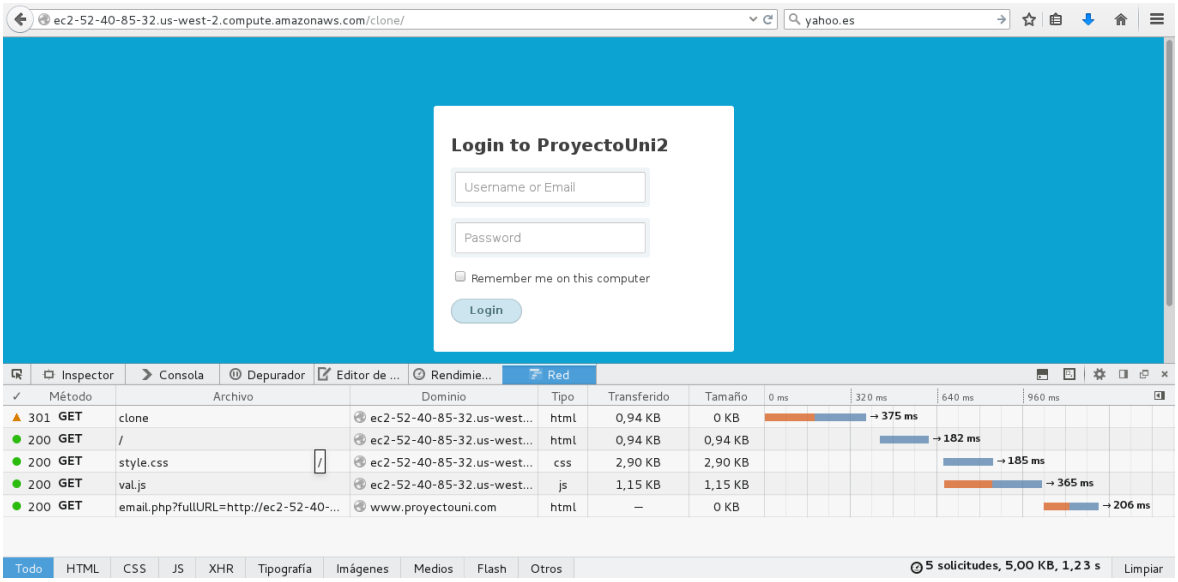



**Ilustración 5-3: Página clonada e invocación a sistema de alertas**

Esta alerta consistirá en el envío de un correo con un asunto determinado al buzón de correo configurado en el archivo email.php alojado en el servidor proyectouni.com. La Figura 5-4 muestra este tipo de mensaje de alerta.

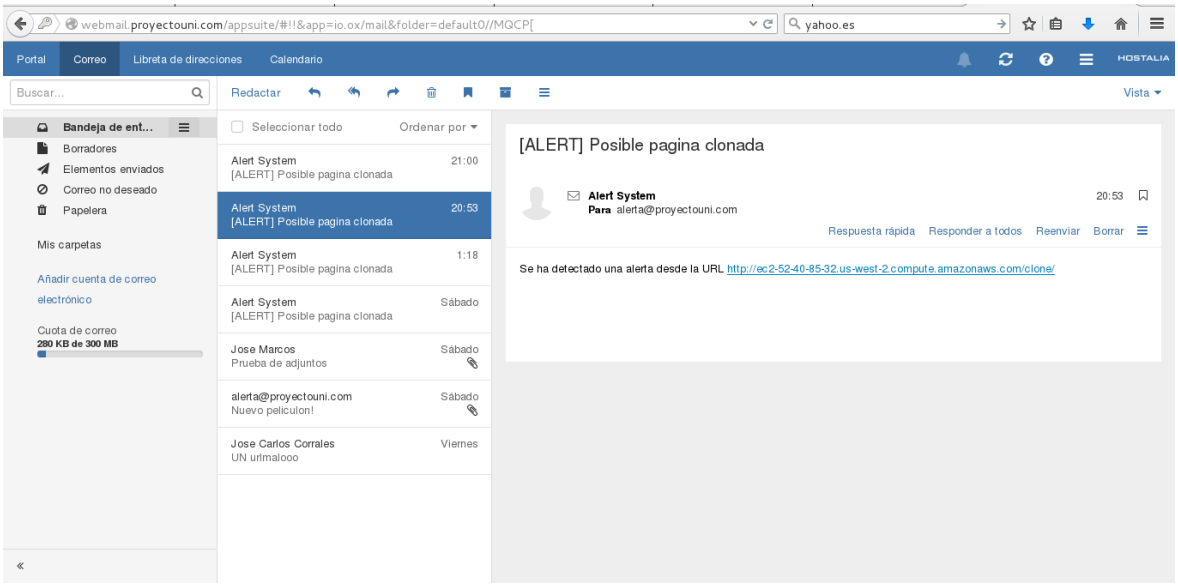



**Ilustración 5-4: Alerta recibida en el buzón de correo**

Como se puede observar, una vez el atacante ha desplegado la copia en el servidor malicioso alojado en amazonaws.com, el *script* de clonado realiza la validación y llama mediante petición GET al archivo email.php que se encarga de enviar de manera satisfactoria la alerta al buzón de correos, adjuntando además la URL atacante.

Posteriormente, y para que todo el proceso sea realizado de manera automatizada, se ejecutará la tarea programada en el cron que se encargará de recoger el correo y procesarlo mostrándolo en la pantalla de la consola del detector.

De este modo tenemos ya identificado un intento de suplantación de identidad y podemos actuar para mitigarlo, bien contactando vía administrativa con los ISPs, bien bloqueando en los sistemas corporativos la resolución al *host* malicioso, o incluso estableciendo reglas en los sistemas de correo que dificulten su distribución entre los potenciales empleados u objetivos dentro de la compañía.

### 5.1.2 *Phishing* manual con referencias al sitio web legítimo

En ocasiones los atacantes se pueden percatar de la existencia de mecanismos anticlonado o simplemente prefieren personalizar desde cero los websites que van a suplantar con el fin de añadir algún campo adicional en los formularios para poder recabar información complementaria de las víctimas (segundo factor de autenticación, detalles personales, etc.). Estos procesos son más costosos aunque evitan una detección tan inmediata como en los casos de los clonados.

En este tipo de ataques es común que, a pesar de no clonar la página entera, los atacantes referencien alguna parte de la web legítima dejando trazas en el servidor que pueden ser detectadas mediante el análisis de los *logs*. Durante el proyecto se ha optado por realizar una web (Figura 5-5) en la que las diferencias con la página legítima son muy evidentes (Roja vs Azul) y para generar las trazas en el servidor legítimo se ha introducido una redirección al mismo cuando la víctima del *phishing* rellena el formulario y le da al botón de *login*.

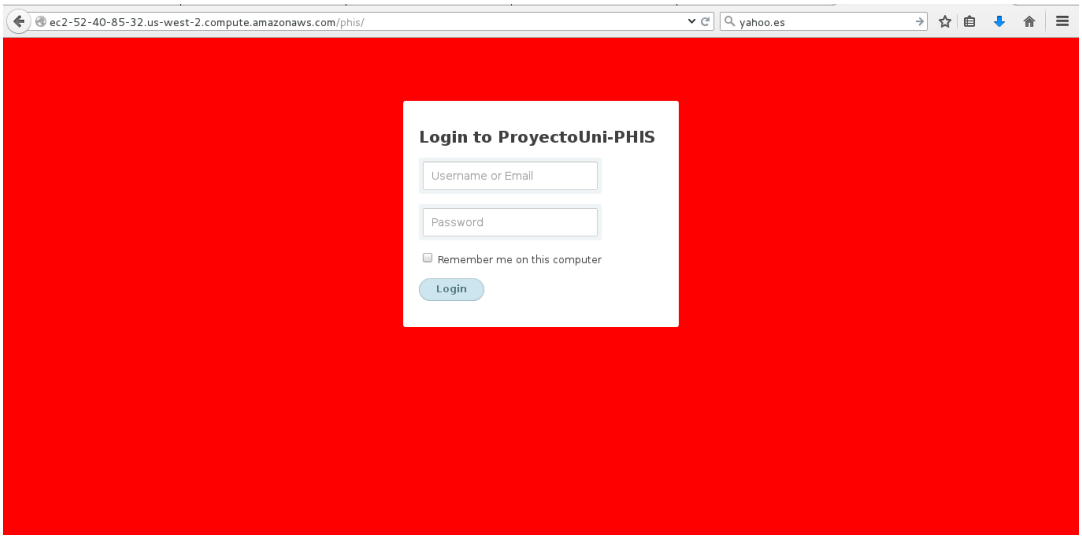



**Ilustración 5-5: Ataque de phishing manual**

Una vez se ejecuta el *cron* que procesa el *syslog*, y tras descartar los dominios incluidos en la lista blanca, se generan las alertas en el detector de fraude online categorizándolos como "Referer Sospechoso".

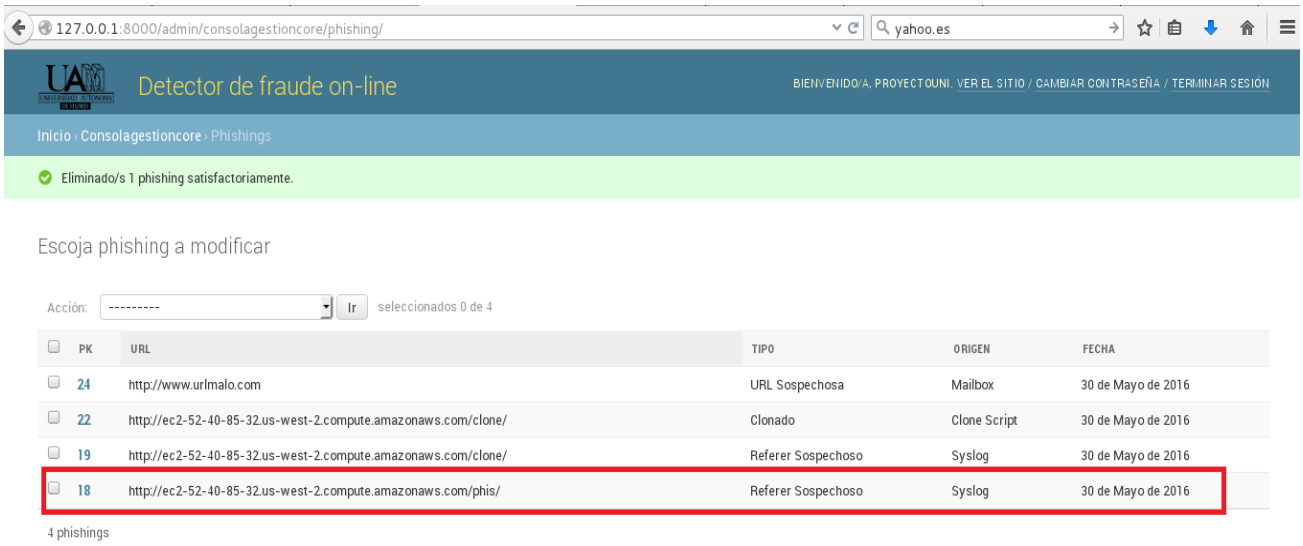



**Ilustración 5-6: Alerta de phishing en el dashboard**

Una vez más se pone de manifiesto la utilidad de la herramienta para detectar potenciales ataques filtrando los falsos positivos alojados en los *logs* de acceso del apache.

A continuación se muestra un pequeño esquema (Figura 5-7) en el que se explica el funcionamiento de los *logs* almacenados por el servidor y por qué es necesario filtrar los falsos positivos para evitar la inundación del sistema con alertas inservibles.

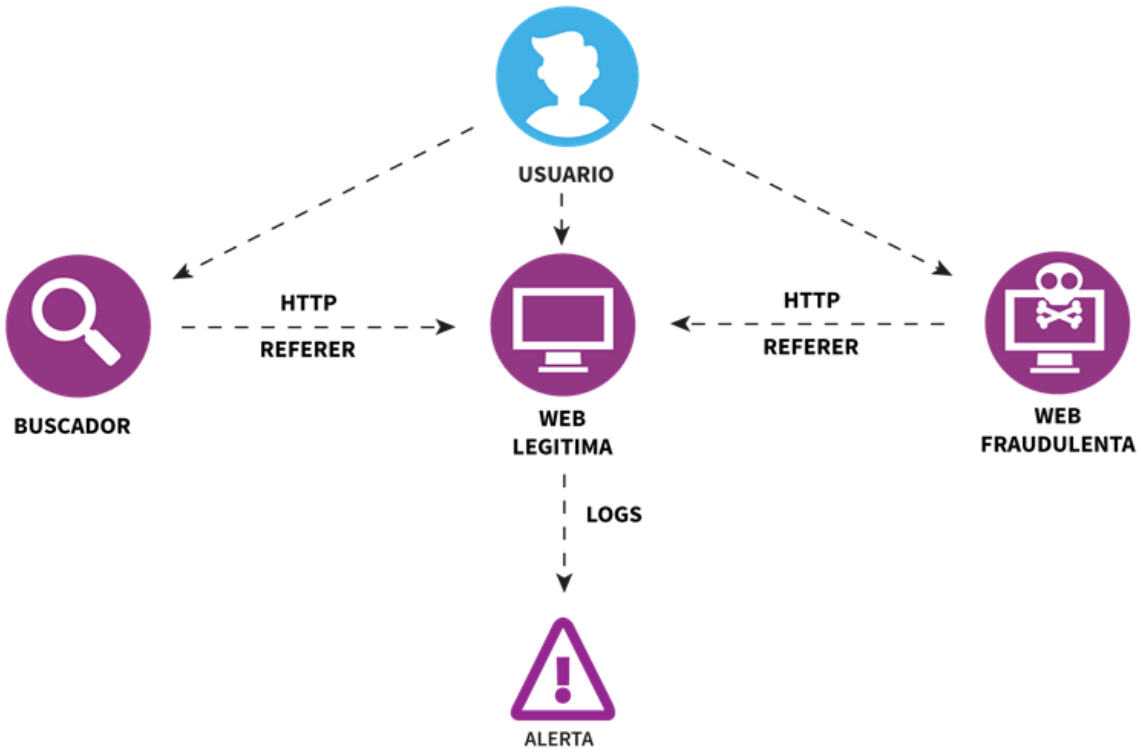



**Ilustración 5-7: Esquema accesos por Referer al servidor**

### 5.1.3 *Phishing* sin trazas recibido en el buzón de correo

Por último, el escenario menos deseado es aquel en el que el atacante no solo no clona la web evitando que el código trampa detecte la amenaza, sino que tampoco referencia en ningún punto del ataque la web legítima.

En estos casos, disponer de buzones trampa distribuidos por el mundo permite ampliar la capacidad de detección de los ataques "fantasma".

A continuación se han realizado pruebas enviando varios correos que contenían URLs al buzón configurado para las pruebas de este proyecto y se han observado los siguientes resultados:

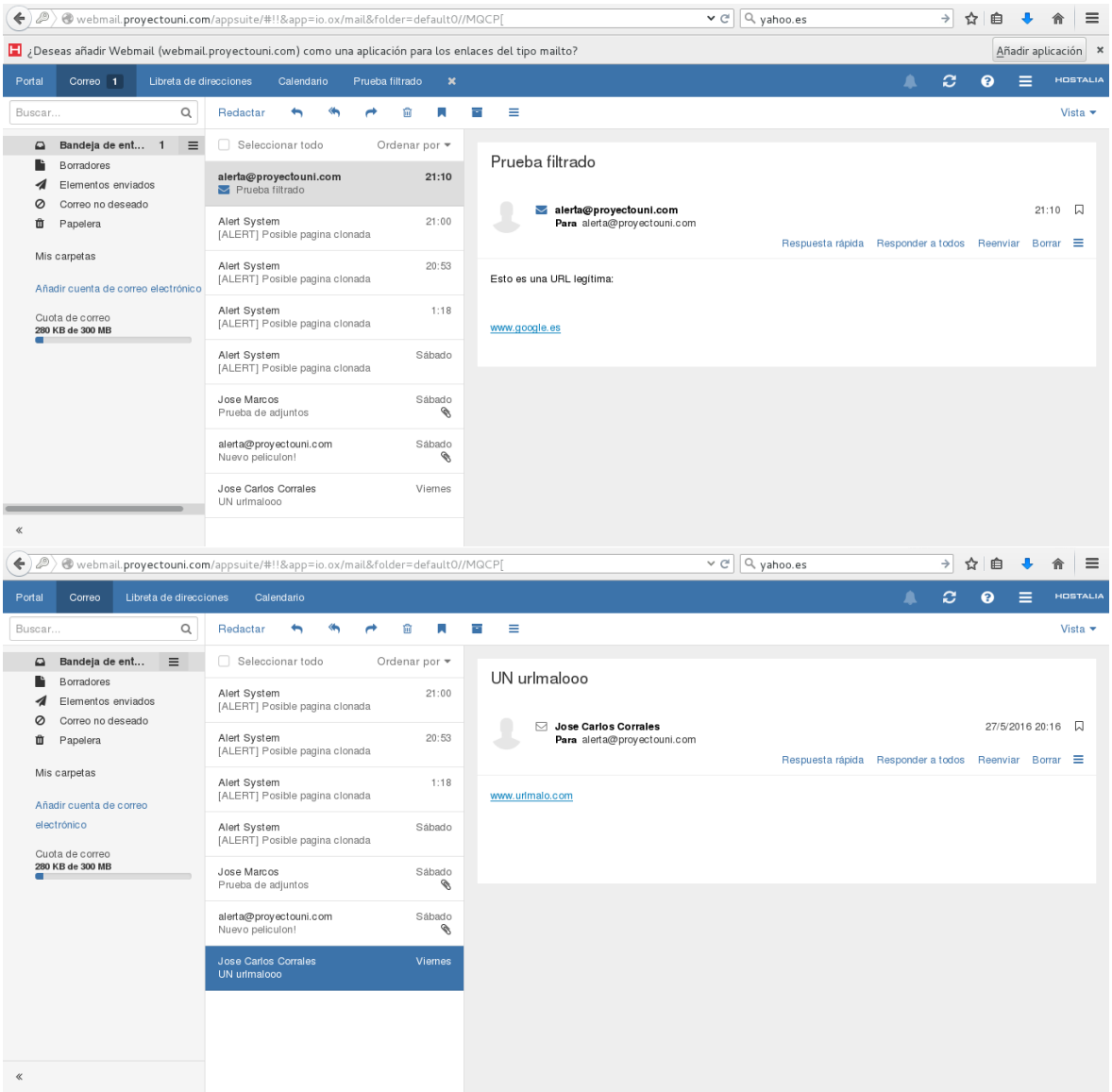



**Ilustración 5-8: Correos de prueba recibidos en el buzón**

Una vez el *cron* ejecuta la función *fetchmail* los correos son descargados y procesados, descartando aquellos que están incluidos en la lista blanca, en este caso *google.es*, y generando alertas para aquellas que no lo están, tales como *urlmalo.com.*

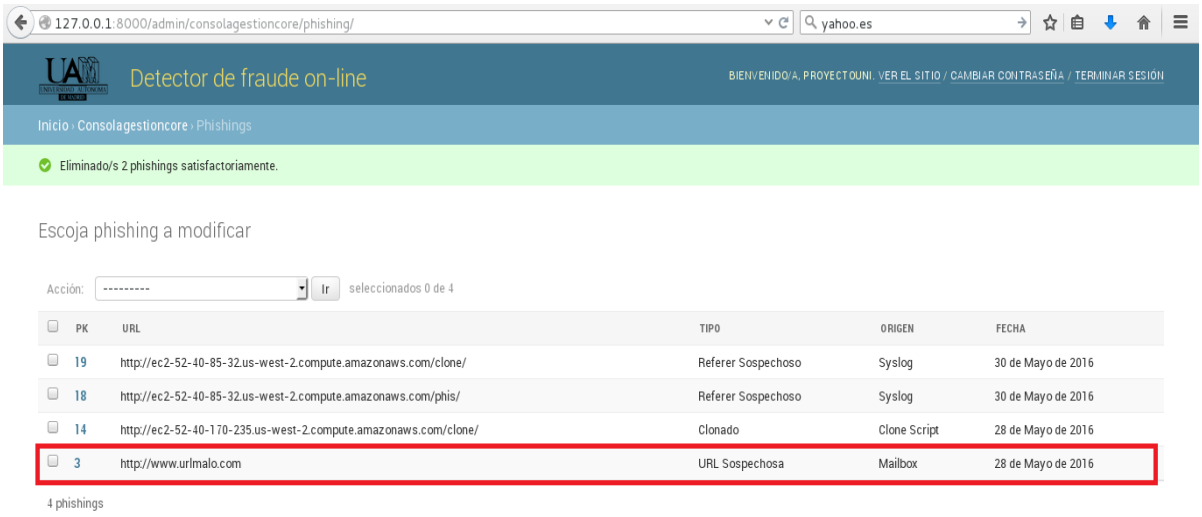



**Ilustración 5-9: Alerta de URL sospechosa en la consola**

.

## 5.2 Ataques de malware

Lo ataques vía *malware* son llevados a cabo por cibercriminales con mayor capacidad técnica y económica y es por ello que son amenazas que suponen un mayor riesgo y cuya detección y mitigación es más complicada.

Si la víctima se ve comprometida por uno de estos ataques, toda la información y documentos alojados en la máquina infectada estarán en riesgo de ser robados o secuestrados.

Por ello, en el proyecto se ha considerado integrar el análisis automatizado de los archivos adjuntos contenidos en los correos con el fin de dotar de visibilidad en fase temprana a los administradores de seguridad.

Para realizar las pruebas se han enviado varios correos en los que se han adjuntado distintos binarios maliciosos descargados de Internet.

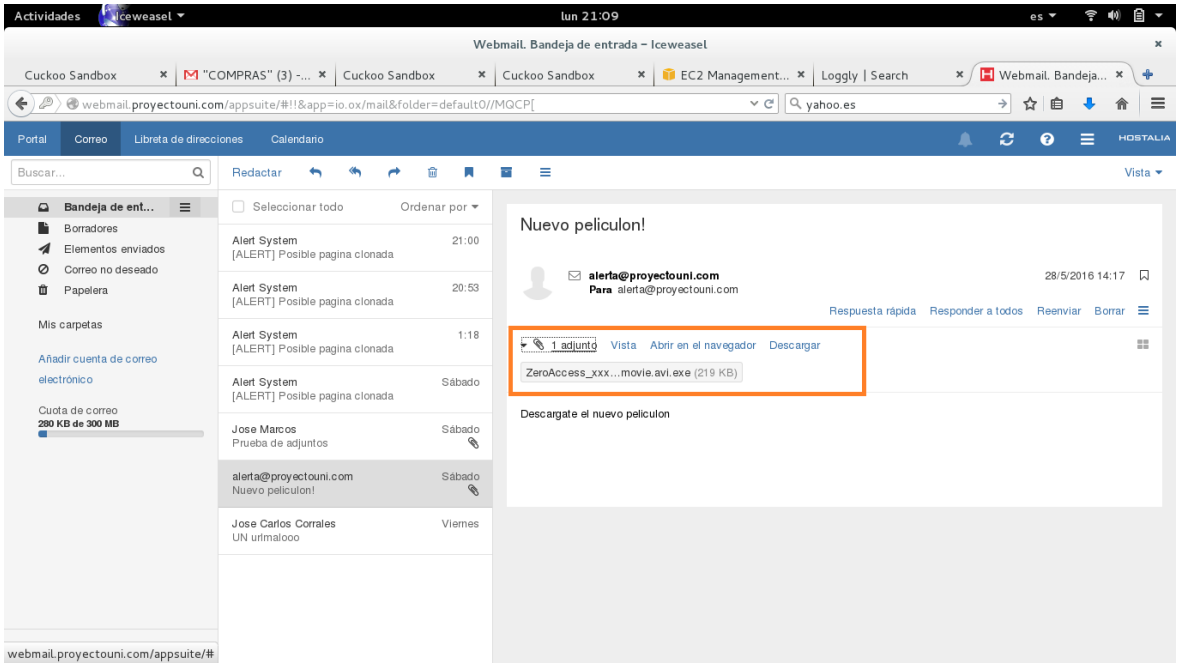



**Ilustración 5-10: Correos de prueba con adjuntos recibidos en el buzón**

Una vez se ejecuta la función *fetchmail* el adjunto es procesado y enviado a analizar a la *Sandbox* de Cuckoo, generando los resultados en la consola de gestión.

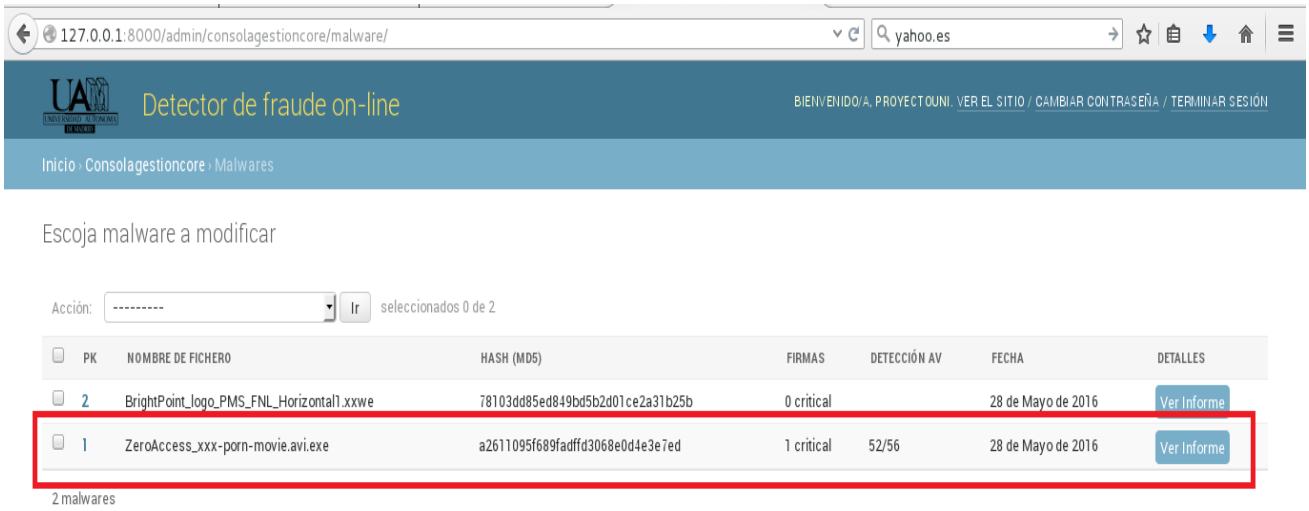



**Ilustración 5-11: Alerta de Malware en la consola**

Se puede observar como de manera automática se ha incluido el análisis en la consola bajo la pestaña *Malware*. También se detalla información que pueden permitir validar o descartar la alerta de manera sencilla en base a los campos Firmas (detalla si existen firmas que revelen criticidad en el análisis) y "Detección AV" (detalla si los antivirus del mercado consideran el adjunto malicioso).

Por último se incluyen datos que permitan monitorizar y bloquear los archivos que contengan el *hash* o el nombre relacionados con el archivo analizado.

Para profundizar en el detalle de la amenaza se incluye también la opción de ver el informe completo. Esto permitirá tomar mejores decisiones para bloquear en la infraestructura de la compañía la amenaza:

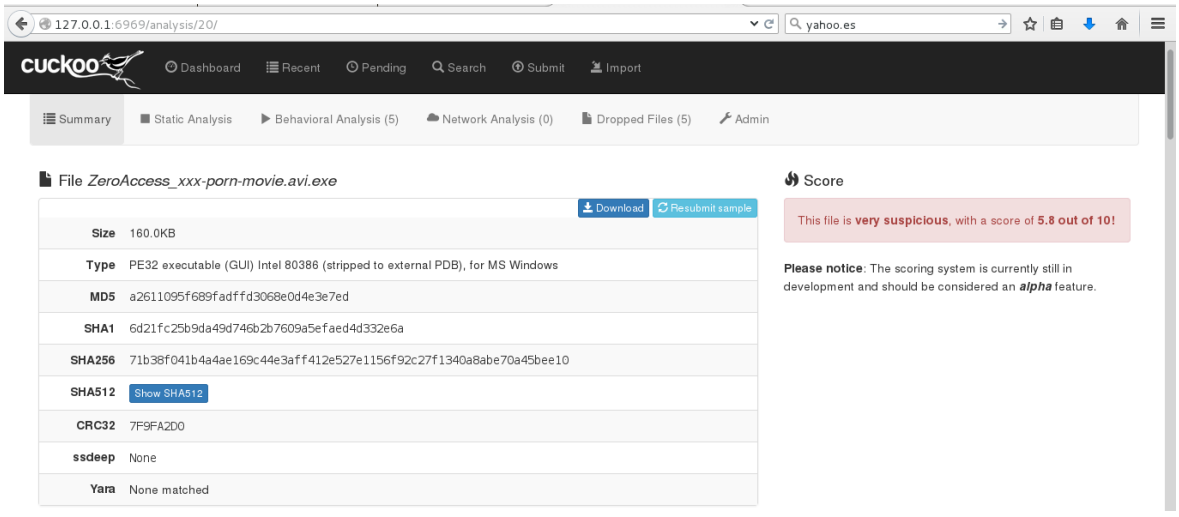
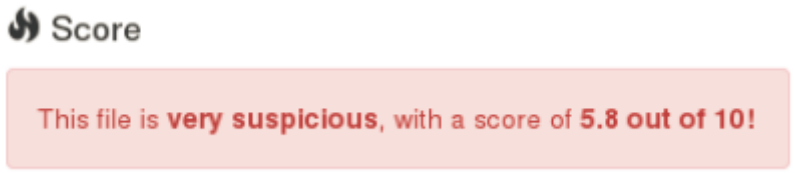



**Ilustración 5-12: Informe de análisis de malware (Pestaña Summary)**

El panel de resumen pone de manifiesto que el adjunto analizado se trataba de un binario malicioso.




Profundizando en la revisión del informe se pueden identificar también las firmas críticas identificadas. Las mismas dan una idea del comportamiento del archivo y permiten entender el riesgo de la amenaza.

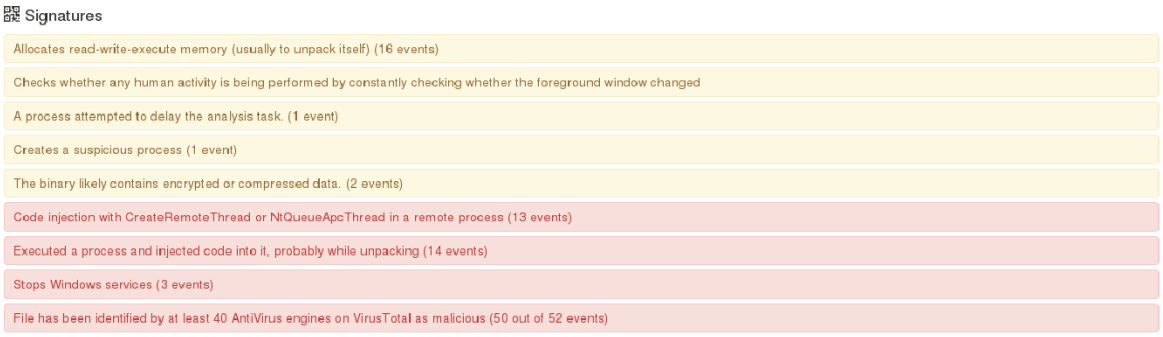



**Ilustración 5-13: Firmas criticas generadas en el análisis**

En dichas firmas se puede observar como menciona que el archivo es detectado por la gran mayoría de antivirus. Esta información también puede consultarse en la pestaña "Antivirus" donde además se detallará la familia identificada, en este caso ZeroAccess.

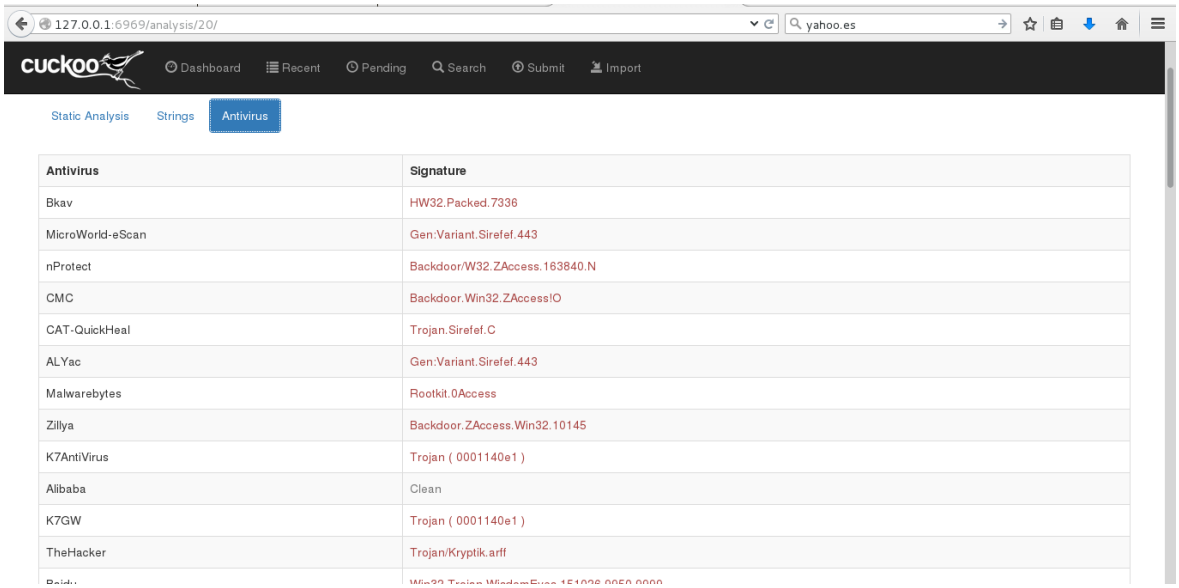



**Ilustración 5-14: Resultados de analizar el binario en los AV**

### 5.2.1 Rendimiento del análisis de archivos automatizado

Los análisis automatizados de archivos requieren de un procesado alto ya que para poder realizar todas las pruebas necesarias requiere de la creación de un entorno virtual y de la ejecución del archivo para comprobar su comportamiento.

Por ello se ha medido durante las pruebas el tiempo de los análisis para archivos maliciosos y para otros adjuntos inofensivos, obteniendo los siguientes resultados:

**Tabla 5-1: Tiempos de procesado de archivos**

| **Tipo de archivo** | **Tiempo medio/ejecución** |
|---|---|
| Malicioso (zeroAccess_xxx-porn-movie.avi.exe) | 139 segundos |
| Inofensivo (brightPoint_logo_PMS_FNL_Horizontal.xxwe) | 24 segundos |
| **TOTAL (Media)** | **82 segundos** |

Estos tiempos medios permitirían analizar con una sola *Sandbox* unos **1050 archivos por día** que son unas volumetrías inalcanzables si tuvieran que ser realizados los análisis manualmente.

## *5.3 Conclusiones*

Las pruebas realizadas con escenarios reales de ataques han demostrado la utilidad de una herramienta de detección como la diseñada durante este proyecto. Los administradores de seguridad tendrían visibilidad sobre los ataques y podrían mitigar, o al menos minimizar, el impacto de los ataques.

Además las pruebas también han demostrado lo eficiente de automatizar todo el proceso ya que manualmente se haría inviable procesar cantidades tan altas de información.

# 6 Conclusiones y trabajo futuro

## *6.1 Conclusiones*

El objetivo principal de este PFC era lograr ofrecer capacidades de detección de fraude on-line en aquellos escenarios en los que los usuarios son más vulnerables.

Las distintas pruebas llevadas a cabo en el capítulo 5 han confirmado las buenas expectativas existentes en el diseño de la herramienta, obteniendo tras la instalación de la misma una visibilidad muy amplia de aquellos ataques dirigidos a los usuarios de una compañía que se encuentran fuera de sus infraestructuras.

Ataques de *phishing* por clonado, ataques de *phishing* dirigidos o ataques de *malware* a través de SPAM pueden ser identificados por la herramienta con el fin de proveer de una capa inicial de alerta para la gestión de este tipo de ataques logrando minimizar el impacto de los mismos.

Además se ha demostrado como el filtrado de falsos positivos también es muy útil para permitir centrar la atención en aquellos eventos que supongan realmente un riesgo de ataque.

Por último, toda la información se centraliza y se envía sobre una consola accesible de manera pública y que nos permitiría visualizar y gestionar los incidentes de nuestra compañía en cualquier lugar del mundo con acceso a Internet.

Todo esto, además, se ha realizado de manera distribuida permitiendo que varios sistemas estén reportando sobre una misma consola, varios buzones puedan estar siendo procesados a un tiempo y varias granjas de análisis de malware se puedan poner en paralelo para incrementar las ya de por sí muy aceptables 1000 muestras analizadas al día.

## *6.2 Trabajo futuro*

Durante el desarrollo del proyecto se han detectado algunos campos de mejora sobre los que el PFC debería evolucionar. Estas áreas de mejora se centran principalmente en la ocultación de las conexiones realizadas a través del script de clonado, el cifrado del envío de los *logs* usando certificados de seguridad y una ampliación de la usabilidad de la consola de detección.

En relación a estas observaciones se adjunta el listado de trabajos futuros identificados:

1. **Ocultamiento del script de clonado**: A pesar de haber incluido el mecanismo trampa de detección en un Javascript y haberlo ofuscado y comprimido se hacían diversas peticiones desde el mismo hacia el exterior que permitirían a un atacante identificar el mismo y eliminarlo. Por ello es necesario establecer un mecanismo que se encargue de evitar que el script sea eliminado por los atacantes mediante técnicas que lo hagan imprescindible para la correcta visualización de la página.

2. **Protección del envío de log**. Durante el montaje del laboratorio se ha observado que el envío de *logs* a través de Rsyslog se realizaba en claro. A pesar de que posteriormente con Loggly se utilizaba un cifrado SSL, es importante que todos los eslabones de la cadena de comunicación cuenten con mecanismos de cifrado que eviten acceder a información tan confidencial como los *logs* de los servidores web. Además, si un atacante interceptara dicha información y pudiera alterarla evitaría la detección de los ataques realizados y seguramente colapsaría a los administradores de seguridad.

3. **Ampliar usabilidad de la consola**. En el proyecto se ha considerado el desarrollo de la herramienta como un mecanismo de detección que permita identificar ataques de suplantación de identidad, o *malware*. Sin embargo la misma podría ser ampliada para poder llevar la gestión de los incidentes desde la propia consola.
    a. Una máquina de estados con ciclos de gestión de los incidentes sería suficiente para determinar lo que ha sido gestionado sin necesidad de eliminar los históricos.
    b. Generar la clave primaria con una tupla (URL, ID) que permita evitar los duplicados. Un mismo ataque puede generar numerosas alertas y podría colapsar la tabla de datos.

4. **Consola de mando de Estadísticas**: con el fin de hacer muy intuitivo y visual el estado de los ataques on-line sufridos por la compañía se puede implementar un pequeño cuadro de mandos que permita filtrar por rangos temporales los estadísticos de los distintos hallazgos. De este modo se puede ver un histórico y analizar potenciales campañas estacionarias.

5. **Ampliar sistema de procesamiento**. Para grandes volúmenes de datos es necesario establecer el sistema de manera redundada y distribuida con el fin de poder soportar un mayor número de análisis. Para ello sería necesario establecer una granja de procesadores con balanceadores de carga que distribuyeran los datos.

# Referencias


[1] Verizon, INC “2015 Data Breach Investigation Report”, http://verizonenterprise.com/DBIR/2015,Abril 2015.

[2] APWG, Antiphishing Working Group “Phishing Activity Trends Report-q4-2015 http://docs.apwg.org/reports/apwg_trends_report_q4_2015.pdf, Marzo 2016

[3] Ponemon Institute LLC “2015 Cost of Cyber Crime Study:Global,” Octubre 2015

[4] https: //es.wikipedia.org/wiki/Phishing, última modificación 6 de Noviembre de 2015.

[5] https://es.wikipedia.org/wiki/Pharming, última modificación 10 Septiembre de 2015

[6] https://es.wikipedia.org/wiki/Malware,última modificación 30 Octubre 2015

[7] Carnegie Mellon University, “Trojan Horses”, http://www.cert.org/historical/advisories/CA-1999-02.cfm, Marzo1999.

[8] Ryan Angelo Certeza, TrendMicro Labs, “Dealing with the Mess of DRIDEX”, 13 de Octubre de 2015

[9] https://es.wikipedia.org/wiki/Keylogger, última modificación 12 de Marzo de 2016

[10] RFC 2616, "Hypertext Transfer Protocol—HTTP/1.1," by R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Mastinter, P. Leach, and T. Berners-Lee., Junio 1999

[11] RFC 1738, "Uniform Resource Locators (URL)," by T. Berners-Lee, L. Masinter, and M.McCahill, Diciembre 1994

[12] RFC 2396, "Uniform Resource Identifiers (URI): Generic Syntax," by T. Berners-Lee, R.Fielding, and L. Masinter., Agosto 1998

[13] Digit Oktavianto, Iqbal Muhardianto, “Cuckoo Malware Analysis”, PACKT, 2013

[14] Bautts, Tony, “Linux: Guía para administradores de redes”, Anaya Multimedia, 2005

[15] Coar, Ken, “Apache”, Anaya Multimedia, 2008

[16] FraudWatch,”using referral logs to detect attacks”, http://blog.fraudwatchinternational.com/using-referral-logs-to-detect-attacks/,2015

[17] Loggly Enterprise, “API: Retrieving Data”, https://www.loggly.com/docs/api-retrieving-data/

[18] RFC 5424, "The Syslog Protocol," by R. Gerhards, Adiscon GmbH, Marzo 2009

[19] Rsyslog, “Rsyslog, The Rocket-fast system for log processing”, http://www.rsyslog.com/

[20] http://resources.infosecinstitute.com/phishing-dangerous-cyber-threat/

[21] http://dean.edwards.name/packer/

[22] https://javascriptobfuscator.com/Javascript-Obfuscator.aspx

[23] https://en.wikipedia.org/wiki/HTTrack, última modificación 17 de Marzo de 2016

# Anexos

## *ANEXO A - Manual de configuración*

## *A.1- Instalación y configuración de Cuckoo*

Durante la memoria se ha explicado el funcionamiento y el uso de una *sandbox* que permita el análisis automatizado de los archivos adjuntos en los correos.

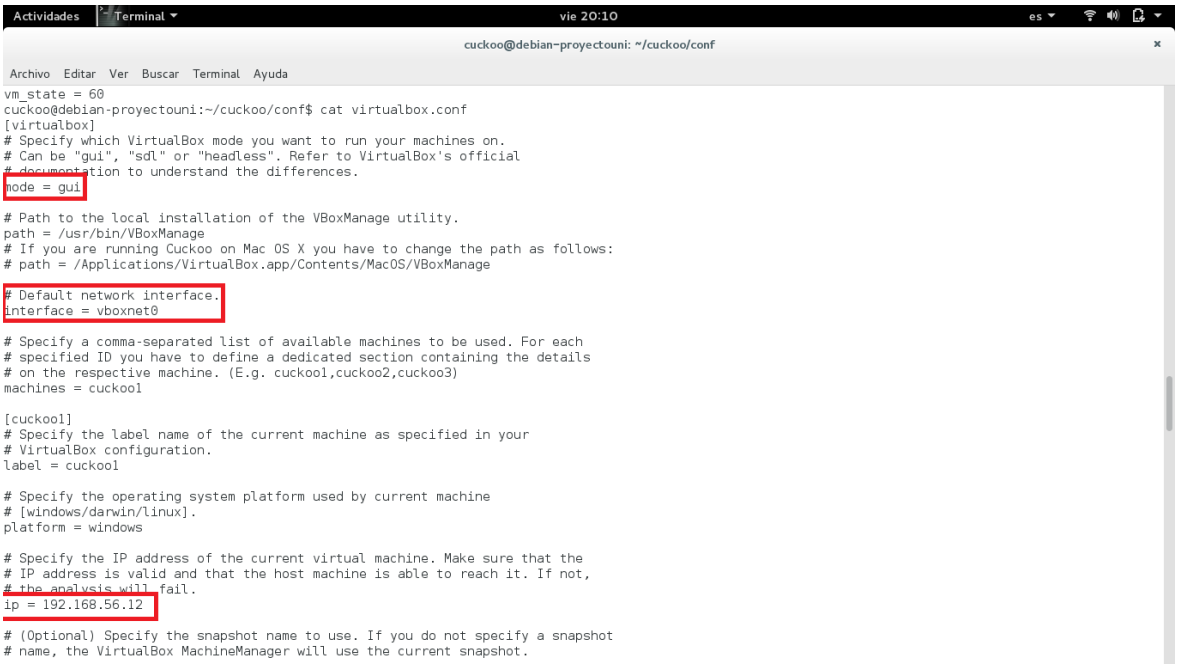
Para poner en funcionamiento ese mecanismo es necesario llevar a cabo la configuración descrita en las siguientes líneas:

1. Instalar un entorno de máquinas virtuales, en el proyecto se ha utilizado Virtualbox.
2. Instalar los módulos adicionales de Yara y volatility para el análisis de binarios.
3. Clonar el repositorio de Cuckoo: https://github.com/cuckoosandbox/cuckoo
4. Crear un usuario cuckoo que será el que "envíe" todos los análisis a la máquina.
5. Incluir el usuario cuckoo en el grupo de usuarios de vbox

   ```
   usermod –G  vboxusers –a cuckoo
   ```

6. Importar o crear una imagen para la máquina virtual.
7. Configurar la opción del adaptador de red para evitar tráfico externo.


8. Configurar el archivo virtualbox.conf: especificando las IPs de la máquina virtual, la interfaz de red, el modo de análisis, el nombre de las imágenes, etc.

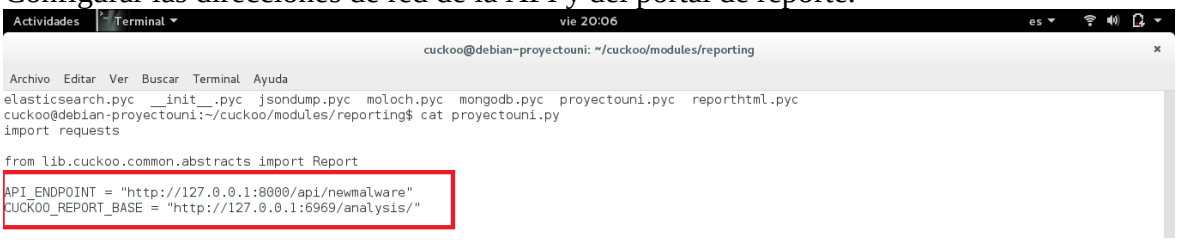


9. Configurar el archivo report.conf incluyendo las siguientes líneas para añadir el módulo proyectouni para el reporting a la consola del detector.

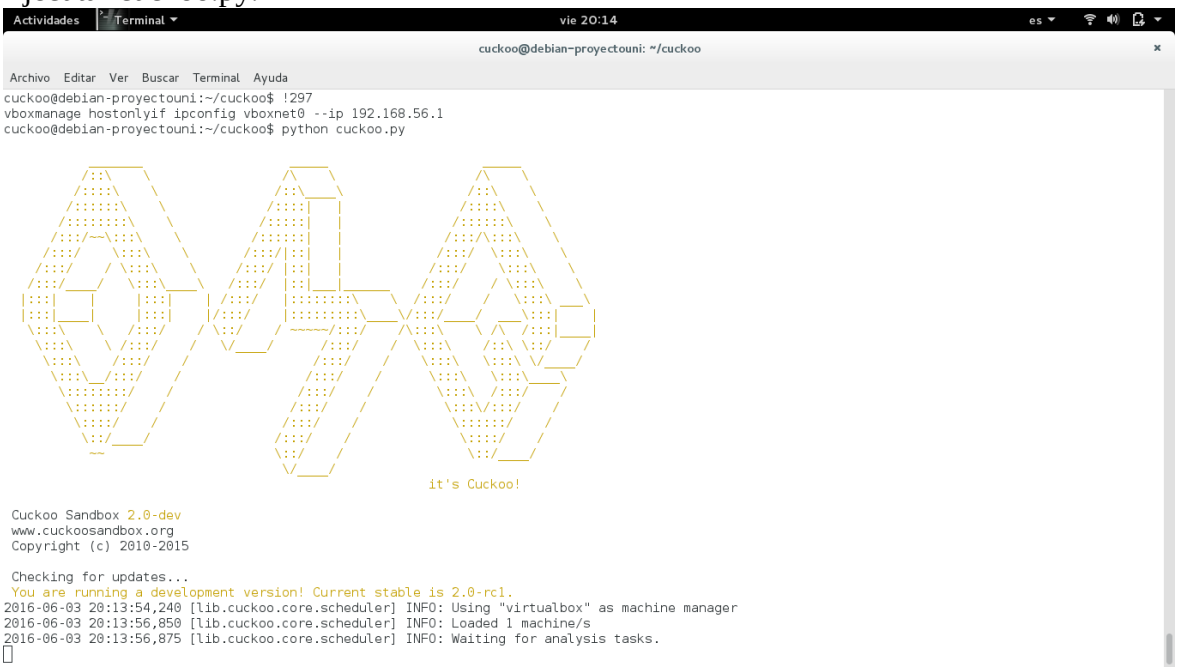
```
# moloch_capture = /data/moloch/bin/moloch-capture
# conf = /data/moloch/etc/config.ini
# instance = cuckoo

[proyectouni]
enabled = yes
```

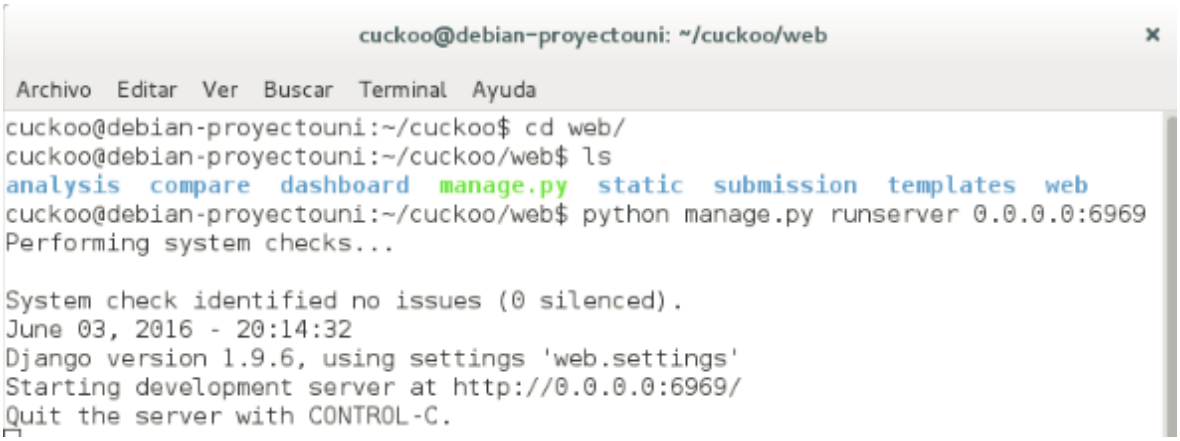
10. Configurar las direcciones de red de la API y del portal de reporte.




11. Ejecutar cuckoo.py.




12. Ejecutar Django y el panel web de Cuckoo (en el ejemplo se configura en la máquina local y en el puerto 6969):




## *A.2- Manual del detector de fraude*

El detector de fraude actúa en 3 etapas: Detección, procesado y representación. Para lograr llevar a cabo las pruebas ha sido necesario

La detección se realiza de manera desatendida por los distintos mecanismos implementados y únicamente será necesario programar el procesado de la información. Para ello, como se ha comentado en la memoria, es necesario ejecutar dos funciones: fetchmail y fetchsyslog.

En ambos casos la instrucción se hará utilizando la versión de python3 y pasando como argumento mínimo un archivo txt que contenga la lista blanca (niano.txt, en el ejemplo más abajo).

*Fetchmail*
En este caso, además, al pasarle el parámetro –A podemos hacer que se procesen todos los correos alojados en el buzón. En caso de no incluirlo, únicamente se procesarían aquellos nuevos.

```
vagrant@debian:/proyectouni/consolagestion$ python3 manage.py  fetchmail -A nian
o.txt
```

*Fetchsyslog*

En este caso se puede observar el output generado por pantalla en el que se constata como se descartan las URls legítimas incluidas en el archivo de lista blanca, evitando que se incluyan esos registros en la consola del detector.

```
vagrant@debian:/proyectouni/consolagestion$ python3 manage.py  fetchsyslog  nian
o.txt
URL legítima http://www.proyectouni.com/
URL legítima http://www.proyectouni.com/
URL legítima http://proyectouni.com/
```

# *ANEXO B – PLIEGO DE CONDICIONES*

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un sistema de detección de fraude online desarrollado en este PFC. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

## Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

## Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.

# *ANEXO C –PRESUPUESTO*

Para poder cuantificar el esfuerzo necesario para la ejecución del presente proyecto y poder realizar la consiguiente valoración económica asociada al mismo se va a proceder a detallar el presupuesto económico del mismo que constará de las siguientes partidas:

- Presupuesto de Ejecución material
- Gastos generales y Beneficio Industrial
- Honorarios por la redacción y dirección del proyecto.
- Presupuesto total

El presupuesto de ejecución material y los gastos generales y beneficio industrial constituyen el denominado *Presupuesto de ejecución por contrata* y junto a los honorarios por la redacción y dirección del proyecto van a suponer el presupuesto total cotizado.

En dicho presupuesto todas las cantidades representadas serán expresadas en Euros (€).

## Presupuesto de Ejecución material

El siguiente presupuesto hace referencia a los Costes de recursos materiales y a los costes de mano de obra básicos. No siendo incluidos en este apartado todos los costes derivados de la redacción y dirección del proyecto.

### Desglose por tareas ejecutadas

En este apartado se va a realizar el desglose pormenorizado de todas las tareas llevadas a cabo de manera que se pueda realizar una estimación precisa del esfuerzo realizado. De este modo se pretende detallar de manera clara y justificada los costes finales del presupuesto.

El proyecto consta de las siguientes tareas:

- ***Tarea 1:***

  *Objetivo:* Estudio del arte del cibercrimen. Cómo los atacantes se comportan, como desarrollan los ataques y cuáles son las víctimas. Además es necesario estudiar los distintos protocolos e identificadores que permitan detectar los puntos donde los ataques son visibles.

  *Duración:* 1,5 meses

  *Esfuerzo:* Ingeniero Superior, 1 persona-mes.

- ***Tarea 2:***
- 

  *Objetivo:* Preparación del entorno de desarrollo, instalación de los sistemas operativos. Estudio de las técnicas de análisis de malware y herramientas disponibles para realizar los mismos.

  *Duración:* 0,5 mes

  *Esfuerzo:* Ingeniero Superior, 1 persona-mes.

- ***Tarea 3:***

  *Objetivo:* Análisis de requisitos. En este punto se evaluarán las alternativas para la detección del fraude, se establecerán los requisitos mínimos para considerar exitoso el proyecto y los requisitos metodológicos necesarios para que la herramienta sea escalable y evolucionable

  *Duración:* 0,5 meses

  *Esfuerzo:* Ingeniero Superior, 1 persona-mes.

- ***Tarea 4:***

  *Objetivo:* Diseño de la herramienta. Flujos de funcionamiento y casos de uso.

  *Duración:* 1 mes

  *Esfuerzo:* Ingeniero Superior, 1,5 persona-mes.

- ***Tarea 5:***

  *Objetivo:* Implantación de la herramienta. Desarrollo en Python de los programas para la interconexión de los módulos de detección y procesado de ataques de fraude on-line.

  *Duración:* 1,5 meses

  *Esfuerzo:* Ingeniero Superior, 2,5 persona-mes.

- ***Tarea 6:***

  *Objetivo:* Implantación de infraestructura de pruebas: servidor web legítimo, servidor web suplantado, buzón de correos, registro de dominios, etc.

  *Duración:* 0,5 mes

  *Esfuerzo:* Ingeniero Superior, 1 persona-mes.

- ***Tarea 7:***

  *Objetivo:* Realización de las pruebas funcionales para garantizar la infraestructura es operativa y se comparta según los esquemas diseñados.

  *Duración:* 0,25 meses

  *Esfuerzo:* Ingeniero Superior, 0,5 persona-mes.

- ***Tarea 8:***

  *Objetivo:* Reconfiguración sistema de análisis de malware e introducción wildcards en la lista blanca.

  *Duración:* 0,25 meses

  *Esfuerzo:* Ingeniero Superior, 0,5 persona-mes.

- ***Tarea 9:***

  *Objetivo:* Realización de las pruebas y obtención de conclusiones.

  *Duración:* 0,5 meses

  *Esfuerzo:* Ingeniero Superior, 1 persona-mes.

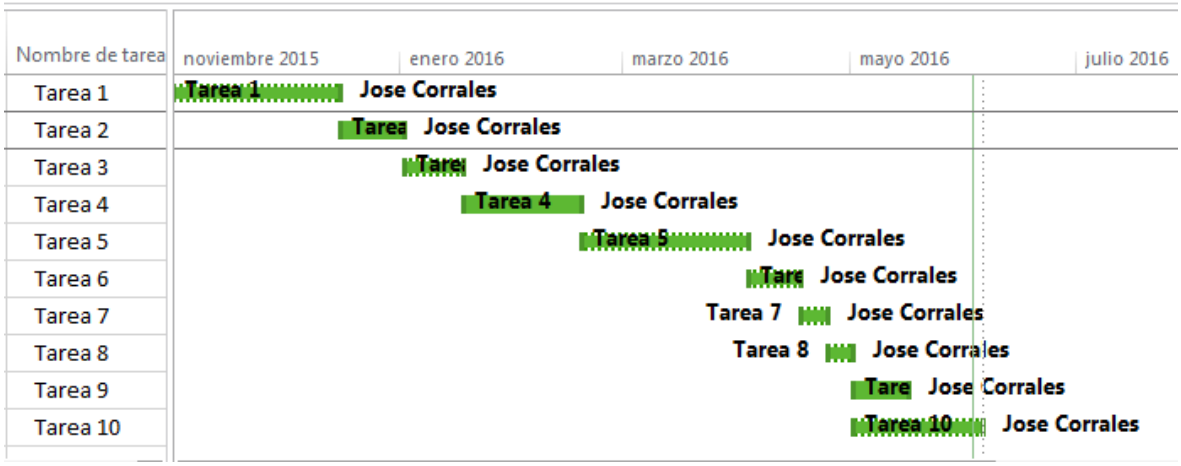
- ***Tarea 10:***

  *Objetivo:* Redacción de la documentación asociada al Proyecto Final de Carrera.

  *Duración:* 1 mes

  *Esfuerzo:* Ingeniero Superior, 1 persona-mes.

El seguimiento del plan de proyecto se puede llevar a cabo consultando el diagrama de Gantt:



### Costes de mano de obra

Para la realización con éxito del proyecto se requieren dos perfiles profesionales distintos:

- Un Ingeniero Superior de Telecomunicación encargado del planteamiento, desarrollo e implementación del trabajo técnico así como de la redacción, presentación y encuadernación del proyecto.

La estimación de los costes será llevada a cabo en base a los siguientes datos:

- Cotizaciones según el Régimen General de la Seguridad Social. Un ingeniero pertenece al grupo 1.

- Jornada laboral de 8 horas/día y 21 días laborables/mes.

El coste de la mano de obra para la ejecución del proyecto, considerando los datos anteriores y la distribución de esfuerzos definidos en el desglose de tareas asciende a 25.980 €.

Este coste salarial se descompone detalladamente en la siguiente tabla:

| **Costes Salariales** | |
|---|---|
| **Concepto** | **Ingeniero Superior** |
| Base Cotizable máxima mensual (01.01.2016) | 3.642,00 € |
| - Contingencias comunes (23,6%) | 860,22 € |
| - Desempleo (5,5%) F.G.S (0,2%) y Formación Profesional (0,6) (Total 6,3%) | 229,459 € |
| Coste de la Seguridad Social (€/mes) | 747,5 € |
| Salario bruto mensual (€/mes salario base 30.000€) | 2.500 € |
| Coste salarial mensual (€/mes) | 3.247,5 € |
| - Numero de meses | 8 meses |
| **Coste total de la mano de obra** | **25.980 €** |

### Coste de recursos materiales

En la siguiente tabla constan los costes de los recursos materiales empleados, considerando un periodo de amortización para el hardware y el software de 3 años.

En primer lugar se indicarán los costes totales y a continuación se imputarán las cuantías correspondientes a la amortización de los recursos durante su periodo de utilización en el desarrollo del proyecto (8 meses sobre 36 total, suponen un 23%).

Se ha considerado desglosar las partidas de costes materiales en aquellos costes derivados de la necesidad de adquirir recursos hardware y de aquellos que han supuesto costes en software o en licencias de hosting en cloud:

| **Recursos Hardware** | | | |
|---|---|---|---|
| **Concepto** | **Coste total** | **Meses** | **Coste Real** |
| Equipo de desarrollo (Lenovo Thinkpad T440p) | 1.577,40 € | 8 | 362,81 € |
| Servidor Dell PowerEdge R730 - 2.3 GHz - 32 GB RAM | 5.073,17 € | 8 | 1.166,83 € |
| **Total Recursos hardware** | | | **1.529,64 €** |

A continuación se detallan los costes en recursos software/cloud:

| **Recursos Software/Cloud** | | | |
|---|---|---|---|
| **Concepto** | **Coste total** | **Meses** | **Coste Real** |
| Loggly Enterprise | 307,06 € /mes | 8 | 2.456,48 € |
| Sistema Operativo Debian 8- Jessie | - € | 8 | - € |
| Oracle VM VirtualBox | - € | 8 | - € |
| Microsoft Office 2014 | 269,00 € | 8 | 61,87 € |
| Microsoft Windows 10 PRO | 279,00 € | 8 | 64,17 € |
| Cuckoo Sandbox 2.0 | - € | 8 | - € |
| Librería desarrollo Python 2.7 | - € | 8 | - € |
| **Total Recursos software/cloud** | | | **2.582,52 €** |

Por lo tanto el coste total unificado de este apartado asciende a: **4.112,16** €

| **Recursos Materiales** | |
|---|---|
| **Concepto** | **Coste** |
| Recursos Hardware | 1.529,64 € |
| Recursos Software/cloud | 2.582,52 € |
| **Total** | **4.112,16 €** |

### Coste Total de los recursos

El presupuesto de ejecución material se constituye de la suma de los costes en recursos materiales totales y los costes por mano de obra. En este caso asciende a:

| Presupuesto de Ejecución Material | |
|---|---|
| **Concepto** | **Coste** |
| Coste de mano de obra | 25.980 € |
| Coste de recursos materiales | 4.112,16 € |
| **Total** | **30.092,16 €** |

## Gastos generales y Beneficio Industrial

Bajo Gastos Generales se incluyen todos aquellos gastos derivados de la utilización de instalaciones, cargas fiduciarias, amortizaciones, gastos fiscales, etc. Igualmente, se incluye el beneficio industrial. Con esto, el Presupuesto de Ejecución por contrata queda como sigue:

| Presupuesto de ejecución por contrata | |
|---|---|
| **Concepto** | **Coste** |
| Presupuesto de ejecución material | 30.092,16 € |
| Gastos generales (16% del PEM) | 4.814,75 € |
| Beneficio industrial (6% del PEM) | 1.805,53 € |
| **Total** | **36.712,45 €** |

## Honorarios por redacción y dirección del proyecto

Los Honorarios calculados tanto para la redacción como para la dirección del proyecto son los asociados a Trabajos tarifados por tiempo empleado, con un valor de un 5.6%.

## Presupuesto Total

Finalmente sumando todas las partidas anteriores y aplicando la tasa impositiva del 21% IVA, se obtiene el presupuesto total final del proyecto.

| **Presupuesto Total** | |
|---|---|
| **Concepto** | **Coste** |
| Presupuesto de ejecución por contrata | 36.712,45 € |
| Honorarios por dirección | 2.055,90 € |
| Honorarios por redacción | 2.055,90 € |
| Subtotal | **40.824,25 €** |
| IVA (21%) | 8.573,10 € |
| **Coste Total Final** | **49.397,35 €** |

El presupuesto total del proyecto asciende a CUARENTA Y NUEVE MIL TRESCIENTOS NOVENTA Y SIETE Euros y TREINTA Y CINCO céntimos.

Madrid, julio de 2016

José Carlos Corrales Casas
El ingeniero proyectista