UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Proyecto de Grado

Estado del Arte

Autores:
Federico Nicolás Pernas
Javier Agustín Sanchez
Nicolás Zeballos

Supervisores:
Gustavo Betarte
Rodrigo Martinez
Marcelo Rodríguez

27 de diciembre de 2020

Índice

1. Introducción

Los Honeypots son dispositivos de seguridad que se describen como "cebos" para los atacantes, con el objetivo de alejarles de los activos principales de una organización, capturando información sobre las diferentes metodologías que conforman sus ataques. En la mayor parte de la literatura donde se plantea el uso de Honeypots, se les presenta como *dispositivos de red*, lo cual conlleva a definirlos conceptualmente en dicho contexto.

En la actualidad la mayor cantidad de ataques presentes se relacionan al uso de *aplicaciones web*, dado que las mismas se han posicionado dentro de los activos más importantes de cualquier organización, y por consiguiente a ser objetivo de los atacantes. Por lo tanto, este documento tiene la finalidad de realizar una investigación acerca del desarrollo y especificación de los *Honeypots de Aplicaciones Web*, buscando una asociación a las áreas de generación de *Indices de Compromiso (IOC)* y *seguimiento de atacantes*.

En la sección Contexto Teórico se presentan definiciones de Honeypots y sus principales características. En la siguiente sección Implementaciones de Honeypot se presenta los avances en la actualidad respecto a la implementación de este tipo de dispositivos. La cuarta sección Honeypots de gestores de contenido (CMS) introduce el concepto de gestor de contenidos (CMS por sus siglas en inglés) y se presentan las opciones disponibles de Honeypots para estos sistemas. En la quinta sección Honeytokens y breadcrums se describe y expone determinados recursos de sumo interés para identificar la presencia de atacantes. Finalmente en la sexta sección Sesiones de Usuarios se exhiben los mecanismos más utilizados para el manejo de sesión.

2. Contexto Teórico

Esta sección presenta una introducción a los conceptos primarios al trabajo de investigación abordado, los cuales son los **Honeypots** y las **Aplicaciones Web**. La finalidad es brindar un conocimiento acerca de los conceptos teóricos enmarcados en los **Honeypots** y las **Aplicaciones Web** para tomarlos como base en el resto de la documentación.

2.1. Honeypots

En la literatura actual referente a **Honeypots**, se presentan diferentes definiciones para estos sistemas, todas haciendo énfasis en diversas características. A continuación se presentan algunas de las definiciones que se considera más representativas conceptualmente.

En la definición establecida por Eric Cole, Ronald Krutz y James Conley en su libro *Network Security Bible* [1], se determina a un Honeypot como *"un sistema diseñado para parecerse a algo que un intruso pueda atacar"*, donde a su vez se menciona que *"son construidos para muchos propósitos, pero el principal es desviar los ataques y aprender de estos sin comprometer la seguridad de la red"*.

Joseph Migga Kizza, en su libro *Computer Network Security* [2], establece que un Honeypot es *"un mecanismo de señuelo monitoreado que se utiliza para mantener a un hacker alejado de recursos valiosos de la red y proporcionar una indicación temprana de un ataque"*.

Por otro lado, la definición de Lance Spitzner de su libro *Honeypots: Tracking Hackers* [3] hace mención a la gran cantidad de definiciones sobre Honeypots presentes en la literatura, definiendo a estos dispositivos como *"un recurso de seguridad cuyo valor radica en ser sondeado, atacado o comprometido"*.

En base a las definiciones presentadas, se puede definir a un Honeypot como :

Un dispositivo de seguridad, que funciona como "cebo" llamativo para los atacantes, lo cual provoca que se mantengan alejados de los activos principales y que se pueda generar información sobre sus ataques.

2.1.1. Propósitos y clasificación de Honeypots

Una primera clasificación para estos dispositivos es la referida *al propósito*, es decir, *se clasifica según el uso que se le dará*. Dentro de la literatura de Joseph Migga Kizza [2] y Lance Spitzner [3] se establecen las siguientes categorías junto con las funcionalidades que se espera cubrir en cada una.

Honeypots de Producción

Como su nombre lo indica, son Honeypots presentes en los ambientes de producción[1] de una organización. En particular, de este tipo de Honeypots se espera que permitan mitigar los problemas de seguridad que se podrían llegar a producir. Las funcionalidades esperadas de estos sistemas son:

Prevenir ataques

En términos de seguridad, la prevención de un ataque se puede expresar como *"mantener alejados a los atacantes"*. El fin de un Honeypot en este aspecto es que *atraiga a los atacantes* y estos se *mantengan interesados en investigarlo y atacarlo*, lo cual producirá que *consuman tiempo y recursos* en esta tarea.

Detectar ataques

La funcionalidad de detección esta asociada a *reconocer y alertar actividad sospechosa*. Este es uno de los trabajos más difíciles de realizar, debido a que en muchos sistemas de detección se producen, principalmente en etapas tempranas, una gran cantidad de *falsos positivos*[2].

Los Honeypots agregan una simplificación al problema anterior, dado que éstos no se basan en analizar todo el tráfico presente sobre una red o el recibido por un host (como si hacen los dispositivos NIDS y HIDS[3]), sino que asumen que *el tráfico que llega a ellos es categorizado como malicioso*. De esta manera, la posibilidad de tener falsos positivos es casi nula.

A su vez, la hipótesis anterior sobre los Honeypots ayuda a reducir los *falsos negativos*[4]. El principal problema en evitar falsos negativos son los *"nuevos ataques"*, dado que no se tiene información previa sobre ellos, pero los Honeypots pueden ayudar a recolectar dicha información sobre el nuevo ataque y a partir de la misma generar mecanismos para prevenirlos a futuro.

[1] Un *ambiente de producción* se define como el conjunto de componentes físicas y lógicas las cuales están expuestas para ser accedidas por los usuarios finales

[2] Comportamiento normal que se toma como malicioso

[3] Los *Host Intrusion Detection System (HIDS)* y *Network Intrusion Detection System (NIDS)*, son sistemas que analizan la integridad de un dispositivo o una red y en caso de descubrir un comportamiento anómalo envían una alarma

[4] Acciones maliciosas que se interpretaron como normales

Responder ante ataques

Comúnmente en una organización, un ataque perpetuado sobre un activo importante puede significar un riesgo elevado, debido a que la mayor parte de las veces se debe apartar al activo afectado de la red principal para el análisis del ataque y así también evitar que se continúe su propagación. Esta contramedida puede llegar a provocar que se pierda la disponibilidad de algún servicio brindado.

Sin embargo, si un atacante está actuando sobre un Honeypot, sus acciones podrían ser analizadas de manera *"offline"*, es decir, descargando los datos desde el Honeypot hacia otro equipo, permitiendo analizar el ataque y definir una contramedida sin interrumpir la operativa del Honeypot ni la de la organización, dado que los servicios accedidos por los clientes no han sido involucrados en el problema.

Honeypots de Investigación

Otra de las líneas de trabajo relevantes para los Honeypots es la referida a la "investigación". Este aspecto hace referencia a la capacidad que tienen estas herramientas de *mantener la información que generan los atacantes al realizar acciones sobre el dispositivo.*

La información recolectada permite que expertos en seguridad puedan definir perfiles de los atacantes, descubrir ataques desconocidos hasta el momento (denominados *"Zero-day Attacks"*), nuevas herramientas utilizadas por parte de los atacantes o incluso analizar el comportamiento de algún *malware* utilizado.

A su vez, el modelado de estos datos permite la definición de estructuras de datos que pueden ser utilizadas por otros dispositivos para reconocer los mismos ataques que se generaron en el Honeypot, las cuales se denominan **Indices de Compromiso (IOC)**.

Otra manera en que podrían clasificarse los Honeypots se basa en la *interacción que tienen con un atacante*. Esto permite definir una *"escala"* para comparar diferentes implementaciones de Honeypots. Cuanto mayor sea la interacción, más acciones podrá realizar el atacante sobre el dispositivo, permitiendo recabar mayor cantidad de información sobre sus acciones. Sin embargo, mayor interacción implica mayores riesgos, ya que el atacante podría llegar a comprometer el equipo donde se encuentra el Honeypot y usar el mismo como entrada a los sistemas de la organización.

Los niveles de interacción se clasifican en:

- **Baja Interacción**

  Estos Honeypots simulan ser un componente sencillo, el cual brinda respuestas simples sin permitir mucha comunicación con el atacante. Un ejemplo de este tipo de dispositivos sería el de exponer un servicio del cual se conocen vulnerabilidades (lo cual es atractivo para un atacante), pero la interacción con dicho servicio siempre retorna la misma respuesta.

  A pesar de que este tipo de Honeypots puede no llegar a ser muy atractivos para un atacante, tienen la ventaja de reducir el costo de puesta en producción, y gracias a su simplicidad se reduce el riesgo de comprometer la organización.

  Estos dispositivos están diseñados específicamente para capturar *comportamiento conocido*, dado que su determinismo no permite conocer nuevos tipos de ataques.

- **Alta Interacción**

  Este tipo de Honeypot, a diferencia del anterior, presenta un alto costo en la puesta en producción, dado que pretende simular completamente un servicio, sistema operativo, o cualquier otro tipo de componente complejo.

  A pesar del alto costo, los Honeypots de "alta interacción" son más atractivos para los atacantes ya que permite que se realice un mayor número de acciones sobre él. Esto deriva en la obtención de grandes volúmenes de datos del ataque generado, que es conocimiento de nuevos tipos de ataques, herramientas o malware.

  Sin embargo, esta mayor interacción involucra un riesgo elevado asociado a las vulnerabilidades que se puedan presentar en el Honeypot, de manera que el atacante logre comprometer el dispositivo.

- **Media Interacción**

  Los Honeypots de "interacción media" buscan una combinación entre las particularidades de los tipos mencionados anteriormente. De esta forma, se espera obtener un Honeypot con complejidad mínima para que sea desplegado en poco tiempo, y que a su vez genere la mayor interacción posible con los atacantes, para maximizar la obtención de información.

  Este tipo de Honeypot puede llegar a presentar la problemática de tener riesgos asociados por su nivel de interacción, pero sigue aportando más valor que un Honeypot de "baja interacción".

2.2. Principios de los Honeypots

Dentro del libro de Lance Spitzner, *Honeypots: Tracking Hackers* [3], en diferentes secciones se hace mención de algunas "propiedades" que deben cumplir los Honeypots. Estas propiedades pueden definirse como las siguientes:

No interferencia

Esta propiedad refiere a que los Honeypots tengan la capacidad de *ser agregados dentro de la infraestructura de la organización sin alterar el funcionamiento previo.*

No exposición

Los Honeypots son activos que *no deben ser expuestos* para reducir considerablemente la posibilidad de que un usuario normal de la aplicación llegue al mismo.

No establece comunicación con usuarios legítimos

En conjunto con la propiedad anterior, para evitar la mayor cantidad de falsos positivos es condición que *los usuarios legítimos no tengan acceso al Honeypot*, por lo tanto, para acceder al mismo se debe realizar una actividad sospechosa.

2.3. Aplicaciones Web

En los inicios de la Web, como menciona OWASP [4], la mayor cantidad de contenido estaba basado en páginas estáticas, las cuales eran simplemente texto plano con la finalidad de compartir información.

Al pasar los años, muchas empresas tuvieron la iniciativa de impulsar su negocio hacia la Web, dado que la misma brinda una conexión inmediata con los diferentes clientes, proveedores y socios, siendo una ventaja comercial importante. Este contexto desencadenó el desarrollo de las denominadas **Aplicaciones Web**.

La cantidad de componentes involucrados en la arquitectura de estos sistemas es muy amplia. Esto refiere a que no solamente se tiene código compilado ejecutando en una computadora local, como ocurría comúnmente con las aplicaciones de escritorio, sino que las Aplicaciones Web requieren de un *Servidor Web* para su funcionamiento, el cual se encuentra en un lugar remoto y puede ser accedido por diferentes usuarios mediante un *Navegador Web*, instalado en sus respectivas computadoras.

A su vez, las Aplicaciones Web se conectan con *bases de datos* para poder persistir datos de los usuarios y del sistema en sí. Al igual que con los servidores web, las bases de datos se pueden encontrar instaladas en un lugar remoto e independiente del cual se encuentra el servidor web, por lo que se puede deducir, que las *Aplicaciones Web trabajan normalmente en un ambiente distribuido*.

Se han inventado diversas tecnologías que permiten mejorar el rendimiento, ampliar la cantidad de funcionalidades y brindar comunicación entre diferentes aplicaciones. Un ejemplo de esto son los *Web Services*, los cuales se pueden ver como una parte del código de las Aplicaciones Web que mediante un lenguaje estándar denominado *XML*, permite que dos aplicaciones implementadas en diferentes tecnologías se puedan comunicar.

Las *Máquinas Virtuales* se involucran dentro del desarrollo de Aplicaciones Web y en la actualidad son utilizadas para crear *Contenedores*. Estos Contenedores permiten albergar un sistema operativo en el cual puede disponibilizar un servidor web con una aplicación contenida en él. Esto podría llevar a tener la misma aplicación distribuida en partes y cada parte ejecutar en un contenedor distinto.

Como se puede apreciar, las Aplicaciones Web involucran un conjunto variado y amplio de tecnologías y mecanismos que llevan a que las mismas sean cada vez más grandes y complejas. Este fenómeno lleva a que se tenga un trabajo constante y evolutivo para poder simplificar el desarrollo, puesta en producción y que a su vez se mejore la performance y aumente la disponibilidad de las mismas.

2.4. Seguridad sobre Aplicaciones Web

Las Aplicaciones Web presentan un gran desafió respecto a la seguridad. En gran parte por no contemplar un único punto de riesgo, sino por la potencial presencia de varias debilidades en diferentes sistemas que forman parte de su arquitectura.

Es de vital importancia para cualquier organización poder mantenerse actualizada y al pendiente de las recomendaciones de los expertos en seguridad informática, tener actualizados los sistemas de terceros que utilizan y tener conocimiento sobre nuevas violaciones a la seguridad de su empresa.

Open Web Application Security Project (OWASP) [5] es una organización *"dedicada en asistir a diferentes organizaciones para que puedan concebir, desarrollar, adquirir, operar y mantener aplicaciones en las que se pueda confiar"*, en donde todas sus herramientas, materiales e información son de acceso gratuito y están públicos para toda la comunidad.

OWASP presenta un proyecto conocido como **OWASP Top Ten Project** [6], el cual se describe como *"un poderoso documento que brinda conciencia sobre la seguridad en las aplicaciones web"*, en donde agregan que *"se incita que todas las organizaciones lo adopten para poder minimizar el riesgo en el desarrollo de aplicaciones web"*.

Dicho informe se presenta público en versión del año 2017 [7] (el cual también se encuentra traducido al español [8]). En el contenido del documento se encuentra información sobre las diez vulnerabilidades presentes en las Aplicaciones Web más explotadas por los atacantes dentro de los datos recolectados hasta el momento de publicación.

En la actualidad la lista está compuesta por las siguientes vulnerabilidades:

Injection: Los defectos de inyección se producen cuando se envían datos no confiables a un intérprete como parte de un comando o consulta. El atacante puede engañar al intérprete para que ejecute comandos no deseados o acceda información sin la autorización pertinente.

Broken Authentication: Autenticación y administración de sesiones se suelen implementar de manera incorrecta, lo cual permite a los atacantes comprometer contraseñas, claves o tokens de sesión, o asumir las identidades de otros usuarios de manera temporal o permanente.

Sensitive Data Exposure: Muchas aplicaciones web y APIs no protegen adecuadamente los datos confidenciales, como los financieros e historiales clínicos. Los atacantes pueden robar o modificar dichos datos para realizar fraudes con tarjetas de crédito, robo de identidad u otros delitos.

XML External Entities XXE: Muchos procesadores XML antiguos o mal configurados evalúan referencias de entidades externas dentro de documentos XML. Las entidades externas se pueden usar para revelar archivos internos, escaneo de puertos internos, ejecución remota de código y ataques de denegación de servicio.

Broken Access Control: La gestión de privilegios de usuarios autenticados suele hacerse de forma incorrecta. Los atacantes pueden explotar esos errores para acceder a funcionalidades y/o datos no autorizados, tales como cuentas de otros usuarios, ver archivos confidenciales, modificar los datos de otros usuarios o cambiar los derechos de acceso.

Security Misconfiguration: Suele ser el resultado de configuraciones predeterminadas inseguras, configuraciones incompletas o para un caso concreto, almacenamiento en la nube abierta, encabezados HTTP mal configurados y mensajes de error detallados que contienen información confidencial. Además de tener una configuración segura en sistemas operativos, frameworks, bibliotecas y aplicaciones, también deben ser parcheados y/o actualizados a la fecha.

Cross-Site Scripting XSS: Se produce cada vez que una aplicación incluye datos no confiables en una nueva página web sin una validación, o actualiza una página web existente con datos proporcionados por el usuario utilizando una API de navegador que puede crear HTML o JavaScript. XSS permite a los atacantes ejecutar scripts en el navegador de la víctima para secuestrar sesiones de usuario, modificar sitios web o redirigir al usuario a sitios maliciosos.

Insecure Deserialization: Siendo la deserialización el pasaje de datos seriales que van por la red a su formato como objeto manipulable por la aplicación. Por lo general derivan en ejecución remota de código pero incluso cuando no es así, pueden usarse para realizar ataques, incluidos ataques de repetición, ataques de inyección y ataques de escalada de privilegios.

Using Components with Known Vulnerabilities: Los componentes, como bibliotecas, se ejecutan con los mismos privilegios que la aplicación. Si se explota un componente vulnerable, dicho ataque puede facilitar la pérdida grave de datos o la adquisición del servidor. Las aplicaciones y APIs que usan componentes con vulnerabilidades conocidas pueden debilitar las defensas implementadas y permitir múltiples ataques.

Insufficient Logging & Monitoring: En conjunto con la falta o ineficiencia en la respuesta a incidentes, permite a los atacantes prolongar y mejorar sus ataques contra el sistemas, llegar a más sistemas a través de la red y manipular, extraer o destruir datos.

3. Implementaciones de Honeypot

3.1. Noción vigente y primeros desarrollos

Los primeros desarrollos de Honeypots, como se menciona en la sección Contexto Teórico, estaban enfocados a ser dispositivos de protección y vigilancia a nivel de red, donde se mantenían apartados de los activos principales.

En *Honeypots: Tracking Hackers* [3] se dispone de una sección dedicada a preguntarse que es lo que se espera de un honeypot, de manera que se simplifica o encamina a la decisión final de cual sera la opción mas acorde a desarrollar. La pregunta *¿Quieres proteger o quieres aprender?* allí planteada, permite seleccionar el objetivo principal del Honeypot a implementar, si será de producción o de investigación.

Cuando se implementa un Honeypot de producción, puede ser destinado a una de las siguientes tres categorías: prevención, detección y reacción. Para definir en cual de ellas se ubica mejor, se plantea la pregunta *¿Qué valor se esté buscando?*. Conjuntamente, se describen tres escenarios que permiten identificar el área más adecuada: si se busca *engañar a los atacantes, potencialmente confundiéndolos o ralentizándolos* esta área será *prevención*; si se espera *detectar ataques o identificar blackhats*[5] *penetrando su firewall o sus redes*, el área será **detección**; si se espera *mejorar su reacción a los compromisos del sistema*, entonces el área será **reacción**.

Cuando se opta por implementar un Honeypot de investigación se debe definir que es lo que se desea aprender. Dos preguntas ayudan en esta decisión: *¿Desea capturar código malicioso, aprender la técnica de un atacante o desarrollar habilidades de análisis forense?*, y *¿Está buscando identificar vulnerabilidades en la última versión de su servidor web?*.

En el libro de *Lance Spitzner* se presentan también tres criterios a determinar para la implementación de un Honeypot, el primero previamente mencionado en la sección Contexto Teórico es el nivel de interacción, cuanto mayor sea la interacción, más se puede aprender de los ataques. Luego se decide entre obtener un Honeypot COTS[6] o construirlo en su totalidad. Por último la plataforma de ejecución, elegir entre un sistema operativo u otro puede impactar en las prestaciones dadas y la performance.

Otro punto a tener en cuenta es la cantidad de Honeypots a instalar en una misma red, ya que *para organizaciones muy grandes con múltiples redes, un solo honeypot puede no ser suficiente*. En cuanto a la distribución de los mismos, se debe tener en cuenta que no se puede dejar vulnerable a la red interna. Esta razón lleva a que la red DMZ[7] sea una red candidata a contener un Honeypot ya que generalmente, esta red alberga servidores web, de email o DNS, y

[5] Se refiere a un hacker que irrumpe en un sistema informático o red con intención maliciosa.

[6] Los sistemas COTS (Commercial Off The Shelf) son sistemas comerciales, los cuales deben ser configurados previamente a su uso.

[7] DMZ: demilitarized zone (zona desmilitarizada). Una red DMZ es una red local que se ubica entre la red interna de una organización y la red externa [9]

suelen ser objetivo principal de los blackhats. Se recomienda además que un Honeypot destinado a protección no cubra la misma red que un Honeypot destinado a aprendizaje, pues de lo contrario se estaría interfiriendo en el grado de interacción del mismo.

3.1.1. Ejemplos clásicos

A continuación se presentan cuatro implementaciones de Honeypots presentes en *Honeypots: Tracking Hackers* [3], listando ventajas y desventajas de cada uno de ellos.

Honeyd

Honeyd es un Honeypot "open-source" diseñado para ser una solución dedicada a sistemas Unix. Fue creado en abril de 2002 por Niels Provos de la Universidad de Michigan, y aún cuenta con mantenimiento. Se categoriza como un Honeypot de bajo nivel de interacción pues solo emula servicios y no brinda un sistema operativo al cual el atacante pueda acceder.

En sus inicios se trató de un Honeypot de producción que apuntaba a detectar ataques o accesos no autorizados, emulando solo cinco servicios: QPOP 2.53, POP3, Microsoft IIS, FTP y SMTP. Pero debido a que es una herramienta open-source, la cantidad de servicios soportados aumento gracias a la colaboración de diferentes desarrolladores. Un aporte importante en su época fue la capacidad de detectar actividad en cualquier puerto que opte por el protocolo TCP para establecer una comunicación.

Honeyd tiene la capacidad de monitorear inmensas redes con IP falsas o que no corresponden a un sistema real. Por lo general, una limitante sobre la cantidad de redes monitoreadas suele estar en el rendimiento de su propia red que será el *cuello de botella*.

Como último punto a destacar se tiene su capacidad de emular múltiples sistemas operativos más allá de la capa de aplicación, ya que puede modificar el stack IP.

Ventajas	**Desventajas**
Puede monitorear cualquier puerto UDP o TCP y redes enteras.	Como solución de baja interacción, no puede proporcionar un sistema operativo real con el cual los atacantes interactúen.
Como solución open-source, es gratis y creció rápidamente con el apoyo de interesados en la seguridad informática.	Como solución open-source, no proporciona soporte formal para mantenimiento y resolución de problemas.
Evade dejar fingerprints que delaten la presencia del Honeypot gracias a emular servicios que llegan a modificar el stack IP.	No hay ningún mecanismo incorporado para alertar, ni capaz de capturar gran cantidad de información.

ManTrap

Con el objetivo de tener una opción más flexible y aumentar el nivel de interacción, Recourse Technologies creo ManTrap.

Esta herramienta brinda un ambiente completamente controlado, en el cual se permite emula varios sistemas operativos funcionales en su totalidad gracias al manejo de contenedores virtuales, en donde se destaca la capacidad de crear hasta cuatro ambientes virtuales sobre un único sistema físico. Su flexibilidad radica en que es capaz de adaptarse a casi todos los requisitos exigidos sobre un Honeypot.

Cubre las funcionalidades esperadas tanto por un Honeypot de producción como de investigación, sumando características funcionales como las de testeo y validación de mecanismos de seguridad.

Ventajas	**Desventajas**
Detecta actividad en cualquier puerto usando un dispositivo incorporado en sniffer[8].	Se asumen grandes riesgos relacionados con la alta interacción, pues los atacantes pueden usar el sistema para causar daños internos o externos a la organización.
Despliega un sistema operativo completo dedicado a asegurar alta interacción con el atacante.	Una vez que un atacante ingresó a uno de los contenedores de ManTrap, podrá reconocer fingerprints y dejar el contenedor.
Captura toda la actividad del atacante a través de espacio en el core del sistema, incluido el tráfico encriptado como SSH.	Limitado al sistema operativo Solaris, usando la versión completa para desarrolladores.

BackOfficer Friendly (BOF)

Uno de los Honeypot más fáciles de utilizar, pero por su simplicidad es poco recomendable por profesionales de la seguridad. De todas formas es una buena introducción al uso de Honeypots dado que es de baja interacción (mas simple de configurar) y permite ser ejecutado tanto en sistemas Unix como en Windows, para el cual fue creado en un principio.

Marcus Ranum junto al equipo de Network Flight Recorder lo diseñaron como una herramienta que responde al troyano "Back Orifice". Este Troyano se caracterizó por atacar los sistemas Windows 95 y Windows 98, y su fortaleza radica en su flexibilidad, la cual mejoraba gracias a la creación de los denominados BUTTplugs, plugins creados por la comunidad para introducir al malware con mayor facilidad. BOF pasó a ser la alarma que se disparaba al detectar que algún intruso estaba escaneando un sistema.

Tiene la capacidad de detectar y vigilar siete servicios: la irrupción del troyano Back Orifice y las manipulaciones en los protocolos FTP, Telnet, SMTP, HTTP, POP3 e IMPA.

[8]Es una aplicación creada para la captura de paquetes que viajan a través de la red

Ventajas	**Desventajas**
Fácil de instalar, configurar y mantener.	Limitado a siete servicios sobre siete puertos únicamente.
Funciona tanto en sistemas Windows como Unix incluyendo equipos de escritorio o personales.	Los puertos no se pueden personalizar, lo que aumenta la posibilidad de dejar fingerprints.
Por su simplicidad su uso implica un riesgo bajo.	Sin registro remoto, alertas o configuración de funcionalidades; por lo tanto, no es apropiado a nivel empresarial.

Specter

Otro caso de baja interacción pero que en comparación a BOF extiende la capacidad de detección y alerta sumando la recolección de información. Fue creado y mantenido por NetSec, una compañía de seguridad radicada en Suiza, con el objetivo de manejar una mayor cantidad de servicios simulados, lo cual sería más atractivo para el ambiente empresarial. Además tiene la capacidad de simular trece sistemas operativos, de manera que puede reconocer como aparecen distintos tipos de ataques en múltiples ambientes.

Una de sus debilidades está en la compatibilidad, dado que solo ejecuta en determinadas versiones de Windows por lo que pierde atractivo para el sector de la comunidad informática más a fin al software libre. Como funcionalidad destacada, se elige el potencial no solo para emular un servicio y su interacción, sino para llegar a emular incluso sus vulnerabilidades. Esto hace creer al atacante que logró su cometido con éxito, distrayendo al mismo o tan solo haciendo que pierda su tiempo.

Ventajas	**Desventajas**
Fácil de instalar, configurar y desplegar.	Permite monitorear un número bajo de puertos (únicamente catorce).
Simula varios servicios y monitorea el doble de puertos que BOF.	Los servicios emulados pre programados se limitan a interactuar ante comportamientos conocidos.
Excelente manejo de notificaciones y acepta el manejo remoto.	Discrepancias entre el stack IP y el sistema operativo simulado pueden generar fingerprints.

3.2. Pasaje a aplicaciones web

A comienzos de la segunda década de este siglo comenzó la investigación e interés en trasladar el concepto de Honeypot hacia las aplicaciones web. Una aproximación interesante se observa en el paper *A Generic Toolkit for Converting Web Applications Into High-Interaction Honeypots,* [10] donde se presenta una implementación capaz de modelar determinadas aplicaciones web desarrolladas en PHP como Honeypots, partiendo de la pregunta *¿Cómo se puede construir Honeypots de aplicaciones web para tener la flexibilidad de los Honeypots de alta interacción combinados con el consumo de recursos de los honeypots de baja interacción?*. En dicho paper se presenta un conjunto de requerimientos esperados para un Honeypot de aplicaciones web:

Funcionalidad invariada

En caso de modificar una funcionalidad, no debe verse perjudicada la interacción con el usuario. El Honeypot contendrá un subconjunto de funcionalidades modificadas, las cuales serán visibles por un atacante, ya que (Según la hipótesis planteada en Contexto Teórico), solo los atacantes acceden a un Honeypot.

Disponibilidad no perjudicada

El usuario no puede verse perjudicado por fallas asociadas al cambio arquitectónico en todo el sistema. La presencia del Honeypot no debe alterar la disponibilidad de la Aplicación Web.

Acceso restringido

El Honeypot no puede presentar vulnerabilidades que el atacante explote para tomar el control del sistema y a su vez se debe restringir el acceso desde el honeypot al sistema.

Además se destacan recomendaciones que definen el uso y manejo de la información recabada por un Honeypot:

Cobertura total de la información que involucra al atacante

Los mensajes y las acciones del atacante permiten definir perfiles de atacantes para clasificar e identificar accesos malintencionados. Cuanto mayor sea la obtención de información, más específico y enriquecido será el conjunto de datos que se creará para comunicar al resto de las aplicaciones sobre el reconocimiento de nuevos ataques.

Log externo

El registro de los datos externos al Honeypot permite un análisis y manejo de los mismos desacoplado, asegurando su integridad. Esto abre paso a la posibilidad de ver al Honeypot como componente externo donde el análisis queda desligado del seguimiento al atacante.

Soporte al análisis de los datos recabados

El procesamiento de datos es necesario para brindar, además de reconocimiento de tipos de ataques, conclusiones a partir de diversas estadísticas que serán el sustento para tomar decisiones de seguridad en la evolución de la aplicación web.

Extensible al registro de nuevos ataques y patrones

La evolución exacerbada de los ataques obliga a ser flexibles ante su reconocimiento. Un Honeypot sin esta capacidad tendrá una vida útil sumamente reducida.

Universal para toda aplicación web

Si se tiene el potencial de un Honeypot universal se abre paso a una defensa a gran escala contra los intrusos. Su adhesión a cualquier aplicación web será sencilla, dado que la base de conocimiento sobre su uso será tan extensa como los interesados en su mantenimiento, y a mayor obtención de datos, mayor velocidad para clasificar perfiles además de mayor cantidad de perfiles generados.

3.3. Desarrollos en la actualidad

En esta sección se abordan varias implementaciones de Honeypots de aplicación web, se presentan en síntesis y al final algunas estadísticas sobre ellos.

Cada referencia a los creadores del software es en base a su identificación de la plataforma GitHub que es el medio por el cual se publicó su implementación, a menos que se haya podido acceder al nombre real del creador.

La descripción de cada uno consta de: nombre del Honeypot, referencia al creador, motivo de creación y uso, noción temporal sobre cuando fue creado y cuando se realizo el último commit en su respectivo repositorio, que es un indicio de su actualización o mantenimiento. También se hace referencia al lenguaje de programación elegido para su implementación.

A continuación se lista cada una de las herramientas encontradas:

1. **Bukkit Honeypot** [11]

   Creado por *Argomirr*. Se trata de un Honeypot plugin para Bukkit, proyecto destinado a la creación de plugins para el popular juego MineCraft. En este contexto, el Honeypot permite la generación de una trampa para identificar y desviar de sus objetivos a los jugadores deshonestos. Fue creado en Java hace más de 9 años.

2. **EoHoneypotBundle** [12]

   Creado por *eymengunay*. El proyecto Symphony2 es un framework PHP destinado a facilitar el desarrollo web. La necesidad del Honeypot radica en generar una lista negra con

web robots malintencionados que buscan acceder a las web creadas con dicho framework. Fue creado hace más de 6 años y según el registro su último commit fue en abril des 2019.

3. **Glastopf** [13]

   Creado por *Lukas Rist* quien optó por Python para su implementación. Glastopf es por lejos el Honeypot con más documentación y con más opiniones de usuarios, además de ser la base para varias de las opciones que veremos. Apunta a recolectar datos de múltiples ataques tanto manuales como automatizados, lo cual permite la investigación detallada además de aprovechar cada dato para generar reglas de firmwall o firmas IDS[9]. Fue creado hace más de 7 años y su último commit fue a principios de Julio 2019.

4. **Tanner & Snare** [14]

   Creado por *Lukas Rist* junto al grupo de *mushmush.org* utilizando Python 3 para su implementación. Snare es un Honeypot de aplicación web que permite clonar paginas web y posteriormente publicar dichas paginas clonadas. Para generar las respuestas HTTP que debe realizar la página clonada, Snare trabaja en conjunto con Tanner. Tanner es un servicio de clasificación y análisis de datos remotos, evalúa las solicitudes y compone las respuestas. Para su funcionamiento requiere de un conjunto de diversas herramientas: Redis, PHP Sandbox, Docker y algún manejador de base de datos como SQLite. Ambas herramientas fueron creadas hace mas de 4 años y el último commit de Snare es de Mayo 2019 mientras que para Tanner es de Julio 2019.

5. **Google Hack Honeypot** [15]

   Creado por los profesionales de *Google* utilizando el lenguaje PHP. Diseñado para proporcionar reconocimiento contra atacantes que usan motores de búsqueda como una herramienta de hacking contra sus recursos. Fue creado hace más de 14 años, es predecesor al Honeynet Project, y su última actualización ronda el año 2013.

6. **Laravel Application Honeypot** [16]

   Creado por *Maksim Surguy*. Se trata de un paquete simple de prevención de spam para aplicaciones Laravel, que es un proyecto destinado a la creación web desarrollado en PHP. Este paquete crea un DIV oculto con dos campos, un campo honeypot arbitrario y un campo honeytime ocultos al usuario, una marca de tiempo cifrada que marca el momento en que la página fue entregada al usuario. Cuando el formulario retorna a la aplicación, un elemento de validación personalizado que viene con el paquete verifica que el campo honeypot esté vacío y también verifica el tiempo que le tomó al usuario completar el formulario. Si el formulario se completó demasiado rápido o si se puso un valor en el

[9]Las firmas IDS son firmas que identifican ataques conocidos, para distinguir el uso normal del fraudulento.

campo honeypot, se considera que esté envío es de un bot de spam. Fue creado hace más de 6 años y su último commit fue en Marzo 2019.

7. **Nodepot** [17]

   Creado por *Markus Schmall*, implementado en NodeJs. Honeypot destinado a dar soporte a las aplicaciones web desarrolladas en NodeJs. Su ambiente de implantación es de pequeño-mediano porte tal como un Raspberry-Pi. Fue creado hace más de 6 años y carece de indicios que nos permitan confirmar la mantenibilidad del mismo.

8. **Servletpot** [18]

   Creado por *Markus Schmall*, implementado en Java. Su objetivo principal es dar soporte al desarrollo web en base al uso de los servlets de Java. Fue creado hace más de 8 años y no se encuentra registro de actualizaciones. A pesar de ser los servlets una tecnología en vías de obsolescencia, es de interés evaluar como maneja los métodos GET y POST.

9. **Shadow Daemon** [19]

   Creado por *Hendrik Buchwald*. Es un Honeypot de alta interacción tanto para aplicaciones PHP como para aplicaciones Perl y Python. Apunta a módulos WAF (firewall de aplicación web), utiliza pequeños conectores a nivel de aplicación para interceptar. Garantiza que los datos analizados sean exactamente los mismos que los datos de entrada de la aplicación web, una tarea que muchos firewalls no pueden realizar correctamente. Fue creado hace más de 4 años y carece de actualizaciones posteriores.

10. **StrutsHoneypot** [20]

    Creado por *Nir Krakowski* y *Imri Goldberg*. Es un honeypot basado en Struts Apache 2, contiene un módulo de detección para servidores Apache 2. Struct Apache 2 es un framework destinado a la creación de aplicaciones web con Java. StrutsHoneypot busca detectar y bloquear los ataques que explotan la vulnerabilidad CVE 2017-5638 del framework. Fue creado hace más de 2 años y desde entonces no se realizo un commit posterior en el repositorio de GitHub.

11. **WebTrap** [21]

    Creado por *Dolev Ben Shushan*. Fue diseñado para crear páginas web engañosas que permitan la redirección de los atacantes fuera del sitio web real sin ser percibido. Tiene dos grandes componentes, WebCloner, para clonar las páginas web reales; y Deceptive Web-Server, encargado de publicar las páginas web y responder con reportes al servidor donde se mantiene el log, ambos desarrollados en Python. Fue creado hace más de 2 años y hace un año se realizo la ultima actualización.

12. **basic-auth-pot (bap)** [22]

    Creado por *bjborn*. Honeypot dedicado a las HTTP Basic Authentication desarrollado en Python. Fue creado hace más de 5 años y nunca se realizó otro commit.

13. **bwpot** [23]

    Creado por *graneed*. Honeypot para aplicaciones web vulnerables, dichas aplicaciones son creadas para ser fáciles de atacar y comprometer. A partir de las mismas se obtienen registros que se transfieren a Google BigQuery. Se opto por PHP para su desarrollo. Fue creada a principios de 2019 y no se a ha encontrado más actividad en el repositorio.

14. **django-admin-honeypot** [24]

    Creado por *Derek Payton*. Pantalla de inicio de sesión de administrador falsa de Django para notificar a los administradores de intentos de acceso no autorizado. Fue creada hace más de 8 años y se tiene un commit de mayo 2019.

15. **drupot** [25]

    Creado por *d1str0*. Honeypot para Drupal, el cual es un sistema de gestión de contenidos o CMS libre, modular, multipropósito y configurable para publicación de artículos, imágenes, archivos. También ofrece la posibilidad de otros servicios añadidos como administración de usuarios y permisos. Fue creado en GO a principios de este año y su último commit es de julio 2019.

16. **honeyhttpd** [26]

    Creado por *bocajspear1*. HoneyHTTPD es un honeypot de servidor web basado en Python. Facilita la configuración de servidores web falsos y registra las solicitudes que se le hacen. Fue creado hace más de 3 años y su último commit ronda finales de febrero 2019.

17. **phpmyadmin_honeypot** [27]

    Creado por *bgfoss*. Honeypot para phpMyAdmin, que es un software libre dedicado a la gestión de bases de datos MySQL en la web. Dicho honeypot falsifica la pantalla de inicio de administrador usando Django. Fue creada hace 6 años y su último commit es anterior a 2015.

18. **shockpot** [28]

    Creado por *jatrost*. Honeypot para detectar intentos de explotación de Shell Shock, desarrollado en Python. Fue creado hace 5 años y su último commit es de hace 4 años.

19. **smart-honeypot** [29]

    Creado por *jfreak3dot*. PHP Script aplicado a un honeypot. Fue creado en 2014 y no se realizó otro commit desde entonces.

20. **stack-honeypot** [30]

    Creado por *jChristoph Hochstrasser*. Inserta una trampa para bots de spam en las respuestas HTTP. Fue creado en 2013 utilizando PHP para su desarrollo. No se realizó otro commit desde su creación.

21. **tomcat-manager-honeypot** [31]

    Creado por *Helospark*. Honeypot que imita los endpoints del administrador de Tomcat. Registra las solicitudes y guarda el archivo WAR del atacante para su posterior estudio. Fue creado en Java hace más de 2 años y no se registraron commits posteriores.

22. **HonnyPotter** [32]

    Creado por *Martin Ingesen*. Honeypot de inicio de sesión de WordPress, desarrollado en PHP, para recopilar y analizar intentos fallidos de inicio de sesión. Fue creado hace más de 4 años y no se registraron commits posteriores.

23. **HoneyPress** [33]

    Creado por *dustyfresh*. Honeypot de WordPress basado en Python en un contenedor Docker. Fue creado hace 3 años y el creador comenta que por temas de tiempo no puede mantenerlo.

24. **wp-smart-honeypot** [34]

    Creado por *Ryan Johnston*. Complemento de WordPress desarrollado en PHP para reducir el spam de comentarios con un honeypot más inteligente. Fue creado hace más de 6 años y su último commit fue hace más de 2 años.

25. **wordpot** [35]

    Creado por *Gianluca Brindisi*. Wordpot es un honeypot de Wordpress que detecta irregularidades en complementos, temas, timthumb y otros archivos comunes utilizados para la creación de fingerprints en la instalación de Wordpress. Fue creado en Python hace más de 7 años y su último commit es de Noviembre 2018.

3.3.1. Caso destacado: Proyecto Honeypot de OWASP

La elección de destacar este proyecto por fuera de los listados es la importancia de ser un proyecto impulsado y avalado por OWASP, referencia en el mundo de la seguridad informática. A su vez es el único caso que apunta a un acceso universal a la información recabada, contando con el apoyo de la comunidad informática para desplegar OWASP-Honeypot sobre diversas redes.

El Proyecto Honeypot de OWASP, liderado por la Universidad Anglia Ruskin, tiene como objetivo identificar ataques contra aplicaciones web y posteriormente informar al respecto a

toda la comunidad. Gracias a este aporte se simplifica la implantación de protección contra tales ataques.

Distribuir el o los honeypots que creará el proyecto, a través de toda la comunidad, permite obtener mayor cantidad de información para lograr una retroalimentación más precisa sobre el actuar del atacante. Dicha retroalimentación posibilita la creación de reglas y técnicas capaces de definir buenas prácticas en el desarrollo web, prácticas que lograrán confrontar las vulnerabilidades que explotan los atacantes.

La recolección de datos será centralizada, en gran parte, en la Universidad Anglia Ruskin donde también se va a persistir dicha información para su posterior análisis. Por su lado Ali Razmjoo junto al grupo de ZDResearch son quienes dan seguimiento al avance en la creación del honeypot. En la actualidad se está en fase de investigación y desarrollo del honeypot, por tal motivo se avisa a cada interesado que se esperan errores y se agradece que sean reportados.

Se optó por Pyhton para su implementación y tiene una actividad continua en el repositorio de GitHub, registrando más de un commit en lo que va del año. Provee una interfaz web intuitiva que facilita la visualización de la información recabada en base a gráficas tal como las diez direcciones IP que reportaron más eventos intrusivos.

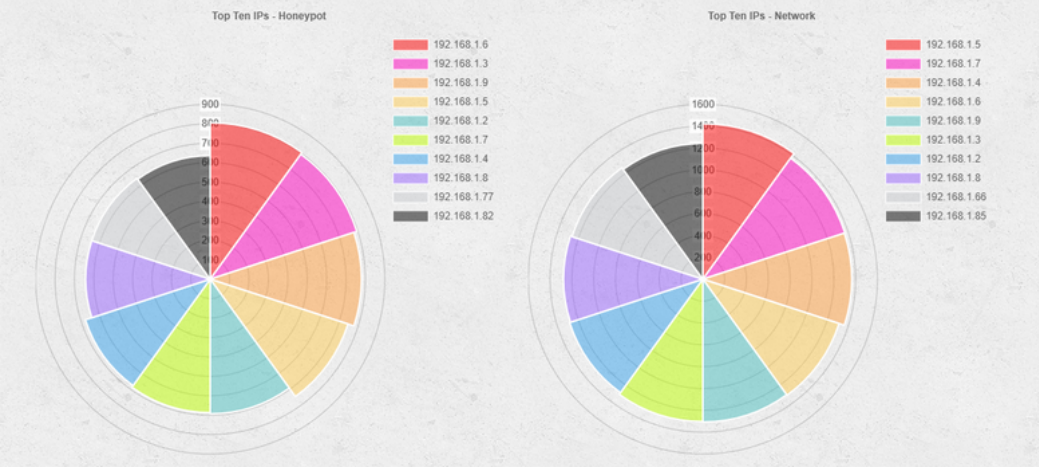

Figura 1: Interfaz web de OWASP Honeypot

3.3.2. Resumen y comparación

- **Generalidad**

  Una porción para nada despreciable del conjunto de honeypots se pensó para fines sumamente específicos, su creación fue con un objetivo "a medida" tal como *Bukkit Honeypot* que apunta a una funcionalidad de determinado juego; o *StrutsHoneypot* que bloquea los ataques que explotan determinada vulnerabilidad en *Apache 2*. De todas formas se logra identificar casos sumamente interesantes que brindan funcionalidades con potencial para ser generalizadas.

Sin embargo, se cuenta con honeypots que facilitan la configuración de servidores web falsos, como lo es *honeyhttpd*, o incluso honeypots capaces de clonar páginas web y luego redireccionar al atacante a dichas páginas, como el caso de *WebTrap*.

- **Vigencia y mantenimiento**

  El mayor debe esta en el mantenimiento, donde más de la mitad de los honeypot mencionados carece de algún commit en el último año. Este punto es de suma importancia si se considera como una reacción desalentadora al uso de honeypots de aplicación web. De todas formas varios de esos casos son demasiado acotados a una realidad específica, e incluso en uno de ellos el creador aclaro que por falta de tiempo no podía realizar el mantenimiento necesario.

Figura 2: Actividad en repositorios

Por las fechas de creación se puede asumir que existió un auge de investigación entre los años 2012-2014. Algunos de los pocos casos vigentes, considerando su vigencia por contar con mantenimiento activo, se sitúan próximos a dicha franja de años. Uno de esos casos es *Glastopf*, un honeypot de baja interacción que abrió paso a varias implementaciones destacadas como son *Snare* y *Taner*, reconocidas en el rubro.

Figura 3: Fechas de creación

- **Lenguajes de desarrollado**

  Tanto Python como PHP lideran las elecciones de cada desarrollador para la creación de sus honeypots, seguido por Java y un conjunto mínimo donde se destaca la presencia de una implementación con NodeJs un lenguaje sumamente usado hoy en día. En cuanto a compatibilidad se dispone de una implementación tan flexible que admite tres lenguajes de programación, es el caso de *Shadow Daemon*, capaz de dar sustento con un honeypot de alta interacción para aplicaciones web desarrolladas en *PHP, Python* y *Perl*.

Figura 4: Lenguajes de programación

4. Honeypots de gestores de contenido (CMS)

Un **gestor de contenidos** (**C**ontent **M**anagement **S**ystem) es una plataforma que permite crear y mantener un sitio web por parte de los diferentes usuarios a los cuales se le asigna un rol para controlar su labor sobre la plataforma [36]. Cabe destacar que los usuarios que emplean estos sistemas no necesitan de un amplio conocimiento técnico sobre informática.

4.1. Seguridad en un CMS

Muchos sitios creados con CMS pueden contener potenciales vulnerabilidad de seguridad, como por ejemplo, la mala validación de los campos utilizados en los formularios del sitios. Sumado a esto, los usuarios maliciosos o malintencionados, considerados atacantes pueden utilizar diferentes herramientas para realizar daños perjudicando la disponibilidad de los sitios, como lo es el uso de bots[10].

Para ilustrar los avances en seguridad presentes en los CMS, se mostrarán tres de los casos mas reconocidos: Drupal, WordPress y Joomla!.

4.2. Investigando honeypots en Drupal

Drupal es un sistema multiplataforma de gestión de contenido, escrito en PHP y de código abierto[38]. Fue creado en enero de 2001 y se encuentra actualmente en la versión 8.6.

En el área de seguridad, Drupal introduce los honeypots como medida para combatir el "spam"[39].

Dada la proliferación de bots que consumen recursos en las web, es de principal interés proteger los formularios web de un sitio Drupal, siguiendo el objetivo de realizarlo de la manera menos invasiva y transparente para el usuario. Una primera aproximación es la inclusión de campos de tipo *captcha*, los cuales emplean técnicas que requieren a un usuario "humano" para ser completadas. El problema de estos campos es que puede resultar invasivo para los usuarios que no presentan una amenaza a un sitio.

En 2011, Jeff Geerling crea un módulo para Drupal en base a lo que denominó *el método honeypot*. El punto de partida de esta idea fue la necesidad de proteger los formularios de registro de usuarios e inicio de sesión, los cuales recibían "cientos o miles de ataques de bots"[40]. El método emplea dos herramientas que funcionan de manera complementaria. Por un lado, se agrega a cada formulario un campo oculto que permita el ingreso de texto (un input, por ejemplo). Ya que el campo no es visible, un usuario bien intencionado no tenderá a modificar este campo, pero los bots sí, ya que se encargan de buscar en una página todos los campos que puedan ser editados y los llenan con información. El sistema verifica que el campo oculto quede siempre vacío. En caso que éste contenga información, no se permitirá realizar el envío del formulario.

[10] Un bot es un programa informático que efectúa automáticamente tareas repetitivas [37]

Por otro lado, se emplea una técnica basada en *timestamps (marcas temporales)* para poder identificar de mejor manera, si quien está completando un formulario es una persona o no. Sabiendo que los bots tienen como tarea completar la mayor cantidad de formularios en el menor tiempo posible, el método honeypot establece un tiempo mínimo de 5 segundos en los que un formulario no podrá ser enviado, ya que de lo contrario se sospechará que el llenado lo ha realizado una máquina.

El método anterior cuenta con algunas ventajas a su favor:

- Almacena información en los archivos de *log* del sistema.
- Contiene una API que permite la fácil integración de un honeypot en un sistema de producción.
- No requiere de instalación adicional ya que forma parte del "core" de Drupal.
- Es posible realizar un "baypass" al honeypot para ciertos usuarios, como administradores.

A pesar de estas ventajas, el método tiene la limitante de que permite saber únicamente si el atacante en un sitio es un bot. No permite, a priori, detectar a un usuario humano malintencionado.

Existen otras dos herramientas que aportan un nivel adicional de seguridad a un sitio Drupal: *drupo* y *fail2ban*. drupo es un honeypot que ya fue nombrado en la sección Implementaciones de Honeypot. Por su parte, *fail2ban* es una herramienta de tipo firewall que permite filtrar direcciones IP y prohibir que accedan a los sistemas de una organización. Integrado por Drupal, permite rechazar el acceso a todas aquellas direcciones que han sido rechazadas repetidamente por el honeypot.

4.3. Investigando honeypots en Wordpress

WordPress es un sistema de gestión de contenidos creado en mayo de 2003. Desarrollado en el lenguaje PHP, WordPress fue inicialmente de gran uso para crear sitios de tipo *blog*, siendo luego una de las herramientas más importantes para la creación de sitios web. Este CMS se destaca por la gran comunidad de desarrolladores y diseñadores que posee, los cuales se encargan de programar ya sea en el "core" de la herramienta, o creando distintas extensiones (*plugins*). En marzo de 2019, WordPress era usado por el 33.4 % de todos los sitios en Internet, y un 60.3 % de todos los sitios basados en gestores de contenido.[41]

Pasando al nivel de seguridad, las herramientas de tipo "honeypot" se presentan a modo de plugin, de los cuales se encuentran 10 de ellos [42]. De cada uno se presenta su nombre, número de versión actual, fecha aproximada de última actualización y una descripción de dicho plugin.

- **Honeypot for Contact Form 7 [43]**
  Versión del plugin: 1.41.1. Fecha de ultima actualización: setiembre 2019.

  Este plugin complementa a Contact Form [11], agregando funcionalidades básicas para evitar el spam, consiguiendo evitar la intrusión de bots sin la necesidad de agregar un captcha a un formulario. El funcionamiento y objetivo principal es el mismo que el módulo honeypot de Drupal.

- **Blackhole for Bad Bots [45]**
  Versión del plugin: 2.6. Fecha de ultima actualización: agosto 2019.

  Este plugin, como el anterior, tiene como objetivo detener el avance de los bots en un sitio web, de manera de no desperdiciar los recursos de los servidores de una organización.

  El funcionamiento es simple: el plugin inserta enlaces ocultos en los "footer" de cada sitio. Cada uno de ellos tiene un archivo *robots.txt* donde se agregan reglas de comportamiento para los bots. En particular, se agrega una que indica que los bots no deben ejecutar nunca los enlaces ocultos. Si uno de estos enlaces es accedido es porque algún bot no obedeció la regla establecida, por lo que el acceso a cualquier parte del sitio WordPress será restringido.

- **Formidable Honeypot [46]**
  Versión del plugin: 0.2. Fecha de ultima actualización: hace 5 años.

  Es un plugin con el mismo funcionamiento que **Honeypot for Contact Form 7**, con la particularidad que funciona únicamente con formularios Formidable. [12]

- **AP HoneyPot WordPress Plugin [48]**
  Versión del plugin: 1.4. Fecha de ultima actualización: hace 6 años.

  Basado en el plugin *http:BL WordPress Plugin* [49] [13], permite verificar las direcciones IP de quienes se conecten a un blog WordPress haciendo uso de la base de datos de Project Honey Pot [14]. De esta manera, es posible rápidamente verificar si quien accede es algún tipo de ente malicioso.

[11] Plugin que permite administrar múltiples formularios [44]

[12] Los *formidable forms* son un tipo especial de formularios de WordPress [47]

[13] Http:BL ([50]) es un sistema que permite a administradores de sitios web aprovechar los datos generados por el Project Honey Pot de manera de evitar el acceso a sus sitios por parte de bots malintencionados.

[14] Project Honey Pot es un sistema distribuido que permite identificar "spammers" y los "spambots" que estos emplean para obtener direcciones de un sitio web [51]

- **Honeypot Toolkit [52]**
  Versión del plugin: 4.0.9. Fecha de ultima actualización: setiembre 2019.

  Honeypot Toolkit funciona en base a Project Honey Pot como el anterior, y permite bloquear las direcciones IP que se encuentren en la lista Http:BL de dicho proyecto. Además, impide que los bots realicen ataques de fuerza bruta en formularios de inicio de sesión. Esto se realiza bloqueando el acceso de cualquier entidad que falla un número repetido de veces en el intento de acceso o que intenta ingresar con cuentas de usuario ya bloqueadas. Es posible también bloquear direcciones IP que generen una alta tasa de respuestas HTTP 404.

- **Honeypot for WP Comment [53]**
  Versión del plugin: 1.0.0. Fecha de ultima actualización: setiembre 2019.

  Este plugin permite filtrar los comentarios considerados "spam". En la descripción de la herramienta, se estima que su efectividad es del 50 %.

- **WP Spam Question Filter [54]**
  Versión del plugin: 1.0.1. Fecha de ultima actualización: hace 2 años.

  Como el anterior, este plugin funciona para prevenir los mensajes de spam en los comentarios de un blog WordPress.

- **Contact Form 7 Honeypot Plug [55]**
  Versión del plugin: 1.0.0. Fecha de ultima actualización: agosto 2019.

  El funcionamiento del honeypot es el mismo que **Honeypot for Contact Form 7**.

- **SpamPot [56]**
  Versión del plugin: 0.34. Fecha de ultima actualización: hace 3 años.

  SpamBot apunta a evitar spam en formularios de acceso y registro. Como algunos anteriores, funciona agregando un campo oculto al formulario en que actúa.

- **HoneyPotter [57]**
  Versión del plugin: 1.2. Fecha de ultima actualización: hace 4 años.

  HoneyPotter fue mencionado en Desarrollos en la actualidad.

4.4. Investigando honeypots en Joomla!

Joomla! es un CMS multiplataforma que permite desarrollar sitios web dinámicos e interactivos. Es un software de código abierto, desarrollado en PHP [58]. La primera versión fue lanzada en agosto de 2005, y se encuentra actualmente en la versión 3.9.

Respecto a seguridad, Joomla! cuenta con un gran número de plugins que sirven para distintos propósitos[59]. Si bien ninguno de ellos aparece bajo el concepto de "honeypot", las funciones de estos plugins son similares a los vistos para WordPress. Entre los objetivos de estos plugins, se encuentran:

- Detener el avance de bots a los sitios Joomla! mediante la adición de captchas.
- Proteger los formularios de inicio de sesión y registro.

Los plugins de este CMS aparecen bajo el concepto de "extensiones", las cuales se listan a continuación. Si bien ninguna de ellas aparece bajo el concepto de "honeypot", si lo hacen bajo el concepto de seguridad. Además, sirven para propósitos similares a los plugins presentados para WordPress. Es por esta razón que se listan las extensiones para Joomla!, de las cuales se presenta su nombre, número de versión actual, fecha aproximada de última actualización y una breve descripción de ellas.

- **RSFirewall! [60]**
  Versión del plugin: 2.11.26. Fecha de ultima actualización: Mayo 2019.

  RSFirewall! es una extensión que permite proteger un sitio Joomla! de intrusos y ataques de usuarios malintencionados. Contiene defensas contra ataques conocidos, como SQL injection, XSS o DoS.

- **Securitycheck Pro [61]**
  Versión del plugin: 3.1.8. Fecha de ultima actualización: Agosto 2019.

  Esta extensión ofrece seguridad a distintos niveles, protegiendo un sitio Joomla! sin afectar los recursos del servidor.

- **Antispam by CleanTalk [62]**
  Versión del plugin: 1.0. Fecha de ultima actualización: Agosto 2019.

  Esta extensión protege todo tipo de formularios Joomla! contra spam.

- **SpambotCheck [63]**
  Versión del plugin: 3.1.0. Fecha de ultima actualización: Noviembre 2018.

  Esta extensión impide el funcionamiento de bots, impidiendo que éstos logren realizar el registro de un nuevo usuario o el inicio de sesión de uno existente.

- **jSecure Lite [64]**
  Versión del plugin: 1.0. Fecha de ultima actualización: Agosto 2018.

  jSecure impide el acceso hacia la página de administración de un sitio Joomla! a menos que el usuario actuante posea una clave de acceso apropiada.

- **Pre Registration Email Validation [65]**
  Versión del plugin: 1.0.1. Fecha de ultima actualización: Abril 2015.

  Esta extensión permite que una dirección de correo sea verificada antes de realizar el registro en un sitio Joomla!. Permite además verificar errores de tipeo en la dirección ingresada.

- **Registration Validation [66]**
  Versión del plugin: 1.0.1. Fecha de ultima actualización: Abril 2015.

  La extensión agrega validación por Ajax al sistema de registro por defecto de Joomla!

- **ECC+ - EasyCalcCheck Plus [67]**
  Versión del plugin: 3.1.6. Fecha de ultima actualización: Agosto 2019.

  Esta extensión protege los formularios principales de Joomla! y extensiones de terceros mediante la adición de servicios contra spam.

- **jSecure [68]**
  Versión del plugin: 3.5. Fecha de ultima actualización: Agosto 2018.

  jSecure es un componente que permite seguridad en distintos niveles de un sitio Joomla!.

- **Anti-bot [69]**
  Versión del plugin: 1. Fecha de ultima actualización: Marzo 2016.

  Esta extensión está escrita en PHP, y permite diferenciar entre bots malintencionados y usuarios normales de un sitio, enviando documentos en blanco cada vez que detecte que un bot está ingresando al sitio.

- **Aimy Captcha-Less Form Guard [70]**
  Versión del plugin: 9.2. Fecha de ultima actualización: Marzo 2019.

  Extensión que provee seguridad en formularios, evitando el uso de campos captcha.

  Existe una versión "PRO" de esta extensión [71].

- **JoomReporter [72]**
  Versión del plugin: 1.1. Fecha de ultima actualización: Enero 2017.

  Extensión que agrega botones de tipo "Reportar", "Recomendar", "Reportar página" o "Reportar contenido".

- **n3t Seznam Captcha [73]**
  Versión del plugin: 4.0.0. Fecha de ultima actualización: Abril 2019.

  Campo captcha con opción de audio en idioma checo.

- **Spam Protect Factory [74]**
  Versión del plugin: 1.2.2. Fecha de ultima actualización: Julio 2019.

  Antes de que un formulario de ingreso (inicio de sesión o registro de un nuevo usuario) sea enviado, esta extensión lo investiga y actúa en base a los datos que contiene y la configuración que se le ha dado.

- **OSpam-a-not [75]**
  Versión del plugin: 1.1.13. Fecha de ultima actualización: Julio 2019.

  OSpam-a-not permite proteger los formularios de un sitio Joomla! contra la intrusión se bots.

- **Antivirus Website Protection [76]**
  Versión del plugin: 5.3. Fecha de ultima actualización: Abril 2019.

  Esta extensión permite prevenir, proteger y remover virus maliciosos y código sospechoso, analizando todos los documentos que un sitio Joomla! contiene.

- **Lab5 Captcha [77]**
  Versión del plugin: 4.0. Fecha de ultima actualización: Febrero 2019.

  Extensión que permite la adición de captchas a un formulario.

- **CedSecurityImages [78]**
  Versión del plugin: 6.0.4. Fecha de ultima actualización: Marzo 2015.

  Permite agregar campos captcha en formularios.

- **text2image [79]**
  Versión del plugin: 1.0.1. Fecha de ultima actualización: Febrero 2015.

  Esta extensión permite convertir cualquier texto en una imagen, de manera de proteger información sensible ante bots.

- **Stop Spammer For Community Builder [80]**
  Versión del plugin: 1.0. Fecha de ultima actualización: Julio 2016.

  En casos en que se use Community Builder [81], esta extensión permite que, al momento de registro de un usuario, los datos sean validados contra distintas bases de datos de spam, previniendo el registro en aquellos casos que sea necesario.

- **RCP User [82]**
  Versión del plugin: 1.0.2. Fecha de ultima actualización: Agosto 2019.

  Extensión que permite administrar usuarios.

- **JaM Contact [83]**
  Versión del plugin: 1.1. Fecha de ultima actualización: Marzo 2019.

  Extensión con facilidades Ajax en formularios de registro de usuarios e inicio de sesión.

- **JV-MathCaptcha [84]**
  Versión del plugin: 1.0. Fecha de ultima actualización: Julio 2019.

  Extensión que agrega seguridad a formularios mediante la resolución de un problema matemático generado de manera aleatoria. Solo contiene operaciones de suma y producto.

- **Vik Secure [85]**
  Versión del plugin: 1.1. Fecha de ultima actualización: Septiembre 2018.

  Extensión que permite proteger y sanitizar información en un sitio Joomla!.

- **BadBot Protection [86]**
  Versión del plugin: 1.1. Fecha de ultima actualización: Abril 2019.

  La extensión impide que atacantes logren inhabilitar el acceso a un sitio, así como cualquier intento de hackeo al mismo. También permite incrementar la "performance" de un sitio Joomla!.

- **SiteLock Security [87]**
  Versión del plugin: 1.0.8. Fecha de ultima actualización: Mayo 2019.

  Permite configurar distintos niveles de seguridad en un sitio de manera sencilla para sus administradores.

- **Google reCaptcha v3 for Chronoforms v6 [88]**
  Versión del plugin: 1.2. Fecha de ultima actualización: Abril 2019.

  Permite agregar captchas con la tecnología reCaptcha V3 de Google en formularios Chronoforms [89].

La empresa *itoctopus*[90] llevó a cabo un experimento a base de honeypots, denominado *The Joomla Honeypot Project Experiment* [91]. Los objetivos de esta prueba fueron:

- Llegar a un sistema de protección de tipo honeypot.
- Encontrar nuevas maneras de proteger un sitio Joomla!
- Obtener una solución que permita registrar información sobre los accesos y que pueda ser analizada.

La idea principal consiste en capturar todas las llamadas que se hacen a un sitio, almacenar esta información para luego investigarlas. El código creado fue localizado en el archivo *defines.php* de Joomla!, ya que éste es el primer archivo que se carga de la plataforma.

Luego de crear el código, este fue puesto en un sitio Joomla! activo. Al ver los resultados, fue posible observar:

- "user agents" maliciosos
- parámetros extraños en solicitudes GET
- incontables intentos de cargar archivos maliciosos al servidor

Este proyecto permitió, de una manera sencilla y fácil de implementar, la investigación de las llamadas HTTP que se realizan en un sitio web.

5. Honeytokens y breadcrums

5.1. Honeytokens

Los honeytokens pueden ser entendidos como recursos *dummy* de IT ubicados en sistemas o redes con el propósito de atraer la atención de cibercriminales y poder obtener información sobre las vulnerabilidades en sus sistemas. Estos recursos *dummy* resultan atractivos para los atacantes, de manera que son atraídos hacia ellos. Pero el verdadero valor de los honeytokens es hacia quienes lo crean y administran, ya que les permite monitorear sus sistemas y redes, así como obtener información sobre los atacantes que logran burlar las medidas de seguridad dispuestas.

Los honeytoken son similares a los honeypots en el sentido en que nadie debería interactuar con ellos, pero cuando alguien lo hace, se sabe que esta interacción es maliciosa o no permitida.

5.1.1. Tipos

Existen diferentes tipos de honeytokens[92]. A continuación se ilustra cada uno de ellos junto a una breve descripción:

- **Direcciones de email inválidas**

  Una empresa crea cuentas de correo que no serán utilizadas por nadie, aunque sus datos pueden representar tanto a personas internas o externas a la misma. Estas cuentas inactivas se alojan en los servidores de correo de la empresa, o en algún sistema que posea un servidor web de acceso público.

  Si alguna de las cuentas activas de correo de la empresa contiene algún email en su casilla de entrada proveniente de alguna de las cuentas inactivas, se entiende que algún usuario mal intencionado se ha hecho con las credenciales de alguno de estos emails inválidas, lo que indica que la seguridad de los servidores de correo puede haber sido comprometida.

- **Información inválida en una base de datos**

  Uno de los objetivos de los atacantes son las bases de datos de los sistemas empresariales, donde algunas empresas insertan datos ficticios. Estos datos resultan atractivos para los atacantes, quienes tratan de hacerse con ellos. Cuando un atacante accede a estos datos, los administradores pueden conocer como fueron ellos capaces de explotar vulnerabilidades o "loopholes"[15] en sus sistemas y redes.

[15] Un *loophole* es un error o una vulnerabilidad en el código de un programa que le permite ser manipulado o explotado.

- **Archivos ejecutables inválidos**

  Este tipo de honeytokens son creados con activadores "phone home", los cuales están encargados de recolectar información y enviarla al propietario del ejecutable cuando un atacante accede a este archivo. De esta manera, estos tokens permiten conocer, dentro de lo posible, el origen del ataque. El activador se llama "phone home" dado que la información que recolecta la envía hacia los dueños del archivo ("home").

  El problema con estos tokens es que pueden ocasionar daños al sistema del atacante, o pueden infringir violaciones de privacidad o leyes de ciberseguridad. La eficiencia de estos tokens reside en que el atacante se encuentre desprotegido.

- **Enlaces embebidos con activadores "phone home"**

  Estos tokens son documentos reales dentro de un sistema de archivos de una organización pero con enlaces ocultos o embebidos con activadores "phone home".

- **Web beacons**

  Un *web beacon* es un enlace embebido dentro de un objeto muy pequeño (por ejemplo, una imagen transparente) en el contenido de una aplicación web. Al ser un objeto tan pequeño pasa desapercibido a un usuario regular, pero no por un atacante. Cuando el enlace es encontrado, realiza un "phone home" con información del atacante.

- **Cookies de navegación**

  Se diseñaron para agregar el manejo de estados a la comunicación entre navegador y servidor web, dado que el protocolo HTTP carece de la noción de estados y es el protocolo utilizado. Los cookies son tratadas ampliamente en Sesiones de Usuarios.

- **Canary traps**

  Estos tokens funcionan de manera que, cuando alguien dentro de una organización filtra información, envían un "aviso" mencionando que la información ha sido comprometida. Es decir, estos tokens permiten realizar una asociación entre la persona que filtra los datos y la información filtrada.

  Un ejemplo claro de este tipo de tokens es el empleado por Screen Actors Guilad de Hollywood (Sindicato de Actores de Cine, SAG por las siglas en inglés), quienes tienen un éxito notable exponiendo a personas de dentro de su organización que filtran copias de películas o las nominaciones a los Oscar de ellas. Cada copia de una película que es enviada a cada miembro contiene una marca única, asociada con el receptor. Si uno de los miembros que recibieron la película realiza una copia, estará copiando también la marca única que el objeto original contiene.

- **Claves Amazon Web Services (AWS[93])**

  AWS emplea claves firmadas digitalmente para permitir o bloquear el acceso a distintas partes de su infraestructura. Estas claves pueden ser empleadas como honeytokens por los administradores, colocándolas por distintas partes de la infraestructura. De esta manera, cuando un atacante obtiene alguna de estas claves tratará de usarla para saber hasta donde puede llegar en la infraestructura, y como las claves poseen un sistema de registro, los administradores pueden saber hasta donde han llegado los atacantes en sus sistemas.

  En base a las claves de AWS, en marzo de 2018 se ideó el *Proyecto Spacecrab*, una plataforma que permite a los usuarios administrar claves AWS, así como configurar distintos niveles de alertas [94].

5.2. Implementaciones actuales

A continuación se presentan cinco implementaciones de honeytokens, donde para cada una se detalla su nombre, el nombre del creador, lenguaje en que se desarrolló, fecha de última actualización y una descripción de la herramienta. Cabe destacar que la fecha de última actualización se toma la fecha en la cual se ha realizado el último "commit" en la plataforma GitHub.

1. **Honeybits [95]**
   Autor: Adel Karimi. Lenguaje: Python. Última actualización: marzo 2019.

   Honeybits se encarga de automatizar la creación y despliegue de breadcrumbs y honeytokens a través de servidores de producción y equipos de trabajo, de manera de encaminar a los atacantes hacia los honeypots instalados. Estos señuelos plantados contienen:

   - Historial (no real) de ejecuciones de comandos de una consola bash.
   - Credenciales AWS no existentes.
   - Entradas inválidas en tablas como la de redirección, ARP.
   - Historial de navegación, favoritos y contraseñas almacenadas, todos ellos con datos inválidos.

   Existe una versión desarrollada para Windows, denominada **Honeybits-win**.

2. **honeyλ [96]**
   Autor: Adel Karimi. Lenguaje: Python. Última actualización: octubre 2018.

   honeyλ permite crear y monitorear URLs sobre AWS Lambda[97] y Amazon API Gateway[98]. Las referencias a los endpoints pueden ser ubicadas en alguna fuente que se sospeche puede ser atacada (por ejemplo: documentos, historial de navegación). Esta herramienta está basada en el framework Serverless[99] y permite que honeyλ sea instalada como solución en la nube.

3. **honey**ku [100]
   Autor: Adel Karimi. Lenguaje: Python. Última actualización: abril 2019.

   honeyku es un proyecto similar a honeyλ, desarrollado en Python y basado en Heroku[101].

4. **DCEPT [102]**
   Autores: Joe Stewart y James Bettke. Lenguaje: Python. Última actualización: septiembre 2016.

   Domain **C**ontroller **E**nticing **P**assword **T**ripwire es un tipo de honeytoken diseñado para Active Directory de Microsoft, y tiene como finalidad crear credenciales de acceso que serán accesibles únicamente si son obtenidas desde la memoria del sistemas. Cualquier intento de acceso con ellas informará a los administradores que su seguridad ha sido burlada.

5. **Canarytokens [103]**
   Autor: Thinkst Applied Research. Lenguaje: Python. Última actualización: Junio 2019.

   El proyecto Canarytokens es un servidor que permite crear honeytokens y poder monitorear la red.

5.3. Canarytokens

Los *canarytokens* son un tipo especial de honeytokens. La empresa Thinks[104] se ha especializado en este tipo de tokens, creando y ofreciendo herramientas e infraestructura suficiente que permiten trabajar con estos tokens. Uno de sus productos principales es **Canary**[105], hardware que es conectado en la red interna de una empresa y permite monitorear la misma, habilitando una consola web para ver su actividad.

Por otro lado, Thinkst contiene la plataforma **Canarytokens by Thinkst [106]** la que permite crear, de manera gratuita, distintos tipos de canarytokens. La infraestructura necesaria para su funcionamiento es provista por Thinkst, aunque es posible configurar todo este sistema en servidores locales a una empresa, ya que el código fuente del sistema está disponible en GitHub. Este sistema permite crear los siguientes canarytokens ([107] contiene documentación completa sobre ellos):

- **Token URL:** se alerta cuando una URL es visitada.
- **Token DNS:** se alerta cuando un "hostname" es requerido.
- **Dirección de correo única:** se alerta cuando un email es enviado a una dirección de correo única.
- **Imagen web personalizada:** se alerta cuando una imagen que se sube es visualizada.
- **Documento de Microsoft Word:** se alerta cuando se abre un documento en Microsoft Word.

- **Documento de Acrobar Reader PDF:** se alerta cuando un documento PDF es abierto en Acrobat Reader.
- **Carpeta de Windows:** se alerta cuando un directorio de Windows es visto desde el Explorador de Windows.
- **Archivo ejecutable:** se alerta cuando un archivo ejecutable (EXE o DLL) es ejecutado.
- **Sitio web clonado:** se alerta cuando un sitio web propio es clonado.
- **SQL Server:** se alerta cuando una base SQL Server es accedida.
- **Código QR:** permite generar un código QR como token físico.
- **SVN:** alerta cuando se realiza un "checkout" de un repositorio SVN.
- **Claves AWS:** se alerta cuando una clave AWS es usada.
- **Redirección rápida:** se alerta cuando una URL es visitada, y el usuario es redirigido.

5.4. Breadcrumbs

El término *breadcrumbs* referencia al conjunto de *pistas* que una plataforma esparce por un sistema organizacional para llamar la atención de potenciales atacantes [108]. Estas pistas crean un camino falso que lleva a los atacantes hacia sistemas señuelos, logrando proteger los activos de una organización. Para que los breadcrumbs sean efectivos, deben verse como información real para el atacante, de manera que decida seguir el camino trazado por ellos.

Los breadcrumbs son una herramienta que complementa perfectamente a los honeytokens, ya que pueden verse como "evidencia" sobre un honeytoken, logrando que un atacante crea que éste realmente es información valiosa y real de la organización, logrando al final que dicho atacante caiga en los sistemas de defensa de la organización para este caso, en un honeypot.

Existen cuatro tipos de breadcrumbs que pueden ser combinados en su uso:

Breadcrumbs de credenciales y Active Directory [108]

Uno de los objetivos principales de un atacante es lograr obtener acceso a cuentas de usuario de una organización. Es posible emplear honeytokens para fingir la existencia de cuentas que sean objetivo de estos ataques. Pero para que tengan real atractivo para un atacante, es recomendable crear breadcrumbs con estos honeytokens. Estos tokens funcionan en distintos sistemas señuelo ubicados dentro de una organización, y acceden cada cierto tiempo a ellos, de manera de darle a un atacante la sensación de que son cuentas reales y activas.

Cuando un atacante accede a un sistema señuelo a través de los breadcrumbs dejados, un sistema de validación es el encargado de alertar a los administradores de esta intrusión.

Breadcrumbs de archivos e información [109]

Los breadcrumbs de archivos son una de las técnicas más simple y versátil disponible, los cuales incluyen documentos, correos electrónicos, registros en una base de datos o enlaces a una lista de "archivos recientes" que apunte a una carpeta compartida con un sistema falso.

Es ideal que estos señuelos estén construidos de forma que parezcan reales respecto a la organización. Es decir, se recomienda seguir los lineamientos de la organización respecto a la creación y mantenimiento de documentos (respetar la convención para los nombres o agregar el logo de la organización).

Algunos señuelos que pueden resultar atractivos son:

- Un documento de texto (ej: README) que contenga instrucciones sobre la instalación o uso de un sistema o aplicación y que contenga credenciales de usuario.
- Un documento técnico común a cualquier organización, como puede ser las instrucciones para conectarse a una VPN corporativa.

breadcrumbs de red [110]

Para generar breadcrumbs de red, los sistemas señuelos deben comunicarse con componentes de la organización (por ejemplo, un servidor DNS) de manera de generar información atractiva para los atacantes. Los señuelos emplean distintos protocolos para comunicarse con los demás sistemas, lo que lleva a los atacantes a realizar ataques de tipo MitM [16]. Por ejemplo, se agregan entradas en las tablas ARP con conexiones abiertas hacia los señuelos. Cuando los atacantes investigan esta información, son dirigidos hacia estos sistemas.

Breadcrumbs de aplicación [110]

Los breadcrumbs de aplicaciones deben ser idealmente muy variados. Por ejemplo, breadcrumbs de aplicaciones de sesión dejan credenciales SSH, FTD, entre otras, para que encuentren los posibles atacantes. Breadcrumbs de navegadores web dirigen a los atacantes hacia sistemas señuelo a través de historial, cookies, y marcadores y contraseñas almacenadas. Esta información es visible por los atacantes.

[16] man-in-the-middle es un ataque en el que se adquiere capacidad de leer, insertar y modificar información a voluntad [111]

6. Sesiones de Usuarios

Como se comentó en Contexto Teórico, una de las funcionalidades y características esperadas de los Honeypots es la de recolectar información sobre los ataques generados. El problema es que alguna de estas implementaciones no logran reconocer al atacante o generar un perfil del mismo a partir de esta información.

Se considera importante comentar sobre algunas técnicas utilizadas para el reconocimiento y seguimiento de usuarios relacionadas a las Aplicaciones Web para poder generar perfiles de atacantes.

6.1. Cookies

Las **cookies** han sido diseñadas para agregar el manejo de estados a la comunicación establecida entre el navegador y un servidor web, *debido a que el protocolo HTTP utilizado para dicha comunicación no presenta este manejo*. Los estándares definidos para la implementación de las Cookies están presentes en RFC 2109 [112] y RFC 2965 [113].

Al agregar el manejo de estados, el protocolo HTTP obtiene mayor flexibilidad, debido a que se permite *manejo de transacciones* que mantienen cierta información durante la comunicación entre el cliente y el servidor.

6.1.1. Funcionamiento de las cookies

Para que un cliente HTTP pueda intercambiar información mediante una cookie con el servidor, se deben realizar los siguientes pasos:

1. El usuario desde su cliente web se conecta al servidor y en uno de los cabezales del paquete de datos se envía un valor alfanumérico que oficia de identificador para la cookie de dicho usuario. En caso que no exista una cookie almacenada, se envía el cabezal correspondiente vació.

2. Si el valor de cookie enviado por el cliente este almacenado en el servidor web, se consultará por la información asociada a dicha cookie y retornará al cliente la vista web en función de los datos almacenados. Cabe destacar que el valor del identificador de la cookie será enviado en cada una de las solicitudes que emita el cliente al servidor.

3. Por último, el cliente recibe los ficheros HTML, CSS y JavaScript almacenado en el servidor pertenecientes al sitio web al que accede, y obtiene de servidores de terceros los recursos faltantes (imágenes, animaciones o vídeos, otros ficheros css o JavaScript).

El último punto define dos tipos de transacciones: *first-party* son las comunicaciones que un navegador establece con el servidor al cual un usuario solicitó conectarse, y *third-party*, que son

las comunicaciones que los navegadores establecen con los servidores de terceros para traer otros recursos.

Mediante estos dos tipos de comunicaciones se pueden definir dos tipos de cookies:

- **First-Party Cookies** Están asociadas al sitio que el cliente desea visitar. Existen los cabezales Cookie, Cookie2 y Set-Cookie2, que contienen el estado de la información entre cliente, servidor y user-agent.

  Estos parámetros se basan en una colección de tipo "Nombre=valor", separados por ";", y los atributos comment, domain, max-age, httponly y secure.

- **Third-Party Cookies** Estas cookies son las utilizadas por empresas de terceros, comunmente empresas que brinden avisos. Las *third-party cookies* pueden ser recibidas por el navegador mientras el usuario esta visitando una página web con contenido *third-party* (imágenes, vídeos o publicidad).

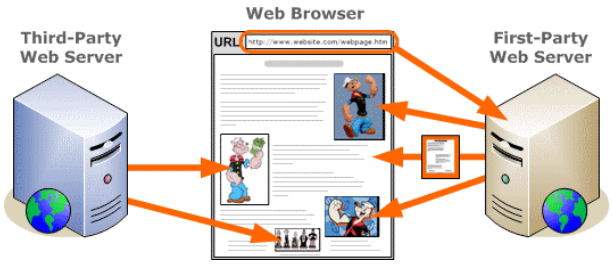

Figura 5: Ejemplo de diagrama de comunicación *First-Party* y *Third-Party* Cookies. Imagen obtenida de `https://www.grc.com/cookies/operation.htm`

6.1.2. Manejo de información mediante cookies

Como ya se ha mencionado, las cookies son utilizadas para mantener una sesión de un cliente HTTP. Es deseable que el servidor almacene algunos datos que son de utilidad para simplificar su uso.

Según ENISA [114], dentro de los datos almacenados se encuentran:

- Credenciales (nombre de usuario y contraseñas).
- Preferencias de usuario e interface web.
- Información de sesión y datos del sitio (información en cache).
- Información de *tracking* de usuarios.

El manejo de los datos mencionados anteriormente es el que permite que se puedan implementar características funcionales dentro de los sistemas que utilizan el protocolo HTTP para su comunicación, lo cual le da cierta robustez al mismo. Algunos ejemplos de estas funcionalidades, clasificados según el origen de la comunicación son:

- **Perspectiva funcional**
  - Identificación y autenticación de usuarios.
  - Estadísticas de números de visitas.
  - Almacenamiento de preferencias y configuraciones.
- **Perspectiva de marketing y publicidad online**
  - Calificar y evaluar la eficiencia en los avisos.
  - Crear perfiles de usuarios y utilizarlos para definir objetivos de los avisos.
  - Mejorar la gestión de avisos.

Un aspecto importante a descartar de las cookies es que no solamente es necesario que se almacene información del lado del servidor, sino que del lado del cliente (navegador web) se deberán persistir ciertos datos que serán enviados por el navegador web al momento en que se establece una comunicación entre ambas partes.

Del lado del cliente HTTP se definen los siguientes tipos de Cookies:

- **Non-Persistent Cookies**

  Las cookies pertenecientes a este grupo permanecen activas mientras se esté utilizando el navegador. Cuando el mismo es cerrado la cookie es eliminada o expira luego de un *time-out*.

- **Persistent Cookies**

  Las cookies de esta categoría son almacenadas por el navegador una vez que la sesión es finalizada. Gracias a este mecanismo, cuando el usuario vuelve a conectarse nuevamente con el sitio relacionado con la cookie, los datos son enviados automáticamente. De todas formas, estas cookies contienen una fecha de expiración en caso de no ser utilizadas por un determinado tiempo.

A modo de resumen, se tiene que las cookies no persistentes son las utilizadas durante la navegación entre páginas del mismo sitio, mientras que las persistentes son las encargadas de almacenar información del usuario (preferencias, configuración lenguaje, entre otros datos) relacionada a un sitio.

6.1.3. Tracking y privacidad

Uno de los problemas presentes en el contexto del uso de las cookies es *la facilidad que brindan las mismas al momento de identificar usuarios*. Esto no se debe solamente a que mediante una comunicación directa con un servidor una cookie puede generar un identificador único para un cliente, sino que las comunicaciones con las anteriormente citadas *third-party cookies*, permiten que dicho identificador pueda ser enviado a diferentes servidores.

A muchas empresas de publicidad les es de utilidad poder tener un *"tracking"* de los usuarios que consumen sus avisos, lo cual es muy utilizado para el *análisis de comportamiento (User Behavior)*. Por otro lado, muchas veces estos mecanismos tienen un fin que puede llegar a violar la privacidad de los usuarios, la vigilancia.

En el trabajo titulado *"Cookies That Give You Away: The Surveillance Implications of Web Tracking"* [115] se presenta la implementación de un "ataque" en el cual se realiza una escucha pasiva (*"eavesdropper"*), en la cual se recolecta información dentro de las *third-parties Cookies* y se generan *"clusters"* en donde se clasifica a los usuarios mediante el uso de determinados datos (como por ejemplo geolocalización, versión del navegador o aspectos de su configuración).

Se puede deducir que el uso indebido de las cookies puede producir que se este brindando información sensible de los usuarios de forma indiscriminada, por lo que es importante considerar los métodos que se presentan para poder reducir el envió pasivo de información.

6.2. SuperCookies

Con el correr del tiempo, se vio necesaria la implementación de algunas mejoras en aspectos funcionales de las cookies que permitan potenciar su uso. Sin embargo, algunas de estas "mejoras" vienen con características funcionales y de diseño que han permitido facilitar el *tracking* de los usuarios. A estas cookies se les llaman **SuperCookies**.

A continuación se presentan las **SuperCookies**.

6.2.1. Flash Cookies

Adobe Flash Player ([116]) es una aplicación que tiene la capacidad de reproducir contenido multimedia en un navegador. Pero, como se menciona en [117], para poder mejorar el intercambio de este tipo de información con los clientes web utiliza un tipo de dato denominado *"Local Shared Object"* (LSO) ([118]), o también conocidos comúnmente como **Flash Cookies**.

Las **Flash Cookies** se caracterizan por ser de un tamaño mayor a las cookies convencionales definidas en los RFC, dado que pueden alcanzar un tamaño de 100KB en su uso estándar, mientras que las tradicionales llegan a los 4KB. De todas formas, la característica más relevante de este tipo de estructura de datos es que puede almacenarse en un directorio dentro del *filesystem*

el cual no llega a ser accedido por los navegadores web, generando una mayor dificultad sobre el control de esta información, y por lo tanto, una mayor dificultad al borrarlos desde el mismo.

Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas y Chris Jay Hoofnagle [117], realizaron un análisis sobre el uso de este tipo de cookies y su interacción con los sitios que presentan contenido *third-party*. En este estudio se destaca cómo los sitios de publicidad pueden relacionar a los usuarios mediante el uso de las Flashs Cookies con el contenido que estos brindan y gracias a que son difíciles de eliminar es posible que las mismas sean regeneradas, comportamiento que se denomina *Re-Spawning*.

Sin embargo, el paper que se referencia es del año 2009, lo que se destaca debido a que hoy en día existen diferentes mecanismos para poder controlar el uso de estas cookies dado que en Páginas de Ayuda de Flash Player [116] [118] se explica como manejar la configuración de las mismas, e incluso eliminarlas. Pero sumado a esto, los navegadores web (como por ejemplo Firefox y Google Chrome) tiene configurado por defecto bloquear el contenido Flash, dejando a consideración del usuario la habilitación del mismo.

6.2.2. Zombies Cookies

Se le da la denominación de Zombie Cookies ([119]) a las cookies que tienen la capacidad de permanecer almacenadas en el equipo del cliente y no interactuar con ningún otro sitio, es decir que *"dan la sensación de estar muertas"*, pero una vez que se conectan con el sitio proveniente, puede ejecutarse y así recuperar su funcionamiento, lo que se puede ver como una analogía de *"despertar de la muerte"*, de ahí es que surge su nombre.

Como es aclarado en 6.2.1, las Flash Cookies permanecen almacenadas en el *filesystem* de un cliente y son utilizadas a demanda por los sitios relacionados a ellas, por lo que se podrían clasificar como un tipo de Zombie Cookie.

Sin embargo, las Zombie Cookies no solamente se asocian a las Flash Cookies, sino que las mismas pueden ser implementadas para cualquier sitio. Para su implementación, existe una API desarrollada en JavaScript llamada **evercookie** ([120]), que permite facilitar el proceso de desarrollo de las mismas.

6.2.3. PermaCookies

Las empresas de telefonía móvil estadounidenses **AT&T** y **Verizon** implementaron un mecanismo para tener un identificador único para cada usuario perteneciente a la compañía, que se denominó **PermaCookie** ([119]).

Básicamente, las **PermaCookies** [119], utilizan un *header HTTP* denominado **Unique Identifier Header (UIDH)**, el cual contiene el identificador único para dicho usuario.

En el caso particular de la empresa Verizon ([121]), el header HTTP no solamente es inyectado para las comunicaciones en los sitios de la empresa, sino que es enviado hacia servidores de terceros gracias al contenido *third-party*, lo que permite a dichos involucrados identificar a cada usuario en la comunicación.

A pesar de lo novedoso y eficaz del mecanismo, existen algunos métodos para poder evitar el uso de estas Cookies (como se menciona en el articulo de Electronic Frontier Foundation [121]), los cuales consisten básicamente en el uso de VPNs o proxies intermedios en la comunicación.

6.3. Manejo de información de Sesión en HTML 5

HTML es un lenguaje de etiquetas (similar al lenguaje XML) que es utilizado por los navegadores para poder presentar las páginas web. HTML hoy en día está disponible en su versión 5, el cual contiene atributos y un conjunto amplió de tecnologías que "permiten a los sitios Web y a las aplicaciones ser más diversas y de gran alcance" ([122]).

Esta nueva versión también incorpora el uso de almacenamiento local de información, en variables JavaScript del tipo *Storage*. Particularmente, para el uso de las sesiones se presentan dos tipos de variables accedidas desde la variable global **window**:

- **SessionStorage** ([123]) Da la posibilidad de acceder a un objeto de tipo *Storage* asociado a la **sesión actual**. Esta propiedad permite que la información que almacene posea tiempo de expiración, siendo eliminada al finalizar la sesión de la página.

  Dicha sesión perdura mientras el navegador se encuentra abierto, y se mantiene por sobre las recargas y reaperturas de la página. Abrir una página en una nueva pestaña o ventana iniciará una nueva sesión, lo que difiere en la forma en que trabajan las cookies de sesión.

- **LocalStorage** ([124]) Es una propiedad de sólo lectura que permite acceder a un objeto local de tipo *Storage* en donde los datos son almacenados entre de las diferentes sesiones de navegación, es decir, entre ventanas y *tabs* con el mismo origen.

La finalidad de estas variables es el de solucionar los siguientes problemas que presenta el uso de las cookies ([125]):

- Envió de información extra en cada solicitud.
- Espacio limitado en el tamaño de las cookies.
- Límite en el uso de *Third-Party Cookies*.
- Certeza que del lado del cliente se utiliza dicha información.

Se realizaron pruebas para manipular la información de los datos en localStorage de la web `http://www.maestrosdelweb.com/tutorial-local-session-storage/`, en donde al cambiar los datos, los mismos era regenerados al momento de iniciar una nueva pestaña. Sin embargo, al utilizar el navegador Google Chrome y acceder a la URL `chrome://settings/siteData`, se puede encontrar toda la información almacenada por los diferentes sitios web, lo cual fue posible realizar para el sitio en el cual se estaban ejecutando las pruebas.

6.4. Cached Data Fingerprinting

La información mantenida en "caché" es muy utilizada por los recursos web para poder minimizar la congestión de solicitudes en la red, de forma que al reutilizar la información persistida del lado del cliente, el servidor no tiene que entregar nuevamente el recurso solicitado.

A continuación se describen algunos métodos en los cuales se puede apreciar como el uso de información en caché puede llevar a identificar a los usuarios.

6.4.1. Last modified

En el documento RFC 7232 [126], se presenta el cabezal `Last-Modified` como un *cabezal de respuesta el cual contiene un valor de "timestamp"*[17] *en donde el servidor menciona cuando fue la última vez que se modificó el recurso accedido.*

Básicamente, el servidor al recibir una petición para un recurso, envía adjunto el cabezal `Last-Modified` con el valor asociado a la última modificación de dicho recurso. Cuando posteriormente el cliente solicita el recurso nuevamente, el servidor comparará el valor del cabezal `If-Modified-Since` enviado en la solicitud del cliente contra la última modificación realizada sobre el recurso. Si el mismo no fue modificado, responderá con `304 Not Modified`, en caso de que el recurso cambie, se envía nuevamente.

Si bien a simple vista no parece haber problema con el uso de estos cabezales, Patrick Verleg en su tesis *"Cache Cookies: searching for hidden browser storage"* [127] menciona que en muchos clientes no se controla el valor de dicho cabezal, por lo que el mismo podría componerse de cualquier valor que sea utilizado para el reconocimiento de los usuarios.

Sumado a esto, Robert Heaton en su artículo *"Cookieless user tracking for douchebags"* [128] presenta un ataque en donde a través de un archivo JavaScript asociado al recurso utilizado, se genera un identificador alfanumérico que existirá hasta que el usuario limpie sus datos en caché.

Algunos mecanismos para poder evitar el uso de esta información en caché para reconocer usuarios sería limpiando periódicamente la memoria o utilizando proxies o plugins[18] del navegador (como sugiere Patrick Verleg).

[17] Un valor "timestamp" refiere a un dato en el cual se presenta una fecha y una hora.

[18] Los *plugins* son componentes de software que ejecutan sobre los navegadores web para brindar funcionalidades extras sobre el mismo.

6.4.2. ETags

ETag es un cabezal de respuesta HTTP, como se especifica en el RFC 7232 [126], el cual *es utilizado para poder definir un recurso de manera única en el sistema y de esta forma identificar cuando el recurso es modificado*. Esto quiere decir que el valor del cabezal es calculado del lado del servidor para representar de forma unívoca (o lo más unívocamente posible) un recurso accedido por el cliente, en donde el método para el cálculo de dicho valor puede ser desconocido.

El cabezal ETag se emplea para *validar datos en caché*, siendo un mecanismo más eficiente para el manejo de la misma que el análisis del cabezal `Last-Modified`, en el cual solamente se presenta información asociada a la fecha en la que se solicitó el recurso.

Esta idea se basa en que un cliente al consultar por un recurso dentro del servidor (como por ejemplo "index.html") recibe dentro del mensaje de respuesta el valor alfanumérico asociado a dicho recurso. Posteriormente cuando el cliente vuelva a solicitar el mismo recurso, se enviará el valor de ETag en el cabezal `If-None-Match`[19]. El servidor nuevamente calcula el valor del ETag asociado al recurso solicitado y si no encuentra cambios envía un mensaje `304 Not Modified`, en caso de que el recurso cambie, se envía nuevamente.

Patrick Verleg en su tesis [127], también menciona que este mecanismo presenta un problema de privacidad. Su fundamento se basa en que la mayoría de las veces *no se conoce como se calcula el valor asociado al recurso*, lo cual potencialmente podría ser calculado con la información que envía el cliente en su solicitud. Por lo tanto, del lado del servidor, se podrían crear métodos en los cuales utilicen dicha información y el identificador *puede ser único para cada cliente*.

Dentro de la especificación de los ETags se menciona que el uso del cabezal es opcional por parte del cliente, pero como menciona Patrick Verleg, es necesario utilizar plugins o proxies intermediarios en la conexión para desactivar su uso, aunque aclara que en casos donde la conexión es cifrada puede que se dificulte la posibilidad de remover el cabezal.

6.5. Browser Fingerprinting

En el presente capítulo se han analizado diferentes mecanismos para poder generar un reconocimiento de los usuarios, pero no se ha comentado sobre los problemas de privacidad presentes en los clientes HTTP más comunes de los usuarios de Internet de hoy día, **los navegadores web**.

Como menciona Jessamyn West en su artículo *"Library Privacy in an Age of Browser Fingerprinting"* [129], un usuario suele desconocer toda la información que es enviada a través de su navegador para conectarse a un sitio y tener una navegabilidad eficiente y sin problemas. Este

[19]El cliente también podría enviar el valor del ETag en el cabezal `If-Match`, el cual tiene un funcionamiento similar a `If-None-Match`. Sin embargo, no es muy utilizado debido a que no todos los dispositivos en el camino de la comunicación lo implementan (como se menciona en [126])

punto refiere a que en la mayoría de las comunicaciones se intercambia información sobre versión del sistema operativo, plugins instalados y hasta incluso el listado de las fuentes soportadas por el dispositivo del cliente.

Jessamyn también hace referencia a los problemas presentes al momento en que un usuario acepta políticas de una página web al comenzar a utilizarla, dado que en la mayoría de los casos no son leídas y en estas siempre están presentes todas las pautas establecidas por la organización sobre la información que intercambiará con el cliente, incluso si se utilizarán identificadores únicos durante la comunicación para el reconocimiento de usuarios.

Krishna.V.Nair y Elizabeth RoseLalson en el paper *"The Unique Id's You Can't Delete: Browser Fingerprints"* [130] mencionan algunos de los métodos más comúnes de reconocimiento de usuarios a través de la información en sus navegadores, los cuales son:

- **Header Based Fingerprinting**, el cual refiere a la información transimitida por el navegador en los cabezales HTTP emitidos por el cliente.
- **JavaScript Based Fingerprinting**, a partir de la existencia de características precisas sobre cada instancia de navegador, en la cual mediante un hash utilizando funciones JavaScript se podría crear un identificador único para el usuario conectado.
- **Plugin Based Fingerprinting**, haciendo uso de los plugins que suelen dejar ciertas marcas que son trasmitidas durante las comunicaciones.
- **Extension Based Fingerprinting**, haciendo uso de las extensiones[20], que al igual que los plugins, suelen mantener cierta información que es utilizada para poder reconocer usuarios.
- **Cross Browser Fingerprinting**, en donde fuentes instalados en el sistema, los cuales son independientes del navegador web, son utilizados para generar un identificador único.

Un método muy utilizado hasta el momento para realizar fingerprinting de navegadores es **"Canvas Fingerprinting"**. Canvas es una etiqueta HTML utilizada para dibujar gráficos a través de código JavaScript (como menciona el grupo de desarrollo de Firefox en [131]). Por ejemplo, se puede usar para hacer gráficas, composiciones fotográficas, crear animaciones, o incluso procesado o renderizado de vídeo en tiempo real.

En el artículo *"The Web Never Forgets: Persistent Tracking Mechanisms in the Wild"* [132] se presenta la descripción de una metodología utilizada para realizar reconocimiento de usuarios a través del uso de la etiqueta Canvas junto a utilidades de JavaScript. La herramienta desarrollada utiliza los métodos `fillText` y `ToDataURL` para dibujar texto y leer datos respectivamente.

[20] Las *extensiones* son componentes de software utilizadas para modificar o mejorar funcionalidades del navegador.

Luego, cuando el usuario visita la página, la herramienta de fingerprinting realiza un dibujo con fuente, texto y color de fondo elegidos por la misma. Posteriormente, al realizar una llamada a la operación `ToDataURL` de la API de Canvas, se retorna un valor en codificación Base64, el cual es una representación binaria de los pixeles. Por último, al realizar un hash de estos valores, se obtiene el identificador único para el usuario.

Sin embargo, Krishna.V.Nair y Elizabeth RoseLalson presentan en su artículo ([130]) un método más eficiente que *"Canvas Fingerprinting"* partiendo de la base que navegadores como Firefox permitén el bloqueo de la utilización de HTML Canvas. El método de fingerprinting desarrollado se basa en obtener diferentes datos de los mensajes HTTP de los clientes para crear un identificador único a través de una función de hash llamada "Murmur Hash" (justificando su uso debido a la eficiencia de la función, ya que no es utilizada con fines de seguridad). En las conclusiones del trabajo (probado en clientes con computadoras con sistema Windows) se menciona que los usuarios pudieron ser identificados únivocamente.

Pierre Laperdrix, Walter Rudametkin y Benoit Baudry en su trabajo *"Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints"* [133] y Peter Eckersley en su trabajo *"How Unique Is Your Web Browser?"* [134], realizaron un gran aporte en el área de Browser Fingerprinting, brindando como resultados las herramientras web *AmIUniq?* [135] y *PanoptiClick* [136] respectivamente, las cuales brindan información sobre el grado de posibilidad en reconocer a un usuario a través de la información que brinda mediante su navegador web.

El funcionamiento de ambas herramientas es similar, el cual se basa en *delimitar valores que pueden ser únicos y mediante los datos obtenidos por el usuario determinar si el mismo puede ser reconocido o no.* En los datos analizados de las solicitudes se destacan:

- **Headers HTTP**, de los cuales se incluye *User Agent*, *Language*, *Content-type*, entre otros.
- **Platform**, valor que se encuentra accediendo a la variable `navigator.platform`.
- **Fuentes del sistema**.
- **Plugins utilizados**, en particular se considera destacable el uso de *"Ad-Blockers"*.
- Uso de **WebGL** y **Canvas**.
- **Cookies**.
- **Contenido Flash**.

Posteriormente a la recolección de los datos, las herramientas se encargan de calcular la entropía de una variable aleatoria discreta, particularmente se utiliza la fórmula de *Shannon*. Los valores obtenidos son normalizados generando de esta manera un indicador que ilustra que probabilidad tiene el usuario de ser reconocido.

6.6. TCP Tracking

Las metodologías y herramientas descritas anteriormente están basadas en el tracking de usuarios a nivel del protocolo HTTP. Sin embargo, se descubrieron que existen herramientas que permiten reconocer usuarios a partir de sus conexiones TCP hacia un servidor.

A continuación se mostrarán ejemplos de algunas de estas herramientas:

6.6.1. p0f

p0f [137] es una herramienta creada por Michal Zalewski. Para su funcionalidad se usa una serie de sofisticados mecanismos de fingerprinting puramente pasivos [21] para identificar a los interlocutores detrás de cualquier comunicación TCP/IP.

Algunas de sus utilidades abarcan identificación altamente escalable y rápida de sistemas operativos y software en ambos puntos de una conexión TCP estándar, medición del tiempo de actividad del sistema y la conexión de red (incluida la topología detrás de NAT o los filtros de paquetes), detección automatizada de conexiones compartidas o NAT, equilibrio de carga y configuraciones de proxy de nivel de aplicación.

Incluyen el reconocimiento durante las pruebas de penetración, rutina de monitoreo de red, detección de interconexiones de red no autorizadas en empresas ambientes, proporcionar señales para herramientas de prevención de abuso y análisis forense.

6.6.2. tcptrack

tcptrack [138] es un sniffer que permite analizar el estado e información de las conexiones TCP establecidas en una interfaz.

Básicamente es una herramienta que permite reconocer las conexiones establecidas en un servidor, y en caso de detectar algún comportamiento extraño se la podría utilizar para reconocer cierta información de la conexión, como lo es puertos de origen y destino, estado de la conexión, tiempo de inactividad y uso del ancho de banda.

[21] Se basa en el escaneo de la red, se comporta como sniffer, en donde únicamente visualiza el tráfico de la red sin alterar el mismo.

Referencias

[1] James W. Conley Eric Cole, Ronald Krutz. *Network Security Bible*. Wiley Publishing, Inc, Indianapolis, Indiana, 2005.

[2] Joseph Migga Kizza. *Computer Network Security*. Springer Science+Business Media, Inc, Chattanooga, Tennessee, 2005.

[3] Lance Spitzner. *Honeypots: Tracking Hackers*. Addison Wesley, Boston, Massachusetts, 2002.

[4] OWASP Foundation. Owasp | what are web applications? `https://www.owasp.org/index.php/What_are_web_applications%3F`. [Accedido: 21-AUG-2019].

[5] OWASP Foundation. About us | the owasp foundation. `https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project`. [Accedido: 21-AUG-2019].

[6] OWASP Foundation. Owasp top ten web application security risks. `https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project`. [Accedido: 21-AGO-2019].

[7] OWASP Foundation. Owasp top ten web application security risks 2017 - pdf file. `https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf`. [Accedido: 21-AGO-2019].

[8] OWASP Foundation. Owasp los diez riesgos más críticos en las aplicaciones web 2019 - pdf file. `https://www.owasp.org/images/5/5e/OWASP-Top-10-2017-es.pdf`. [Accedido: 21-AGO-2019].

[9] Wikipedia. Tzona desmilitarizada (informática) - wikipedia, la enciclopedia libre. `https://es.wikipedia.org/wiki/Zona_desmilitarizada_(informatica)`. [Accedido: 02-FEB-2020].

[10] Thorsten Holz and Jeanna Matthews. A generic toolkit for converting web applications into high-interaction honeypots. 2008.

[11] GitHub. Github - argomirr/honeypot: A honeypot plugin for bukkit. `https://github.com/Argomirr/Honeypot`. [Accedido: 26-AUG-2019].

[12] GitHub. Github - eymengunay/eohoneypotbundle: Honeypot type for symfony forms. `https://github.com/eymengunay/EoHoneypotBundle`. [Accedido: 26-AUG-2019].

[13] GitHub. Github - mushorg/glastopf: Web application honeypot. `https://github.com/mushorg/glastopf`. [Accedido: 26-AUG-2019].

[14] GitHub. Github - mushorg/snare: Super next generation advanced reactive honeypot. `https://github.com/mushorg/snare`. [Accedido: 26-AUG-2019].

[15] Google Hack Honeypot. Ghh - the google hack honeypot. http://ghh.sourceforge.net/. [Accedido: 26-AUG-2019].

[16] GitHub. Github - msurguy/honeypot: Simple spam prevention package for laravel applications. https://github.com/msurguy/Honeypot. [Accedido: 26-AUG-2019].

[17] GitHub. Github - schmalle/nodepot: A nodejs web application honeypot. https://github.com/schmalle/Nodepot. [Accedido: 26-AUG-2019].

[18] GitHub. Github - schmalle/servletpot: Webapplication honeypot. https://github.com/schmalle/servletpot. [Accedido: 26-AUG-2019].

[19] Shadow Daemon Open. Shadow daemon open - source web application firewall. https://shadowd.zecure.org/overview/introduction/. [Accedido: 26-AUG-2019].

[20] GitHub. Github - cymmetria/strutshoneypot: Struts apache 2 based honeypot as well as a detection module for apache 2 servers. https://github.com/Cymmetria/StrutsHoneypot. [Accedido: 26-AUG-2019].

[21] GitHub. Github - illusivenetworks-labs/webtrap: This project is designed to create deceptive webpages to deceive and redirect attackers away from real websites. https://github.com/IllusiveNetworks-Labs/WebTrap. [Accedido: 26-AUG-2019].

[22] GitHub. Github - bjeborn/basic-auth-pot: bap - http basic authentication honeypot. https://github.com/bjeborn/basic-auth-pot. [Accedido: 26-AUG-2019].

[23] GitHub. Github - graneed/bwpot. https://github.com/graneed/bwpot. [Accedido: 26-AUG-2019].

[24] GitHub. Github - dmpayton/django-admin-honeypot: A fake django admin login screen page. https://github.com/dmpayton/django-admin-honeypot. [Accedido: 26-AUG-2019].

[25] GitHub. Github - d1str0/drupot: Drupal honeypot. https://github.com/d1str0/drupot. [Accedido: 26-AUG-2019].

[26] GitHub. Github - bocajspear1/honeyhttpd: Honeyhttpd is a python-based web server honeypot builder. https://github.com/bocajspear1/honeyhttpd. [Accedido: 26-AUG-2019].

[27] GitHub. Github - gfoss/phpmyadmin_honeypot: A simple and effective phpmyadmin honeypot. https://github.com/gfoss/phpmyadmin_honeypot. [Accedido: 26-AUG-2019].

[28] GitHub. Github - pwnlandia/shockpot: Webapp honeypot for detecting shell shock exploit attempts. https://github.com/threatstream/shockpot. [Accedido: 26-AUG-2019].

[29] GitHub. Github - freak3dot/smart-honeypot: Php script demonstrating a smart honey pot. https://github.com/freak3dot/smart-honeypot. [Accedido: 26-AUG-2019].

[30] GitHub. Github - chh/stack-honeypot: Inserts a trap for spam bots into responses. https://github.com/CHH/stack-honeypot. [Accedido: 26-AUG-2019].

[31] GitHub. Github - helospark/tomcat-manager-honeypot: Honeypot that mimics tomcat manager endpoints. logs requests and saves attacker's war file for later study. https://github.com/helospark/tomcat-manager-honeypot. [Accedido: 26-AUG-2019].

[32] GitHub. Github - martiningesen/honnypotter: Wordpress honeypot. https://github.com/MartinIngesen/HonnyPotter. [Accedido: 26-AUG-2019].

[33] GitHub. Github - dustyfresh/honeypress. https://github.com/dustyfresh/HoneyPress. [Accedido: 26-AUG-2019].

[34] GitHub. Github - freak3dot/wp-smart-honeypot: Wordpress plugin to reduce comment spam with a smarter honeypot. https://github.com/freak3dot/wp-smart-honeypot. [Accedido: 26-AUG-2019].

[35] GitHub. Github - gbrindisi/wordpot: A wordpress honeypot. https://github.com/gbrindisi/wordpot. [Accedido: 26-AUG-2019].

[36] What CMS. What is a cms? - what cms? https://whatcms.org/What-Is-A-CMS. [Accedido: 09-OCT-2019].

[37] Wikipedia. Bot - wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Bot. [Accedido: 09-OCT-2019].

[38] Wikipedia. Drupal - wikipedia, la enciclopedia libre. https://www.itoctopus.com/the-joomla-honeypot-project-experiment. [Accedido: 09-OCT-2019].

[39] Jegg Geerling. Introducing the honeypot form spam protection module for drupal | jeff geerling. https://www.jeffgeerling.com/blogs/jeff-geerling/introducing-honeypot-form-spam. [Accedido: 08-OCT-2019].

[40] acquia. Drupal 8 module of the week: Honeypot. https://dev.acquia.com/blog/drupal-8-module-of-the-week/drupal-8-module-of-the-week-honeypot/25/02/2016/9766. [Accedido: 08-OCT-2019].

[41] Wikipedia. Wordpress - wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/WordPress. [Accedido: 09-OCT-2019].

[42] WordPress. Plugins en la categoría honeypot | wordpress.org español. https://es.wordpress.org/plugins/tags/honeypot/. [Accedido: 08-OCT-2019].

[43] WordPress. Honeypot for contact form 7 – plugin wordpress | wordpress.org español. https://es.wordpress.org/plugins/contact-form-7-honeypot/. [Accedido: 08-OCT-2019].

[44] WordPress. Contact form 7 – wordpress plugin | wordpress.org. https://wordpress.org/plugins/contact-form-7/. [Accedido: 08-OCT-2019].

[45] WordPress. Blackhole for bad bots – plugin wordpress | wordpress.org español. https://es.wordpress.org/plugins/blackhole-bad-bots/. [Accedido: 08-OCT-2019].

[46] WordPress. Formidable honeypot – plugin wordpress | wordpress.org español. https://es.wordpress.org/plugins/formidable-honeypot/. [Accedido: 08-OCT-2019].

[47] Formidable FORMS. Formidable forms - the most advanced wordpress forms plugin. https://formidableforms.com. [Accedido: 08-OCT-2019].

[48] WordPress. Ap honeypot wordpress plugin – plugin wordpress | wordpress.org español. https://es.wordpress.org/plugins/ap-honeypot/. [Accedido: 08-OCT-2019].

[49] WordPress. http:bl wordpress plugin – wordpress plugin | wordpress.org. https://wordpress.org/plugins/httpbl/. [Accedido: 08-OCT-2019].

[50] Project Honey Pot. Http:bl overview | project honey pot. https://www.projecthoneypot.org/httpbl.php. [Accedido: 08-OCT-2019].

[51] Project Honey Pot. The web's largest community tracking online fraud & abuse | project honey pot. https://www.projecthoneypot.org/. [Accedido: 13-OCT-2019].

[52] WordPress. Honeypot toolkit – plugin wordpress | wordpress.org español. https://es.wordpress.org/plugins/honeypot-toolkit/. [Accedido: 08-OCT-2019].

[53] WordPress. Honeypot for wp comment – plugin wordpress | wordpress.org español. https://es.wordpress.org/plugins/honeypot-for-wp-comment/. [Accedido: 08-OCT-2019].

[54] WordPress. Wp spam question filter – plugin wordpress | wordpress.org español. https://es.wordpress.org/plugins/wp-spam-question-filter/. [Accedido: 08-OCT-2019].

[55] WordPress. Contact form 7 honeypot plus – plugin wordpress | wordpress.org español. https://es.wordpress.org/plugins/cf7-honeypot-plus/. [Accedido: 08-OCT-2019].

[56] WordPress. Spampot – plugin wordpress | wordpress.org español. https://es.wordpress.org/plugins/spampot/. [Accedido: 08-OCT-2019].

[57] WordPress. Honnypotter – plugin wordpress | wordpress.org español. https://es.wordpress.org/plugins/honnypotter/. [Accedido: 08-OCT-2019].

[58] Wikipedia. Joomla - wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Joomla. [Accedido: 09-OCT-2019].

[59] Joomla! Joomla! extensions directory. https://extensions.joomla.org/tags/spam-protection/?start=0. [Accedido: 09-OCT-2019].

[60] Joomla! Rsfirewall!, by rsjoomla! - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/rsfirewall/. [Accedido: 13-OCT-2019].

[61] Joomla! Securitycheck pro, by texpaok - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/securitycheck-pro/. [Accedido: 13-OCT-2019].

[62] Joomla! Antispam by cleantalk, by cleantalk - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/antispam-by-cleantalk/. [Accedido: 13-OCT-2019].

[63] Joomla! Spambotcheck, by aicha vack - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/spambotcheck/. [Accedido: 13-OCT-2019].

[64] Joomla! jsecure lite - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/jsecure-lite/. [Accedido: 13-OCT-2019].

[65] Joomla! Pre registration email validation - joomla! extension directory. https://extensions.joomla.org/extension/access-a-security/site-security/pre-registration-email-validation/. [Accedido: 13-OCT-2019].

[66] Joomla! Registration validation, by function90 - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-access/registration-validation/. [Accedido: 13-OCT-2019].

[67] Joomla! Ecc+ - easycalccheck plus, by viktor vogel - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/easycalccheck-plus/. [Accedido: 13-OCT-2019].

[68] Joomla! jsecure, by jsp team - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/jsecure/. [Accedido: 13-OCT-2019].

[69] Joomla! Anti-bot - joomla! extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-access/anti-bot/. [Accedido: 13-OCT-2019].

[70] Joomla! Aimy captcha-less form guard, by aimy extensions - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/aimy-captcha-less-form-guard/. [Accedido: 13-OCT-2019].

[71] Joomla! Aimy captcha-less form guard pro, by aimy extensions - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/aimy-captcha-less-form-guard-pro/. [Accedido: 13-OCT-2019].

[72] Joomla! Joomreporter - joomla! extension directory. https://extensions.joomla.org/extensions/extension/authoring-a-content/joomreporter/. [Accedido: 13-OCT-2019].

[73] Joomla! n3t seznam captcha, by pavel poles - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/n3t-seznam-captcha/. [Accedido: 13-OCT-2019].

[74] Joomla! Spam protect factory, by thephpfactory - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/spam-protect-factory/. [Accedido: 13-OCT-2019].

[75] Joomla! Ospam-a-not, by joomlashack - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/ospam-a-not/. [Accedido: 13-OCT-2019].

[76] Joomla! Antivirus website protection, by safetybis ltd. - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/antivirus-website-protection/. [Accedido: 13-OCT-2019].

[77] Joomla! Lab5 captcha, by lab5 - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/lab5-captcha/. [Accedido: 13-OCT-2019].

[78] Joomla! Cedsecurityimages - joomla! extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/security-images/. [Accedido: 13-OCT-2019].

[79] Joomla! text2image - joomla! extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/text2image/. [Accedido: 13-OCT-2019].

[80] Joomla! Stop spammer for community builder - joomla extension directory. https://extensions.joomla.org/extensions/extension/extension-specific/community-builder-extensions/stop-spammer-for-community-builder/. [Accedido: 13-OCT-2019].

[81] JoomlaPolis! Community builder - joomla social networking. https://www.joomlapolis.com/. [Accedido: 13-OCT-2019].

[82] Joomla! Rcp user - joomla social networking. https://extensions.joomla.org/extensions/extension/clients-a-communities/user-management/rcp-user/. [Accedido: 13-OCT-2019].

[83] Joomla! Jam contact, by grusha - joomla extension directory. https://extensions.joomla.org/extensions/extension/contacts-and-feedback/contact-forms/jam-contact/. [Accedido: 13-OCT-2019].

[84] Joomla! Jv-mathcaptcha, by jv-extensions - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/jv-mathcaptcha/. [Accedido: 13-OCT-2019].

[85] Joomla! Vik secure, by e4j extensions for joomla - joomla extension directory. https://extensions.joomla.org/extensions/extension/vik-secure/. [Accedido: 13-OCT-2019].

[86] Joomla! Badbot protection, by safetybis ltd. - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/site-security/badbot-protection/. [Accedido: 13-OCT-2019].

[87] Joomla! Sitelock security, by sitelock dev team - joomla extension directory. https://extensions.joomla.org/extensions/extension/sitelock-security/. [Accedido: 13-OCT-2019].

[88] Joomla! Google recaptcha v3 for chronoforms v6, by sky spider - joomla extension directory. https://extensions.joomla.org/extensions/extension/access-a-security/google-recaptcha-v3-for-chronoforms-v6/. [Accedido: 13-OCT-2019].

[89] ChronoEngine. Chronoforms. https://www.chronoengine.com/chronoforms. [Accedido: 13-OCT-2019].

[90] itoctopus. itoctopus. https://www.itoctopus.com. [Accedido: 13-OCT-2019].

[91] itoctopus. The joomla honeypot project experiment | itoctopus. https://www.itoctopus.com/the-joomla-honeypot-project-experiment. [Accedido: 09-OCT-2019].

[92] Finjan Blog. Honeytokens | how they are used to track cybercriminals. https://blog.finjan.com/honeytokens-used-to-track-cybercriminals/. [Accedido: 30-AUG-2019].

[93] AWS. Aws | cloud computing - servicios de informática en la nube. https://aws.amazon.com/es/. [Accedido: 13-OCT-2019].

[94] DARKReading. Hunting cybercriminals with aws honey tokens. https://www.darkreading.com/cloud/hunting-cybercriminals-with-aws-honey-tokens/d/d-id/1331342?elq_mid=83984&elq_cid=20964238&_mc=NL_DR_EDT_DR_daily_20180323&cid=NL_DR_EDT_DR_daily_20180323&elqTrackId=d061ff22ada64d4d8a4b2d6de1463809&elq=5dbac308e0dd4236ace. [Accedido: 09-OCT-2019].

[95] GitHub. Github - 0x4d31/honeybits: A poc tool designed to enhance the effectiveness of your traps by spreading breadcrumbs & honeytokens across your systems to lure the attacker toward your honeypots. https://github.com/0x4D31/honeybits. [Accedido: 20-SEP-2019].

[96] GitHub. Github - 0x4d31/honeylambda: honeyλ - a simple, serverless application designed to create and monitor fake http endpoints (i.e. url honeytokens) automatically, on top of aws lambda and amazon api gateway. https://github.com/0x4D31/honeylambda. [Accedido: 20-SEP-2019].

[97] AWS. Aws | lambda - gestión de recursos informáticos. https://aws.amazon.com/es/lambda/. [Accedido: 13-OCT-2019].

[98] AWS. Amazon api gateway | api management | amazon web services. https://aws.amazon.com/es/api-gateway. [Accedido: 13-OCT-2019].

[99] serverless. The serverless application framework | serverless.com. https://serverless.com. [Accedido: 13-OCT-2019].

[100] GitHub. Github - 0x4d31/honeyku: A heroku-based web honeypot that can be used to create and monitor fake http endpoints (i.e. honeytokens). https://github.com/0x4D31/honeyku. [Accedido: 20-SEP-2019].

[101] heroku. Cloud application platform | heroku. https://www.heroku.com. [Accedido: 13-OCT-2019].

[102] GitHub. Github - secureworks/dcept: A tool for deploying and detecting use of active directory honeytokens. https://github.com/secureworks/dcept. [Accedido: 20-SEP-2019].

[103] GitHub. Github - thinkst/canarytokens: Canarytokens helps track activity and actions on your network. https://github.com/thinkst/canarytokens. [Accedido: 20-SEP-2019].

[104] thinkst. Thinkst home. https://thinkst.com. [Accedido: 13-OCT-2019].

[105] Thinkst Canary. Thinkst canary. https://canary.tools. [Accedido: 20-SEP-2019].

[106] Canarytokens. Canarytokens. https://canarytokens.org/generate. [Accedido: 20-SEP-2019].

[107] Thinkst Canary. Canarytokens. https://docs.canarytokens.org. [Accedido: 20-SEP-2019].

[108] Fidelis Cybersecurity. 4 kinds of breadcrumbs in intelligent deception | fidelis cybersecurity. https://github.com/zdresearch/OWASP-Honeypot. [Accedido: 08-OCT-2019].

[109] Fidelis Cybersecurity. Using file & data breadcrumbs for intelligent deception | fidelis cybersecurity. https://fidelissecurity.com/threatgeek/deception/using-file-data-breadcrumbs-intelligent-deception/. [Accedido: 08-OCT-2019].

[110] Fidelis Cybersecurity. Using network & application breadcrumbs for intelligent deception | fidelis cybersecurity. https://fidelissecurity.com/threatgeek/deception/using-network-application-breadcrumbs-intelligent-deception/. [Accedido: 08-OCT-2019].

[111] Wikipedia. Ataque de intermediario - wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Ataque_de_intermediario. [Accedido: 13-OCT-2019].

[112] IETF. Rfc 2109 - http state management mechanism. https://www.ietf.org/rfc/rfc2109.txt. [Accedido: 13-SEP-2019].

[113] IETF. Rfc 2165 - http state management mechanism. https://www.ietf.org/rfc/rfc2965.txt. [Accedido: 13-SEP-2019].

[114] Demosthenes Ikonomou Rodica Tirtea, Claude Castelluccia. Bittersweet cookies. some security and privacy considerations. 2011.

[115] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. Cookies that give you away: The surveillance implications of web tracking. 2015.

[116] Adobe. Adobe - flash player : Settings manager - global storage settings panel. http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager03.html. [Accedido: 09-SEP-2019].

[117] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagl. Flash cookies and privacy. 2009.

[118] Adobe. Manage local shared objects in flash player. https://helpx.adobe.com/flash-player/kb/disable-local-shared-objects-flash.html. [Accedido: 09-SEP-2019].

[119] Reputation Defender. How to delete flash cookies, permacookies, and zombie cookies | reputationdefender. https://www.reputationdefender.com/blog/privacy/how-to-delete-flash-cookies-permacookies-and-zombie-cookies. [Accedido: 09-SEP-2019].

[120] Samy Kamkar. Samy kamkar - evercookie - virtually irrevocable persistent cookies. https://samy.pl/evercookie/. [Accedido: 09-SEP-2019].

[121] Electronic Frontier Foundation. Verizon injecting perma-cookies to track mobile customers, bypassing privacy controls | electronic frontier foundation. https://www.eff.org/es/deeplinks/2014/11/verizon-x-uidh. [Accedido: 09-SEP-2019].

[122] MDN. Html5 - html | mdn. https://developer.mozilla.org/es/docs/HTML/HTML5. [Accedido: 09-SEP-2019].

[123] MDN. Window.sessionstorage - referencia de la api web | mdn. https://developer.mozilla.org/es/docs/Web/API/Window/sessionStorage. [Accedido: 09-SEP-2019].

[124] MDN. Window.localstorage - referencia de la api web | mdn. https://developer.mozilla.org/es/docs/Web/API/Window/localStorage. [Accedido: 09-SEP-2019].

[125] W3C. Web storage (second edition). https://www.w3.org/TR/webstorage/. [Accedido: 09-SEP-2019].

[126] IETF. Rfc 7232 - hypertext transfer protocol (http/1.1): Conditional requests. https://tools.ietf.org/html/rfc7232. [Accedido: 09-SEP-2019].

[127] Patrick Verleg. Cache cookies: searching for hidden browser storage. 2014.

[128] Robert Heaton. Cookieless user tracking for douchebags | robert heaton. https://robertheaton.com/2014/01/20/cookieless-user-tracking-for-douchebags. [Accedido: 09-SEP-2019].

[129] Jessamyn West. Library privacy in an age of browser fingerprinting - practical technology. 2019.

[130] Krishna.V.Nair and Elizabeth RoseLalson. The unique id's you can't delete: Browser fingerprints. 2018.

[131] MDN. Api canvas - html: Lenguaje de etiquetas de hipertexto | mdn. https://developer.mozilla.org/es/docs/Web/HTML/Canvas. [Accedido: 20-SEP-2019].

[132] JGunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. 2014.

[133] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. 2016.

[134] Peter Eckersley. How unique is your web browser? 2010.

[135] AmIUnique. Amiunique. `https://amiunique.org/`. [Accedido: 20-SEP-2019].

[136] EEF. Panopticlick. `https://panopticlick.eff.org/`. [Accedido: 20-SEP-2019].

[137] Kali Tools. p0f | penetration testing tools. `https://tools.kali.org/information-gathering/p0f`. [Accedido: 05-OCT-2019].

[138] die.net. tcptrack(1): Monitor tcp connections on network - linux man page. `https://linux.die.net/man/1/tcptrack`. [Accedido: 05-OCT-2019].