UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Proyecto de Grado

Documento de Análisis-Diseño

Autores:
Federico Nicolás Pernas
Javier Agustín Sanchez
Nicolás Zeballos

Supervisores:
Gustavo Betarte
Rodrigo Martinez
Marcelo Rodríguez

27 de diciembre de 2020

Índice

1. Introducción

Durante el transcurso de los años, el uso de las aplicaciones web ha incrementado en gran escala, alcanzando a formar parte de varios aspectos de nuestra vida cotidiana, como por ejemplo, en la compra de ropa, solicitud de un vehículo para traslado, pedido de comida a domicilio, transferencias bancarias y otra gran cantidad de acciones que normalmente requerían trasladarse o llamar a un local.

Este grupo de sistemas informáticos no es ajeno a los usuarios malintencionados, los cuales buscan fallas que puedan ser utilizadas para realizar acciones fraudulentas, robo o incluso extorsión. Para lograr detectar y aprender cómo dichos usuarios malintencionados realizan sus acciones, durante mucho tiempo se utilizaron dispositivos de seguridad llamados **Honeypots**. Sin embargo estos dispositivos fueron concebidos para el análisis de tráfico de determinados servicios conocidos, como SSH, FTP o SMTP, y no se tiene un gran avance de los mismos en el área de las aplicaciones web.

Por lo tanto, el objetivo de este documento es *presentar el proceso de análisis y diseño para el desarrollo de un Honeypot de aplicaciones web que pueda ser utilizado en sitios en producción, teniendo en cuenta evitar interrumpir la operativa con los usuarios de dicha aplicación.* Se considera que un camino para alcanzar los objetivos establecidos es el desarrollo de un *Honeypot Embebido de Aplicaciones Web* en base a *Honeytokens*, los cuales estarán *dentro* del activo principal y son los utilizados al momento de definir si el acceso de un usuario puede ser catalogado como sospechoso o malicioso.

El documento se compone de las secciones Marco Teórico, donde se presentan las definiciones necesarias y una visión general acerca de los Honeypot de aplicaciones web para poder comprender en profundidad el tema; Análisis, en donde se muestra paso a paso el proceso de análisis realizado para la especificación del sistema a elaborar; y por último Diseño, en donde se detalla las diferentes decisiones dentro de la etapa de diseño para la elaboración del producto final.

2. Marco Teórico

Antes de introducirse el análisis y diseño elaborado, se considera apropiado presentar las definiciones pertinentes para entender en detalle los Honeypots en el *contexto de las Aplicaciones Web*. Estas definiciones fueron elaboradas durante la investigación para poder establecer el área de trabajo y donde dirigir el mismo.

A continuación, se realizará una asociación entre las definiciones y características de los Honeypots en su contexto clásico, establecidas en el documento del **Estado del Arte** [1], y se presentará su pasaje al contexto de las aplicaciones web. A su vez, se darán definiciones acerca de **Honeypots Embebidos** y **Honeytokens**, los cuales son dos conceptos con gran impacto sobre la herramienta a elaborar. Finalmente, se presenta una descripción acerca de una **representación genérica** de un Honeypot de aplicaciones web y los requerimientos necesarios.

2.1. Honeypot de Aplicaciones Web

Para contextualizar a los Honeypots dentro del área de las Aplicaciones Web se requiere modelar la definición previamente establecida en el Estado del Arte [1] para Honeypot dentro del ámbito de las Aplicaciones Web. Por lo tanto, para comenzar se considera la siguiente definición de Honeypots:

"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".

Esta definición es lo suficientemente genérica, con lo cual logra abarcar diferentes contexto de trabajo. Gracias a esto, es posible ajustar algunos conceptos dentro de esta definición para llevarlos al contexto de las Aplicaciones Web:

- ***"cebo" llamativo para los atacantes***

  Es importante determinar que será un *"cebo" llamativo* dentro de una Aplicación Web. Considerando que los Honeypots suelen ser puertos de red en escucha o inclusos sistemas operativos completos, se podría definir que en el contexto de las Aplicaciones Web, un *"cebo" llamativo* puede ser cualquier componente dentro del protocolo HTTP (el cuál es el utilizado para la comunicación en este tipo de aplicaciones) o incluso una Aplicación Web completa.

- ***activos principales***

  Se comprende por *"activos principales"* a aquellos recursos o sistemas que son críticos para el funcionamiento de una organización. En el contexto de Aplicaciones Web, se considera que los activos principales de una organización son las Aplicaciones Web que exponen diferentes servicios para los clientes.

Por lo tanto, se puede inferir la siguiente definición de Honeypot de Aplicaciones Web a partir de las consideraciones establecidas:

"Un Honeypot de Aplicaciones Web es un dispositivo de seguridad, presentado como un recurso asociado al contexto de los sistemas web, que tiene la finalidad de atraer a los atacantes y mantenerlos alejados de las Aplicaciones Web dentro de cualquier Organización y que a su vez generen información sobre los ataques recibidos".

2.1.1. Principios de los Honeypots

En conjunto a la definición de Honeypot de Aplicaciones Web, es necesario considerar más aspectos definidos dentro del mundo de los Honeypots para no dejar fuera consideraciones claves al momento de realizar el pasaje de contexto. Para esto, se tienen definidos los siguientes principios sobre los Honeypots:

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.

Estas tres propiedades pueden ser llevadas al contexto de las Aplicaciones Web de la siguiente manera:

No interferencia

Se espera que al utilizar un Honeypot de Aplicaciones Web, la operativa y la comunicación entre los clientes y los servidores donde se encuentran las Aplicaciones Web no se vea interferida por el uso del dispositivo.

No exposición

Los Honeypots de Aplicaciones Web no podrán ser recursos directamente accesibles o visibles para los usuarios finales al momento de comunicarse con la aplicación, sino que deberán estar ocultos dentro de la misma Aplicación Web o en algún sitio de la Organización.

No establece comunicación con usuarios legítimos

Se considera que cualquier usuario que acceda al recurso utilizado como Honeypot, ya sea una componente del protocolo HTTP (como un header, cookie o incluso código dentro del cuerpo del mensaje), o una aplicación web completa, es considerado un usuario malicioso dado que al realizar acciones sobre el Honeypot se considera su actividad como sospechosa.

2.1.2. Clasificación de Honeypot de Aplicaciones Web

Comprender cómo se clasificarán los Honeypots es importante para conocer que limitantes puede tener un sistema de este tipo. Los mismos están clasificados de la siguiente forma:

* **Según el propósito**

  Para este caso, los Honeypot son clasificados como **de Producción**, cuando su finalidad es proteger los activos expuestos de la organización; y como **de Investigación**, cuando su finalidad consiste en recolectar información sobre los ataques generados por los usuarios maliciosos.

* **Según el Grado de Interacción**

  Esta clasificación consiste segmentar los Honeypots según *el tipo de respuestas que le dan a los atacantes*, con lo cual se definen los Honeypots de **Baja Interacción**, los cuales brindan una respuesta sencilla, pero tiene un bajo costo de puesta en producción. En contra parte a este tipo, se encuentra los Honeypots de **Alta Interacción**, que permiten simular sistemas enteros, brindando respuestas más elaboradas las cuales generan una mayor atracción de los atacantes. Sin embargo, este tipo de Honeypot son más costosos y en caso de no ser configurados de forma correcta podría llegar a ser comprometidos por los atacantes. Por último, en un punto intermedio, se encuentran los Honeypot de **Media Interacción**, los cuales intenta combinar las mejores características de los Honeypots de Baja Interacción y Alta Interacción, tratando de conseguir respuestas lo más factible posible a un bajo costo de puesta en producción y garantías altas respecto a características de seguridad.

Para el desarrollo de un Honeypot de Aplicaciones Web es importante llevar estas definiciones al nuevo contexto de trabajo para poder clasificar la herramienta a elaborar. Para esto, se plantean las siguientes definiciones:

* **Según el propósito**

  - **Honeypot de Producción**

    Se considera un Honeypot de Producción como un recurso que es utilizado para quitar la atención a un atacante de las aplicaciones web dentro de la organización, las cuales se pretenden proteger.

- **Honeypot de Investigación**

  Se considera un Honeypot de Investigación como un recurso que es utilizado para recolectar información sobre ataques conocidos (como los establecidos en el documento **OWASP Top Ten** [2]) o descubrir nuevos caminos y herramientas para vulnerar este tipo de aplicaciones.

* **Según el Grado de Interacción**

- **Baja interacción**

  Un Honeypot de Baja Interacción se considera como un recurso estático dentro del protocolo HTTP o un sitio web que no muestre respuestas elaboradas y tenga un bajo costo de puesta en producción.

- **Alta interacción**

  Un Honeypot de Alta Interacción se considera como una aplicación *funcionalmente completa*, es decir, se espera que sus respuestas tengan la capacidad de simular un comportamiento esperado por dicha aplicación.

  Este Honeypot podría ser cualquier servidor con una aplicación que se tenga operativo y se pretenda utilizar para ser vulnerado o incluso una instancia replicada de la aplicación web que se pretende proteger. En ambos casos, se puede deducir que se tiene que considerar un costo de puesta de producción y un riesgo de seguridad alto, dado que el mismo podría llegar a ser completamente comprometido por los atacantes y podría ser utilizado para generar ataques hacia otros objetivos.

- **Media interacción**

  Un Honeypot de Media Interacción debe brindar una respuesta atractiva para el atacante pero presentar un bajo costo al realizar su puesta en producción. En particular, no se tienen características particulares de los Honeypot de Aplicaciones Web para que sean clasificados como Media Interacción. Algunas de las implementaciones realizadas hasta el momento, presentan sistemas donde se muestra una página web estática, la cual no responde ante intentos de interacciones con el backend, o incluso tienen la capacidad de detectar ataques conocidos, y ante esto retornar una misma respuesta dependiendo del tipo de ataque, pero considerando mantener un mecanismo sencillo de instalación.

2.2. Honeypot Embebido

A partir de la definición de Honeypot de Aplicaciones Web, presentada en la sección Honeypot de Aplicaciones Web se puede introducir la definición de Honeypot Embebido en Aplicaciones Web. Se destacan los conceptos que serán ajustados o extendidos acorde a los cambios que requiere un Honeypot Embebido:

- ***"Tiene la finalidad de atraer a los atacantes"***

  Los Honeypot Embebidos se despliegan en el mismo activo principal, de manera que buscan alcanzar un mejor nivel de atracción haciéndose pasar por un elemento del objetivo real del atacante. Por lo tanto, se extiende este concepto incorporando como este nuevo tipo de Honeypot que busca atraer a los atacantes simulando ser parte de una componente real.

- ***"Mantenerlos alejados de las Aplicaciones Web"***

  A partir de cómo se realiza el despliegue de los Honeypots Embebidos, se puede deducir que *"se busca alejar al atacante del activo principal desde el mismo activo principal"*. El atacante es atraído al Honeypot Embebido, dado que sus ataques potencialmente serán dirigidos a los elementos propios de este Honeypot que está implantado en la aplicación web. Por lo tanto, se modifica este concepto con el fin de aclarar cómo este tipo de Honeypot aleja a los atacantes de las aplicaciones web. Se espera que los usuarios malintencionados no reconozcan la frontera entre el objetivo real y el elemento o conjunto de elementos en los cuales solo malgasten tiempo y esfuerzo.

En consecuencia se puede inferir la siguiente definición de Honeypot Embebido en Aplicación Web a partir de las consideraciones establecidas:

"Es un caso particular de Honeypot de Aplicación Web donde se busca incrementar la atracción del atacante y dirigir sus ataques al Honeypot desde el interior del mismo activo principal."

2.3. Honeytoken

Según como se presentó el concepto en el estado del arte, un Honeytoken refiere a *determinado recurso de IT que se despliega en la red o en sistemas con el propósito de atraer la atención de atacantes y obtener información sobre las vulnerabilidades que ellos intentarán explotar al manipular o abusar de los mismos*. El único detalle que varía en esta definición, por tratarse de un contexto de aplicación web, es el interés en que sean desplegados a nivel de un servidor web o el tráfico web que dicho servidor maneja.

Según *Spitzner* en **The Other Honeypot** [3], los Honeytokens son entidades digitales también son considerados Honeypots. En particular su valor radica en su abuso, no en su uso. Esto implica que estos elementos deben ser alterados para que se pueda inferir información en base a ellos.

Se entenderá “abuso” como sinónimo de “manipulación” , pues la simple interacción con un Honeytoken será suficiente para evidenciar que se está en presencia de un usuario malintencionado. Y es necesario definir que por *manipulación* se entiende a la *acción de modificar el estado en que se presenta un Honeytoken al usuario del sistema*. Es decir que si un usuario altera el contenido de un Honeytoken, así como si elimina un Honeytoken, ese comportamiento será suficiente para catalogar su interacción como malintencionada.

2.4. Representación del sistema

El sistema se modela empleando una **Máquina de Estados**, la cual permite *describir todo el proceso en el que un atacante interactúa con el Honeypot y como éste responde ante los diferentes estímulos recibidos.*

Esta máquina de estados tiene la característica de que los **Estados** se pueden ver como *"diferentes etapas del proceso de trabajo del Honeypot"*, lo cual mejora el entendimiento del sistema al *analizar su comportamiento en etapas* y a su vez *describir las condiciones de transición entre dichas etapas.*

La máquina de estados elaborada es la siguiente:

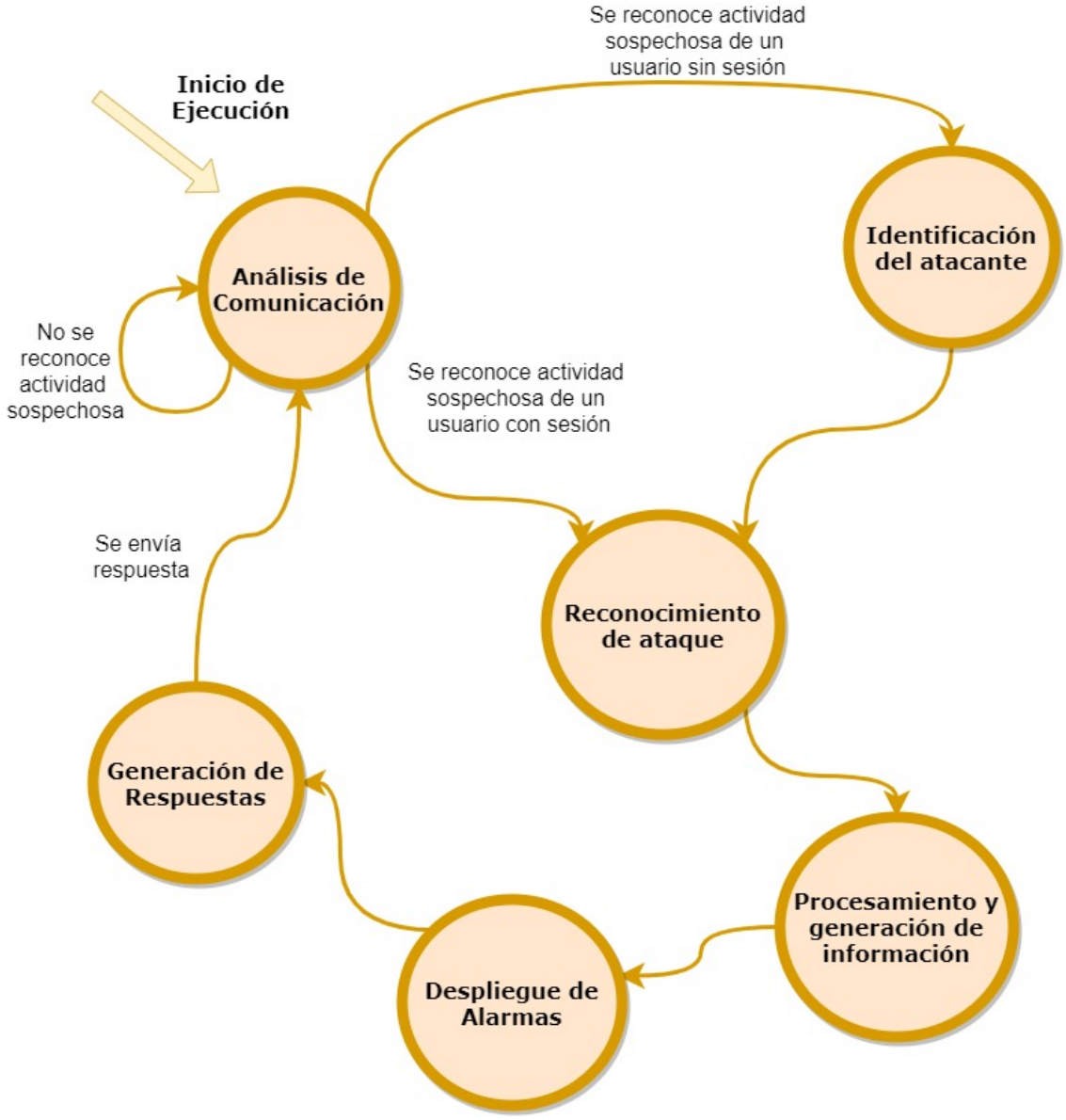

Figura 1: Máquina de estados para representar el funcionamiento de un Honeypot de Aplicaciones Web

A continuación se describe el significado de cada estado junto con las funcionalidades esperadas a ejecutarse en cada uno:

- **Estado 1: Análisis de comunicación**

  Este estado refleja la etapa en que un usuario se comunica con el sistema y *realiza una acción que puede ser considerada maliciosa*. Existen dos caminos posibles a tomar: *mantenerse en el estado y continuar analizando nuevos mensajes* o *ejecutar una acción a partir de un comportamiento sospechoso*.

  En este estado se espera cumplir con las siguientes funcionalidades:

  1.1 **Análisis de mensajes HTTP**

  Debido que el marco de estudio comprende a las Aplicaciones Web, es necesario tener la capacidad de analizar y manipular mensajes del **Protocolo HTTP**, el cual es utilizado para la comunicación en este tipo de aplicaciones.

  1.2 **Detección de comportamiento sospechoso**

  Para poder diferenciar de forma preliminar los mensajes emitidos por un usuario entre aquellos mensajes que pueden ser identificados como ataques de los que no, es necesario que el sistema pueda reconocer si el mensaje contiene comportamiento sospechoso.

  1.3 **Identificación de un usuario malintencionado**

  Cuando el sistema recibe mensajes de parte de un usuario, se debe tener la capacidad de diferenciar si dicho usuario ha sido identificado como malintencionado anteriormente.

- **Estado 2: Identificación del atacante**

  Este estado representa una etapa en donde se *asigna un identificador único para reconocer al usuario ya clasificado previamente como malintencionado*.

  En este estado se espera cumplir con la siguiente funcionalidad:

  2.1 **Generación de sesión para un atacante**

  Se deberá contar con un mecanismo de asignación y manejo de sesiones para los diferentes usuarios que ya fueron reconocidos previamente como atacantes.

- **Estado 3: Reconocimiento de ataque**

  En este estado ya se sabe que se está procesando un *mensaje catalogado como ataque*, emitido por *un usuario malintencionado hacia la Aplicación Web que se está tratando de proteger*. Es posible *reconocer el ataque emitido*, es decir, en este estado es donde se realiza una clasificación del ataque emitido.

  En este estado se espera cumplir con la siguiente funcionalidad:

3.1 **Clasificación de ataque**

Se deberá clasificar que tipo de ataque emitió el usuario malintencionado.

- **Estado 4: Procesamiento y generación de información**

Una forma de enriquecer las funcionalidades del Honeypot es generar la mayor cantidad de información posible y disponibilizarla para que otros expertos o técnicos puedan informarse sobre los ataques, o incluso normalizarla para que pueda ser consumida por otros sistemas de seguridad. Es importante considerar una etapa en la cual se realicen tareas necesarias para manipular la información recibida desde el Honeypot y distribuirla posteriormente.

En este estado se espera cumplir con la siguiente funcionalidad:

4.1 **Normalización de datos**

Es deseable que la información procesada en este estado sea normalizada y de esta forma pueda ser consumida por un sistema externo con el cual se deseé compartirla.

4.2 **Loggeo de información**

Se espera registrar todos los datos emitidos por el atacante mediante sus mensajes HTTP, los mensajes de respuesta de la Aplicación Web y la información que el Honeypot genere relacionada a las comunicaciones catalogadas como alarmantes.

4.3 **Envío de información a un servidor remoto**

Se espera tener la capacidad de enviar la información generada a un servidor remoto en donde sea más seguro de mantenerla en caso de que el servidor donde se encuentra el Honeypot sea comprometido.

4.4 **Generación de datos para compartir**

Es deseable que se cuente con una funcionalidad que permita crear estructuras de datos estandarizada, las cuales contengan información sobre el comportamiento del atacante y puedan ser compartida con otros sistemas.

- **Estado 5: Despliegue de alarmas**

Una funcionalidad importante y esperada por un Honeypot es que pueda dar aviso a los responsables de una Aplicación Web al momento en que se está efectuando un ataque. Se debe considerar una etapa en la cual se realicen estos avisos para proteger proactivamente el sitio.

En este estado se espera cumplir con la siguiente funcionalidad:

5.1 **Configuración de alarmas**

Se debe permitir la configuración de alarmas para dar aviso a los administradores del sitio que se está generando un ataque, y de esta forma ellos puedan actuar de forma acorde.

5.2 **Activación de alarmas**

Una vez que un usuario malintencionado realiza una actividad categorizada como sospechosa, es necesario activar las alarmas previamente configuradas para que los administradores sean alertados.

- **Estado 6: Generación de respuesta**

Este estado representa una etapa en donde se realiza la creación de respuestas que aseguren mantener la comunicación del atacante con el sistema, simulando los efectos de un ataque exitoso en caso de ser necesario e incrementando la interacción con el mismo para atraerlo más hacia el Honeypot y, de esta forma, alejarlo del activo principal.

En este estado se espera cumplir con las siguientes funcionalidades:

6.1 **Generación de respuesta**

Es esperable que ante un ataque se brinde una respuesta que sea lo más aproximada a un resultado esperado por el atacante, y así generar mayores garantías de que el ataque seguirá fluyendo en dirección al Honeypot y no al sitio original.

6.2 **Simulación de ataque en un ambiente controlado**

Para potenciar las garantías de que una respuesta se asemeje a un resultado real y esperable por un atacante, es conveniente replicar los ataques sobre un "ambiente controlado", donde la ejecución de estos ataques no signifique un posible daño para la organización.

6.3 **Incremento del nivel de interacción**

Otra de las formas de mantener la atracción de un atacante hacia el Honeypot es brindarle más elementos que lo atraigan hacia el Honeypot. Esto se puede ver como agregar más Honeytokens o exponer otras vulnerabilidades en el sitio a medida que se detecta que el usuario genera cada vez más actividad sospechosa.

2.4.1. Requerimientos Generales

Una vez finalizada la máquina de estado anteriormente descrita, es posible definir los requerimientos generales para un Honeypot de Aplicaciones Web:

Manejo de protocolo HTTP

Es necesario que el sistema comprenda y procese información enviada mediante el protocolo HTTP, para que de esta manera pueda analizar y procesar los mensajes intercambiados en la comunicación.

Detección de ataques

El sistema debe discriminar entre usuarios normales de la aplicación y atacantes. Para esto, se deberá reconocer cuando un usuario intenta atacar el sitio y si dicho usuario ya lo había intentado realizar antes, utilizando sus atributos de sesión. No es esperado que un usuario normal que utiliza la aplicación pueda llegar al Honeypot, dado que por definición se asume que *"todo usuario que llega a un Honeypot es un usuario malicioso"*.

Reconocimiento de ataques

Se espera que el sistema sea capaz de reconocer ataques conocidos y no conocidos. Para los ataques conocidos, un posible camino es tomar como base el documento **OWASP Top Ten** [2], donde se tiene un conjunto de los ataques web más frecuentes. Como ataque no conocido se entiende aquel en que no se tiene conocimiento ni documentación pública divulgada hasta el momento.

Loggeo de información

El sistema debe ser capaz de persistir toda la información recolectada en un archivo de log con un previo proceso de normalización. Dicho archivo tendrá la capacidad de ser almacenado localmente o enviado hacia un servidor remoto.

Generación de datos para compartir

Se considera importante que el sistema tenga la capacidad de compartir datos sobre el comportamiento de los atacantes con otros sistemas.

Creación de respuestas

El sistema debe tener la capacidad de generar respuestas que resulten suficientemente atractivas para el atacante, de manera que éste centre su atención en el Honeypot y no en el sitio original.

Configuración y activación de alarmas

Ante la detección de actividad sospechosa, el sistema debe alertar a los administradores del sitio mediante alarmas que fueron previamente configuradas por ellos.

Simulación de vulnerabilidades

Una vez que el atacante se encuentra en el Honeypot es necesario *que se mantengan interactuando con él*. Para ello, el Honeypot deberá exponer ciertas vulnerabilidades que al momento en que el atacante las intente explotar, considere que las respuestas brindadas son atractivas.

3. Análisis

En la siguiente sección se presenta el análisis realizado para alcanzar el desarrollo de la herramienta final.

En Análisis de HTTP y HTML se realiza un análisis del protocolo HTTP y el lenguaje HTML para describir la importancia de estas dos componentes en el análisis. En Honeypot a desarrollar se presentan los objetivos del desarrollo y la máquina de estados que modela el comportamiento del Honeypot a construir, la cual es una instancia particular de lo presentado en Representación del sistema. Los distintos componentes que integran el Honeypot se presentan en Componentes del Sistema. Por último, en la sección Interacción con el Honeypot se definen las funciones que expondrá este Honeypot.

3.1. Análisis de HTTP y HTML

Las aplicaciones web están compuestas de diferentes tecnologías, tanto en componentes de tipo hardware como software, pero en el estudio abordado en este proyecto se hace énfasis en dos componentes: **protocolo HTTP** y **lenguaje HTML**.

3.1.1. Protocolo HTTP

El protocolo HTTP es el que permite *recibir* todos los elementos de una aplicación web (mediante mensajes denominados *response*) para ser mostrada por un navegador, y *enviar* las peticiones (mediante mensajes denominados *request*) realizadas desde navegador hacia el servidor. Más allá del funcionamiento, es interesante analizar la anatomía de los mensajes, los cuales pueden ser divido en tres secciones [4]: primera línea (cuyo contenido varía según si el mensaje es un request o un response), los encabezados (o *headers*) y el cuerpo (o *body*), siendo éste opcional. El nombre de cada elemento puede verse en la siguiente figura:

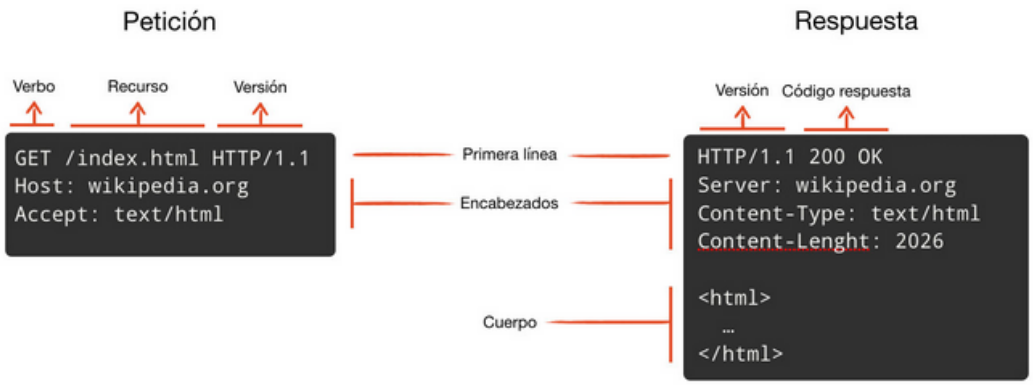

Figura 2: Anatomía de un mensaje HTTP

En el contexto del proyecto se define que las aplicaciones a las cuales se hará referencia serán

aquellas basadas en el protocolo HTTP, y las secciones de interés son aquellas que transportan metadata y contenido de la aplicación.

- **Headers** Es la sección que provee información o metadata sobre el mensaje HTTP en forma de parámetros con su respectivo valor (único o lista de valores), subdivido en cuatro categorías pues se tienen headers generales, para request y response.
- **Body** Es la sección donde esta el cuerpo (si lo hay) de un mensaje HTTP, por lo tanto se utiliza para transportar dicho contenido.

3.1.2. Lenguaje HTML

HyperText Markup Language (HTML por sus siglas en inglés) es un lenguaje que permite la visualización de aplicaciones web (junto con tecnologías como CSS y JavaScript) basado en el uso de *etiquetas*, las cuales permiten la definición de distintos elementos y propiedades de la interfaz visual de la aplicación. Por ejemplo, la etiqueta *form* permite la definición de un *formulario* en una aplicación web.

Otro ejemplo destacado son los inputs que son elementos HTML que pueden contener y transportar datos entre un usuario y una aplicación web. Muchos usuarios malintencionados suelen emplear herramientas automáticas que escanean el código HTML de estas aplicaciones e intentan explotar cada elemento, ejecutando todos los enlaces que encuentran en su camino, así como inyectar información en todos los inputs a los que accede. Como es una herramienta, puede llegar a no distinguir entre campos ocultos o visibles.

3.2. Honeypot a desarrollar

El objetivo a alcanzar es el *desarrollo de un Honeypot de aplicaciones web que pueda ser utilizado para detectar posibles intrusiones a sitios en producción, teniendo en cuenta evitar interrumpir la operativa con los usuarios de dicha aplicación.* A su vez, considerando que el comportamiento del mismo debe permitir extensiones y cubrir una amplia variedad de aplicaciones, se establece que será un framework [1] que cumpla con los siguientes objetivos:

1. **Extensible** El diseño del framework debe permitir la adición de nuevos componentes que permitan implementar tanto nuevas funcionalidades como disponer de mejores versiones o alternativas a las ya existentes.
2. **Genérico** El framework debe ser lo suficientemente genérico para ser utilizado por diferentes aplicaciones web.
3. **Obtención de información** Debe contar con capacidad para recolectar información útil sobre ataques y/o atacantes.

[1] Se llama *framework* [5] al esquema o estructura que se establece y que se aprovecha para desarrollar y organizar un software determinado

4. **Detección de intrusos** Debe proveer un mecanismo para discernir entre usuarios legítimos y maliciosos.

5. **Seguimiento de atacantes** Es deseable que cuente con algún mecanismo que permitan dar seguimiento al proceder de cada atacante.

6. **Sin interrupción operativa de la aplicación web** Es necesario que no se interrumpa en la operativa habitual de la aplicación web, ya que su presencia debe pasar desapercibida.

Un camino posible para lograr cumplir los objetivos anteriores es el desarrollo de un *Honeypot Embebido de Aplicaciones Web* en base a *Honeytokens*, los cuales estarán *dentro* del activo principal y serán los utilizados al momento de definir si el acceso de un usuario puede ser catalogado como sospechoso o no.

Cabe recordar que en Honeytoken se definieron los Honeytokens, su importancia y por qué son de utilidad en el contexto de Aplicaciones Web. Con el uso e inclusión de estos honeytokens *dentro* de la Aplicación Web, se asegura que el Honeypot solo procesa dichos artefactos, teniendo en cuenta la información que estos contienen para poder cumplir con los objetivos propuestos. *Cualquier otra información del sitio no será procesada ni analizada por el Honeypot.*

Se consiera apropiado aclarar que el término **Honeypot** refiere al framework a desarrollar. A su vez, el mismo es considerado un **Honeypot Embebido** debido a que los **Honeytokens** son inyectados por dicho framework sobre componentes de la Apicación Web.

El Honeypot a desarrollar debe ser una instancia particular de la máquina de estados presentada en Representación del sistema. Dado que el objetivo principal de la herramienta es poder detectar la intrusión de un usuario y registrar información relevante al mismo, se desarrolla la siguiente instancia *reducida*:

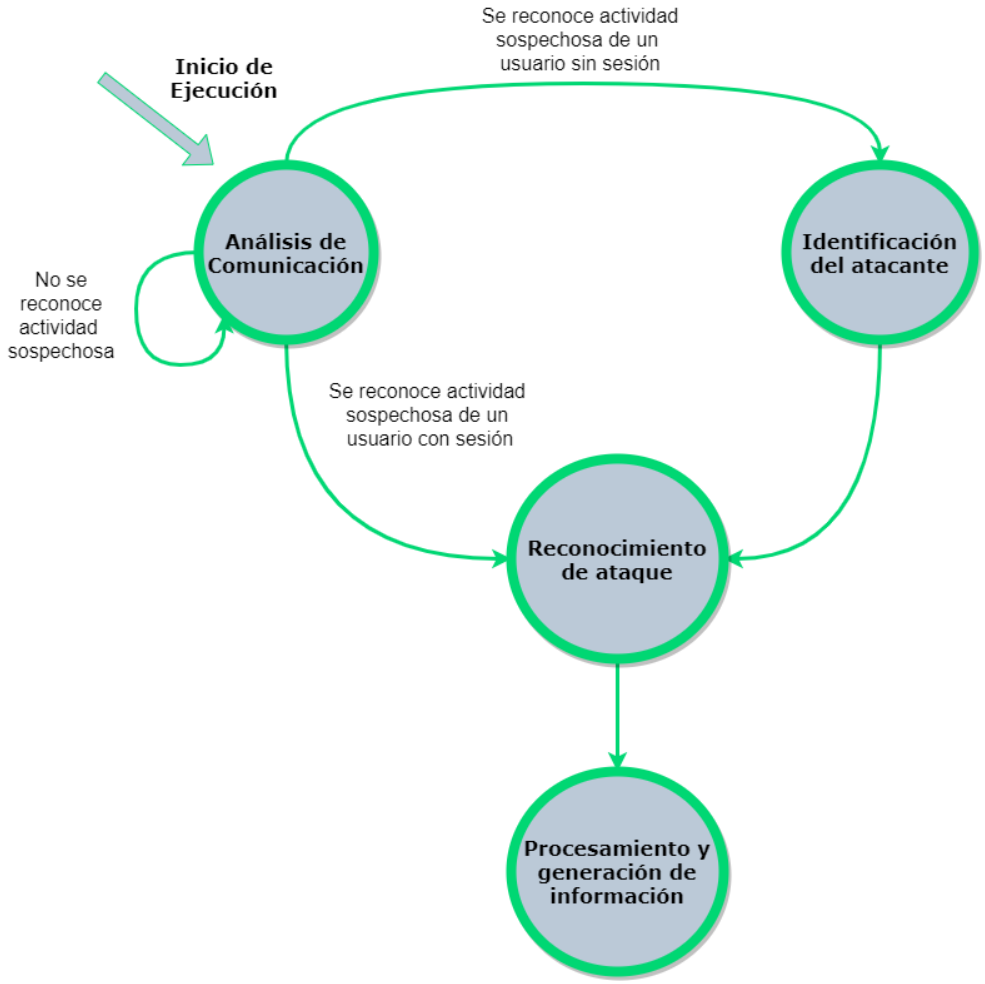

Figura 3: Máquina de estados del Honeypot Embebido a desarrollar

Así como para la máquina de estados genérica, se describe el comportamiento de los estados y las funcionalidades relacionadas a cada uno de ellos:

- **Análisis de comunicación**

  En este estado se analizan los pedidos (request) en busca de indicios de *comportamiento sospechoso*.

  Por *comportamiento sospechoso* se entiende a la *manipulación de Honeytokens* tal como fue definida con anterioridad.

  En este estado se definen las siguientes funcionalidades:

  - **Análisis de mensajes HTTP**

    Se toma el mismo concepto que en la máquina de estados general.

  - **Detección de comportamiento sospechoso**

    Se entiende que un mensaje contiene "comportamiento sospechoso" si alguno de los Honeytokens allí presentes ha sido modificado o eliminado por el usuario.

- **Identificación de un usuario malintencionado**

  Se toma el mismo concepto que en la máquina de estados general.

- **Identificación del atacante**

  Se define este estado de igual manera que para la máquina de estados genérica.

  En este estado se define la funcionalidad:

  - **Generación de sesión para el usuario**

    Se toma el mismo concepto que en la máquina de estados general

- **Reconocimiento del ataque**

  El Honeypot tendrá como finalidad detectar si se está en presencia de un usuario malicioso mediante el análisis de los Honeytokens presentes en el request. Este objetivo no representa que se esté necesariamente frente a un ataque, por lo que si bien el estado se representa en la máquina de estados, no se reconocerán los ataques perpetuados por un usuario malicioso.

- **Procesamiento y generación de información**

  La información recabada por el Honeypot debe ser disponibilizada para que otros puedan acceder a ella utilizando algún mecanismo para normalizarla y compartirla con otros técnicos especializados o dispositivos de seguridad.

  En este estado se definen las siguientes funcionalidades:

  - **Normalización de información**

    Se toma el mismo concepto que en la máquina de estados general

  - **Loggeo de información**

    Se toma el mismo concepto que en la máquina de estados general

  - **Envío de información a un servidor remoto**

    Se toma el mismo concepto que en la máquina de estados general

3.3. Componentes del Sistema

El siguiente diagrama es una representación de alto nivel del despliegue del Honeypot:

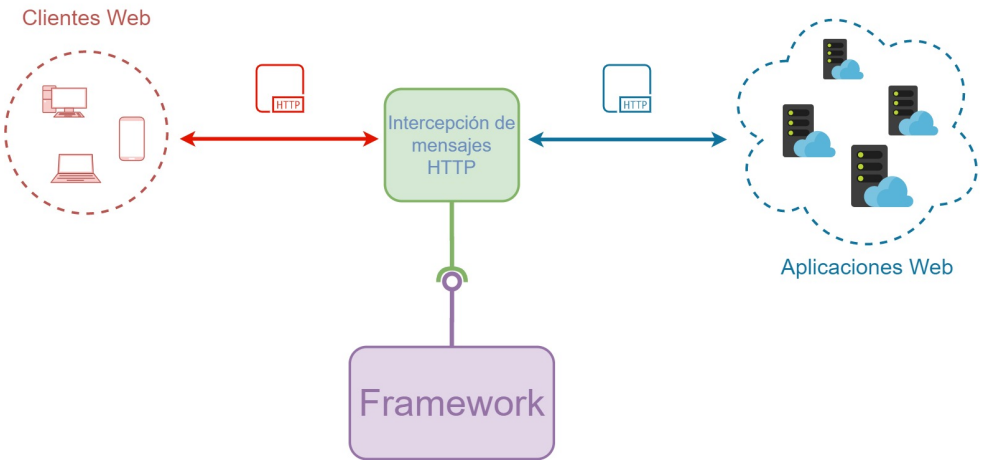

Figura 4: Diagrama de componentes del Honeypot

Como se observa, **Aplicaciones web** es el conjunto de aplicaciones que ofrecen servicios que consumirán los distintos **Clientes web** a demanda. Algunos de estos clientes web buscan *explotar vulnerabilidades* sobre estas aplicaciones que les permitan obtener datos confidenciales o incluso tomar control de la infraestructura. De esta manera se incorporará un **Interceptor de mensajes HTTP** en la comunicación entre cliente y aplicación. Este *interceptor* puede ser cualquier elemento que pueda interponerse en la comunicación entre cliente y servidor (un proxy, algún tipo de *plugin* en caso de tratarse de un CMS, o incluso la aplicación misma), y estará encargado de redirigir los mensajes hacia el **Framework**.

Será responsabilidad del Honeypot el manejo de los Honeytokens presentes en el mensaje que disponibiliza el *interceptor*. **El Honeypot no deberá leer, obtener o eliminar de ninguna manera ningún tipo de información que no esté contenida y/o sea parte de un Honeytoken.**

El siguiente diagrama ilustra el comportamiento del Honeypot:

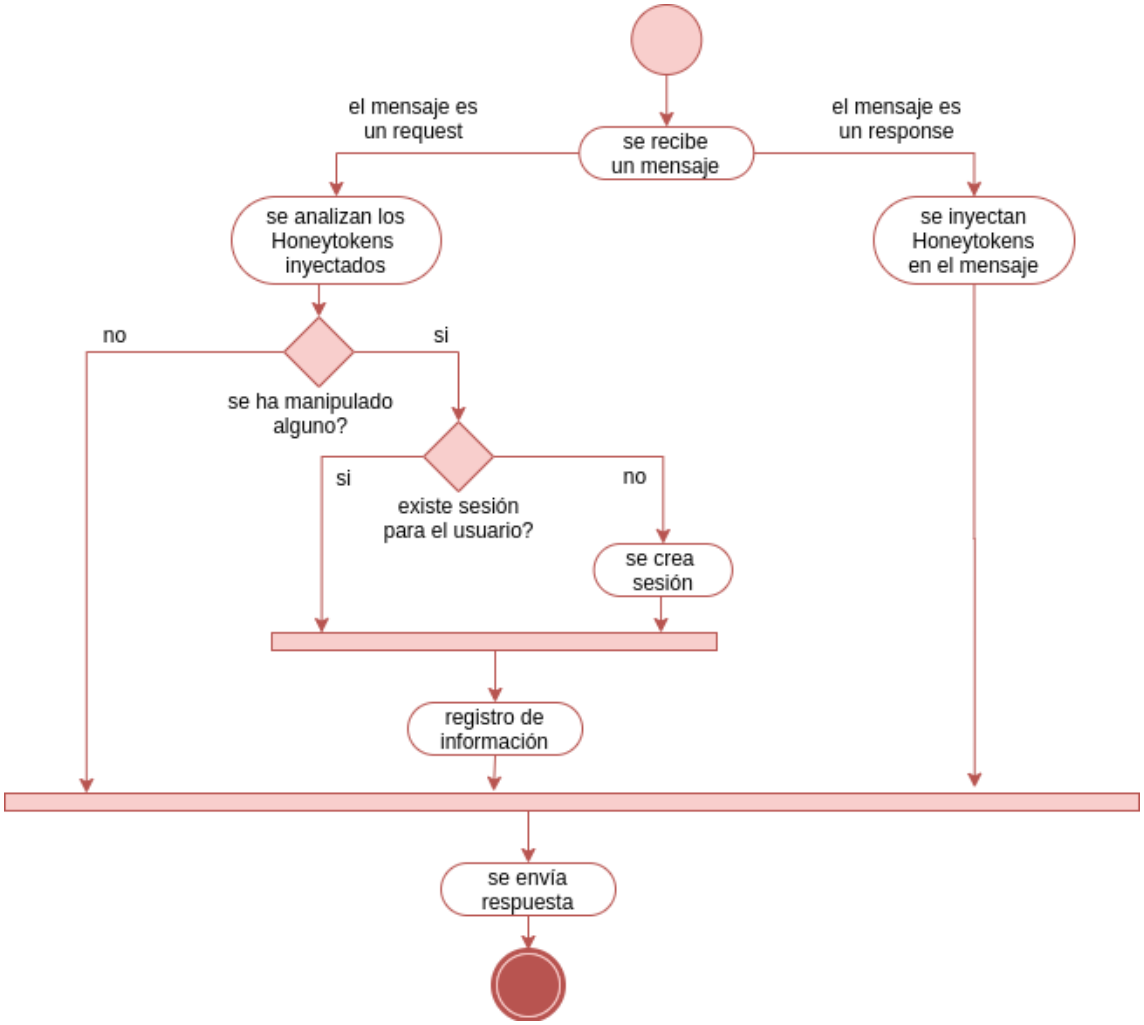

Figura 5: Diagrama de flujo del Honeypot

La ejecución del flujo comienza al recibir un mensaje HTTP. Estos mensajes deberán ser catalogados en *mensajes emitidos por la aplicación web (response)* o *mensajes emitidos por un cliente (request)*.

Los mensajes response son los mensajes a los que se deberán *inyectar* Honeytokens. Los mensajes request recibidos serán analizados para determinar si los Honeytokens inyectados en el response correspondiente han sido manipulados. En caso de que se detecte una manipulación se deberá dar seguimiento al usuario responsable del request procesado. Es decir, se le deberá crear una *sesión* a ese usuario (si es que no tenía una previamente). Además, se almacenará información con el objetivo de tratar de identificar al usuario y su accionar catalogado como intrusivo.

3.4. Interacción con el Honeypot

Para que el sistema comience su funcionamiento, es necesario que exista una entidad o componente encargado de entregar al Honeypot los mensajes emitidos por la aplicación web y los clientes. El siguiente diagrama de secuencia ilustra esa comunicación:

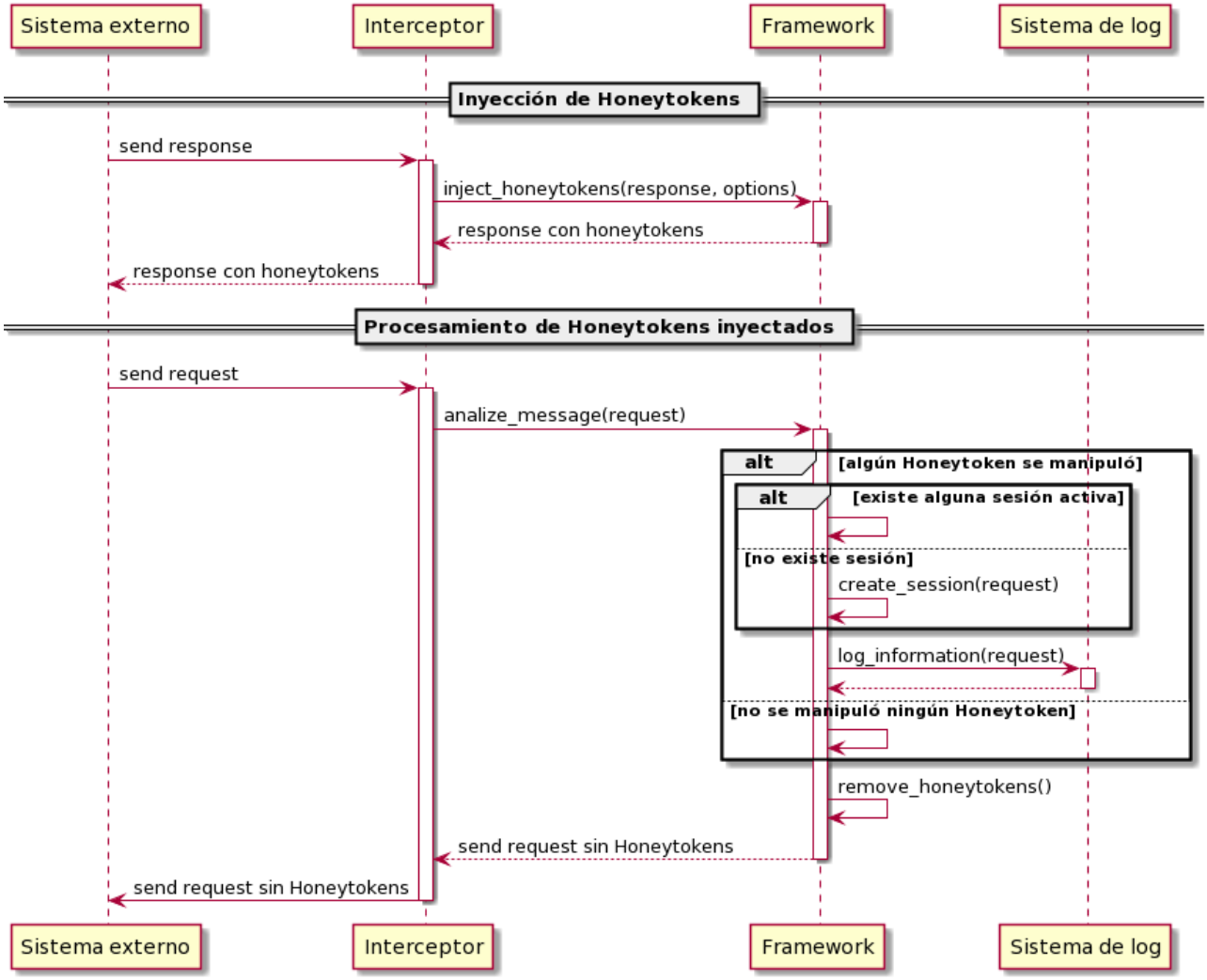

Figura 6: Llamadas al Honeypot

La entidad **sistema externo** hace referencia al emisor de los correspondientes mensajes (para el caso de un request sería un usuario, y para el response la aplicación web), mientras que **interceptor** hace referencia a esa entidad capaz de dirigir los mensajes hacia el **framework**. Se deduce que esta entidad debe ser capaz de ubicarse *en medio* de la comunicación entre cliente y aplicación web.

Para el caso en que se precise inyectar Honeytokens, se dispondrá de la operación *inject_honeytokens*, la cual recibirá el mensaje response. En el caso del análisis de un request, se dispondrá de la operación *analize_message* que recibirá el mensaje request a procesar. An-

tes de que el framwork retorne el mensaje a la entidad intermediaria, los Honeytokens deberán ser eliminados de manera que la aplicación web procese los mensajes sin estar alterados por el Honeypot.

El diagrama anterior establece claramente las responsabilidades de cada entidad involucrada en el funcionamiento del Honeypot. La entidad intermediaria será responsable únicamente de funcionar como *pasaje* entre los mensajes y el **framework**, este último estará a cargo de saber qué Honeytokens se han inyectado en que response, además de cuál es el request asociado a ese response. Así como el manejo de sesiones que también estará a cargo del **framework**.

4. Diseño

En esta sección final se pretende abordar el desarrollo de la etapa de diseño del proyecto. Esta etapa se compone de la construcción y especificación de las últimas características necesarias para dar lugar a la comprensión total del framework.

En Diagrama de Despliegue se presenta la disposición de los distintos componentes que integran el Honeypot. En la sección Diagrama de Clases se describen las clases que componentes al framework mediante el uso de un Diagrama de Clases, seguido de la sección Vista de operaciones, donde se ilustran las operaciones que involucradas en el framework y a que componente corresponden.

4.1. Diagrama de Despliegue

Se emplea un **diagrama de despliegue**[6] como presentación de la disposición física en alto nivel de la solución junto a sus diferentes componentes:

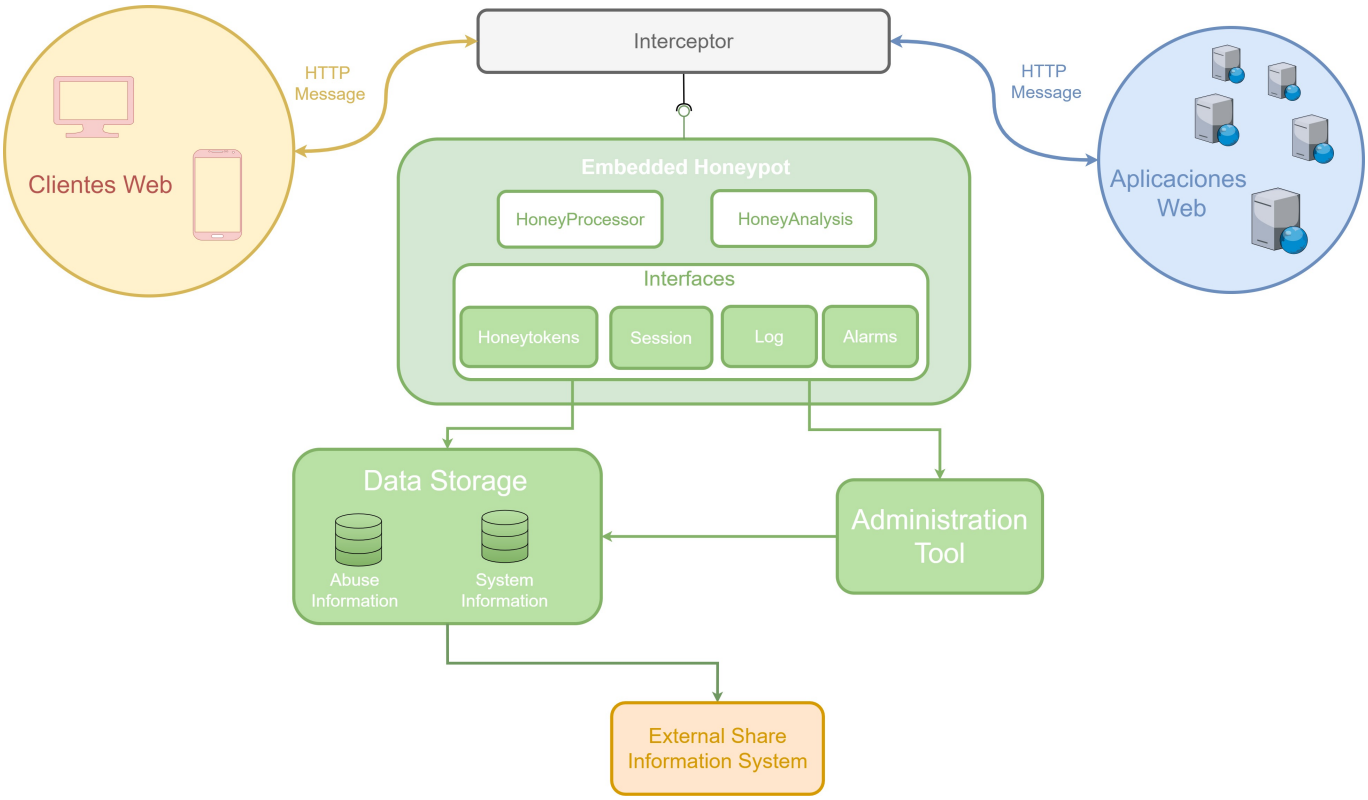

Figura 7: Diagrama de despliegue del framework

Este diagrama se compone de las siguientes partes:

Clientes Web

Los *Clientes Web* representan a *todos los usuarios que pueden conectarse a las Aplicaciones Web*. Con "*todos los usuarios*" se hace referencia tanto a usuarios normales como malintencionados.

Los clientes podrán conectarse a las aplicaciones utilizando una conexión mediante un navegador web en cualquier dispositivo que lo permita (PC, dispositivo móvil, etc).

Aplicaciones Web

Las *Aplicaciones Web* hacen referencia al *sistema final al que el usuario desea conectarse*, es decir, a la aplicación en la que el framework agregará los Honeytokens para kuego analizar el abuso sobre ellos.

Interceptor

El Interceptor es la componente encargada de *interceptar los mensajes HTTP emitidos entre los clientes y las Aplicaciones Web para luego ser utilizados por* ***Embedded Honeypot*** *para su posterior modificación y análisis*. Básicamente, esta componente es utilizada para ponerse cómo intermediario de la comunicación y consumir del framework para hacer el análisis de los mensajes. Como se menciona anteriormente, esta componente puede ser cualquier tipo de software que capture los mensajes y pueda comunicarse con el framework, como por ejemplo un proxy, un *plugin* o la aplicación web misma.

Framework

El Framework puede ser dividido en tres grandes componentes, las cuales a su vez se subdividen en otras subcomponentes:

1. **Embedded Honeypot**

   Embedded Honeypot es la componente fundamental del framework, dado que se encarga de *procesar los mensajes HTTP que se envían entre los Clientes Web y las Aplicaciones Web para poder detectar actividad sospechosa y agregar la información necesaria en base a la actividad sospechosa detectada*. Estos mensajes, *son recibidos* desde el **Interceptor** a través de una interfaz.

   Las subcomponentes en las que se divide el mismo son:

   1.1 **Honey Processor**

   Esta subcomponente es la que se encarga de *agregar Honeytokens y los valores de sesión en cada mensaje* HTTP response recibido por el **Interceptor**.

1.2 **Honey Analysis**

Esta subcomponente se encarga de *definir si un request HTTP emitido por un cliente contiene un abuso*. En caso de detectar un abuso, la misma componente deberá *crear un identificador de sesión para ser utilizado para darle seguimiento al usuario malicioso* y también *almacenar la información acerca del comportamiento del usuario y dar un aviso a los administradores, respectivamente*.

1.3 **Interfaces**

Esta subcomponente contiene las cuatro interfaces principales del framework: **Honeytokens**, **Session**, **Log** y **Alarms**. Estas interfaces son las que definen las operaciones necesarias para realizar la operativa de **Honey Processor** y **Honey Analysis**. En la sección Vista de operaciones se ilustra con mayor profundidad las operaciones involucradas en dichas interfaces.

Cabe destacar que al presentarse como interfaces, estas componentes permiten al framework ser extensible, lo cual ayuda que se pueda tener diferentes implementaciones de las operaciones definidas para cada una. En la sección Vista de operaciones se ilustra con mayor profundidad las operaciones involucradas en dichas interfaces.

2. **Administration Tool**

La Herramienta de Gestión es la componente *encargada de brindarle una interfaz a los administradores para que puedan comunicarse con el framework y así realizar las configuraciones necesarias sobre el mismo*. Estas configuraciones involucran información para el funcionamiento de todas las entidades (Honeytokens, Sessions, Alarms y Log) y valores requeridos para que el framework funcione apropiadamente (como las rutas o conexiones a bases de datos para obtener los datos de las entidades).

3. **Data Storage**

La componente de *Data Storage* alberga la información necesaria para el funcionamiento del framework. Básicamente, contiene un repositorio encargado de mantener los datos del framework y sus entidades y otro con los datos de los ataques detectados. Esta componente implementa mecanismos para comunicarse con *External Share Information System*.

External Share Information System

La componente *External Share Information System* esta asociada a sistemas externos al framework a construir, los cuales brindan la posibilidad de compartir información generada por *Embedded Honeypot* hacia otros sistemas.

4.2. Diagrama de Clases

En la siguiente imagen se presenta un diagrama de clases reducido del framework[2]

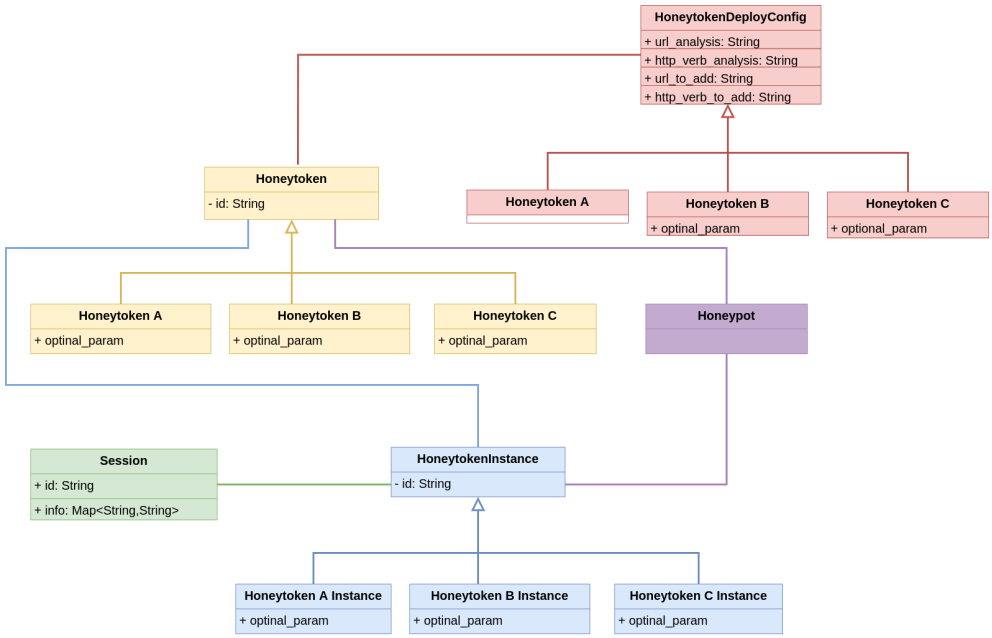

Figura 8: Diagrama de clases del framework

En este diagrama se pueden ver las entidades más importantes del framework: **Honeytoken**, **HoneytokenInstance**, **Session**, **HoneytokenDeployConfig** y **Honeypot**.

Se puede apreciar la entidad **Honeytoken**, la cuál representa los *datos estáticos definidos para los Honeytokens*, es decir, muestra la entidad que almacena los datos que brinda el administrador de la aplicación al momento de definir los Honeytokens a utilizarse. Esta entidad se define como una jerarquía, *con lo cual se brinda un mecanismo para que los diferentes desarrolladores que utilicen el framework puedan crear distintos tipos de Honeytokens*, brindado de esta forma la *posibilidad de extender esta arquitectura*. Es considerable aclarar que estos **Honeytoken** contienen atributos comunes a todos los tipos de Honeytokens que se puedan definir, pero cabe destacar que es posible definir atributos particulares para cada tipo de Honeytoken.

Los **HoneytokenInstance** persiguen el objetivo de *establecer el concepto de instancia de Honeytokens definidos para el framework*, es decir, *mostrar una entidad que tiene la responsabilidad de presentarse dentro de los mensajes HTTP intercambiados entre los clientes y las aplicaciones web, que a su vez es una instancia de un Honeytoken previamente definido en el framework por los administradores de la aplicación*. Esta idea ayuda a separar a la entidad presente dentro de las interacciones entre el cliente y las aplicaciones web con entidad que se define a partir de los datos brindados por los administradores de la aplicación web. También se puede apreciar que

[2]Se considera que el diagrama es reducido dado que el mismo no presenta los manejadores, controladores y datatypes involucrados, solamente se basa en las entidades necesarias para el funcionamiento.

esta entidad se presenta como una jerarquía, esto se debe a que está asociada directamente a la entidad **Honeytoken**, por lo que es necesario crear por cada tipo de **Honeytoken** con un tipo de **HoneytokenInstance**.

En el momento en que se introducen los **HoneytokenInstance** se presenta el problema de *identificar un Honeytoken dentro de un mensaje enviado por un cliente* y *reconocer en cuál mensaje debe ser enviado el Honeytoken*, para lo cual se crea la entidad **HoneytokenDeployConfig**. Esta entidad contiene los datos necesarios para que dado un mensaje HTTP se reconozca que Honeytokens analizar o agregar en el mismo. Nuevamente, es importante remarcar que esta entidad requiere de una jerarquía. Esto es debido a que cada tipo de **Honeytoken** puede requerir algún dato particular (además del dato de la URL y el verbo HTTP que se encuentran presentes para todos en el diagrama) para ser analizado o agregado a los mensajes intercambiados.

Session es una entidad particular, utilizada para *identificar usuarios maliciosos*. Es decir, una vez que se reconoce un abuso sobre un **HoneytokenInstance**, se *crea una sesión para darle seguimiento a las acciones del usuario que generó el abuso*. Cada **Session** contiene un identificador único para el framework y de esta forma se asocia la información de la manipulación de uno o varios **Honeytoken** con una **Session**.

Honeypot es la entidad principal que funciona como puerta de entrada a la ejecución del sistema. Esta entidad es la encargada de exponer los métodos que serán invocados para agregar Honeytokens en un response y detectar la manipulación en alguno de ellos.

4.3. Vista de operaciones

En el capítulo Diagrama de Clases se establece una visión sobre la información contenida dentro de cada entidad del framework. Una vez que se logra establecer los atributos de cada entidad, es necesario conocer cuáles son sus responsabilidades. Es decir, *cuáles son las operaciones que cada entidad ejecutará*.

A continuación, se muestra un diagrama que contiene algunas de las componentes del framework ya vistas en Diagrama de Despliegue que serán de utilidad para presentar las operaciones mas relevantes del framework:

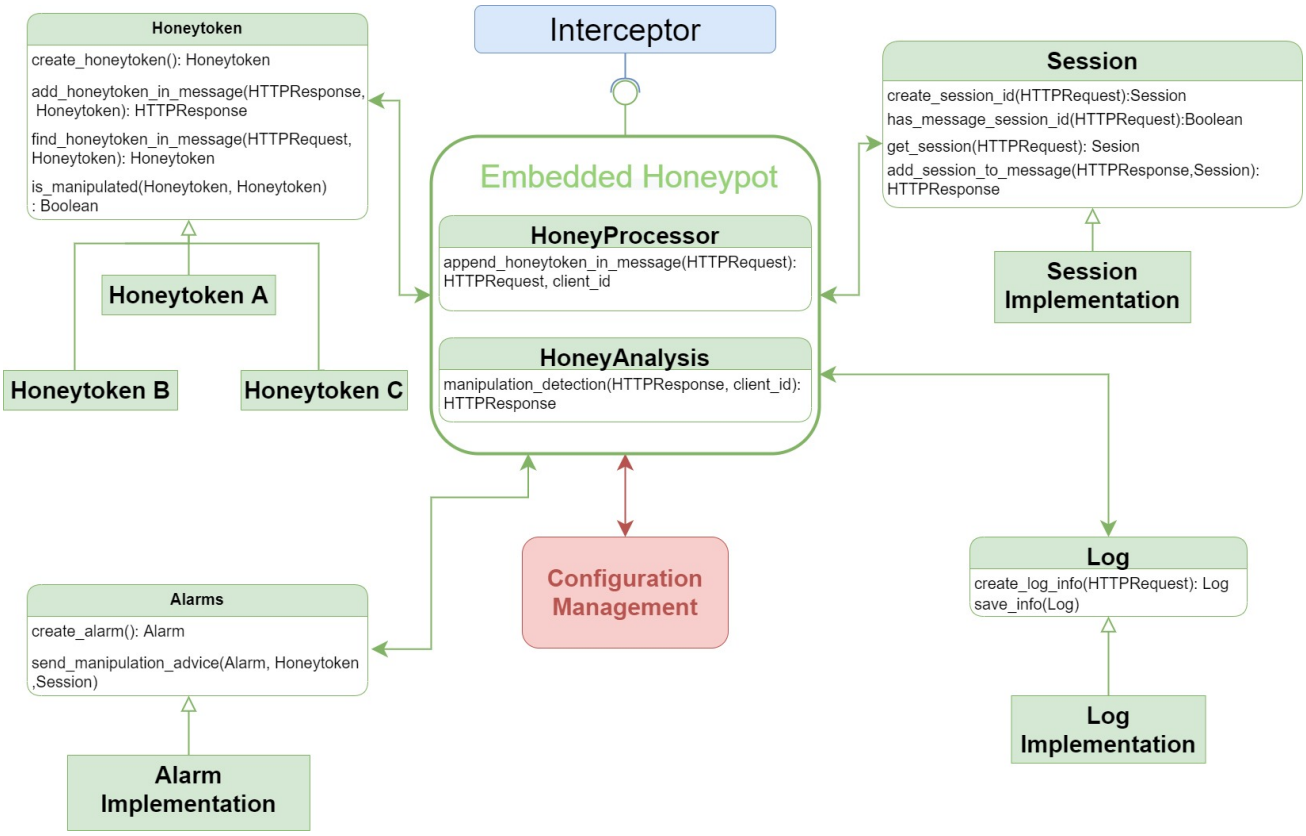

Figura 9: Diagrama con operaciones del framework

En el diagrama se puede apreciar nuevamente que el **Interceptor** se conecta a una interfaz habilitada por **Embedded Honeypot**. En dicha interfaz se encuentran las dos operaciones principales del desarrollo:

- **append_honeytoken_in_message**: Esta operación toma un mensaje HTTP response y le agrega los HoneytokenInstance que estén asociados a la URL de donde proviene la solicitud a la cual se le está respondiendo.
- **manipulation_detection**: Esta operación recibe un mensaje HTTP request y detecta si los *HoneytokenInstance* asociados a la URL dentro del mensaje han sido manipulados.

Las componentes **HTTP Processor** y **HTTP Analysis**, los cuales implementan las operaciones anteriormente mencionadas, requieren de otras componentes del framework para poder realizar la operativa asociada a cada operación. **HTTP Processor** se comunica con **Honeytoken** y **Session** para poder obtener información de cada una de estas entidades y así agregar los *HoneytokenInstance* a los mensajes HTTP response. A su vez, la componente **HTTP Analysis** requiere de las componentes **Honeytoken**, **Session**, **Alarms** y **Log**, para reconocer, almacenar información y dar aviso a los administradores al momento de detectar un abuso sobre un *HoneytokenInstance*.

Las componentes **Honeytoken**, **Session**, **Alarms** y **Log** son presentadas como **Interfaces**, es decir, *se requiere de una nueva componente que implemente su definición*. De esta forma, se permiten agregar nuevos mecanismos para realizar cualquiera de las operaciones que se definan en estas clases, logrando que los diferentes desarrolladores que se quieran involucrar a futuro en el proyecto puedan agregar nuevos tipos de Honeytokens, nuevas formas de aviso a los administradores, mecanismos de sesión o métodos de registro de información con facilidad.

Por otro lado, se presenta una comunicación con una componente llamada **Configuration Management**. La finalidad de dicha componente en este caso es disponer la información relacionada a la configuración general del framework y los datos de las entidades (incluyendo los datos de la clase *HoneytokenDeployConfig* presentada anteriormente).

Referencias

[1] Agustín Sanchez Federico Pernas and Nicolás Zeballos. Entregable: Estado del Arte. 2020.

[2] OWASP Foundation. Owasp top ten web application security risks | owasp. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project. [Accedido: 18-AGO-2020].

[3] Lance Spitzner. Honeytokens: The other honeypot. https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=74450cf5-2f11-48c5-8d92-4687f5978988&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments. [Accedido: 25-AGO-2020].

[4] Germán Escobar. El protocolo http. https://blog.makeitreal.camp/el-protocolo-http/. [Accedido: 29-AGO-2020].

[5] NeoAttack. ¿qué es un framework y para que sirve? - neo wiki | neoattack. https://neoattack.com/neowiki/framework/. [Accedido: 27-NOV-2020].

[6] Wikipedia. Diagrama de despliegue - wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Diagrama_de_despliegue. [Accedido: 28-JUN-2020].