Network Safe Box

Memoria trabajo fin de grado del Grado en Ingeniería Informática

Facultad de Informática

Universidad Complutense de Madrid

Curso 2015 - 2016

AUTORES

Rodrigo Arranz López

Rafael Delgado Meana

DIRECTORES

Jose Luis Vazquez-Poletti

José Manuel Velasco Cabo

UNIVERSIDAD COMPLUTENSE MADRID

AUTORIZACIÓN PARA LA DIFUSIÓN DEL TRABAJO FIN DE GRADO Y SU DEPÓSITO EN EL REPOSITORIO INSTITUCIONAL E-PRINTS COMPLUTENSE

Los abajo firmantes, alumno/s y tutor/es del Trabajo Fin de Grado (TFG) en el Grado en Ingeniería Informática de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el Trabajo Fin de Grado (TF) cuyos datos se detallan a continuación. Así mismo autorizan a la Universidad Complutense de Madrid a que sea depositado en acceso abierto en el repositorio institucional con el objeto de incrementar la difusión, uso e impacto del TFG en Internet y garantizar su preservación y acceso a largo plazo.

Periodo de embargo (opcional):

- ☐ 6 meses
- ☐ 12meses

TÍTULO del TFG: NetworkSafeBox

Curso académico: 2015 / 2016

Nombre del Alumno/s:
Rodrigo Arranz López
Rafael Delgado Meana

Tutor/es del TFG y departamento al que pertenece:
Jose Luis Vazquez-Poletti
José Manuel Velasco Cabo

Firma del alumno/s Firma del tutor/es

Resumen

En la actualidad estamos conectados a Internet de forma permanente, ya sea en casa o en el trabajo. Cualquier persona puede acceder a Internet de forma rápida y sencilla desde casi cualquier dispositivo electrónico para buscar información, jugar, ver contenido multimedia, etc. El problema es que los ciberdelincuentes se aprovechan de las personas que no se preocupan por su seguridad de la red. Los ciberdelincuentes suelen tener un amplio conocimiento en informática, pero la mayoría de la gente tiene pocos conocimientos sobre la seguridad informática. ¿Cómo puede esta gente defenderse de los ciberdelincuentes? Para resolver este problema estamos desarrollando el proyecto Network Safe Box. Network Safe Box añade una capa de seguridad a su conexión a Internet. Sin mucho conocimiento sobre seguridad informática un usuario puede controlar el tráfico entrante y saliente a través de una interfaz web en unos pocos clics. Network Safe Box crea una red de máquinas virtuales que controla las conexiones a Internet en base a las necesidades del usuario y protege contra el acceso malintencionado a su red doméstica o de trabajo.

Palabras clave

Seguridad informática, máquinas virtuales, ciberdelincuentes, red doméstica, red de trabajo, Virtualbox, PHP, Python.

Abstract

At present we are connected to the Internet permanently either at home or at work. Anyone can access the Internet quickly and easily from almost any electronic device to search for information, play games, watch multimedia content, etc. The problem is that cyber criminals are taking advantage of people who do not care about their network security. Cyber criminals usually have extensive computer knowledge, but most people have little knowledge about computer security. How can these people defend themselves from cyber criminals? To solve this problem we are developing the project NetworkSafeBox. NetworkSafeBox adds a layer of security to your Internet connection. Without much knowledge about computer security an user can control the incoming and outgoing traffic through a web interface in a few clicks. NetworkSafeBox creates a network of virtual machines that controls Internet connections based on user needs and protects against malicious access to your home or work network.

Keywords

Computer security, virtual machines, cybercriminals, home network, network, Virtualbox, PHP, Python.

Índice

1. Introducción

En este capítulo abordaremos la historia y las motivaciones para realizar este proyecto.

1.1. Antecedentes

Hoy en día podemos afirmar que no podemos vivir sin Internet. Internet ha llegado a nuestras vidas para quedarse para siempre, y cada vez más elementos de nuestra vida cotidiana están conectados a Internet, como ordenadores, móviles, televisores, *tablets*, electrodomésticos, vehículos de transporte, ropa, etc, etc. La lista es interminable e irá aumentando según vaya avanzando la tecnología. Además, con todos estos dispositivos accedemos a cuentas bancarias, realizamos declaraciones de la renta, hacemos compras, realizamos ventas, montamos negocios, etc, etc. Es decir, estamos constantemente enviando y recibiendo información sobre nosotros mismos y sobre quien nos rodea constantemente a través de Internet.

Todos los dispositivos antes mencionados se han popularizado en los últimos años gracias a su facilidad de uso. Antes, los que tenían un ordenador y se conectaban a Internet solían ser personas con conocimientos informáticos ya que no era tan simple como encender el ordenador y navegar. Hoy en día puedes ver como un niño aprende a manejar una *tablet* antes que aprender a sumar o como los abuelos se atreven a comprarse un móvil con acceso a Internet para mandar mensajes a sus seres queridos. Es decir, existe un abanico de edad muy grande que utilizan Internet en su vida diaria, puede que sin ser conscientes, pero eso lo abordaremos más adelante.

Debido a esa gran demanda de dispositivos, las empresas se han preocupado de que sus productos sean fáciles de usar para poder venderlos al mayor número de personas posibles. El problema es que se han centrado mucho en este sentido y han dejado en un segundo plano la seguridad de los mismos. ¿Por qué? Porque la seguridad al fin y al cabo consiste en ir añadiendo capas de protección. Esto conlleva a que la usabilidad se resienta. Debería existir un equilibrio entre seguridad y

usabilidad, pero lo cierto es que la balanza se inclina, la mayoría de las veces, hacia la usabilidad.

¿Esto quiere decir que ningún dispositivo tiene medidas de seguridad? No, la mayoría de los dispositivos vienen con muchas medidas de seguridad, lo que pasa es que muchas de ellas no vienen activadas por defecto y se deja esta responsabilidad en manos de los usuarios. Como se ha mencionado antes, cualquier persona tiene acceso a estos dispositivos y los usa en su vida diaria. El problema es que la gran mayoría de los usuarios tienen entre pocos conocimientos informáticos y ninguno. Por lo cual, utilizan dicho dispositivos con las configuraciones por defecto sin tener la más mínima idea de a lo que se exponen.

Este desconocimiento de los usuarios es aprovechado por ciberdelincuentes y agencias de espionaje (gubernamentales o no) para sacar provecho de ello. Al contrario que los usuarios, estos disponen de altos conocimientos en seguridad informática así como el material y/o la financiación necesaria para lograr sus fines. Tanto unos como otros se aprovechan de esas configuraciones de seguridad por defecto, que suelen ser bastante livianas y fáciles de evadir, o de las continuas vulnerabilidades que aparecen tanto en los *softwares* como en los *hardwares* de los distintos dispositivos que usamos habitualmente para obtener información de los usuarios o incluso para acceder directamente al dispositivo y realizar acciones desde el mismo sin que el usuario se dé cuenta ni sea consciente. Esto puede provocar el robo de datos confidenciales, robos de cuentas bancarias, robos de cuentas sociales, chantajes, suplantaciones de identidad, etc.

Debido a esta “guerra de seguridad” desigual entre usuarios y ciberdelincuentes, se decidió desarrollar un sistema de seguridad que intentase equilibrar usabilidad con seguridad. Este sistema aplica una capa de seguridad extra a nuestra red doméstica o de trabajo con una interfaz simple para que esté al alcance de cualquiera el poder proteger sus conexiones a Internet sin tener unos amplios conocimientos en seguridad informática.

1.2. Objetivos y justificación

Este proyecto tiene como objetivo principal el desarrollo de un sistema de seguridad de red que busque el equilibrio entre usabilidad y seguridad. Es decir, que cualquier usuario sea capaz de utilizarlo independientemente de sus conocimientos en seguridad informática, para así poder añadir una capa de seguridad extra a su red doméstica o de trabajo sin muchos quebraderos de cabeza. Este sistema hará de filtro entre *Internet* y la red local del usuario.

Para conseguir este objetivo podemos enumerar tres subobjetivos importantes:

El primero es desarrollar una interfaz web sencilla que permita a los usuarios configurar de manera intuitiva y en pocos pasos su sistema de seguridad. De esta manera el usuario podrá tener el control de su red y saber qué es lo que está pasando en ella.

El segundo sub-objetivo es que el sistema sea adaptable a las necesidades de cualquier usuario y a los constantes cambios que se dan en el campo de la seguridad informática. Para ello el sistema creará una red de máquinas virtuales por las que se repartirá el tráfico con destino a *Internet* y se filtrará dicho tráfico según las pautas introducidas por el usuario. Se ha elegido un entorno virtualizado porque así el sistema es fácilmente escalable y adaptable a las necesidades del usuario, gracias a la facilidad que nos ofrece un entorno virtualizado para crear, modificar o eliminar máquinas.

Por último, hay que unir la interfaz web con la creación del entorno virtualizado para que por una parte se recoja la configuración de usuario y se cree el entorno virtualizado y por otra el entorno virtualizado informe de su estado al usuario a través de la interfaz web.

Analizando los objetivos, se puede apreciar que el proyecto abarca ampliamente gran parte del programa estudiado a lo largo de la carrera. Con el desarrollo de este sistema se ponen en práctica conocimientos de programación y desarrollo web (FP, TP, EDA, AW, SC), bases de datos (BD, ABD), redes y sistemas operativos (RED,

SO, ASOR), seguridad informática y virtualización de sistemas (RyS), organización y planificación de desarrollo *software* (IS) y diseño de sistemas interactivos (DSI), entre otros.

Las expectativas de uso del sistema son altas, ya que cada vez más las personas se están dando cuenta de que la seguridad informática es casi tan importante como la seguridad física. Por ello, este sistema está orientado a uso doméstico como para pequeñas y medianas empresas, que quieran proteger a sus seres queridos como a sus negocios de las amenazas que pueda haber en *Internet* respectivamente.

El sistema ha sido bautizado con el nombre de **Network Safe Box**.

1.3. Plan de trabajo

El proyecto se llevó a cabo entre los meses de octubre de 2015 hasta febrero de 2016, abarcando las fases de especificación de requisitos y planificación, diseño e implementación, pruebas y redacción de la memoria.

1.3.1. Actores involucrados en el proyecto

Jose Luis Vazquez-Poletti, profesor de Redes y Seguridad del departamento de Arquitectura de Computadores y Automática, es el director del proyecto.

José Manuel Velasco Cabo, codirector del proyecto, del Departamento de Arquitectura de Computadores y Automática.

El proyecto se lleva a cabo por dos alumnos del grado en Ingeniería Informática, **Rodrigo Arranz López y Rafael Delgado Meana**.

1.3.2. Definición de requisitos y fases del proyecto

Debido a que los alumnos querían presentar el trabajo de fin de grado en la convocatoria extraordinaria de febrero se tuvo que compactar los tiempos de las fases del proyecto. En vez de disponer de nueves meses (Octubre-Junio) que es lo normal se tenían cincos meses (Octubre-Febrero). Para ello se mantuvieron reuniones regularmente entre los integrantes del grupo y de manera periódica con el director del trabajo para llevar un control sobre los avances y aconsejar en lo que fuera necesario.

En las primeras reuniones se especificaron los requisitos mínimos del sistema que permitiese cumplir los objetivos establecidos. Además se establecieron varios objetivos opcionales que se desarrollarían en función de cómo fuera el proyecto así como del tiempo disponible. Los requisitos iniciales eran crear un sistema capaz de montar el entorno virtual mediante una interfaz web. Este contaría con unas máquinas por defecto con varias opciones de configuración en cada una.

En reuniones posteriores, se decidió que había que dar al usuario la opción de añadir sus propias máquinas virtuales ya que para un usuario avanzado podría serle de utilidad poder agregar sus propias máquinas virtuales así como añadir nuevas configuraciones a las máquinas por defecto como a las creadas por el usuario.

De la misma manera, se llegó a la conclusión de que había que incorporar un apartado a la interfaz web en el que se viera información sobre el estado de las máquinas virtuales así como de los posibles ataques que se habían recibido.

Tras la especificación de los requisitos se desarrolló el plan de trabajo y las fases que abarcaría y se decidieron las tecnologías y entornos que se utilizarían.

La primera fase de desarrollo del sistema fue la **fase de diseño**. Esta fase abarca el diseño de la interfaz web (posteriormente fue modificado para que resultase más sencillo su uso), el diseño de la red virtual y la unión de la interfaz web con el creador de las redes virtuales.

Tras la fase de diseño se inició la **fase de implementación**. Esta fase es la más larga y con mayor carga de trabajo. Durante esta etapa se desarrolla la interfaz web, el creador de las redes virtuales así como la unión de las mismas. Tanto la interfaz web como el creador de las redes virtuales tuvieron varios ciclos de mejora para corregir posibles fallos, mejorar el rendimiento o mejorar el estilo y/o estética.

La última fase sería la **fase de pruebas y redacción de la memoria**. En esta fase se sometió al sistema a varias pruebas para detectar los fallos que se pudieran dar en ejecución y ultimar la estética. Paralelamente, se redacta el borrador de la memoria del proyecto que desemboca en la memoria final tras su corrección gracias a los apuntes del director del proyecto.

1.4 Mapa conceptual.

La aplicación se describe en los puntos 3,4 y 5, se ha divididos en estos tres grupos diferenciando entre, primero, la interfaz web por un lado, el punto 3, donde se describe la estructura que se ha implementado como se ha implementado. Por otro lado la creación de entornos virtuales, punto 4, donde se detalla la metodología de creación del entorno virtual. Y por último, la estructura general de la aplicación, punto 5, donde se detalla la aplicación en global, haciendo un gran hincapié en las diferentes pantallas que componen la aplicación, sus funcionalidades y modos de uso.

2. Tecnologías empleadas

Antes de comenzar a explicar el desarrollo del proyecto, es interesante conocer las herramientas (soporte software) que se han utilizado para poder llevarlo a cabo. Este proyecto no ha necesitado de ningún soporte hardware especial. Se ha trabajado sobre ordenadores personales con las tecnologías descritas en este apartado. Sólo en la fase final se ha utilizado un portátil personal para montar el sistema y para así poder emular lo que sería una **Network Safe Box** ya operativa.

2.1. Lenguajes, librerías y otras tecnologías de desarrollo

Como ya se ha mencionado anteriormente el sistema consta de dos partes, la interfaz web y el creador del entorno virtual.

La parte de interfaz web la podemos dividir en 3 subcategorías, la primera la interfaz, desarrollada en **HTML5** y **javascript** y usando un framework de diseño CSS, **Foundation**, la segunda parte, la API(anexo II) desarrollada en **PHP** y haciendo uso de la librería **nuSOAP**, esta API se conecta a la tercera parte, la base de datos **mySQL**(anexo III), haciendo que todos los cambios que hagamos sean persistentes, más adelante hablaremos de la conexión de estas tres partes y la conexión con el creador de entornos virtuales.

En cuanto al creador del entorno virtual se ha utilizado principalmente el lenguaje de programación **Python** (en su versión 2.7 incluido en la mayoría de distribuciones Linux. Se utilizó dicho lenguaje para crear un *script* que recogiese la configuración introducida en la interfaz web y crear el fichero de configuración que luego se pasaría a Vagrant (mirar sección 2.2). Además de crear dicho fichero ejecuta los comandos necesarios en Vagrant para iniciar el proceso de creación del entorno virtual.

Para los distintos *scripts* de configuración de las máquinas virtuales se ha utilizado **Bash**, ya que es el lenguaje que usa el intérprete de Linux. Para dichos *scripts* se utilizan los comandos de **Iptables, Route, Ifconfig** y demás comandos relacionados

con la configuración y gestión de los recursos de red incluidos en la mayoría de las distribuciones Linux.

2.2 Herramientas externas

Con herramientas externas nos referimos al software que se ha utilizado, no para desarrollar la aplicación, sino como soporte o complemento de la misma.

El *software* utilizado para virtualizar ha sido **Oracle VM VirtualBox**. Se eligió este *software* por su facilidad de uso y porque es multiplataforma. Además, los componentes ya sabían utilizarlo porque es el que se usa durante el Grado de Ingeniería Informática para montar entornos en los que hacer las prácticas en varias asignaturas.

Adicionalmente se integró la herramienta **Vagrant**. Esta es una herramienta para la creación y configuración de entornos de desarrollo virtualizados. Aunque Vagrant se ha desarrollado en Ruby se puede usar en multitud de proyectos escritos en otros lenguajes, tales como PHP, Python, Java, C# y JavaScript. Además soporta varios proveedores de virtualización como VirtualBox, Amazon EC2, LXC, DigitalOcean, etc. Su uso es sencillo ya que consiste en crear un fichero de configuración del entorno virtual a crear y al pasárselo a Vagrant este se encarga de toda la configuración. Además, una vez creado el entorno virtual te da muchas opciones de interacción con las máquinas virtuales.

Además, se ha hecho uso de **Maltrail**. Es una herramienta de código abierto, hecha en Python, destinada a analizar el tráfico de red con el fin de detectar y registrar posibles amenazas. Maltrail destaca por la categorización de las amenazas, las diversas fuentes de las que obtiene información sobre direcciones IP con mala reputación, la inteligencia para identificar actividades sospechosas, la identificación de malware que intenta hacer conexiones reversas, así como la presentación de toda esta información en un panel de detección web.

2.3. Entornos de trabajo

Se debe diferenciar en este apartado las plataformas o programas que se han utilizado para desarrollar y depurar el código del sistema.

En cuanto a los servidores se refiere, la mayor parte del trabajo de desarrollo lo se ha llevado a cabo en un servidor local, facilitado por la herramienta de software libre **XAMPP**, que proporciona una base de datos MySQL (a través de PhpMyAdmin), el servidor web Apache y los intérpretes para lenguajes de script: PHP y Javascript.

Para la edición de código se ha utilizado el editor de textos **Sublime Text**. Es un editor que aporta muchas características útiles a la hora de programar o editar código. El editor está cargado de funcionalidades útiles y cómodas desde el punto de la usabilidad y eficiencia.

2.4. Control de versiones

Las últimas tecnologías que quedan por mencionar son los controladores de versiones. Nos hemos ayudado de tres plataformas para controlar las versiones de la aplicación.

En un principio la intención era utilizar únicamente la plataforma de desarrollo **GitHub**, sin embargo, debido a el tamaño de las máquinas virtuales recurrimos a otras plataformas complementarias como **Dropbox** o **Mega**. En GitHub almacenamos el código del sistema mientras que las máquinas virtuales eran guardadas en Dropbox en la parte inicial, pero luego tuvimos que migrar a Mega ya que Dropbox solo dispone de 2GB de almacenamiento gratuito en la nube mientras que Mega ofrece 50GB sin ningún coste.

2.5. Tecnologías descartadas

Separamos este apartado en dos, el primero en tecnologías web y el segundo tecnologías para el despliegue y configuración de las máquinas.

2.6. Tecnologías web

Django es un framework de diseño web implementado en python, se descartó por ser un framework demasiado completo para los requerimientos que tenemos en el proyecto, por lo que la curva de aprendizaje nos retrasaría.

Phaser es un motor de videojuegos, si iba a utilizar para crear un mapa interactivo con el que el usuario pudiera crear un mapa de forma muy intuitiva, se descartó por no ser escalable.

Bootstrap es un framework de diseño CSS, se descartó por la elección de otro framework de diseño.

2.7. Despliegue y configuración de máquinas virtuales

Puppet, **Fabric** y **Chef** son aplicaciones que permiten configurar las máquinas virtuales desde la máquina física, se descartó porque Vagrant nos permite hacerlo.

3. Interfaz web

Antes de conocer la estructura general del proyecto, debemos explicar cómo y porque hemos realizado la interfaz web, que detallamos a continuación.

3.1. Diseño

El diseño de la interfaz web se ha realizado siguiendo el framework de diseño CSS **Foundation,** el cual nos permite crear páginas web de forma rápida y que permite al usuario una experiencia de uso agradable. Toda la lógica de la página web está escrita en javascript, ya que sería un método poco seguro tener datos de interés en limpio en el código, se creó una API en php, la cual nos permite de forma inocua a los usuarios acceder a datos de la base de datos operar con ellos y devolver al usuario los datos requeridos, esta API está escrita en PHP y se encuentra alojada en la máquina física, en el servidor Apache, de esta forma tenemos una aplicación más segura.

Por último, tenemos la base de datos mySQL, en la que almacenamos todos los datos necesarios para el despliegue de la aplicación, tales como máquinas base, scripts, máquinas configuradas… (anexo III), a esta base de datos accedemos solamente a través de la API.

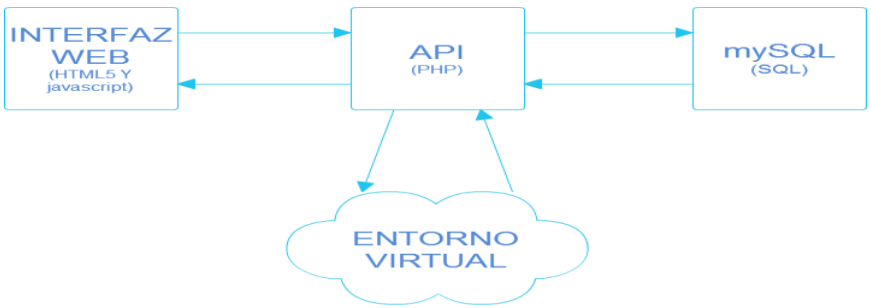

Figura 3.1: Estructura

Como vemos en la figura 3.1 la API, cuando decidamos desplegar, se conectará con el entorno virtual para comenzar a desplegar la configuración decidida en la interfaz web.

3.2. Interconexión entre elementos

En la conexión entre los tres elementos de interfaz web se pueden diferenciar dos conexiones diferentes, la de la interfaz con la API y la conexión de la API con la base de datos, ya que en ningún momento la interfaz hace ninguna petición a la base de datos directamente ni conecta con el entorno virtual directamente, por seguridad.

La primera conexión es javascript-API, se hace mediante el protocolo SOAP(Simple Object Access Protocol)el cual nos permite hacer llamadas a un servidor desconocido para el cliente y obtener datos de forma segura y eficiente. De esta forma no tenemos datos sensibles “hardcodeados” en el código de la aplicación web, tan solo una dirección de acceso al servidor, lo mínimo posible.

La segunda conexión se realiza entre la API y mySQL, dentro de la API, escrita en PHP, tenemos datos de acceso a la base de datos, tales como ubicación, base de datos usuario y contraseña, este usuario tiene los permisos mínimos necesarios para un correcto funcionamiento de la aplicación, esta comunicación se realiza gracias a la extensión de PHP **mysqli**.

4. Creación de entornos virtuales

4.1. Arquitectura del entorno virtual

El entorno virtual se ha definido de la siguiente manera. Toda red virtual creada por el usuario tendrá tres máquinas virtuales por defecto. Dos máquinas virtuales de entrada y salida más una máquina que acepte el tráfico sobrante o que no se filtraría.

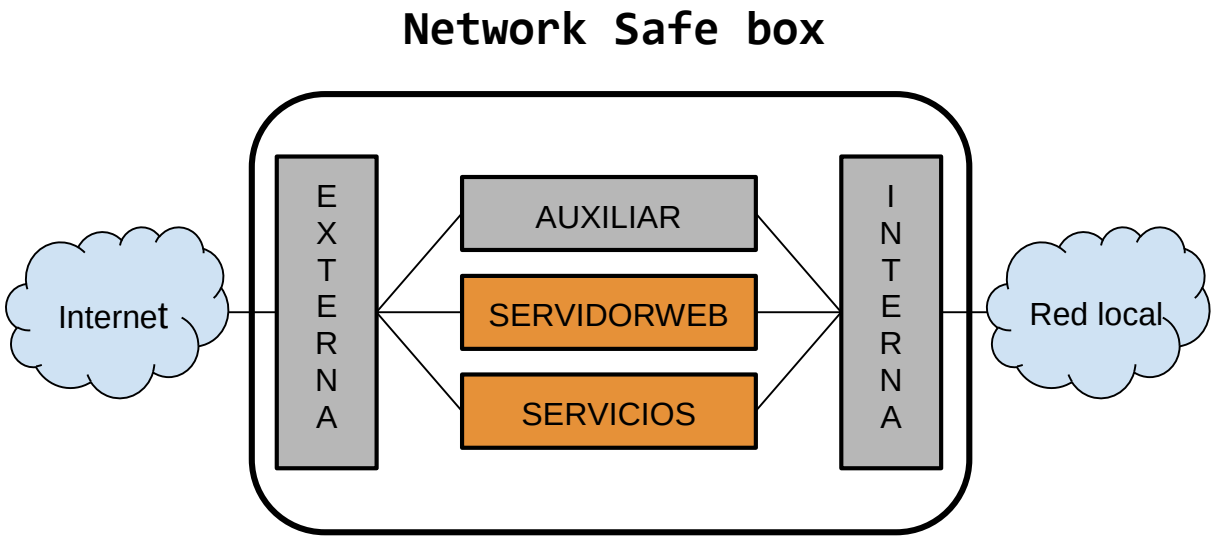

Figura 4.1. Arquitectura interna de la red virtual

Como podemos observar en el diagrama (Figura 4.1.) las máquinas en Externa, Interna y Auxiliar, resaltadas en gris, son fijas independientemente de la configuración que introduzca el usuario, mientras que las máquinas ServidorWeb y Servicios, resaltadas en naranja, han sido añadidas por el usuario.

4.2. Tipos de máquinas virtuales

A continuación se detallan las distintas máquinas virtuales creadas, tanto las fijas como las que el usuario puede añadir según sus necesidades.

4.2.1. Máquina virtual Externa

La máquina virtual Externa es la que da salida a Internet. Tiene una interfaz de red conectada a la interfaz física de la máquina anfitriona que es la que está conectada a Internet. Además, tiene otra interfaz que se conecta con la máquina Auxiliar (se detalla más adelante su función). El número de interfaces restantes depende de cuantas máquinas virtuales cree el usuario, habiendo una interfaz por cada máquina adicional introducida por el usuario. La función de esta máquina es recoger todo el tráfico que reciba por todas las interfaces y encaminarlo hacia Internet para así habilitar la conexión de la red local. Por último, esta máquina es la que ejecuta Maltrail, ya que por ella pasa todo el tráfico y así es capaz de detectar los posibles ataques realizados a la red local.

4.2.2 Máquina virtual Interna

La máquina virtual Interna es la que está conectada a la red interna doméstica o de trabajo. Tiene una interfaz de red conectada a la interfaz física de la máquina anfitriona que es la que está conectada a la red local. Esta interfaz cuenta con un servidor DHCP para que cuando se conecte un dispositivo este reciba una dirección IP directamente. Además, tiene otra interfaz que se conecta con la máquina Auxiliar (se detalla más adelante su función). El número de interfaces restantes depende de cuantas máquinas virtuales cree el usuario, habiendo una interfaz por cada máquina adicional introducida por el usuario. La función de esta máquina es la de repartir el tráfico entrante, desde la red local, por las distintas máquinas en función de los puertos de origen y de destino de las conexiones.

4.2.3. Máquina virtual Auxiliar

La máquina virtual Auxiliar se encuentra entre la máquina virtual Externa e Interna. Dicha máquina cuenta con dos interfaces, una de entrada conectada a la máquina Interna, por la que recibe el tráfico, y otra de salida que se conecta a la máquina

Externa. La función de dicha máquina es aceptar el tráfico restante que el usuario no quiere filtrar.

4.2.4. Máquina virtual ServidorWeb

La máquina virtual ServidorWeb es una máquina que el usuario puede añadir a su red virtual. Dicha máquina viene configurada por defecto para analizar el tráfico con origen o destino a los puertos 80 y 443, correspondientes a los protocolos HTTP y HTTPS respectivamente. En esta primera versión cuenta con varios *scripts* de bloqueo de páginas web conocidas. Esta máquina pertenece a las creadas por defecto para que el usuario pueda añadir a la red virtual o modificarla según sus necesidades.

4.2.5. Máquina virtual Servicios

La máquina virtual Servicios una máquina que el usuario puede añadir a su red virtual. Dicha máquina viene configurada por defecto para analizar el tráfico con origen o destino a los puertos 21, 22 y 23, correspondientes a los protocolos FTP, SSH y Telnet respectivamente. En esta primera versión el usuario cuenta con la posibilidad de permitir o bloquear el acceso a la red local a través de estos protocolos. Esta máquina pertenece a las creadas por defecto para que el usuario pueda añadir a la red virtual o modificarla según sus necesidades.

4.2.6. Máquina virtual Genérica

Esta máquina virtual es una máquina base para que el usuario, creando sus propios *scripts* de configuración pueda crear sus propias máquinas virtuales en función de sus necesidades. Esta máquina es la base con la que se han creado las máquinas virtuales ServidorWeb y Servicios, modificando su configuración para adaptarlas a los protocolos a filtrar.

4.3. Servidor de información del estado de la máquina virtual

Todas las máquinas virtuales ejecutan un mini servidor web al que al hacer una petición *GET* devuelve el porcentaje de uso de la CPU, de la RAM y del disco duro en ese instante. Esta información se muestra al usuario en la pestaña “Información” de la interfaz web.

4.4. Proceso de creación del entorno virtual

El proceso de creación del entorno virtual consta de tres fases: obtención de la configuración introducida por el usuario, creación de los *scripts* de configuración que se ejecutarán en cada una de las máquinas virtuales, creación del fichero de configuración de Vagrant y ejecución de Vagrant con la configuración creada en el fichero de configuración.

El proceso de creación se detalla a continuación. Una vez que el usuario pulsa el botón “Iniciar” en la pestaña “Mapa” de la interfaz web se ejecuta un PHP que crea un fichero de texto con la configuración de la red virtual. Este fichero consta de una línea por cada máquina virtual que se encuentra entre la máquina virtual Externa e Interna.

Una vez creados los *scripts* de configuración de cada máquina virtual y el fichero de texto con la red virtual a crear se ejecuta un *script* llamado *builder.py* escrito en Python que crea el fichero de configuración de Vagrant con todos los comandos necesarios para crear y configurar la red en función de los datos introducidos por el usuario. Además, crea un *script* de configuración para cada máquina virtual con todos los comandos que se deben ejecutar una vez iniciadas las máquinas virtuales. Estos *scripts* de configuración se crean concatenando unos *scripts* creados por defecto.

Por último, una vez creados todos los ficheros y *scripts* de configuración se ejecuta Vagrant. Esta acción también la realiza el *script builder.py*.

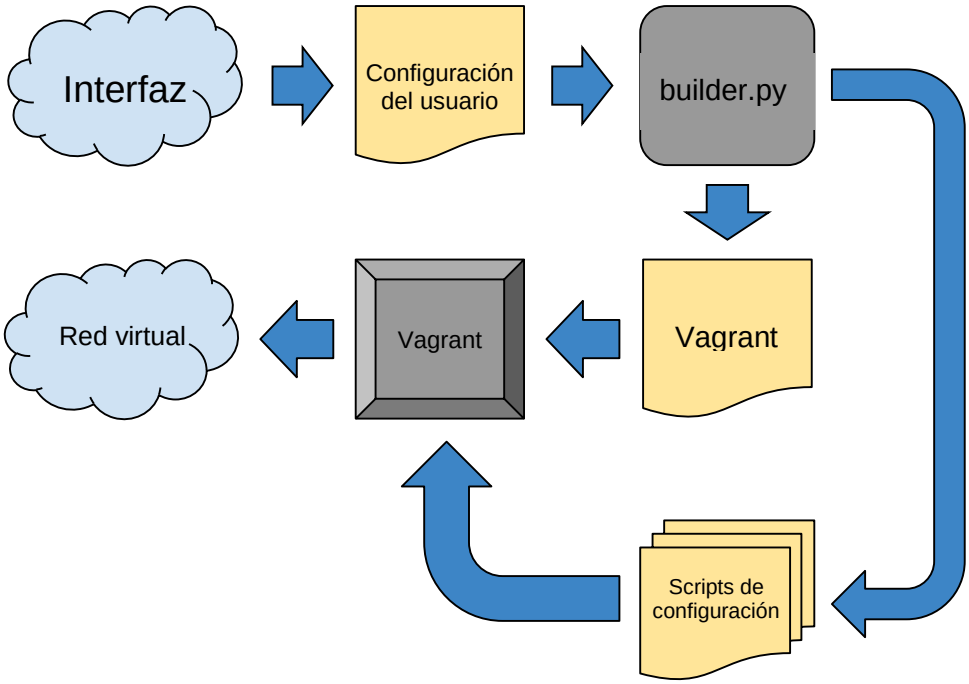

Figura 4.2. Proceso de creación del entorno virtual

5. Network Safe Box

5.1. Estructura del proyecto

En este apartado vamos a describir, en primera instancia la estructura global del sitio web, y posteriormente, la estructura más específica de cada una de las pantallas de control de la aplicación, dentro de estos apartados se describirá de forma detallada el comportamiento de cada uno de los componentes de estas pantallas.

5.1.1. Mapa del sitio

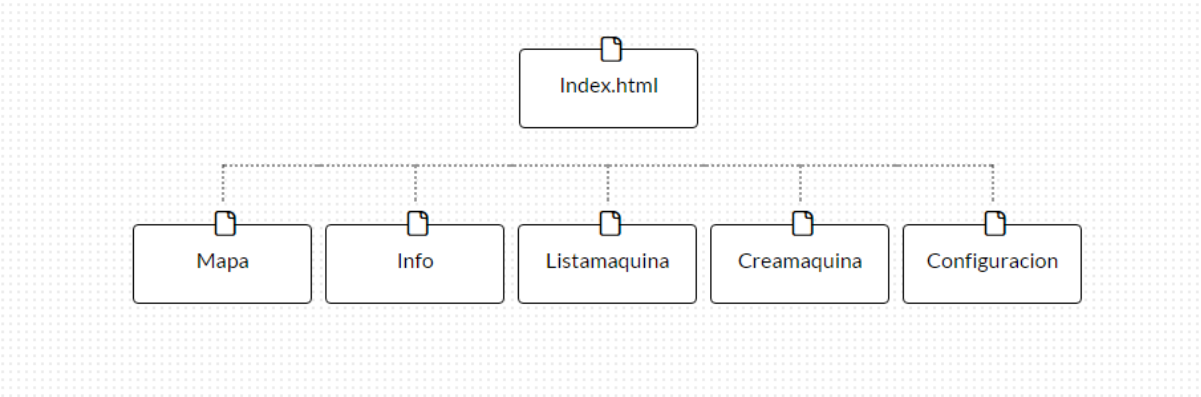

Figura 5.1.1: Mapa del sitio

El sitio está estructurado de tal forma que desde cualquier página tenemos acceso al resto de páginas. Y es accesible desde el menú que se encuentra a la izquierda en todas las pantallas, a excepción de la página de *login*, que en este caso se trata en una pantalla de presentación.

Figura 5.1.2: Menú

Desde este menú se accede a todas las pantallas.

5.2. Interfaz web

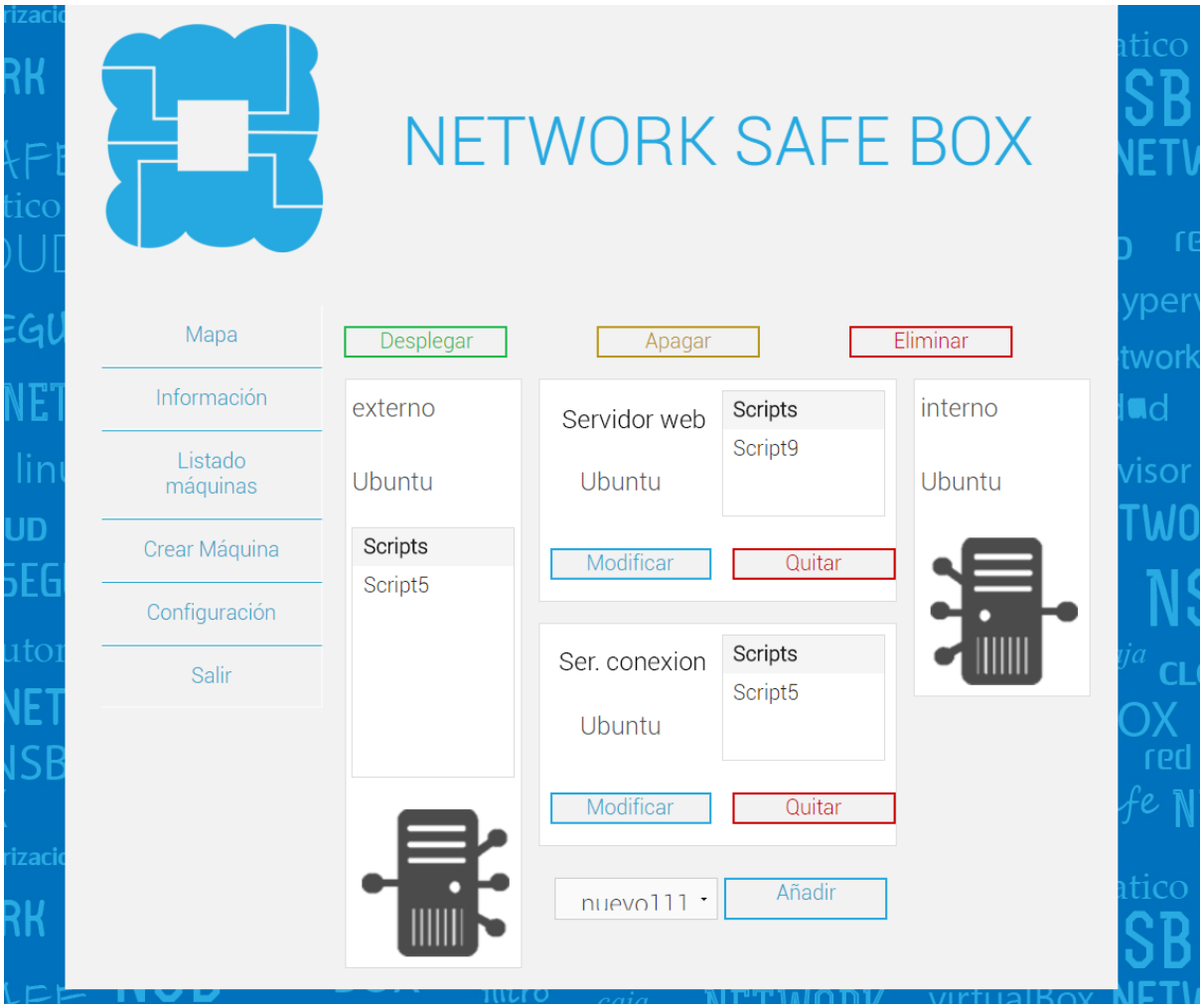

Figura 5.2: Visión general

La imagen 5.2 se corresponde con la vista general de la aplicación. En la cabecera podemos ver el **logotipo**, **nombre** de la aplicación, y a la izquierda, el **menú** del descrito en la imagen 5.1.2, estos tres elementos permanecen fijos en todas las plantillas de la aplicación, que detallamos a continuación.

5.2.1. Pestaña Mapa

Al acceder a la aplicación o al pulsar sobre el menú en “Mapa” visualizamos la siguiente información.

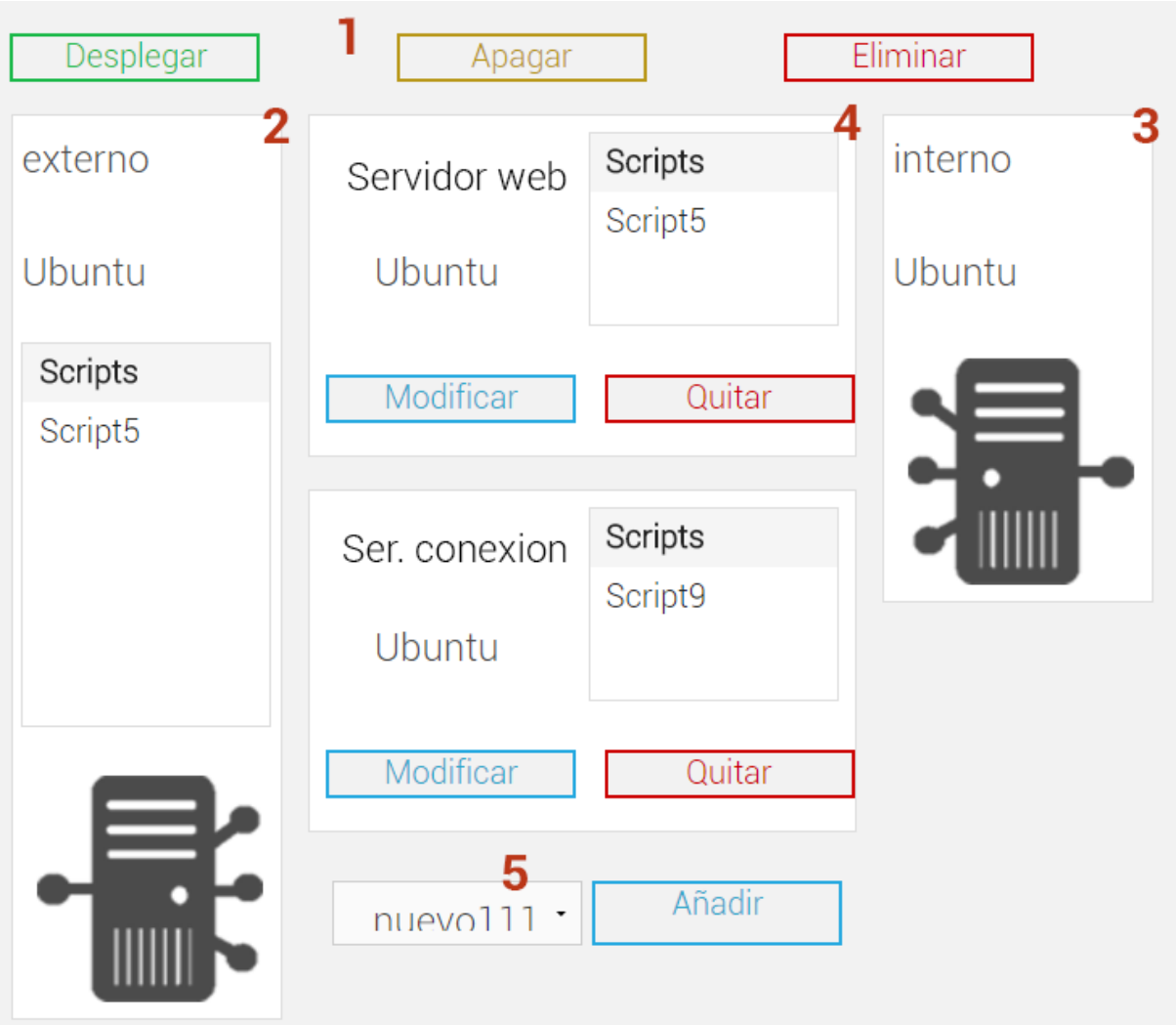

Figura 5.2.1: Mapa

1 - Los 3 botones de acción sobre el despliegue. En verde, “Desplegar” que al pulsarse se arranca el despliegue y empieza todo el proceso de creación, si fuera necesario, de arranque y configuración de las máquinas virtuales.
En naranja, “Apagar” que al pulsarlo conseguimos apagar las máquinas virtuales, pero no eliminarlas, por lo que arrancar un despliegue de nuevo, se haría de una forma más rápida, ya que, se mantiene la configuración de las mismas. Por último, en rojo,

“Eliminar” que, apaga las máquinas virtuales y además las elimina, dejado el despliegue limpio de máquinas.

2 - El cuadro de router externo, siempre fijo e inalterable.

3 - El cuadro de router interno, como el router externo, fijo e inalterable.

4 - Cuadro donde aparecen todas las máquinas añadidas al despliegue. Desde cada máquina se puede acceder a ésta, y modificarla, para adaptarla a nuestras necesidades, o eliminarla del despliegue.

5 - Combo box para añadir nuevas máquinas al despliegue, hasta un máximo de 5. Cada máquina que se añade, se añade también a la base de datos, por lo que todos los cambios quedan reflejados.

5.2.2. Pestaña Información

Al acceder desde el menú, y siempre que esté un despliegue en ejecución, nos aparecerá la siguiente información.

Figura 5.3: Información

1 - Al pulsar sobre este botón, se nos abre un modal, con la información recogida con el software MalTrail.

2 - En El siguiente cuadro tenemos la información de estado de uso de CPU, RAM y HDD de las máquinas levantadas en el despliegue.

5.2.3. Pestaña Lista de máquinas

Al pulsar sobre “lista maquinas” en el menú, accedemos a una tabla con todas las máquinas creadas y que pueden ser añadidas al despliegue.

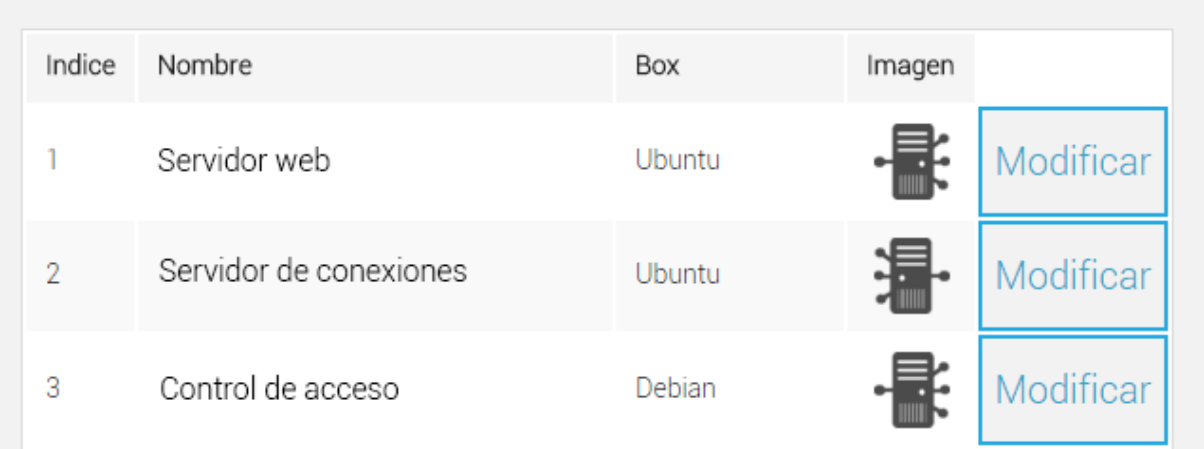

Figura 5.4: Lista de máquinas

En esta lista aparecen todas las máquinas creadas, y podemos acceder a ellas para modificarlas.

5.2.4. Pestaña Crea máquina

Al pulsar sobre “crear máquina” accedemos a la plantilla para crear nuevas máquinas que estén disponibles en el despliegue a partir de los boxes disponibles.

Figura 5.5: Crear máquina

1 - Campo de texto para introducir un nombre a la máquina para que sea identificada del resto.

2 - Combo box para seleccionar el Box que debe usar.

3 - Selecciona la imagen para poder visualizarlo de forma más sencilla, por defecto es la imagen que aparece justo debajo.

4 - Una tabla en la que se seleccionan los scripts que se deben ejecutar al desplegar la máquina.

5 - Al pulsar guarda la máquina en base de datos.
Esta misma pantalla se abre al modificar una máquina ya existente, la única diferencia se observa al acceder, se encuentran los datos cargados de la máquina a modificar.

5.2.5. Pestaña de Configuración

Por último, y no por ello menos importante, la pantalla de configuración, la explicación se va a dividir en pequeños fragmentos dada la extensión de la misma.

1- Puerta de enlace, el dato más necesario para el correcto funcionamiento del proyecto y, si no existiera dato ahí, no se podría desplegar con éxito la aplicación.

Figura 5.6.1: Puerta de enlace

2 - Tenemos una tabla con todos los scripts disponibles y podemos eliminarlos.

Figura 5.6.2: Gestión de scripts

3 - Podemos crear scripts nuevos, solo debemos introducir un nombre, buscar el script que queremos agregar a la base de datos, seleccionar los boxes para los que estará disponible este script y por último, pulsar en “guardar”.

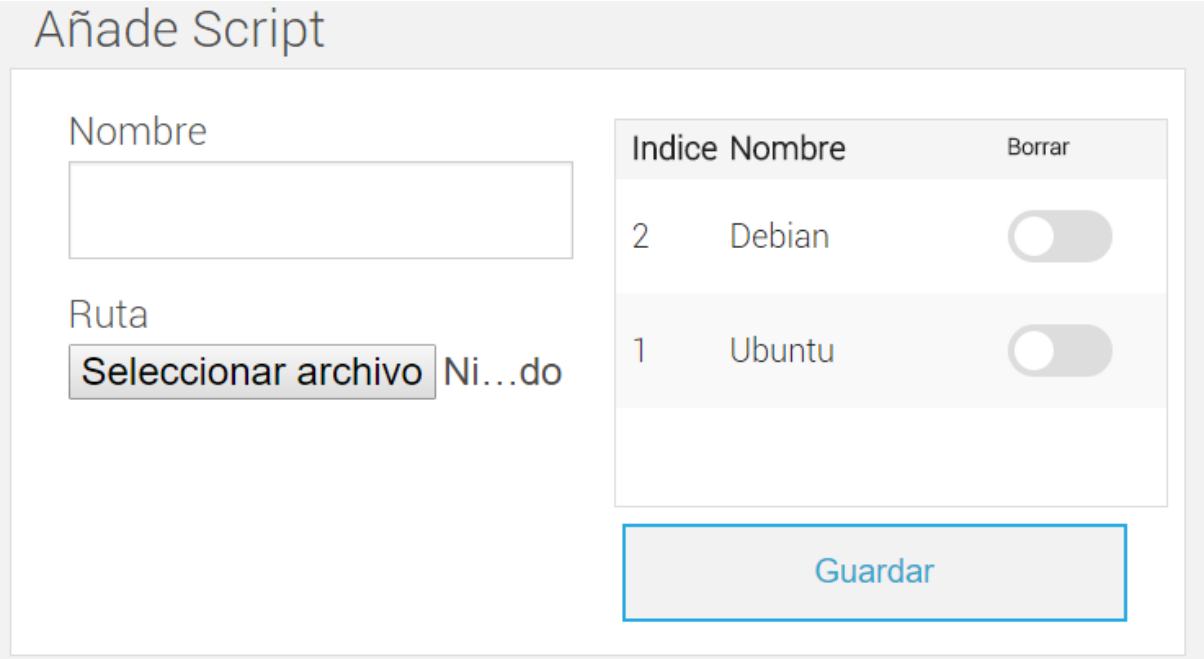

Figura 5.6.3: Crear script

4 - Por último podemos añadir nuevos Box, este es un aspecto de configuración experta de la aplicación y su uso y modificación se recomiendan únicamente a usuarios con conocimientos avanzados.

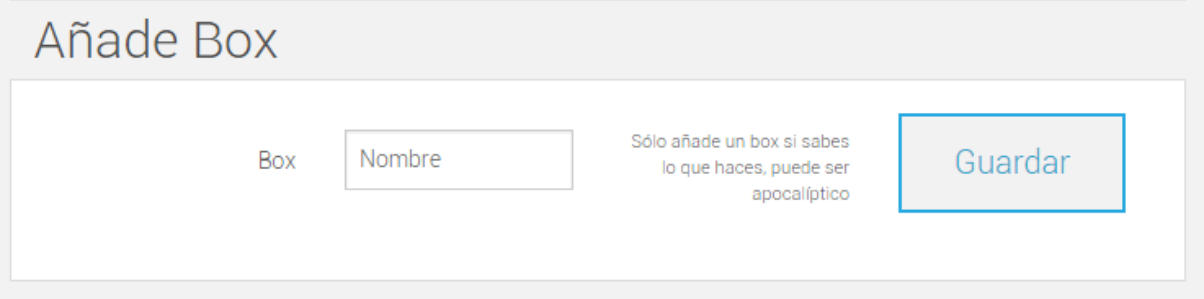

Figura 5.6.4: Añadir box

6. Conclusiones

6.1. Valoración personal

Teniendo en cuenta que la idea del proyecto la aportamos los desarrolladores del mismo, podemos afirmar que el resultado ha sido más que satisfactorio para nosotros.

Hemos logrado crear desde cero un sistema que añade una capa de seguridad extra a una red local doméstica o de trabajo fácil e intuitivo que puede utilizar cualquier persona, incluso sin conocimientos avanzados. Además se han alcanzado los requisitos que nos propusimos que cumpliera el proyecto e incluso hemos añadido alguno más durante la realización del proyecto.

Además, creemos que con este proyecto hemos practicado, hecho buen uso de muchos de los conocimientos aprendidos durante la carrera y adquiridos nuevos conocimientos gracias a la parte de investigación de nuevas tecnologías que luego hemos incluido en el proyecto.

Por último añadir que nuestro objetivo en un futuro es seguir desarrollando esta idea y este sistema para intentar conseguir un prototipo que se pudiera comercializar.

6.2. Trabajo futuro

La creación de este proyecto abre las puertas a un desarrollo de una herramienta mucho más completa y compleja. El proyecto se ha realizado y programado para que sea fácilmente escalable y ampliable tanto por el usuario como por los desarrolladores.

Una posible ampliación del proyecto puede ser añadir más interfaces físicas a la máquina anfitriona para así poder segmentar una red local. De esta forma habría una máquina virtual Interna por cada interfaz física conectadas o no entre sí en función de si se quiere tener acceso desde un segmento a otro.

Otra ampliación podría ser la de distribuir el tráfico entre las máquinas virtuales por las IPs de origen de las conexiones. El sistema ahora reparte el tráfico en función de los puertos de las conexiones, esta ampliación daría la opción de que por cada máquina virtual pasen todas las conexiones de un rango de IPs con sus filtros correspondientes. Es decir, en vez de filtrar por servicios y/o protocolos de conexión se filtraría por las IPs de origen de la red local.

Además, se podría crear una comunidad web en la que los usuarios de Network Safe Box pudieran compartir sus propias configuraciones con los demás usuarios así como las máquinas virtuales creadas por ellos. También se podría incluir la opción de poder buscar las máquinas virtuales y las configuraciones compartidas por los usuarios en la comunidad web desde la misma interfaz de Network Safe Box para poder descargarlas y añadirlas directamente a nuestra lista de máquinas virtuales.

7. Conclusions

7.1. Personal Ratings

Given that the idea of the project developers bring it, we can say that the result was more than satisfactory for us.

We have built from scratch a system that adds an extra security layer to a home LAN or job LAN easy and intuitive that anyone can use, even without advanced knowledge. We have also met the requirements we set out to fulfill the project and have even added some more for the project.

We also believe that with this project we have practiced, I made good use of many of the skills learned during the career and gained new knowledge through the research part of new technologies that have then included in the project.

Finally, our goal in the future is to further develop this idea and this system to try to get a prototype that could be marketed.

7.2. Future work

The creation of this project opens the door to developing a more comprehensive and complex tool. The project was implemented and scheduled to be easily scalable and expandable both the user and by developers.

A possible extension of the project may be to add more physical host machine in order to segment a LAN interfaces. Thus there would be an internal virtual machine for each physical interface connected to each other or not depending on whether you want to access from one segment to another.

Another extension could be to distribute traffic between virtual machines by source IP connections. The system now distributes traffic based port connections, this extension

would give the option for each virtual machine that pass all connections to a range of IPs with corresponding filters. That is, instead of filtering services and / or connection protocols be filtered by source IPs in the local network.

In addition, you could create a web community where users Safe Box Network to share their own settings with other users and virtual machines created by them. It could also include the option to search for virtual machines and shared by users on the web community from the same interface Safe Box Network to download and add them directly to our list of virtual machine configurations.

8. Bibliografía

HTML5:

http://www.w3schools.com/html/html5_intro.asp

PHP:

https://secure.php.net/

nuSOAP:

https://github.com/deviservi/nusoap

Foundation:

http://foundation.zurb.com/

Javascript:

http://www.w3schools.com/js/

MySQL:

https://www.mysql.com/

Python:

https://www.python.org/

VirtualBox:

https://www.virtualbox.org/

Vagrant:

https://www.vagrantup.com/

ANEXO I

En este anexo se detallan las contribuciones al proyecto de cada participante.

Aportación de Rodrigo Arranz López

En las primeras fases del proyecto realizamos conjuntamente los posibles diseños que podría tener el proyecto, tanto a nivel de interfaz gráfica como a nivel de creación de entornos virtuales. Una vez decidido el diseño final, que iríamos modificando para mejorar la experiencia final del usuario, llegamos a la conclusión de que Rafael Delgado se dedicaría a la parte de interfaz gráfica, que en este caso sería una interfaz web y yo me encargaría de la parte de creación de entornos virtuales, para luego una vez terminadas dichas partes unirlas trabajando de manera conjunta. Para realizar la creación de entornos virtuales del proyecto tuve que desarrollar los siguientes elementos.

Un script en Python llamado *builder.py* que al recibir los datos introducidos por el usuario en la interfaz web creara la red de máquinas virtuales pertinente con la configuración deseada. Dicho script consta de dos partes, una parte en la que se procesa los datos recibidos y otra en la que se crea el fichero de configuración, que luego ejecutará *Vagrant*, que contiene toda la configuración de las máquinas virtuales así como los scripts y programas que se deben ejecutar en las mismas. Además, una vez creado el fichero de configuración ejecuta *Vagrant* con dicho fichero.

Las *Boxes* que usa *Vagrant* para montar el entorno virtual. En esta parte tuve que crear cinco *Boxes* llamadas Interna, Externa, ServidorWeb, Servicios y Genérica. Cada *Box* tuvo que ser configurada para que encaminase el tráfico de manera correcta y que admitiese tráfico que no era para sí misma. Esta configuración se realizó mediante modificaciones de la configuración del sistema así como de reglas para *Iptables*. Además desarrollé un script en Python que lee el estado del sistema midiendo el porcentaje de uso del procesador, la memoria RAM y el disco duro. Este script monta un mini servidor web al que si haces una petición GET te devuelve la información del estado del sistema. Dicho script se ejecuta en cada máquina virtual de la red creada por *Vagrant*.

Los scripts de configuración de las máquinas virtuales. Dado que las máquinas se crean en función de las necesidades del usuario tuve que desarrollar varios scripts que se ejecutarán en las máquinas virtuales para filtrar el tráfico. Dichos script se programaron en *bash* y contienen tanto reglas de *Iptables* como reglas de enrutamiento.

El servidor donde se aloja la interfaz web. En este caso tuve que instalar y configurar un servidor Apache con soporte PHP para que la interfaz web pudiera ejecutarse sin problemas. Además tuve que configurar los permisos oportunos en las carpetas y archivos del servidor para que se pudieran acceder y ejecutar desde el usuario "www-data". También tuve que instalar y configurar una base de datos MySQL así como la herramienta phpMyAdmin para poder interaccionar con la base de datos de manera gráfica.

Todo esto lo implementé en un portátil que lleva instalado Ubuntu Server, con una interfaz gráfica sencilla, y configurado para que funcione todo correctamente. En este portátil se instaló la herramienta *Vagrant*, con sus dependencias y librerías, para poder utilizarla sin problemas.

En cuanto a la parte en la que trabajamos de manera conjunta para unir la interfaz con la creación de entornos virtuales mi aportación consistió en desarrollar las funciones en PHP que creaban el fichero con los datos del usuario que luego se mandarán al script de creación del fichero de configuración de *Vagrant*. Además tuve que crear ciertas tablas en la base de datos para guardar los datos necesarios para la creación de la red de máquinas virtuales.

Aportación de Rafael Delgado Meana

Después de la fase de investigación, la cual se realizó de forma conjunta y se atacaron ambas partes, la solución gráfica y la solución lógica de despliegue de máquinas virtuales, una vez decidido y habiendo hecho una búsqueda muy completa de las tecnologías existentes, se decidió dividir las tareas para llevar una línea de trabajo bien marcada, y bien definidas, Rodrigo se encarga del despliegue y yo, Rafael, del apartado gráfico, que detallo a continuación.

Después de investigar posibles framework ya diseñados y funcionales, vimos que no se ajustaban a los que requerimos, entonces se decidió crear un framework nuestro, desde cero, se decidió usar **Foundation**, ya que **Bootstrap** ya habíamos trabajado con él y queríamos aprender nuevos frameworks CSS, en un primer momento se quiso hacer el despliegue gráfico, a partir de un motor de videojuegos llamado **Phaser**, pero tenía un problema, no era escalable, por ello se desestima.

Una vez decidido la tecnología web, había que elegir un método de comunicación con el resto de partes de la aplicación, un método que fuera seguro, se decidió hacer mediante un Servicio Web escrito en **PHP**, la API, que se ha hablado anteriormente, esta API nos permite crear de forma rápida nuevas funcionalidades, esta API se creó mediante el protocolo de comunicación **SOAP**, y al estar en el servidor apache, configurado por Rodrigo, no era visible al usuario con los métodos tradicionales.

La API se comunicaba a su vez con una base de datos de datos **mySQL**, en al que se crearon las tablas necesarias para la persistencia de cualquier cambio realizado en el interfaz web, estas tablas se pueden ver en el anexo III.

Por lo tanto, yo hice la interfaz web con el framework de diseño CSS **Foundation**, la API php con todas las funciones para un correcto funcionamiento de la interfaz web, y a parte todas las cabeceras para que Rodrigo supiera de forma sencilla donde escribir su código para el despliegue y monitorización de las máquinas, es justo en este punto donde se unían las dos partes.

Los mayores retos que he encontrado en el proyecto han sido, por un lado en la investigación saber descartar herramientas muy válidas pero que no se ajustaban a nuestros requerimientos o eran demasiado grandes para nuestros propósitos y que en el caso de intentar trabajar con ellas nos hubiera llevado a ralentizar mucho nuestro trabajo. Por otra parte en la parte de implementación, el mayor reto fue la API y la interconexión con javascript ya que es un método de conexión muy específico y no se puede salir de esa estructura.

ANEXO II

Application Programming Interface (API)

- **function getMachines();**

Función que devuelve las máquinas existentes.

- **function getMachine($id);**

Función que devuelve la máquina “id”.

- **function guardarMaquina($nombre,$box,$imagen,$script);**

Crea una entrada en la base de datos.

- **function actualizaMaquina($id,$nombre,$box,$imagen,$script);**

Actualiza la información de una máquina en base de datos.

- **function getBoxes();**

Función que devuelve los boxes disponibles en la base de datos.

- **function getBox($id);**

Función que devuelve el box “id”.

- **function getScript();**

Función que devuelve todos los scripts disponibles.

- **function guardarScript($nombre,$script,$boxes);**

Función que guarda un script nuevo en base de datos.

- **function borraScript($id);**

Función que borra el script “id” de la base de datos.

- **function getDespliegue();**

Función que devuelve el despliegue guardado en base de datos.

- **function rmDespliegue($id);**

Función que borra el despliegue en base de datos

- **function addDespliegue($id);**

Función que crea un despliegue en base de datos.

- **function getScriptMachine($id);**

Función que devuelve los scripts activos en una maquina “id”.

- **function borraScriptMaquina($id);**

Función que borra un script asignado a una máquina.

- **function getScriptBox($id);**

Función que devuelve los scripts disponibles para un box “id”.

- **function getImages();**

Función que devuelve todas las rutas a las imágenes.

- **function getImagePath($id);**

Función que devuelve la ruta a una imagen “id”

- **function dameConf($clave);**

Función que devuelve el valor de una “clave”

- **function guardarConf($clave,$valor);**

Función que añade una entrada en la tabla de configuración de la base de datos.

ANEXO III

Estructura de la base de datos.