Diseño e implementación de un servicio de videovigilancia seguro en Internet con activación automática de cámaras remotas

Endika Zuazo Sagredo

Diseño e implementación de un servicio de videovigilancia seguro en Internet con activación automática de cámaras remotas

Endika Zuazo Sagredo

Memoria presentada como parte de los requisitos para la obtención del título de Master en Ingeniería de Telecomunicaciones por la Universidad del Pais Vasco.

Director: Juan Jose Unzilla Galan

Índice general

Índice de figuras

Índice de tablas

Resumen

El objetivo de este proyecto es el diseño e implementación de un servicio de acceso a una red de videocámaras de forma remota, automatizando las herramientas de publicación de los servicios. Para ello, se crearán nuevas herramientas de automatización y se diseñará una política de seguridad que garantice que los servicios publicados son accedidos únicamente por los usuarios autorizados, con los privilegios adecuados. Ambas consideraciones, tanto la seguridad como la capacidad de automatización, resultan cada vez más importante en la sociedad actual, debido a que el aumento de servicios publicados requiere de herramientas para simplificarlo y atrae un creciente número de ataques.

The aim of this proyect consists on the design and implementation of a service which allows the remote access of a network made of video cameras, automatizing the process. For that new automatizing tools will be developed and a security policy will be created. The policy will assure that the published servicces are only accesed by the authorized users and with the correct privilege level. Both considerations, the security and the automatization, are getting more and more important this days, because of the increase of the published services that need to guarantee security and an easy way for the deployment.

Proiektu honen helburua videokamera sare baten sarrera diseinatzea eta automatizatzea da, zerbitzuak publikatzeko tresnak sortuz. Horretarako, automazazio erremintak sortuko dira eta segurtasun politika bat diseinatu egingo da, publikatutako zerbitzuak bakarrik baimendutako erabiltzaileak aitzituko dituztela ziurtatuz. Aipatutako bi begiruneak gaur egun gero eta garrantzituago agertzen dira, izan ere, publikatutako zerbitzu zenbakia igotzen jarraitzen du, eta horrekin batera, zerbitzu hauek erasotzeko saiakerak.

1 Introducción

Desde la aparición de Internet en los años 80, la red se ha vuelto un elemento indispensable en nuestro entorno y en nuestras vidas. Mediante los sucesivos avances realizados, Internet se ha consolidado como uno de los pilares de nuestra sociedad. Tanto los individuos como las empresas hacen un uso intensivo de Internet, que se vio fuertemente incrementado con la aparición de los Smartphones, que permiten a los usuarios acceder a Internet en cualquier momento y lugar. El siguiente paso a conectar nuestros teléfonos es conectar todo lo demás. En este ámbito surge el concepto de Internet of the Things (IoT), en el que se prevé conectar a la red una gran variedad de dispositivos y sensores.

Internet, desde su popularización y debido a su uso masivo, ha sido también objetivo de individuos y colectivos que han visto en la red una forma de realizar actividades maliciosas e ilegales. En los años 80, cuando se diseñó Internet, no se preveía el crecimiento ni los usos variados que se le iban a dar, por lo que en su concepción la seguridad no estaba apenas contemplada. Ello permite que existan millones de precedentes de ataques, fraudes, engaños... realizados mediante Internet o contra los servicios prestados por Internet, que a pesar de la creciente conciencia sobre la seguridad, han ido apareciendo y obteniendo éxito. Por tanto, la seguridad es un concepto clave en este entorno. Debido a la variedad de amenazas a las que hay que hacer frente, es un reto sin final, una carrera interminable para estar siempre por delante de aquellos que buscan utilizar la red para fines ilegítimos. A diario por Internet son enviadas, accedidas y almacenadas ingentes cantidades de datos, muchos de ellos de carácter sensible, personal y confidencial. La aparición de IoT genera nuevos y variados escenarios en los que los atacantes pueden realizar actividades dañinas. IoT se caracteriza por la no-homogeneidad de los dispositivos conectados a la red, por la cantidad de distintas aplicaciones ofrecidas, lo que dificulta aún más la securización del entorno. Muchos de los dispositivos conectados ni siquiera están preparados para ofrecer seguridad, ya que se prima la eficiencia, el bajo consumo...En la mayoria de los casos son equipos que no tienen grandes requisitos, por lo que se les dota de poca capacidad, complicando sobremanera la implementación de medidas de seguridad.

A pesar de la dificultad que presenta este nuevo entorno, es necesario hacer un esfuerzo para tratar de garantizar un nivel de seguridad acorde con la criticidad de los servicios y equipos. La forma de alcanzar este objetivo es mediante el diseño y posterior implementación de una política de seguridad que defina claramente cuáles son los activos a proteger, los servicios y las técnicas, metodologías o herramientas que se utilizaran para tal fin.

El caso estudiado en este documento se trata de una red de cámaras que se desean conectar a Internet para ser accesibles desde cualquier lugar con unas garantías de seguridad que eviten el acceso y control de las mismas por parte de terceros no autorizados para tal fin. Las cámaras se hallan distribuidas geográficamente y conectadas entre ellas por una red dedicada. Además, la red de cámaras considerada dispone de un número de equipos conectado suficientemente grande como para que realizar la publicación y securización de los servicios de forma manual no sea un método recomendable. Por ello, es necesario automatizar el proceso mediante un

procedimiento que lo simplifique y permita facilitar su repetición en un futuro. A lo largo del documento se desarrollarán la política de seguridad a implementar, razonando las decisiones tomadas, así como el procedimiento de automatización de la publicación de servicios.

En resumen, se pretende garantizar el acceso de forma segura a una red de videocámaras conectadas a Internet desde cualquier punto de la red.

2 Contexto

En la actualidad la presencia de Internet en nuestro día a día es una constante. El número de dispositivos conectados a Internet ha crecido exponencialmente en los últimos años. En 2015 se calculaba que existían unos 15.4 billones de dispositivos conectados. Se calcula que este número puede duplicarse para 2020 o incluso multiplicarse por diez, según el gigante tecnológico Intel[1]. Buena parte de los nuevos dispositivos que se van añadiendo a la red pertenecen a la categoría del Internet de las cosas.

En 1999 el ingeniero del MIT (Massachusetts Institute of Technology) Kevin Ashton propuso por primera vez el concepto de IoT. El concepto hace referencia a la interconexión digital entre todo tipo de objetos, es decir, que sean capaces de comunicarse entre ellos, mediante Internet. En esta definición se encuadran objetos tan diversos como puede ser un frigorífico, un libro electrónico, un sensor de temperatura en una antena en el monte o la propia ropa (los llamados wereables). El impacto y la transcendencia que puede llegar a tener es prácticamente imposible de predecir, aunque se intuye enorme. Por un lado puede transformar la vida cotidiana de las personas, mediante las casas domóticas, camisetas capaces de avisar al médico si detectan alguna anomalía en los latidos del corazón y otro tipo de novedades. Por otro lado, la revolución también se produce en el ámbito industrial, con lo que se conoce como IIoT (Industrial Internet of the Things), que es la aplicación del IoT al ámbito empresarial e industrial. El objetivo es conseguir una optimización de los procesos industriales y cadenas de manufacturación. Mediante la monitorización extensiva de todos los dispositivos involucrados en el proceso de fabricación es posible aumentar la productividad.

Esto combinado con nuevas tendencias como el Big Data, el almacenamiento y tratamiento masivo de datos para su posterior análisis, permite aplicar el mantenimiento predictivo y mejorar ostensiblemente sus procesos industriales. Entre los múltiples sensores que se utilizados para monitorizar, en cualquier ámbito, podemos encontrar las cámaras IP. Anteriormente a las cámaras IP, la videovigilancia solía llevarse a través de circuitos cerrados, propietarios. Sin embargo, las cámaras IP o cámaras de red, son cámaras tradicionales que permiten emitir el vídeo en tiempo real a través de Internet, además de funcionalidades adicionales, como notificaciones de correo o activación mediante movimiento. El uso de este tipo de cámaras ofrece la posibilidad de monitorizar áreas o zonas de forma remota y centralizada, sin necesidad de intermediarios. Todas las cámaras de un mismo sistema, que puede estar geográficamente distribuido, emiten de forma simultánea y en tiempo real, por lo que es posible acceder a la señal desde cualquier lugar para su visionado.

Sin embargo, al igual que todos los dispositivos conectados a Internet, están expuestas a ataques maliciosos por parte de agentes externos. La relativa novedad que suponen las cámaras IP y los dispositivos IoT en general los convierte en blanco fácil de los ataques, ya que la seguridad no está demasiado bien implementada por ahora. Los dispositivos IoT suelen ser equipos de capacidades reducidas, en el que prima la eficiencia, la duración de la batería... y en los que las funcionalidades de securización no suelen estar incluidas, ya que añaden una gran

complejidad computacional, lo que choca frontalmente contra los requisitos habituales en estos entornos. Existen vulnerabilidades descubiertas en diversos modelos que apuntan que algunos dispositivos almacenan la información capturada sin ningún tipo de cifrado o que realizan la autenticación mediante cifrados débiles y de reconocida vulnerabilidad, lo que podría permitir a un atacante hacerse con las imágenes grabadas por la misma. Muchas de estas cámaras, permiten incluso el acceso mediante usuarios anónimos sin autenticar. Con estos accesos una entidad no autorizada podría también emitir imágenes falseadas o de un momento que no se corresponde con el que debería.

Uno de los ataques más conocidos mediante cámaras IP consiste en infectar las cámaras, creando así una botnet (red de equipos infectados), que posteriormente podría ser usado para realizar ataques de denegación de servicio (DoS, Denial of Service) contra otros servicios. Un ejemplo es la botnet Persirai[2], capaz de atacar a miles de cámaras IP repartidas por el mundo, susceptibles de ser vulnerables.

Por tanto, si queremos beneficiarnos de estas nuevas aplicaciones de Internet, es necesario realizar un esfuerzo para dotar de seguridad a los equipos y a las redes. Mediante la aplicación de políticas de seguridad es posible evitar muchos de estos ataques y dificultar enormemente las cosas para los atacantes.

3 Alcance

El alcance de este proyecto contempla el diseño e implementación de una política de seguridad en una red de cámaras IP(Internet Protocol) accesibles de Internet, además del diseño de un procedimiento para simplificar y automatizar la publicación del acceso a dichas cámaras IP.

Este alcance se ve concretado en un objetivo principal, compuesto a su vez por varios objetivos más específicos.

El **objetivo principal** de este TFM (Trabajo de Fin de Master) es diseñar e implementar un servicio de acceso a una red de videocámaras, distribuida geográficamente, de forma remota y con unas garantías de seguridad que vayan en consonancia con la criticidad del servicio que se desea proveer. Resulta también necesario dimensionar las capacidades requeridas por la red para cubrir la demanda, así como una posterior validación del correcto funcionamiento del servicio. Dicho servicio debe, además, ser flexible y escalable, para permitir su posterior ampliación en caso de ser necesario. Para ello, se ha diseñado una política de seguridad que controla el acceso y el uso de los recursos, en función de diferentes perfiles y privilegios. También se ha desarrollado una metodología que simplifica y automatiza la tarea de añadir nuevas cámaras al servicio.

Por tanto, se dispone de una política de seguridad en la que se definen cuáles son los objetivos de seguridad a alcanzar, en cuanto a accesibilidad, trazabilidad, disponibilidad… En la misma política se identifican los activos a proteger y se especifican las acciones a tomar para garantizar los objetivos definidos. Además, se ha desarrollado una herramienta capaz de automatizar el proceso de publicar de forma segura el acceso a una nueva cámara.

Para la consecución de este objetivo, se han definido otros objetivos secundarios, que juntos, conforman el objetivo principal. La consecución de estos objetivos es secuencial, ya que cada uno de ellos depende directamente de la consecución de uno o más de los objetivos previos. A continuación se definen más en detalle cada uno de los objetivos secundarios del proyecto:

- **Definir los requerimientos:** el primer objetivo marcado consiste en una identificación y definición de los requerimientos del sistema. Es decir, una descripción de aquellos requerimientos de seguridad que se desea alcanzar mediante la implementación de la política de seguridad y los requerimientos esperados respecto del procedimiento de automatización.
- **Diseño de la política de seguridad:** una vez identificados los requerimientos a cumplir por la política, es necesario plasmarlos mediante una política de seguridad concreta.
- **Implementar la política de seguridad:** en este objetivo se lleva a la práctica y a un entorno de pruebas la política que se ha definido en el objetivo anterior.
- **Publicar los servicios:** una vez securizado el procedimiento de acceso a los servicios es necesario probar la publicación de uno de los servicios, para garantizar que se pública de forma correcta y que es accesible desde Internet.

- **Automatizar la publicación de los servicios:** este objetivo comprende el diseño del procedimiento para evitar tener que publicar uno a uno los servicios, repitiendo el mismo procedimiento en varias ocasiones.
- **Evaluar el sistema:** una vez publicados todos los servicios e implementada la política de seguridad es necesario comprobar que el funcionamiento de cada componente es el esperado y que se han cumplido los objetivos marcados.

4 Beneficios

En este aparatado se van a proceder a detallar los beneficios derivados de la realización de este proyecto, es decir del diseño de una nueva arquitectura de seguridad para una red y la publicación de forma automática de los servicios internos de dicha red.

Los beneficios se dividen en tres apartados: económicos, sociales y técnicos, que se detallan a continuación:

4.1 Beneficios Técnicos

De entre los beneficios aportados del proyecto realizado, los que están más relacionados con los objetivos del mismo son los beneficios técnicos.

La securización del acceso a la red, con nuevo equipamiento y nuevas funcionalidades es en si un beneficio. La implementación de una nueva política de seguridad en la que se incluyen perfiles de seguridad de capa de aplicación es una funcionalidad técnicamente novedosa que permite tener un mayor control del tráfico de la red, una mayor visibilidad y una mayor granularidad a la hora de ejecutar acciones. Como es lógico, todo ello, combinado con el diseño de una nueva política de seguridad, redunda en una mejoría sustancial de la seguridad de la red.

Con el nuevo esquema de seguridad y el nuevo equipamiento desplegado, además de la seguridad se aumenta el throughput, eliminando el cuello de botella que era el firewall. Además, el firewall era un punto de fallo único, que se ve sustituido por varios equipos en paralelo, aumentando la disponibilidad. Los balanceadores, que a priori también podrían consistir un punto único de fallo, son instancias replicadas en varios equipos físicos, por lo que tampoco se añade punto de fallo único.

El proyecto también incluye la creación de una serie de herramientas para automatizar ciertas operaciones de publicación de los servicios internos (cámaras de videovigilancia). La creación de estas herramientas permite mejorar los tiempos necesarios para desplegar el servicio a nuevos usuarios o nuevos servicios, minimizando también los recursos y el nivel de competencias técnicas necesarias por los futuros gestores de la red. Además, las nuevas herramientas técnicas pueden servir como base para el desarrollo de diversas funcionalidades que se considere necesario automatizar.

4.2 Beneficios Sociales

En este proyecto se configura, de forma automática, el acceso a servicios internos desde la red externa. Esto es una ventaja para los usuarios de dicha red, que podrán utilizar recursos o

servicios de la red desde cualquier otro lugar que no sea su puesto de trabajo, como su hogar, aumentando la comodidad en el acceso y la disponibilidad de los recursos.

Además, el aumento en la seguridad de la red, redunda directamente en la experiencia de usuario, ya que disminuye sustancialmente las posibilidades de que la red se vea comprometida, evitando todo tipo de inconvenicencias a los usuarios, desde imposibilidad para acceder a la misma a la suplantación de los servicios por parte de terceros malintencionados o robo de información personal sensible.

4.3 Beneficios Económicos

En cuanto a los beneficios económicos de este proyecto, no son la parte más relevante del mismo, ya que el objetivo primordial es securizar la red. Sin embargo, no hay que olvidar que la securización de la red, más allá de un deber, es una inversión.

En el mundo de hoy en día, un ataque informático, como por ejemplo un ataque de denegación de servicio, que evite que usuarios legítimos puedan hacer uso de la red, puede suponer unas perdidas cuantiosas para la empresa u organismo explotador de la red. Es por ello que este proyecto, puede evitar a la larga perdidas derivadas de ataques informáticos. El aumento en la seguridad de la red, es indudablemente positivo a largo plazo para los beneficios de la empresa, ya que el coste derivado de un ataque, más allá de posibles multas de la Administración por no proteger debidamente los datos o similares, sería indudablemente mayor al coste del proyecto. Sin mencionar la degradación de la imagen publica que implica el conocimiento de que la empresa ha sufrido un ataque exitoso o los posibles daños ocasionados a terceros si en el trascurso del ataque o gracias a el, la red empresarial es utilizada como botnet para posteriores ataques.

Por otro lado, la posibilidad de automatizar la publicación de servicios redunda en una disminución de las horas necesarias para llevar a cabo algunas tareas repetitivas, con el coste asociado del salario de un trabajador que las lleve a cabo. Gracias a las nuevas herramientas de automatización, esta necesidad de esfuerzos y recursos disminuye, con el consiguiente ahorro de costes.

5 Análisis de alternativas

Una vez se ha realizado la definición del alcance del proyecto, es necesario proceder con la selección de las alternativas tecnológicas existentes para llevarlo a cabo. La selección se ha realizado en función de una serie de criterios ponderados que han permitido tomar una decisión razonada y justificada de cual es la mejor alternativa en cada uno de los casos estudiados.

Para cada una de las decisiones tomadas, se presenta el porqué de dicha decisión y como encaja en el proyecto, clarificando qué se espera de la solución escogida para esa decisión. A continuación, se detallan las diversas alternativas consideradas, realizando una descripción pormenorizada de las mismas. Seguidamente se muestran los criterios considerados para dicha decisión, ponderándolos en base a su importancia en el caso concreto que aplica. Finalmente se recoge en una tabla resumen las puntuaciones asignadas a cada alternativa en los distintos criterios considerados, facilitando la identificación de la solución más adecuada en cada caso.

En este proyecto se implementa una nueva red securizada mediante su correspondiente pólitica de seguridad, en la que ciertos servicios internos se hacen accesibles desde el exterior, además de permitir a los usuarios acceder a los recursos internos. Para llevar a cabo el diseño de dicha solución, se han identificado dos grupos de alternativas entre las que seleccionar la mejor. Por un lado, como va a ofertarse una solución VPN para el acceso a los recursos internos, era necesario escoger el tipo de túnel que mejor se adapta a los requerimientos del proyecto. Por otro lado, también era necesario escoger la herramienta más adecuada para realizar la automatización de los procesos repetitivos, como la publicación de los servicios internos.

Es destacable también comentar que existen otras decisiones que a priori parecian necesarias, pero no lo eran, ya que en la propia definición del proyecto y sus requisitos vienen fijadas las opciones a escoger. Entre estas alternativas la más destacable es la marca y modelo de los equipos físicos a instalar para implementar la decisión, cuya elección no fue necesario realizar, ya que en las propias especificaciones técnicas del proyecto se indica cuales son los equipos a utilizar, evitando así el estudio de las alternativas en cuanto a equipamiento físico se refiere.

A continuación se presentan las decisiones comentadas, con sus respectivas alternativas y criterios, así como las puntuaciones asignadas.

5.1 Herramienta de automatización

Uno de los objetivos del proyecto contemplaba la necesidad de automatizar la publicación de los servicios de videovigilancia para poder ser accedidos desde el exterior. Esta necesidad surge debido a que el número de cámaras existentes es elevado y motivos de seguridad no se desea permitir acceso a todas en conjunto, sino individualmente. Esto permite un control mucho más granularizado de lo que ocurre, con sus correspondientes ventajas a la hora de auditar o realizar configuraciones más específicas. Sin embargo, la publicación “individual”

de cada servicio y la creación de usuarios es un trabajo tedioso que resulta poco eficiente de realizar de forma manual. Es por ello que se decide aprovechar las nuevas herramientas ofrecidas por los fabricantes para llevar a cabo la tarea. La herramienta en cuestión son las APIs (Application Programing Interface) desarrolladas tanto por los fabricantes de los firewalls (Fortinet) y de los balanceadores (F5).

Estas APIs son un conjunto de funciones propias de los equipos que se ofrecen para ser utilizados desde software externo. Por ejemplo, la función interna de crear una regla de firewall se ofrece para que elementos externos al firewall puedan, cumpliendo una serie de requisitos, llamarla y crear una regla. En ambos casos, las APIs ofrecidas son REST (Representational State Transfer), en las que la forma de comunicarse con el sistema es vía peticiones HTTP (Hypertext Transfer Protocol), de tipo get, put, post... en las que se incluyen elementos XML(eXtensible Markup Lenguage) o JSON (JavaScript Object Notation). Para poder interactuar con estas APIs es necesario desarrollar o utilizar un software que sea capaz de manejar los objetos XML/JSON y de realizar peticiones de HTTP. Se valoran para ello diferentes opciones, incluyendo el uso de herramientas específicamente diseñadas para consumir servicios REST (Postman) y la creación de herramientas propias, mediante la codificación de scripts dedicados (usando Perl o Python).

5.1.1 Alternativas

Postman

En el entorno de pruebas de APIs, la herramienta más conocida y más ampliamente utilizada es Postman. En realidad es una colección de herramientas y utilidades preparadas para probar y consumir servicios REST mediante peticiones web.

Inicialmente Postman surgió como una extensión para el navegador, de forma que era posible realizar peticiones a APIs desde el propio navegador. Esto ofrecía la ventaja de abstraerse utilidades como curl, para cuyo uso era necesario la linea de comandos y conocer la sintaxis. Sin embargo, pronto se vio que únicamente la extensión de navegador quedaba muy limitada, por lo que a día de hoy Postman dispone de aplicación nativa para Windows, MAC y Linux.

La versión nativa permite, al igual que el navegador, escoger la URL(Uniform Resource Location), el método HTTP a utilizar y los parámetros a añadir. También ambas ofrecen la posibilidad de guardar peticiones de forma que sea posible reutilizarlas en el futuro. Pero además añade funcionalidades como el pre y post scripting, que mediante JavaScript permite implementar lógica antes y después de las peticiones para rellenar los valores a enviar de forma dinámica o guardar los valores recibidos como variables que puedan ser usadas posteriormente en otras llamadas o para tomar decisiones. También añade el concepto de colección, que permite agrupar una serie de llamadas a APIs, para posteriormente ejecutarlas secuencialmente (salvo que se indique lo contrario por medio de scripting). Esta funcionalidad resulta muy útil a la hora de automatizar tareas o tests de las APIs. Finalmente incluye otras funcio-

nalidades como la posibilidad de ejecutar las colecciones de forma periódica o sincronizar las colecciones (propias o de terceros) entre dispositivos, así como un modo cloud colaborativo para desarrollar colecciones por equipos de trabajo. Este último modo es de pago, igual que la ejecución periódica a partir de un volumen dado. El resto de las características son gratuitas.

Perl

Perl (Practical Extraction and Report Languaje) es un lenguaje de programación creado por Larry Wall en 1987. Este lenguaje se asemeja a C pero con influencias de lenguajes como el utilizado en la bourne shell (sh) y otros lenguajes orientados a la manipulación de texto. Es por ello que es un lenguaje ampliamente utilizado para el procesado de textos, así como para scripting, debido a su flexibilidad y pocas limitaciones, aunque originalmente fue concebido como una ayuda para la tarea de administración de sistemas UNIX. Es un lenguaje que trata de ser simple, comprensible y con una sintaxis razonablemente fácil de interpretar.

Perl es un lenguaje interpretado, es decir, que necesita de un intérprete propio para poder ser ejecutado. Es el propio interprete quien compila los programas. Es por ello que se habla de scripts en Perl en lugar de programas. En general, Perl permite un desarrollo de scripts relativamente rápida, orientada más al correcto funcionamiento del script en cada momento que a un desarrollo mantenido. En el caso que nos ocupa, nos permitiría desarrollar un script de automatización de forma rápida y sencilla, sin necesidad de estructurarlo demasiado. Es decir, permite una solución de compromiso entre el tiempo de desarrollo y la "calidad" del script.

A pesar de estar inicialmente orientado para el mantenimiento de sistemas UNIX y de ser especialmente indicado para el tratamiento de textos, es un lenguaje muy versátil, capaz de extender sus funcionalidades mediante paquetes utlizables. Para el consumo de servicios REST, Pearl dispone de los paquetes *REST::Client* y *JSON*, diseñados para realizar peticiones HTTP a servicios REST y para manejar objetos JSON respectivamente.

Python

Al igual que Perl, Python es un lenguaje de programación que puede ser utilizado para consumir servicios REST entre otras muchas posibilidades. La principal característica de Python es la importancia que se le otorga a que la sintaxis resultante sea los más legible posible. Python esta orientado a objetos, lo que permite, en el caso que nos ocupa, utilizar objetos definidos por los propios fabricantes para simplificar el acceso a las APIs.

Python como lenguaje, es administrado por la Python Software Foundation, que el la fundación encargada del desarrollo del mismo, aunque el creador del mismo es Guido Van Rossum. A día de hoy, la Python Software Foundation mantiene dos ramas de versiones de Python, la

2.X y la 3.X, que no interoperan entre ellas, por lo que es importante escoger la adecuada a la hora de comenzar un proyecto.

Esto, sin embargo, no es una dificultad para su uso, ya que según el indice TIOBE de Septiembre de 2018 (https://www.tiobe.com/tiobe-index/), Python es el tercer lenguaje de programación más usado en el mundo, solo por detrás de Java y C, lo cual nos permite intuir su versatilidad y su utilidad a la hora de afrontar cualquier tipo de proyecto, incluyendo la interacción con APIs REST.

También, al igual que Perl, dispone de librerías y funciones adicionales que pueden importarse en los scripts desarrollados para extender las funcionalidades. De nuevo, disponemos de librerías especialmente pensadas para permitir consumir servicios REST, como son *json requests*, que realizan unas funciones similares a las descritas en Pearl.

5.1.2 Criterios de selección

Para escoger entre las alternativas presentadas, se definen los siguientes criterios, cada uno de los cuales está ponderado en función del impacto del mismo en la decisión.

- **Sencillez de implementación:** como en cualquier solución a implementar el coste en esfuerzo para realizar la implementación es un factor relevante a tener en cuenta. En ocasiones, es posible obtener resultados similares mediante soluciones más sencillas de implementar, por lo que no hay que perderlo de vista. Además, es habitual que una mayor sencillez de implementación lleve asociada una menor dificultad de configuración, lo que se traduce en la mayoría de los casos en un mantenimiento menos complejo. Sin embargo, no se otorga una puntuación demasiado decisiva a este criterio, ya que, en caso de observarse prestaciones sensiblemente mejores en una alternativa, no ha de ser descartada por un mayor esfuerzo de implementación.
- **Documentación:** para llevar a cabo el desarrollo de una nueva herramienta utilizando un lenguaje de programación no conocido y una serie de módulos desarrollados por terceros es importante que éstos se encuentren bien documentados, con multitud de ejemplos para comprender su uso y funcionamiento, ya que de lo contrario se corre el riesgo de tener que dedicar una gran cantidad de tiempo unicamente para aprender, sin estar realizando avances. Lo mismo puede decirse de las herramientas pre-existentes, en las que es importante que existan tutoriales que simplifiquen la curva de aprendizaje para poder dedicar más tiempo al uso de la herramienta que a su comprensión.
- **Prestaciones:** se considera importante como criterio que una vez escogida la alternativa ésta sea capaz de ofrecer unas funcionalidades lo más ajustadas posibles a las necesidades del proyecto. Una herramienta que disponga de muchas funciones no muy relevantes para el proyecto no es demasiado útil. En este apartado, el desarrollar una herramienta desde cero obtiene una mayor puntuación, pero también se ponderan las facilidades de cada lenguaje para crear la herramienta necesaria.

5.1.3 Selección de la solución

Atendiendo a los criterios comentados anteriormente y ponderándolos según la importancia que se considera para cada uno de ellos, se realiza a continuación una tabla que permite escoger la mejor alternativa:

Criterio	Postman	Pearl	Python
Sencillez de implementación(%20)	6	7	8
Documentación(%40)	7	5	9
Prestaciones(%40)	6	8	8
TOTAL	**64**	**66**	**84**

Tabla 5.1: Criterios de selección de la herramienta para la automatización.

Conclusión: la mejor alternativa en este caso pasa por desarrollar una herramienta propia para las necesidades del proyecto, utilizando Python, que dispone de una extensa documentación en la que se incluyen ejemplos de cómo interactuar con las APIs concretas que se requieren para el proyecto. Además, permite de forma sencilla añadir funcionalidades, como la lectura de datos desde Excel, que se ajustan a los requisitos de la automatización de este proyecto.

5.2 Tecnología del tunel

Para permitir el acceso remoto a aquellos servicios internos que no vayan a ser publicados y accesibles desde Internet, es necesario el uso de otro tipo de mecanismo que permita a aquellos usuarios situados en el exterior alcanzar los recursos propios de la red privada. La forma más habitual de permitir estos accesos es mediante el uso de VPNs.

Una VPN (Virtual Private Network) es una red virtual que permite a dispositivos que no se encuentran en una misma red física conectarse como si lo estuvieran, utilizando para ello una red pública. Esta tecnología, también conocida como túneles, encapsula los datos del cliente remoto, que se envían encriptados hasta el terminador del túnel, que está situado en la red privada, aunque se encuentra accesible desde Internet. Por ello, los firewalls suelen ser una elección habitual para funcionar como terminadores de túneles. Al enviar el tráfico encapsulado, se garantiza que atraviesa la red pública de forma segura, sin poder ser interceptado por ningún atacante.

Hoy en día el uso de túneles VPN está ampliamente extendido. Para el establecimiento de una sesión VPN es necesario un protocolo de tunelado. Actualmente, son dos los protocolos más extendidos para ello. A continuación, se comentará el funcionamiento de ambos, así como

sus principales características, para poder mostrar cual era la opción más apropiada para la consecución del proyecto.

5.2.1 Alternativas

VPN-SSL

SSL son las siglas de Secure Socket Layer. SSL es un protocolo que permite a las aplicaciones transmitir datos de forma segura entre dos equipos. Permite establecer un enlace encriptado entre ambos, de manera que la comunicación no viaja en claro a través del mismo. Para ello, SSL utiliza certificados X.509, basados en criptografía asimétrica (es decir, par de claves pública/privada), para autenticar al otro extremo de la comunicación. De esta forma es posible, por un lado, garantizar que el otro extremo de la comunicación es quien dice ser y al mismo tiempo realizar un intercambio que permita establecer una clave, llamada master secret, que permitirá, una vez establecida la comunicación, realizar el cifrado de la información a intercambiar.

El protocolo SSL funciona sobre el nivel de transporte, pero sin llegar a ser de aplicación, es decir, los datos que cifra son los correspondientes a los protocolos de aplicación (HTTP, FTP…) y se adapta muy bien en los escenarios de comunicaciones móviles. Desde cualquier navegador de hoy en día es posible realizar una negociación de SSL contra un servidor de VPN, facilitando el uso del mismo a los usuarios y evitando la necesidad de instalar software adicional. Esto permite granularidad a la hora de desplegar las soluciones, ya que en la propia pestaña del navegador se presentan al usuario los recursos a los que se desea permitir acceso mediante VPN-SSL. Del mismo modo, ofrece la posibilidad de personalizar y controlar los accesos en base al usuario en cuestión, lo que es una característica muy interesante. También permite realizar chequeos del estado del equipo que va a conectarse, para auditar su situación y en caso de detectarse irregularidades, no permitir la conexión.

Por tanto, SSLVPN es una opción sencilla de configurar y que no requiere pasos adicionales por parte del cliente. Permite un acceso seguro a una serie de recursos limitados, definidos por el administrador.

IPSec

IPsec (Internet Protocol Security) está formado por una serie de protocolos, que al igual que SSL, permiten asegurar las comunicaciones extremo a extremo mediante la autenticación y cifrado de los paquetes IP intercambiados. IPsec trabaja en la capa de red, por lo que es posible garantizar también la securización de protocolos de capa 4, como TCP(Transmission Control Protocol) y UDP(User Datagram Protocol).

El funcionamiento de IPsec se basa en establecer una Asociación de Seguridad (SA, Security

Asociation) entre ambos extremos. Una SA es el conjunto de parámetros (tanto algoritmos a usar, longitud de las claves, las propias claves…) que son necesarias para el cifrado de los datos a transmitir. Mediante una negociación en dos fases, una primera en la que se establecen los parámetros de la otra y una segunda en la que se establecen las claves de sesión, IPsec establece un túnel extremo a extremo entre dos dispositivos.

IPsec permite trabajar en dos modos distintos. Por un lado es posible usar el modo transporte, pensado para comunicaciones ordenador a ordenador, en la que solo la carga útil es cifrada. Por otro lado, el más común modo túnel, en el que todo el paquete es cifrado y autenticado, para posteriormente ser encapsulado en un nuevo paquete IP que viajará por la red. Este modo es habitualmente usado para establecer tuneles entre distintas ubicaciones de una misma empresa, permitiendo extender la propia red a través de una red pública.

5.2.2 Criterios de selección

Para escoger entre ambas alternativas, se definen los siguientes criterios, cada uno de los cuales está ponderado en función del impacto del mismo en la decisión.

- **Sencillez de implementación:** como en cualquier solución a implementar el coste en esfuerzo para realizar la implementación es un factor relevante a tener en cuenta. En ocasiones, es posible obtener resultados similares mediante soluciones más sencillas de implementar, por lo que no hay que perderlo de vista. Además, es habitual que una mayor sencillez de implementación lleve asociada una menor dificultad de configuración, lo que se traduce en la mayoría de los casos en un mantenimiento menos complejo. Sin embargo, no se otorga una puntuación demasiado decisiva a este criterio, ya que, en caso de observarse prestaciones sensiblemente mejores en una alternativa, no ha de ser descartada por un mayor esfuerzo de implementación.
- **Prestaciones:** la adaptación de las caracteristicas propias de cada uno de los protocolos al escenario del proyecto es un criterio relevante a la hora de tomar una decisión. Cada uno de ellos se comporta diferente en cada posible situación, variando la idoneidad en cada caso.
- **Necesidad de software propietario:** el objetivo de la VPN es permitir a usuarios remotos conectarse a la red interna. La variedad de dispositivos con los que se pueden querer conectar los usuarios, asi como de sistemas operativos hace que sea interesante la posibilidad de evitar la necesidad de que los usuarios instalen software en sus dispositivos. Además, en ocasiones el software puede ser de pago o no estar disponible para algunas plataformas. Este criterio se considera bastante relevante, ya que lleva aparejada la simplicidad de acceso a la VPN para los usuarios.
- **Escalado:** cada una de las tecnologías implica unas operaciones y procedimientos de encriptación, que pueden resultar una carga para el equipo que las realice. En este caso, al ser el firewall quien funcione como terminador de VPNs, es necesario conocer la carga aproximada que implica para el firewall la creación de nuevos túneles en función del

protocolo usado. Se considera que este criterio es relevante, pero no demasiado ya que no se espera que el número de túneles simultáneos vaya a escalar mucho.

5.2.3 Selección de la solución

Atendiendo a los criterios comentados anteriormente y ponderándolos según la importancia que se considera para cada uno de ellos, se realiza a continuación una tabla que permite escoger la mejor alternativa:

Criterio	VPN-SSL	IPSec
Sencillez de implementación(%20)	9	7
Prestaciones(%30)	8	5
Necesidad de software propietario(%20)	9	5
Escalado(%30)	5	8
TOTAL	**75**	**63**

Tabla 5.2: Criterios de selección de la herramienta para la automatización.

Conclusión: para garantizar el acceso VPN, el protocolo SSL-VPN es la mejor opción en este caso. Los servicios que se desean publicar se acceden mediante web, por lo que establecer un tunel completo mediante IPSec no resulta tan adecuado en este caso concreto. Además, el poder evitar que los usuarios necesiten descargarse un software de terceros ayuda a que SSL-VPN sea la mejor opción.

6 Diseño de la solución

En este apartado se presenta el diseño de la solución escogida para la implementación del acceso securizado a la red interna, basándose en los criterios establecidos y ponderados anteriormente.

En primer lugar se muestra la red original de la que se parte, para a continuación presentar la topología diseñada para el proyecto, incluyendo direccionamientos y cableado físico. En ambos casos se detalla el funcionamiento de las mismas, así como los flujos de datos que atraviesan cada una de las redes.

Posteriormente se muestran los procedimientos seguidos para automatizar el acceso externo a los recursos internos de la red privada. Son dos procedimientos, el primero permite el acceso mediante VPN, mientras que el segundo expone en Internet los servicios concretos que se desean permitir acceder. En ambos casos, se indica la metodología a seguir para realizar la configuración y el modo en el que se ha automatizado el proceso.

Como se ha comentado anteriormente el objetivo principal es garantizar un acceso seguro a la red interna, en la que se encuentran situadas, entre otros equipos y servicios, una red de videocámaras IP.

6.1 Situación actual

En este apartado se comenta brevemente cual es la situación en la que se encuentra la red en el momento de comenzar el proyecto, cual es el camino que hace cada tipo de tráfico y que equipos la componen.

A continuación, en la figura 6.1, se muestra un esquema con la situación de partida del proyecto.

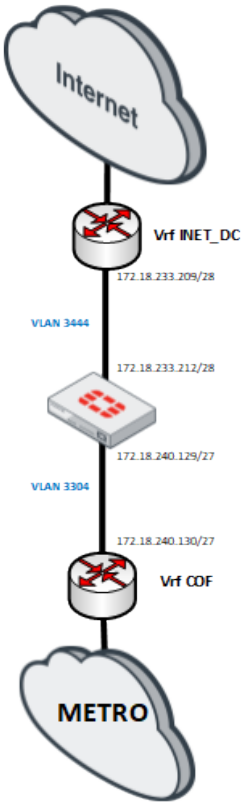


Figura 6.1: Esquema de red de partida

La red privada a securizar accede a Internet a través de un router interno, que conecta mediante un firewall de Fortinet al router de salida a Internet. En realidad, ambos ruters son instancias virtuales de routing alojadas en un mismo equipo físico, utilizando VRFs (Virtual Routing Function), el mecanismo propietario de Cisco para virtualizar varios equipos sobre un mismo hardware. Las dos VRFs son las identificadas en la figura 6.1 como VRF INET_DC y VRF COF. La primera es la VRF a que conecta la salida a Internet con el firewall, mientras que la segunda es el router interno a través del que salen los usuarios de la red.

Todas las direcciones utilizadas en las redes internas son direcciones privadas, utilizándose como dirección pública la del interfaz externo del router de salida a Internet y la siguiente. Para permitir la navegación de los usuarios de la red a través de Internet, se realiza NAT(Network Address Traslation) en el firewall, intercambiando las IPs privadas por las IPs publicas. Por motivos de diseño previos a la realización de este proyecto, todo el tráfico es NATeado a una unica dirección publica, excepto el tráfico FTP(File Transfer Protocol), que se NATea a otra diferente. Esta consideración tiene que ser respetada en la solución propuesta, ya que afecta a servicios externos a la red, que esperan recibir el tráfico con dicha IP origen.

6.2 Solución propuesta

Debido a la antigüedad del firewall instalado y a las nuevas necesidades, tanto de funcionalidades como de troughput, derivadas del aumento de usuarios de la red, se opta por sustituir la salida a Internet por una nueva solución que aporte mayor seguridad y disponibilidad. Con esos objetivos en mente se diseña la siguiente arquitectura de red:

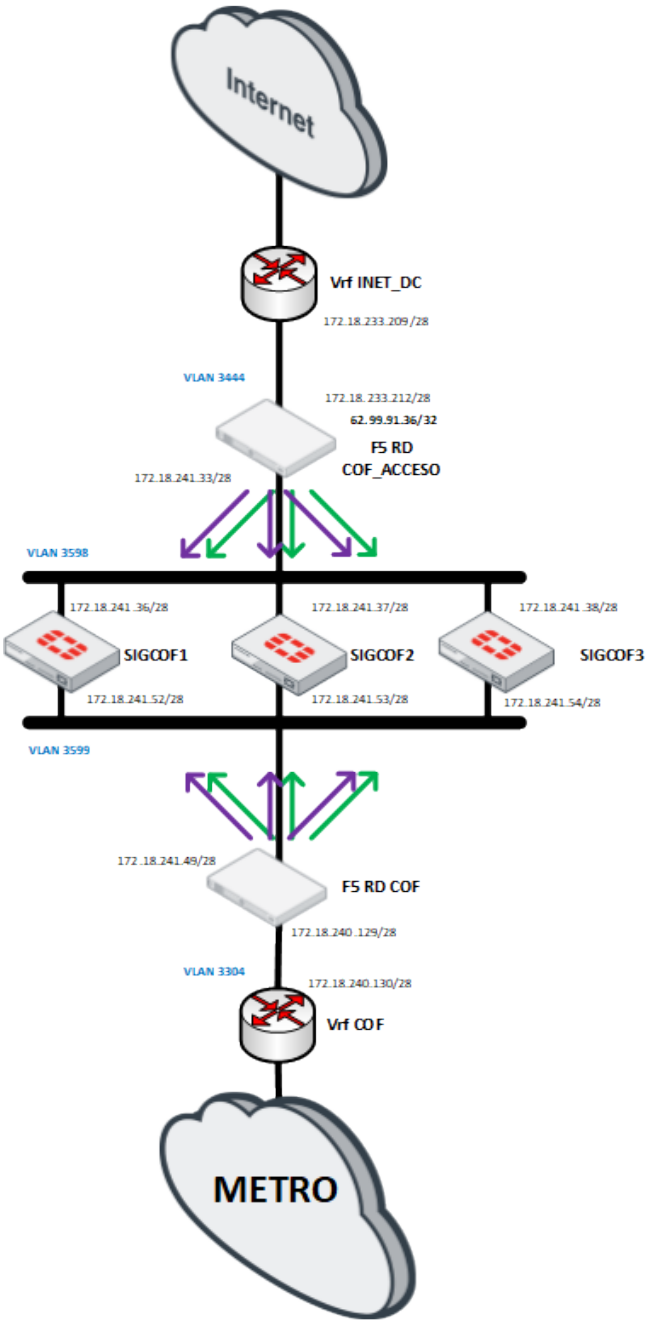


Figura 6.2: Esquema de red de partida

En la nueva arquitectura lógica, el firewall existente previamente se ha sustituido por tres firewalls funcionando en paralelo, cuya distribución de carga queda en manos de dos balanceadores de tráfico, colocados en ambos extremos. En cuanto al equipamiento físico en el que se ha implementado esta solución, consiste en tres equipos FortiGate 1000D[3], adquiridos expresamente para este proyecto y que por ahora no tienen uso en ningún otro proyecto y dos equipos F5 BIG-IP[4] previamente adquiridos y de uso compartido para otros proyectos. Para la configuración de los firewalls se dispone de un equipo FortiManager[5] pre-existente. En el ecosistema de productos de Fortinet, los equipos FortiManager son los encargados de gestionar múltiples firewalls FortiGate u otros equipos de Fortinet. De esta forma, desde un único equipo, es posible realizar el aprovisionamiento, configuración y mantenimiento de un alto número de otros equipos.

La forma de compartir un mismo equipo físico para varios proyectos o clientes de forma simultanea varia de un fabricante a otro. En el caso de los firewalls de Fortinet se hace mediante lo que se denomina VDOM(Virtual Domain). Un VDOM o dominio virtual es el método del que disponen los equipos FortiGate para dividir una sola unidad física en dos o más unidades totalmente independientes. Cada una de ellas dispone de sus propias configuraciones, políticas de firewall, rutas, NATs... Esta solución es muy útil en escenarios en los que se necesitan varios firewalls con requisitos sensiblemente inferiores a los de la unidad física o en entornos de los conocidos como multi-tennant, en los que un proveedor dispone de varios clientes. Utilizando las VDOMs el proveedor puede utilizar un solo equipo para dar servicio a más de un cliente. Para este proyecto se configura en cada uno de los firewalls físicos un VDOM, de forma que en un futuro puedan ser aprovechados para nuevos proyectos. Del mismo modo, Fortinet dispone de ADOMs (Administrative Domain) para los equipos FortiManager. De esta forma es posible asignar equipos completos o VDOMs a cada ADOM, separando la gestión de distintos equipos o redes. Además, es posible restringir el acceso a los ADOMs en función del usuario, por lo que es una solución idonea para proveedores con múltiples clientes.

En los balanceadores de F5, la solución consiste en crear una partición separada, que funciona de forma similar a las VDOMs de Fortinet, permitiendo crear varias instancias virtuales del balanceador funcionando de forma independiente sobre un mismo equipo físico. En cada uno de los balanceadores se crea una partición para este proyecto.

Los firewalls funcionan en modo activo todos ellos, sin ninguna configuración de alta disponibilidad específica, ya que, al realizarse balanceo entre ellos, en caso de caída de cualquiera de los tres equipos, podría seguir cursándose el tráfico de forma transparente excepto para las sesiones establecidas, que si que se verían afectadas.

En los balanceadores, en cambio, se configuran las dos particiones en modo Activo-Pasivo, de forma que se dispone de un solo balanceador virtual. Esto a priorí puede parecer contradictorio con lo que se muestra en el esquema de red, pero no lo es. Dentro de las particiones se configuran dos routing domains diferentes. Los routing domains de F5 son objetos que permiten separar el tráfico de red de una aplicación o red en particular, aplicando distintas tablas de rutas o recursos. De esta forma, mediante los diferentes routing domains podemos separar

las redes internas de las redes externas y de acceso a Internet, pudiendo utilizar una única partición como si estuviera colocada en dos lugares distintos de la red de securización.

Por tanto, el camino de salida a Internet desde las redes internas atraviesa el router Vrf-Cof, que tiene como default gateway el balanceador COF_Accceso. Este utiliza el método de balanceo "least connections", de forma que envía la conexión al firewall que se encuentre en ese momento con el menor número de conexiones activas, de forma que todos los firewalls reciban un volumen de tráfico similar. Los firewalls, una vez comprobado que la conexión ha de ser permitida en base a sus reglas y políticas, envían el tráfico al balanceador F5RD_COF, en el que se realiza el NAT correspondiente para permitir la navegación hacia Internet. Finalmente, el tráfico nateado es enviado al router Vrf Inet_DC, que lo enrutará a Internet del mismo modo que hasta ahora. El caso de los paquetes de vuelta de las conexiones establecidas desde las redes internas, es necesario garantizar que el F5RDCOF los envía al firewall que ha cursado la conexión de salida. En caso de que esto no ocurra así, la comunicación no sería posible, porque tendríamos una situación de routing asimétrico, en la que el firewall que cursa el paquete de ida no es el mismo que el que cursa el paquete de vuelta. El comportamiento de los firewalls cuando reciben un paquete de una conexión que no han visto su inicio y por tanto, no tienen en su tabla de conexiones, es realizar un drop del paquete, impidiendo que se complete la conexión. Para evitar esta situación los equipos de F5 disponen de una funcionalidad que viene automáticamente activada que es la llamada "Auto Last Hop". Es el propio balanceador el que almacena una tabla de conexiones activas y en ella tiene asociado el firewall que está cursando cada conexión, de forma que siempre entrega los paquetes de una misma conexión a un mismo firewall. Gracias a esta función se evita el routing asimétrico y las comunicaciones pueden establecerse de forma bidireccional.

Finalmente, cabe destacar que no es necesario realizar ningún cambio ni modificación de configuraciones en la red interna ni el en los routers, ya que las IPs externas del firewall anterior se mantienen en los extremos externos de los balanceadores, permitiendo que la migración de una configuración a otra sea totalmente transparente.

6.2.1 Política de seguridad

Para configurar estos equipos se ha creado una política de seguridad a implementar en los tres firewalls que securizan el acceso a la red. Esta política permite el tráfico saliente de la red de privada hacia Internet para la navegación, así como los accesos desde dicha red a los servicios que se consideran oportunos para el correcto funcionamiento de la misma. Todo el resto del tráfico es bloqueado.

En la imagen 6.3 puede verse la implementación de la política de seguridad mencionada anteriormente aplicada en los equipos de Fortinet.

Seq.#	Name	From	To	Source	Destination	Service	Action	Security Profiles
1		any	any	Interna_10.0.0.0_14	all	TCP4000-4010	Accept	
2	Datafonos	any	any	Interna_10.0.0.0_14	all	TCP30001 TCP3001 TCP6003 TCP7002 TCP7020-7023 TCP7102 TCP7262	Accept	
3		any	any	Interna_10.0.0.0_14	FQDN-certificados_digitales	TCP8093-8094	Accept	
4		any	any	Interna_10.0.0.0_14	FQDN-TPV_PC_Pago_Digital	HTTPS	Accept	
5		any	any	Interna_10.0.0.0_14	FTP_Server	FTP	Accept	
6		any	any	Interna_10.0.0.0_14	Web_Pedidos	ALL	Accept	
7		any	any	Interna_10.0.0.0_14	Web_Propia	HTTP HTTPS	Accept	
8		any	any	Interna_10.0.0.0_14	FQDN-SMTP_Server	SMTP	Accept	
9		any	any	Interna_10.0.0.0_14	all	ALL	Accept	default Webfilter_Personalizado ApplicationControl_Personalizado default
Implicit (10-10 / Total:1)								
10	Implicit Deny	any	any	all	all	ALL	Deny	

Figura 6.3: Configuración de la política de seguridad

La política de seguridad se traduce en una serie de reglas que permiten o deniegan el tráfico en función de una serie de parámetros. En este punto es necesario diferenciar dos tipos de firewalls distintos, en función de sus capacidades y funcionalidades.

Por un lado, existen los firewalls llamados de capa cuatro. Estos equipos únicamente inspeccionan el tráfico a nivel de red, es decir, el tráfico IP. Por tanto, las reglas que pueden ser configuradas en estos equipos hacen referencia a las direcciones IP origen y destino de los paquetes, los puertos de entrada o salida y el servicio al que esta dirigido, entendiendo como tal el puerto TCP/UDP o el servicio IP.

Los firewalls tradicionales eran firewalls de capa cuatro. Sin embargo, de un tiempo a esta parte han aparecido los firewalls denominados de capa siete. Estos equipos son capaces de inspeccionar el tráfico de nivel de aplicación, ofreciendo muchas ventajas, especialmente en cuanto a granularidad de la seguridad se refiere. Además, el firewall utilizado es un firewall con estado, es decir, que mantiene una tabla de conexiones, gracias a la cual el equipo esta al corriente de las sesiones y comunicaciones que se han establecido a través de él. Esto permite, entre otras cosas, evitar la duplicación de las reglas de firewall. Si el firewall fuera sin estado, para permitir una comunicación bidireccional sería necesario crear dos reglas, una para cada dirección. Sin embargo, al disponer de estado, el número de reglas se reduce a la mitad.

Para poder crear las reglas, es necesario definir primero los objetos que hacen referencia a las direcciones y redes deseadas. Estos objetos serán referenciados en las reglas. Para este caso se han definido los objetos de dos formas distintas, en función de su tipo. La mayoría de ellos los definimos con la combinación de IP/Máscara. En caso de no especificar, el equipo asume que la máscara es 255.255.255.255. Sin embargo, varios de los objetos los definimos de tipo FQDN (Fully Qualified Domain Name[6]). Este nombre incluye el nombre del equipo y el de dominio asociado a el, separados por puntos. Para obtener la IP de este equipo, los firewalls mantienen un cache de todos los DNS records que se han resuelto para los FQDN definidos. De esta forma no es necesario conocer la IP de los servidores externos y se protege contra posibles cambios de la misma. Del mismo modo es útil para todos aquellos servicios que hacen balanceo de tráfico.

En este caso, se han creado 9 reglas para permitir el tráfico más una décima que el propio equipo añade. La última regla, conocida como implícita, tiene como objetivo evitar que en el caso de que la configuración tenga algún descuido y no bloquee un tráfico no legitimo, este sea bloqueado.

Las otras 9 reglas tienen como origen la red interna y diversos destinos y servicios. La primera permite a los equipos internos acceder a los servidores de actualización situados en la red pública para descargarse actualizaciones. La segunda regla, permite a los datáfonos conectarse a través de los puertos conocidos para sus comunicaciones a la hora de realizar los pagos. Las reglas de la 3 a la 8 permiten acceder a distintos servidores situados fuera de la red interna y que es necesario que sean accedidos por los usuarios. Cada una de las reglas permite solamente el acceso a dicho servidor a los puertos que se necesitan. Es decir, la regla 8 permite

acceder al servidor SMTP(Simple Mail Transfer Protocol) únicamente a través del puerto de SMTP, que es el 25 [7]. En las reglas 3,4 y 8 utilizan objetos FQDN para conocer la dirección IP de destino. Como puede verse, todas las reglas definidas tienen Accept como acción, ya que el objetivo es permitir el tráfico.

Las ocho primeras reglas definidas se corresponden con reglas de capa de red, ya que únicamente tienen en cuenta las direcciones IP y el servicio. Sin embargo, la novena regla dispone de unos perfiles de seguridad que inspeccionan las aplicaciones y toman una decisión al respecto. Esta regla es la que da salida a Internet a la red interna y es por eso que se han añadido los perfiles de seguridad.

La regla dispone de tres perfiles de seguridad:

- Antivirus
- Web Filter
- Control de aplicaciones

Antivirus

El módulo antivirus permite al equipo detectar viruses y todo tipo de payload potencialmente malicioso capaz de infectar las maquinas de la red. Para ello, el módulo dispone de una base de datos de firmas de viruses conocidos, de forma que mediante comparación entre la firma de los ficheros que atraviesan el firewall y la base de datos, es capaz de llevar a cabo la acción correspondiente. Para ello, el equipo necesita el fichero completo, por lo que dispone de un buffer en el que va almacenando el fichero completo para poder escanearlo posteriormente. Mientras lleva a cabo la operación, envia información aleatoria al equipo final, de forma que este no crea que la transferencia ha sido fallida. Una vez completado el escaneo se envia el fichero al cliente o si se detecta que es un archivo malicioso, se bloquea y envia una página de aviso al usuario. Si el archivo es mayor que el tamaño del buffer no es posible realizar el escaneo, por lo que es necesario configurar si se dejan pasar esos ficheros sin escanear o se bloquea todo lo que sea mayor que el buffer. En este caso, se bloquean todos los ficheros de un tamaño mayor, para evitar problemas de seguridad. Alternativo a este funcionamiento que se ha comentado, que se conoce como Proxy, existe otro modo de inspección, que es el Flow-Based. Este modo evita las limitaciones del modo Proxy, ya que al inspeccionar cada paquete individualmente, no requiere de ningún tipo de buffer, evitando delays y tamaños máximos de fichero. Sin embargo, no se opta por esta opción, ya que la tasa de infecciones detectada es sensiblemente menor que en el modo Proxy, por lo que se decide primar la seguridad.

Para la regla de salida a Internet de la política de seguridad implementada se aplica el perfil default que viene creado automáticamente para el antivirus en los equipos Fortinet. Este perfil funciona en modo Proxy, como se ha comentado. Una vez detectado en un virus, podemos escoger si bloquearlo o monitorizarlo, se opta por la primera opción. Además, el el propio

perfil se puede escoger que protocolos serán inspeccionados y si se permiten los ejecutables de Windows como viruses si vienen adjuntados a un correo. La configuración de esta regla se puede ver en la figura 6.4.

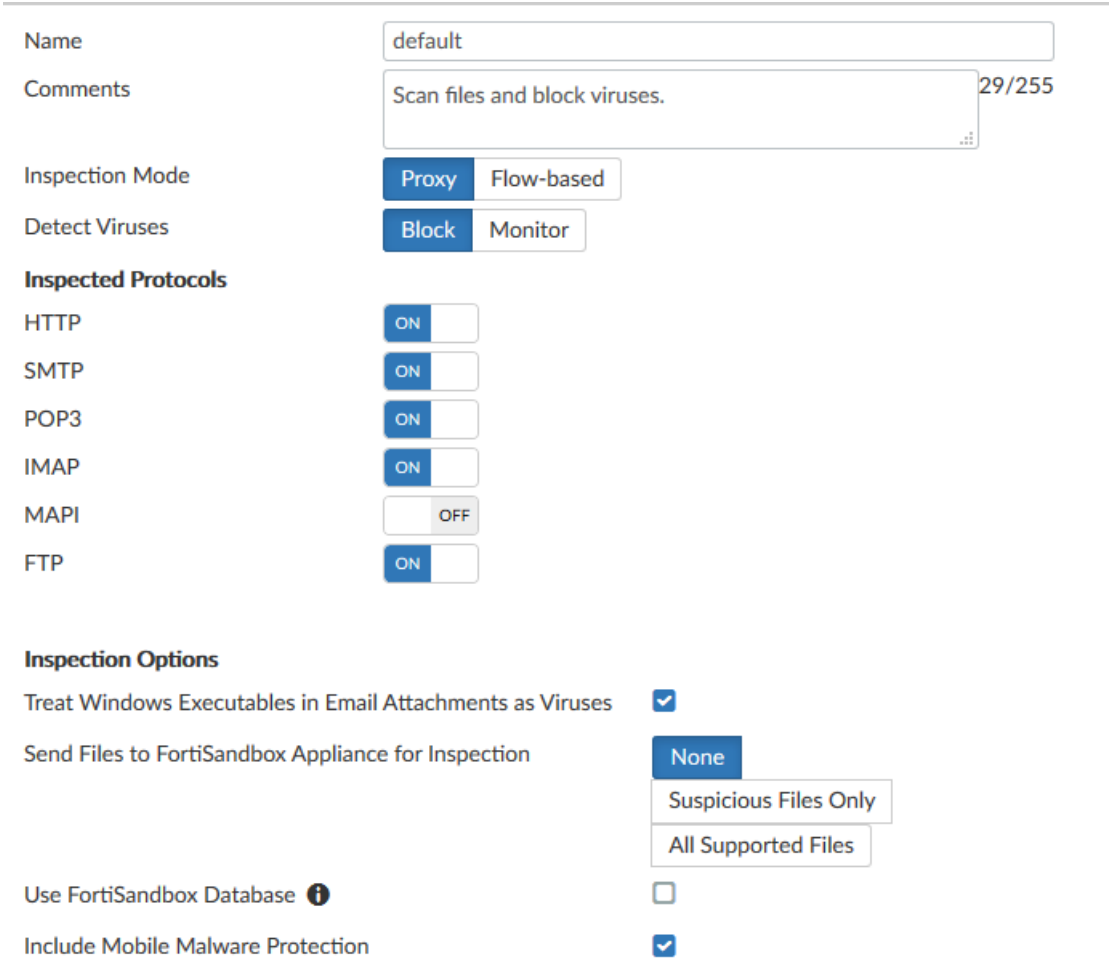


Figura 6.4: Configuración del perfil antivirus por defecto.

Web Filter

Este módulo permite controlar el contenido que esta siendo accedido por los usuarios en Internet. Es importante realizar un seguimiento de las páginas web que están siendo accedidas por distintos motivos, desde perdida de productividad al acceder a páginas de entretenimiento que no guardan relación con la actividad realizada hasta casos más extremos como acceso a contenido o material ilegal. Otra de las preocupaciones más graves es evitar el phising[8], en el que una página web fraudulenta se hace pasar por la página web de algún servicio legitimo para conseguir que el usuario introduzca sus claves. Para llevar a cabo la tarea de identificar cada página web es necesario inspeccionar los paquetes. Del mismo modo que en el antivirus esto puede hacerse de dos formas, almacenando la página web completa (Modo Proxy)

o inspeccionando paquete a paquete (Modo Flow-Based). Igual que en el antivirus, el modo Flow-Based es más rápido pero con un índice de acierto menor. Adicionalmente, Fortinet dispone de una solución accesible via subscripción llamada FortiGuard Web Filter. Este servicio dispone de billones de paginas web categorizadas en función de su contenido. Esto permite bloquear grandes grupos de páginas web de golpe, bloqueando únicamente una categoría, como por ejemplo redes sociales o armas. Adicionalmente, es posible bloquear URLs concretas bajo demanda.

En el caso del proyecto se crea un perfil de Web Filter de nombre Webfilter_Personalizado. Se configura para que realice la inspección en modo proxy y se seleccionan las categorías que se desean bloquear o permitir. Tambíen es posible, en lugar de bloquear o permitir, permitir con un warning, monitorizar o solicitar autenticación para entrar a la página en cuestión. En este caso unicamente se bloquean algunas y se permiten las que se consideran legitimas. Todas aquellas que FortiGuard considere que son un riesgo de seguridad, son bloqueadas. En este caso no se configura ningún bloqueo estático de ninguna URL. La configuración se puede ver en la figura 6.5.

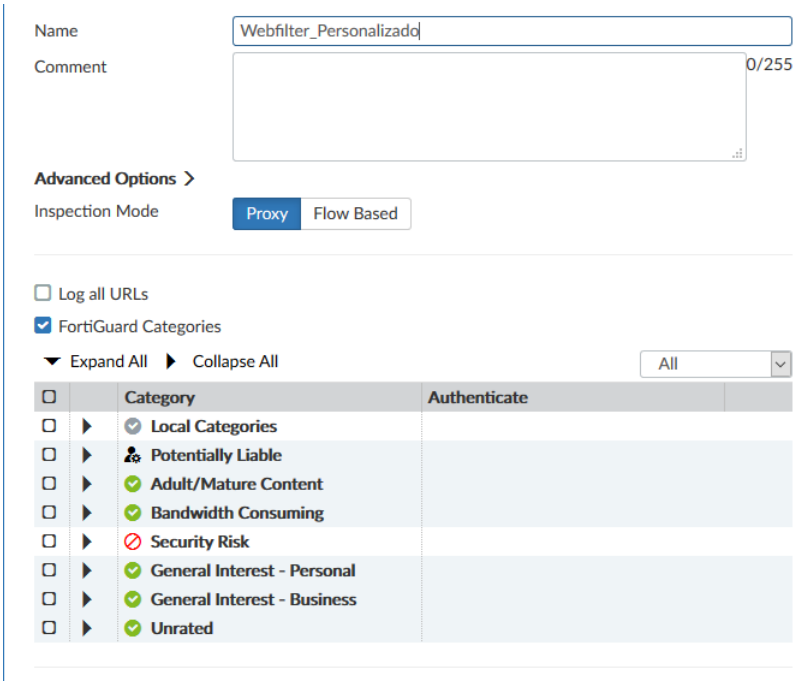


Figura 6.5: Configuración del perfil de web filter creado.

Control de aplicaciones

El módulo de control de aplicaciones es capaz de reconocer el tráfico de una gran cantidad de aplicaciones, de forma que se pueda reconocer el patrón de un ataque por un lado o reaccionar a comportamientos específicos por porta de una aplicación. Esto permite que puedan ser bloqueadas solo ciertas funcionalidades de un protocolo por ejemplo. Este módulo

se interrelaciona estrechamente con el modulo IPS (Intrusion Prevention Sistem), ya que el comportamiento concreto de un protocolo puede corresponderse a una firma de un ataque. Del mismo modo, es posible bloquear aplicaciones enteras, en función de las necesidades. Por defecto, Fortinet categoriza las aplicaciones en una serie de categorías y permite decidir que hacer para cada categoría. Las acciones son:

- Block: bloquea el tráfico de dicha aplicación impidiendo el funcionamiento de la misma.
- Allow: se permite el uso de dicha aplicación.
- Monitor: el tráfico es permitido pero se monitoriza todo lo que pasa.
- Traffic shaping: el firewall permite definir perfiles de Traffic Shaping, con los que limitar el ancho de banda, la cantidad de información transmitida... para cada aplicación.
- Quarentine: bloquea el tráfico durante un periodo de tiempo definido.

Se ha creado el perfil ApplicationControl_Personalizado, en el que se bloquean las categorías de botnet (para evitar ataques desde ordenadores infectados pertenecientes a redes de bots) y proxy (para evitar el tráfico cuyo origen es desconocido. El resto de categorías son monitorizadas, de forma que el tráfico es loggeado para su posterior análisis en caso de ser necesario.

En la regla configurada en este proyecto, se asignan cada uno de los perfiles de uno en uno. Sin embargo, otra opción podria ser crear un grupo de perfiles, de forma que se asignen todos de manera simultanea. Esto es muy útil para crear distintos niveles de privilegios, como navegación básica, avanzada... En este caso, dada la naturaleza de los usuarios no se ha considerado realizar este tipo de distinciones.

Adicionalmente, para aumentar la seguridad y la granularidad, Fortinet, permite realizar la categorización del tráfico en función de inbound-outbound. Esto hace referencia a que es posible añadir los interfaces de entrada y salida a una regla. Por un lado, esto aumenta la seguridad, ya que se garantiza que el paquete proviene de donde debe, evitando el spoofing, es decir, el envío de tráfico usando direcciones ilegitimas. Por otro lado, aumenta la granularidad por que si, por algún motivo una misma dirección o red puede llegar al firewall por dos interfaces distintos, es posible tratarlo de forma diferente. Además, simplifica la visualización de las reglas ordenándolas en función del origen y destino.

6.2.2 SSL-VPN

Para permitir el acceso remoto a los equipos de la red interna, es necesario configurar algún tipo de solución VPN que permita a los usuarios que estén fuera de la red navegar como si estuvieran dentro de la propia red privada. Las utilidades y ventajas de las VPNs son ampliamente conocidas, por lo que no se hará hincapié en el presente documento.

Se ha optado configurar la solución VPN basada en SSL, de forma que los usuarios puedan conectarse accediendo a una pagina web desde el navegador. En dicha página los usuarios introducen sus credenciales de acceso y una vez autenticados, se les muestra un portal personalizable desde el que se permite el acceso a los recursos particulares a los que deba acceder cada usuario.

Esta configuración se lleva a cabo en los firewalls de Fortinet, en los mismos VDOMs en los que se configuran las políticas de acceso, centralizando las operaciones de management. Como se ha comentado anteriormente, los FortiGates son administrados mediante un equipo dedicado únicamente a esa función, llamado FortiManager. La configuración de aprovisionamiento básico, de interfaces, políticas, creación de objetos… se hace desde el FortiManager, lo que simplifica enormemente el despliegue en entornos en los que existe más de un equipo a configurar.

Además, se desea permitir acceso a través de la VPN a un número elevado de usuarios y no se dispone de integración con ningún tipo de servidor TACACS+[9], RADIUS[10] o Directorio Activo, por lo que será necesario ir creando todos los usuarios. Tanto TACACS+ como RADIUS son protocolos de autenticación de usuarios definidos por el IETF en sus respectivas RFCs, mientras que el directorio activo es la implementación de servicio de directorio en una red distribuida de Microsoft. En todos los casos, son diversas formas de autenticar usuarios, con sus correspondientes permisos. Es posible integrar los equipos de Fortinet con servidores de cualquiera de los tres, pero queda fuera del alcance del proyecto.

La operación de configuración de VPN, creación de portales, usuarios y el resto de configuraciones asociadas resulta un trabajo repetitivo y que ha de realizarse en tres equipos. La forma óptima de abordar el problema, es automatizando la operación en su conjunto, permitiendo ahorrar una cantidad considerable de tiempo y simplificando el despliegue de futuros equipos que puedan requerir una configuración similar, además de evitar posibles errores humanos de configuración. Para la automatización, se ha desarrollado un ejecutable, que, con la entrada de datos adecuada, crea los usuarios necesarios, los portales necesarios y ejecuta la configuración de la VPN para permitir el acceso de dichos usuarios a los recursos especificados. Todas las operaciones a realizar en el FortiGate se hacen mediante el uso de la API que el fabricante proporciona en los equipos.

El ejecutable se ha creado a partir de un Script escrito en Python que obtiene los datos de entrada del equipo, un .csv con la información de los usuarios a configurar y se comunica con el FortiGate via API para realizar las configuraciones.

El formato del fichero .csv mediante el que se nutre al script es el siguiente:

```
User;Pass;Resource;DstInt
```

El campo Resource hace referencia a los recursos internos a los que se desea que el usuario tenga acceso. Para cada usuario o grupo se crea una regla que únicamente permite acceso a

los recursos y direcciones identificados en el csv(comma-separated values), evitando que los usuarios de la VPN dispongan de libre acceso a todos los equipos de la red. Del mismo modo en DstInt se hace referencia a la interfaz por la que se va a permitir la salida del tráfico de dichos usuarios. En el caso que nos ocupa será siempre la interfaz interna de los FortiGates, ya que el objetivo de la VPN es permitir acceso a los recurso internos.

El script procesa los datos del csv y del equipo, crea el usuario y lo añade al grupo de usuarios de VPN, a continuación crea el portal personalizado para dicho usuario y se lo asigna. Finalmente crea la política que se ha comentado anteriormente. El pseudocódigo que resume el comportamiento del script es el siguiente:

```
1: Lectura de variables externas y de fichero csv;
2: Apertura de conexión con FortiGate y autenticación;
3: Creación del grupo de usuarios;
4: for linea del csv do
5:     Crear usuario;
6:     Añadir al grupo;
7: end for
8: Crear portal VPN y asignarlo;
9: Crear regla de acceso;
10: Cerrar conexión con FortiGate;
```

El código completo del script puede verse en el anexo I .

Cuando un usuario quiera conectarse a la VPN, tendrá que conectarse contra la IP del FortiGate al puerto en el que se ha definido el log in del portal. Una vez hecho esto, podrá introducir su usuario y password para acceder a su portal personalizado. Desde el portal podrá acceder a los recursos a los que tiene permiso.

Además de permitir el acceso web, los usuarios pueden descargarse el cliente de VPN FortiClient y acceder mediante el cliente pesado. Al igual que en el caso de cliente de navegador, solo tendrán acceso a los recursos contemplados.

6.2.3 Acceso a cámaras IP

Uno de los objetivos del proyecto es permitir el acceso remoto a las cámaras de videovigilancia situadas en la red Interna. Estas cámaras disponen internamente de un servidor web al que se puede acceder mediante un navegador para recibir el vídeo en tiempo real. Sin embargo, las cámaras disponen de un direccionamiento privado y es necesario diseñar algún tipo de mecanismo para acceder a las mismas. La solución implementada pasa por el uso de Virtual Servers en los balanceadores. Como se ha comentado anteriormente, mediante los Virtual Servers es posible tratar el tráfico de formas totalmente personalizadas en función del destino y otras variables, aplicándole perfiles de tráfico, asignándole recursos...

Para cada una de las cámaras de videovigilancia a la que se desea permitir acceso, es necesario crear un nuevo Virtual Server, al que se le asignará como recurso la cámara en sí. A priori, un cuello de botella podría ser el número de virtual servers a configurar. Sin embargo, el límite de virtual servers que puede manejar un equipo viene dado por la memoria de la que dispone. Los equipos de este proyecto no tienen apenas carga, por lo que aun con las estimaciones menos optimistas (4000-5000, según las propias fuentes de F5), no debería ser un problema.

Otra de las ventajas de los Virtual Servers es la posibilidad de configurar health monitors. Mediante esta herramienta el propio balanceador se encarga de monitorizar constantemente el estado del recurso, permitiendo conocer al instante si el recurso no está disponible. En el caso de las cámaras de videovigilancia, el health monitor aplicado es el de http/https, ya que como se ha comentado anteriormente, internamente dispone de un servidor web. También se aplicará el perfil de tráfico de http/https, como es lógico.

La configuración de un Virtual Server resulta sencilla. Es necesario asignarle un nombre y una dupla IP:puerto, de forma que el tráfico con destino a esa dupla será tratado por el Virtual Server. También es necesario indicar el pool de recursos a los que se reenviará el tráfico. Opcionalmente también pueden configurarse más parámetros, como es el caso del health monitor o el perfil de tráfico que se han comentado antes, entre otros.

Para este proyecto se ha optado por crear virtual servers con la IP externa del balanceador y con puertos consecutivos para cada una de las cámaras de videovigilancia, comenzando en el 10100. Sin embargo, el número de cámaras a configurar es tan elevado, superior a 600, que al igual que los accesos VPN, se identifica la necesidad de automatizar el proceso.

Con ese objetivo en mente, se ha creado otro script en Python, que al igual que en el caso anterior, se nutre de un fichero .csv para realizar la configuración.

El formato del fichero es el siguiente:

```
NodeName;Address;PoolName;VirtualServerName
```

El script se encarga de procesar para cada línea del fichero y crear un nuevo Virtual Server para cada una de ellas, que representan las cámaras de videovigilancia. El pseudocodigo del script es el siguiente:

```
1: Lectura de variables externas y de fichero csv;
2: Apertura de conexión con F5 y autenticación;
3: for linea del csv do
4:     Crear recurso;
5:     Crear pool;
6:     Crear virtual server;
7: end for
8: Cerrar conexión con F5;
```

El código completo del script puede verse en el anexo II .

Por tanto cuando el F5 detecte una conexión http o httsp con destino a su IP y con puerto destino uno de los del rango (a partir del 10100), redirigirá la conexión al pool de recursos asociado. En este caso cada pool unicamente tiene un recurso, que será la IP privada de la cámara de videovigilancia objetivo.

6.3 Conexión Física

Los equipos utilizados han de conectarse físicamente para permitir la comunicación. Es importante destacar que la topología física difiere en aspecto de la topología lógica. Todos los equipos, tanto los firewalls como los balanceadores se conectan físicamente a ambos routers físicos. Los routers que pueden verse en la conexión lógica, corresponden a dos routing domains diferentes.

Todas las conexiones se realizan mediante cables de fibra óptica de capacidad de 10Gigabits. Los firewalls disponen de dos puertos capaces de soportar velocidades de 10G, que son los nombrados como A y B. Los puertos se corresponden a los indicados con el número 7 en la figura 6.6. Cada uno de ellos se encuentra en una tarjeta física de puertos diferente por motivos de diseño y fabricación. Esto implica que no es posible configurarlos en modo port-channel, es decir, configurarlos de forma que se traten como un solo puerto lógico, lo que duplicaría la capacidad del enlace y aumentaría la disponibilidad. Al no existir esta posibilidad, que hubiera sido la configuración más adecuada, se opta por cablear ambos interfaces hacia cada uno de los routers, pero configurar únicamente un puerto en cada uno de ellos. Se ha configurado el puerto A en dos de los equipos y el B en otro, es decir, dos de ellos van a un router y el tercero al otro. Esto garantiza la disponibilidad en caso de caída de cualquira de los tres firewalls, de cualquier enlace o de cualquiera de los dos routers, si bien es cierto que en caso de que la caída ocurra en el router 1809, es necesario que el firewall conectado al 2809 pueda cursar el tráfico agregado de los tres firewalls.

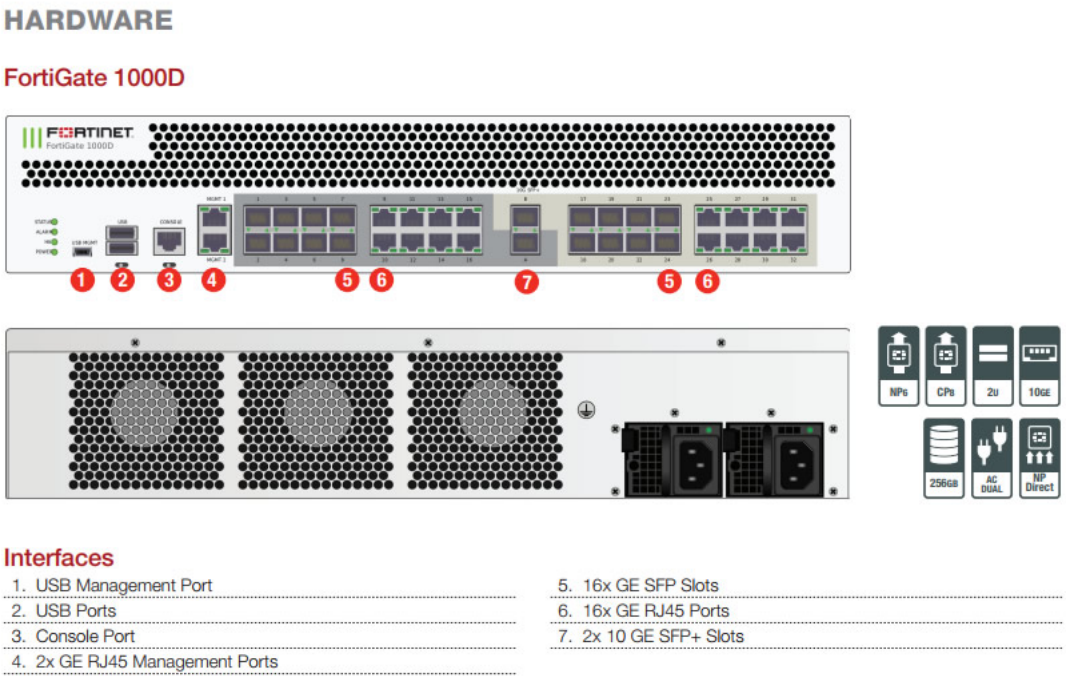


Figura 6.6: Puertos físicos del Fortigate 1000D.

A día de realización del proyecto, los firewalls son de uso exclusivo para este proyecto, pero esto puede cambiar en un futuro. Gracias a esta configuración en el cableado, podrían utilizarse los enlaces redundantes para los nuevos usos que pudieran surgir.

Respecto de los balanceadores F5, estos disponen de cuatro puertos de 40Gb. Cada uno de estos interfaces tienen la peculiaridad de que pueden utilizarse como un solo puerto de 40Gb o como cuatro puertos de 10G, mediante un cable de breakout. Este tipo de cable permite separar una única interfaz física en cuatro interfaces físicas diferentes, repartiendo la capacidad de forma equitativa. Para este proyecto se utilizan los cables de breakout, utilizando dos puertos físicos del F5 y uno de los cuatro extremos, uno para cada router. Quedan otros tres extremos de los cables de breakout libres para futuros usos. La relación de puertos físicos configurados en los balanceadores y firewalls puede verse en la tabla 6.1

6.4 Direccionamiento

Para la implementación de la arquitectura se han definido cuatro VLANs(Virtual Local Area Network), que permiten separar las distintas redes que interconectan los equipos de comunicaciones. La nomenclatura utilizada en cada equipo, el tag asignado y el direccionamiento de cada VLAN puede verse en la tabla 6.2. Las VLANs 3444 y 3304 son las que unen los routers con los balanceadores F5, mientras que las VLANs 3598 y 3599 son las de las redes entre los balan-

Equipo	Puerto	Destino
DC809FW1	A	ACS1809
DC809FW1	B	ACS2809
DC809FW2	A	ACS1809
DC809FW2	B	ACS2809
DC809FW3	A	ACS1809
DC809FW3	B	ACS2809
ADCpp1	1/1.7	ACS1809
ADCpp1	1/1.9	ACS1809
ADCbk1	1/1.7	ACS1809
ADCbk1	1/1.9	ACS2809

Tabla 6.1: Cableado físico de los puertos.

ceadores y los firewalls. Cada equipo tiene una IP en cada una de las redes a las que pertenece (como se ve en la figura), excepto los balanceadores, que al estar en HA(High Availability) tienen dos IPs en cada una de las redes, una propia para identificarlo en esa red y la otra flotante y compartida por ambos para ofrecer alta disponibilidad. En la imagen se muestra la IP virtual en cada caso. Adicionalmente se consideran las redes privadas: 10.0.0.0/8, 192.168.0.0/16 y 172.16.0.0/20 que pertenecen a la red Metro interna. Todo el resto de rangos se enrutan hacia la salida a Internet.

VLAN	Direccionamiento	FortiGate	F5
3444	172.18.233.208/28	-	COF_ACCESO_3444
3598	172.18.241.32/28	COF_ACCESO	VRFINET_DC_COF
3599	172.18.241.48/28	COF	VRFCOF
3304	172.18.240.128/28	-	COF_3304

Tabla 6.2: Direccionamiento y nomenclatura de las VLANs.

7 Metodología

Una vez implementado el diseño de red, cableados y configurados los equipos de la forma que se ha explicado anteriormente, se comprueba el correcto funcionamiento de la solución. Las pruebas realizadas para ello consisten en probar la navegación y los servicios accedidos desde la red interna y que se encuentran fuera de la misma.

Una vez completadas las pruebas de forma satisfactoria, se procede a comprobar el correcto funcionamiento de las herramientas de automatización generadas. Al ser un entorno de producción en el que se ha desplegado la solución, no es recomendable probar las herramientas de automatización directamente sobre el mismo, por lo que se diseña una maqueta de laboratorio sobre la que probar el funcionamiento.

La topología de la maqueta es la siguiente:

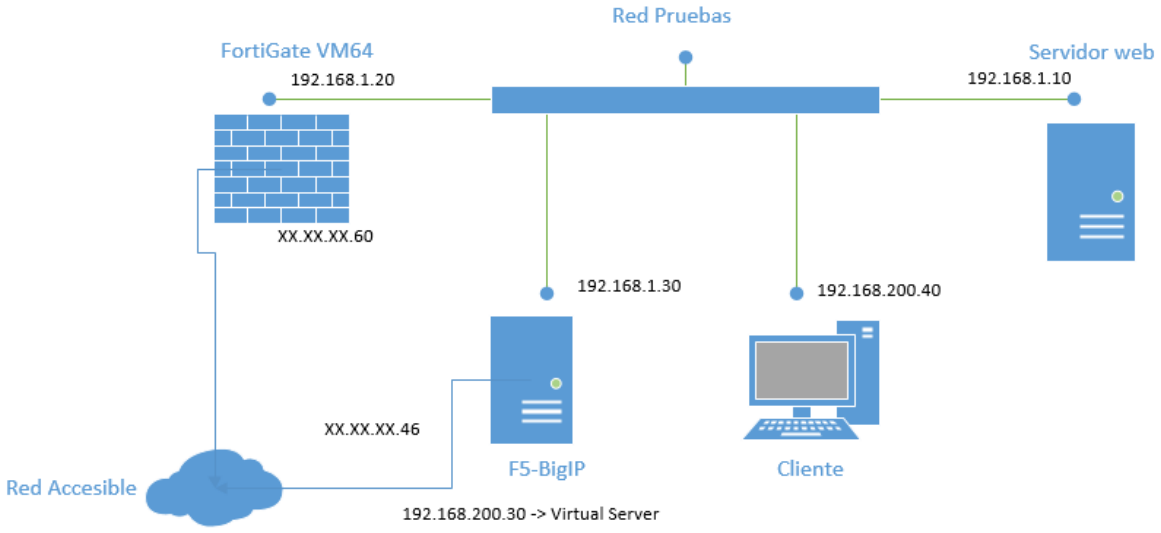


Figura 7.1: Configuración de la politica de seguridad

Se trata de una serie de maquinas virtuales alojadas en un servidor de VMWare. En dicho servidor se han configurado dos switches virtuales, Red_Accesible y RED_PRUEBAS. El primero de los swithes permite la navegación a Internet y tiene conectado un servidor DHCP(Dynamic Host Configuration Protocol), de forma que las maquinas virtuales conectadas al mismo recibirán una dirección IP y podrán ser accedidas de forma remota. El switch RED_PRUEBAS esta aislado, de forma que los equipos conectados al mismo unicamente tendran visibilidad de equipos que tambien se hallen conectados a él. Los switches virtuales en este caso no disponen de ninguna configuración adicional en cuanto a seguridad o VLANs, por lo que no hay ningún tipo de separación de dominios de broadcast o similares.

Como se ve en la figura 7.1, se configuran dos equipos con interfaces en ambas redes y otros dos únicamente conectados al switch de la red de pruebas. Los dos equipos conectados a ambos switches virtuales son dos maquinas virtuales en las que se emulan un equipo F5 BigIP y un firewall Fortigate VM64. Ambos equipos podrán ser accedidos y gestionados desde la red externa, ya que como se ha comentado anteriormente reciben mediante DHCP IPs pertenecientes a un rango rutable desde la red en la que se está trabajando. El resto de maquinas, es decir, el servidor web y el cliente, únicamente se encuentran conectadas a la RED_PRUEBAS, de forma que no se puede acceder a ellos desde el exterior. El servidor web es una maquina Debian en la que se instala un servidor Apache 2.4.25. El cliente es una maquina con Microsoft Windows 7 en el que se instala un navegador para poder acceder a la página web.

En la tabla 7.1 se pueden ver cada uno de los equipos, el sistema operativo instalado y a que switches esta conectado, asi como la IP en cada interfaz.

Nombre	Sistema	Red Accesible	Red Puebas
FortiGate VM64	FortiGate v5.4.8	XX.XX.XX.60	192.168.1.20
F5-BigIP	BIG-IP 12.1.1	XX.XX.XX.46	192.168.1.30
Cliente	Debian	-	192.168.200.40
Servidor Web	Windows 7	-	192.168.1.10

Tabla 7.1: Direccionamiento y nomenclatura de las VLANs.

Como se puede inferir de la tabla, la red desde la que se trabaja es la que se ha referenciado como XX.XX.XX.0/24, ya que son IPs públicas. En los equipos se han configurado direcciones de dos redes distintas. Por un lado se ha configurado el direccionamiento 192.168.1.0/24. En este se encuentran el servidor y los dos equipos (FortiGate y F5 BigIP) que han de poder acceder a él. Adicionalmente, como tanto el cliente como el servidor web se encuentran en el mismo switch, es necesario situarlos en distintas redes para que no puedan comunicarse directamente entre ellos. Para ello se coloca el cliente en la red 192.168.200.0/24. En este mismo rango se coloca el servidor virtual en el F5, como se ve en el esquema de red 7.1. De esta forma desde el cliente es posible acceder al servidor virtual pero no al real.

7.1 Automatización SSl-VPN

Para la automatización de la VPN, se considerará que el script funciona si tras la ejecución del mismo es posible conectarse al servidor web y visualizar la página desde un equipo externo a través de la VPN situada en el FortiGate, usando un usuario de los definidos en él .csv utilizado por el script. Es decir, desde un equipo de red desde la que se está trabajando, inicialmente no se dispone de acceso al servidor web. Sin embargo, si que es posible acceder al firewall, en cuyo puerto 10443 se ha configurado que escuche peticiones de SSL-VPN. Una vez accedido a

dicho puerto se solicita el usuario de la VPN y en caso de autenticación correcta, se permitirá el acceso a la IP del servidor web.

Con el fin de probar el funcionamiento, una vez desarrollado el script, se ejecuta el mismo, en un entorno con Python 2.7 instalado. El script solicita los datos necesarios y realiza las configuraciones del modo que se ha indicado en el el apartado 6.2.2.

Como se ve en la imagen 7.2, al acceder a la IP XX.XX.XX.60 del FortiGate en el que el script ha realizado la configuración, se solicita la introducción del nombre de usuario y la contraseña. En este momento es posible acceder con cualquiera de los usuarios que estén definidos en el .csv que se ha pasado como argumento al script.

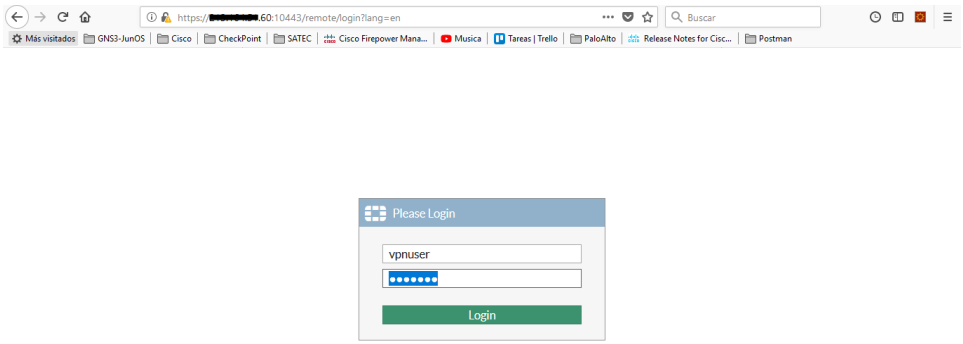


Figura 7.2: Solicitud de autenticación para acceso VPN

Una vez logeados con el usuario, se nos muestra el portal SSL-VPN configurado para ese usuario. En el caso del ejemplo se ha dejado el portal por defecto, pero el script contempla la posibilidad de crear un portal personalizado para cada usuario o grupo de usuarios. El aspecto del portal es el que se ve en la figura 7.3.

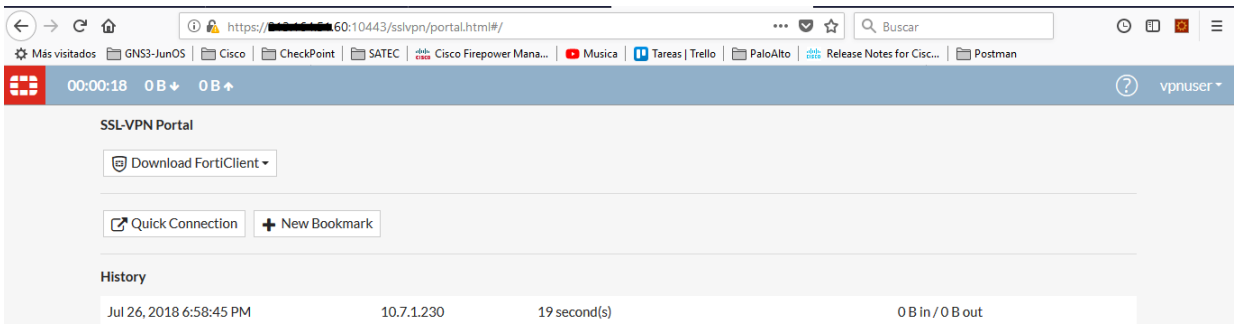


Figura 7.3: Portal SSL-VPN para el usuario vpnuser

Haciendo click en el boton de quick connection se ofrece la posiblidad de escoger el protocolo de conexión y la dirección IP a la que desea conectarse. Para que la conexión sea exitosa, ha de existir una politica de seguridad que permita el acceso del usuario actual a dicho recurso. En las figuras 7.4 y 7.5, puede verse como se rellena la IP del servidor web, que no era accesible inicialmente y sin la VPN y como se accede a la página alojada en el mismo, que no es más que la página web por defecto de Apache con el título modificado.

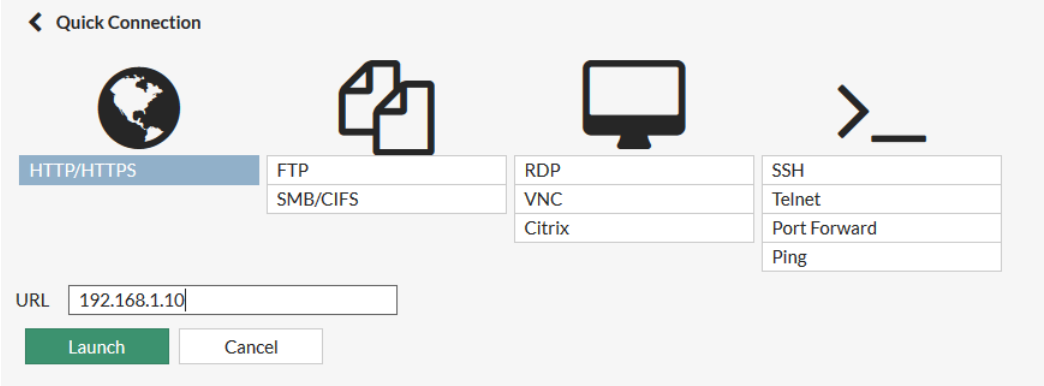


Figura 7.4: Acceso al servidor web

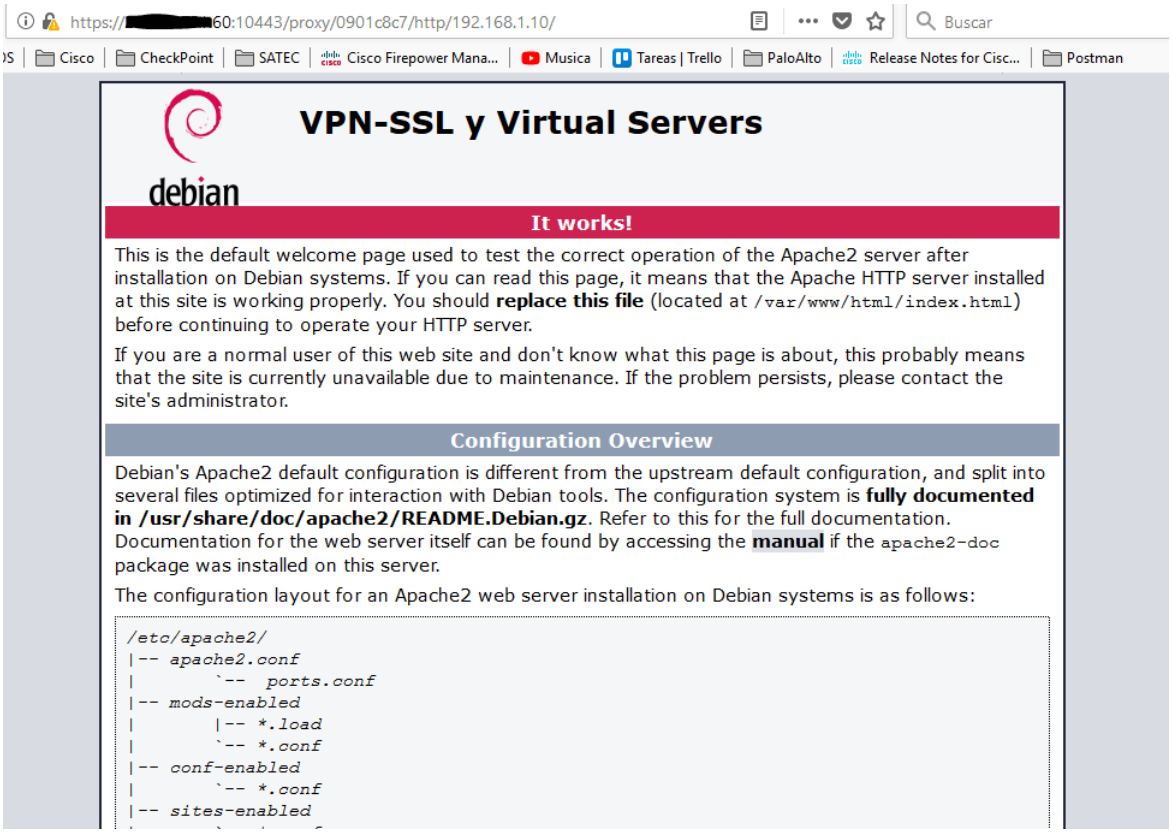


Figura 7.5: Página web accedida a través de la VPN

7.2 Automatización acceso a cámaras IP

Para el caso de la automatización del acceso a cámaras IP, el correcto funcionamiento del script debe permitir que desde la máquina virtual cliente, accediendo a una IP virtual asignada al F5, se pueda acceder a la página del servidor web. Si al realizar una petición HTTP a la IP del servidor virtual del F5 recibimos como respuesta la página alojada en el servidor web, se demuestra que el la tarea a completar por el script ha sido exitosa.

Con el fin de probar el funcionamiento, una vez desarrollado el script, se ejecuta el mismo, en un entorno con Python 2.7 instalado. El script solicita los datos necesarios y realiza las configuraciones del modo que se ha indicado en el el apartado 6.2.3.

Si accedemos al equipo Cliente, que se haya conectado únicamente a la Red de Pruebas, no dispone de acceso directamente al servidor web. El equipo cliente dispone de una interfaz gráfica para simplificar el acceso y las pruebas a realizar.

Sin embargo, desde el equipo cliente es posible realizar la petición HTTP al la IP del servidor virtual alojado en el equipo F5, que corresponde con la IP 192.168.200.30. En este caso, el balanceador recibe la petición que va dirigida a una IP que correspone a un virtual server definido, por lo que la redirige a los recursos asociados a ese servidor virtual. En este caso, el único recurso asociado es un pool de recursos cuyo único miembro es el equipo denominado como Servidor web. En este caso, como se ve en la figura 7.6, el método de distribución de la carga es Round Robin, aunque al haber un solo nodo resulta irrelevante. Si que es interesante destacar que se configura un http_monitor para el pool, que vigila el estado del mismo constantemente, comprobando que responda en el puerto 80. Mientras que todo funcione correctamente en el campo estado se mostrará un botón verde.

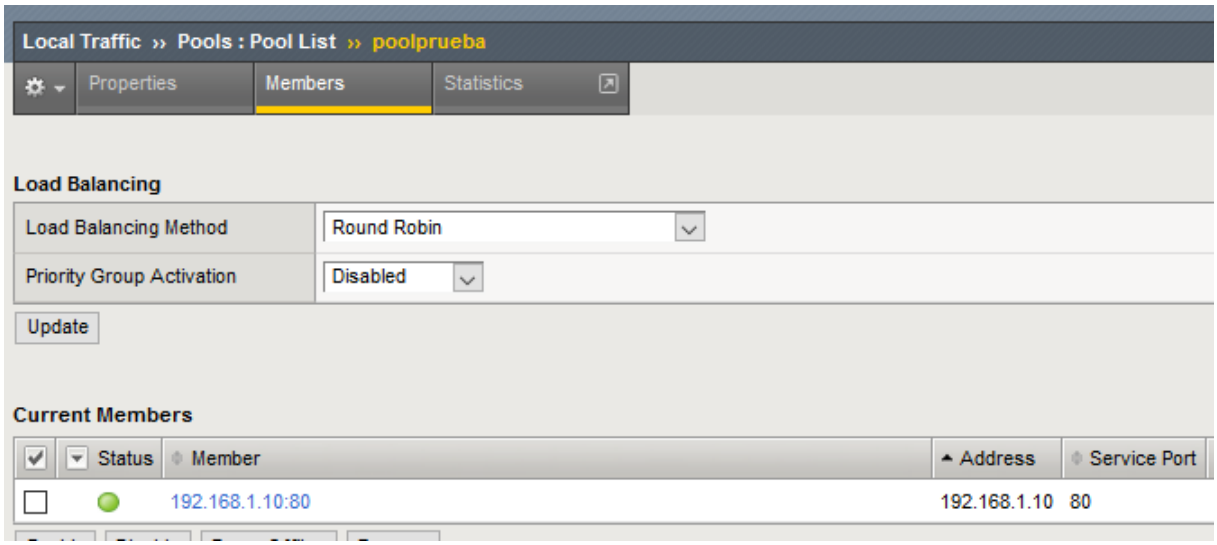


Figura 7.6: Lista de miembros en el pool

De esta forma, la petición es realizada contra el F5 que la redirige al Servidor web, cuya configuración se muestra en la imagen 7.7. Como puede verse, se ha definido en que IP ha de escuchar (192.168.200.30) y en que puerto (12080).

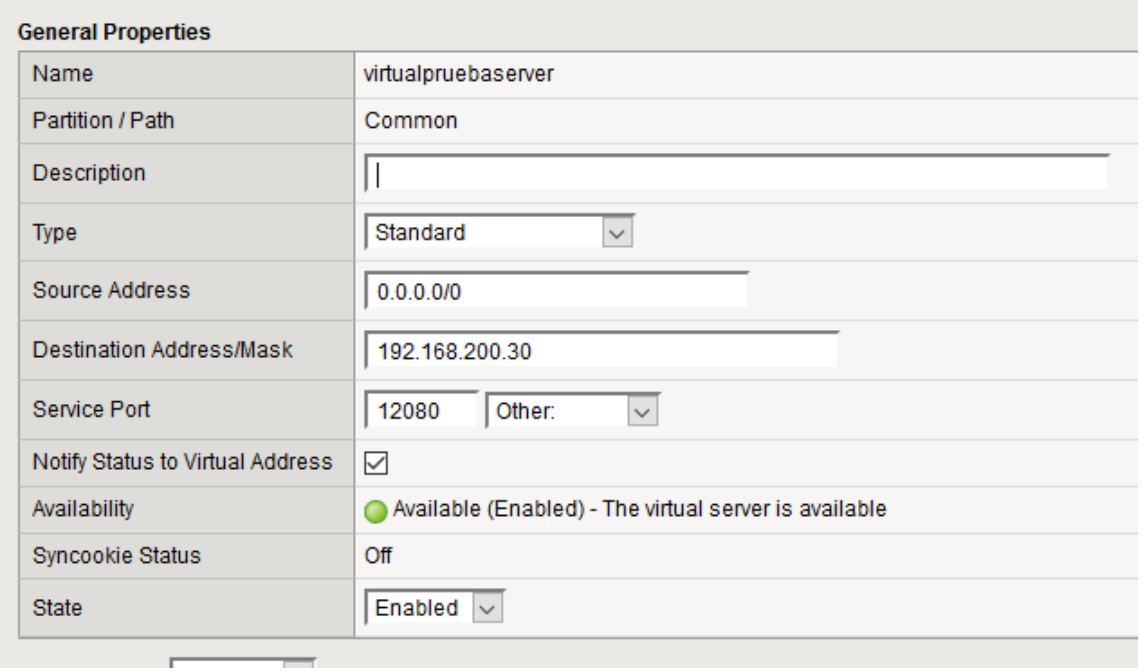


Figura 7.7: Configuración del virtual server

Únicamente con esta configuración no sería posible el funcionamiento del virtual server. La petición realizada por el cliente llegaría al servidor web de forma correcta, pero con la IP origen del Cliente. Esto evitaría que el Servidor web sea capaz de enrutarla de vuelta al cliente, ya que no pertenecen a la misma red y no hay ningún equipo conectándolos ambos que sea capaz de rutar el paquete entre redes. La solución consiste en configurar el servidor virtual para que haga Source Address Traslation, es decir, el balanceador cambiará la IP origen, que no es rutable desde el servidor, por otra que si lo sea.

Esto puede hacerse de dos formas en el caso de F5. Por un lado, existen la opción SNAT(Source Network Address Traslation), en la que se puede definir un pool de direcciones, que serán las que se utilizarán para sustituir la IP original. Como todos los pools de F5, disponemos de control sobre como realizar el balanceo entre las distintas IPs miembros del pool. La otra opción es utilizar el Auto Map. En este caso es el propio balanceador quien elige cual ha de ser la IP adecuado para realizar la sustitución. Para ello escoge una IP de entre las IPs que tiene asignadas para si mismo, que se conocen como Self-IPs. En este caso se ha optado por configurar el servidor virtual en modo Auto Map[11], ya que el propio balanceador dispone de una única Self-IP, la 192.168.1.30, que pertenece a la misma red que el Servidor web, por lo que la comunicación funcionará. La configuración puede verse en la imagen 7.8.

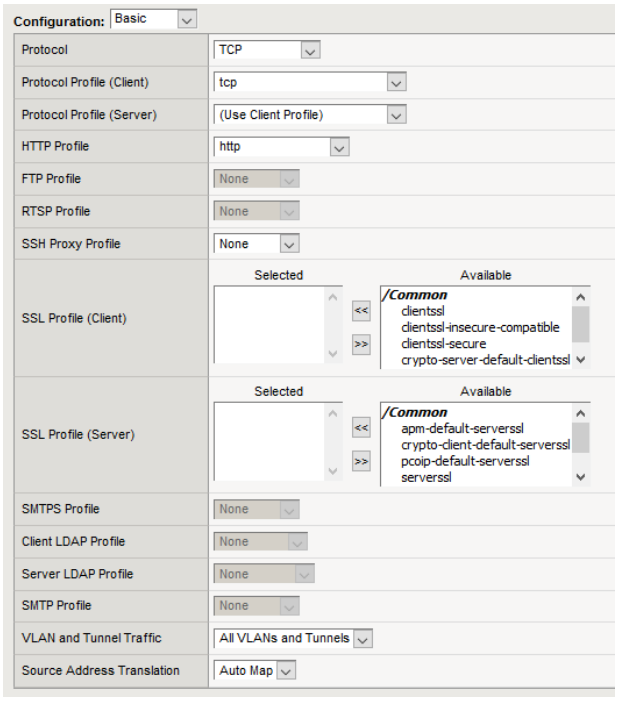


Figura 7.8: Configuración SNAT del virtual server

Todas las configuraciones mencionadas son realizadas por parte del script comentado anteriormente para cada uno de los Virtual Servers a definir en cada caso.

Finalmente, tras ejecutar el script se prueba que el funcionamiento es correcto, mediante el acceso a la pagina web desde el Cliente, conectándose a la IP del virtual server, como se muestra en la figura 7.9.

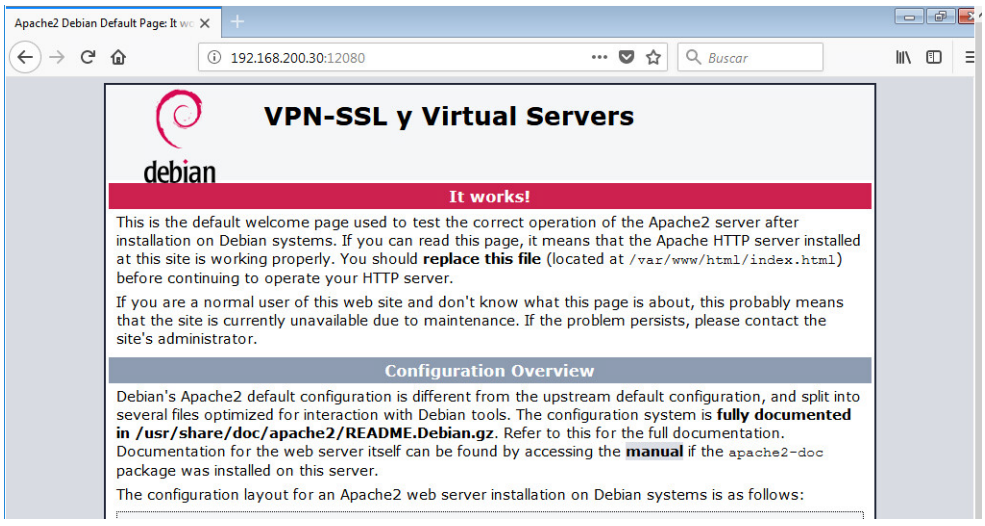


Figura 7.9: Página web accedida a través del virtual server

8 Planificación del proyecto. Gantt

A lo largo del desarrollo de este proyecto se han llevado a cabo una serie de tareas. Estas tareas pueden agruparse en diversos paquetes de trabajo ateniéndonos a su funcionalidad. En este apartado se detallan las tareas que forman el plan de trabajo, indicando para cada una su correspondiente fecha de comienzo y finalización, el responsable de su ejecución y los entregables definidos.

8.1 Equipo de trabajo

Para la realización de este proyecto se ha creado un equipo de trabajo formado por dos personas. Por un lado se requiere un director de proyecto, que se encarga de marcar los objetivos e hitos del proyecto, asi como del asesoramiento técnico y el seguimiento de la evolución del mismo. Por otro lado, se requiere un Ingeniero Junior que será quien lleve a cabo las labores de realización del proyecto, entre las que se incluyen el diseño, las configuraciones y las codificaciones necesarias.

A continuación se detallan los miembros del equipo

- **Ingeniero Junior:** Endika Zuazo
- **Director de proyecto:** Juan José Unzilla

8.2 Paquetes y Tareas

El proyecto es desglosado en una serie de tareas, que en agrupadas en bloques lógicamente relacionados forman los paquetes de trabajo. En este apartado se comentan los bloques planificados.

PT**1** **Contextualización del proyecto. Adquisición de conocimientos**

T**1.1** Adquisición de conocimientos relacionados con Fortinet y F5

Duración: 15 días
Descripción: El ingeniero Junior adquiere los conocimientos necesarios para el diseño de una red usando equipos Fortinet y F5 y su posterior configuración
Recursos humanos: Ingeniero Junior
Recursos técnicos: Ordenador, material fungible

T**1.2** Adquisición de conocimientos sobre las APIs

Duración: 15 días

Descripción: El ingeniero Junior adquiere los conocimientos necesarios para el manejo de las APIs propias de Fortinet y F5, asi como del lenguaje de programación a utilizar
Recursos humanos: Ingeniero Junior
Recursos técnicos: Ordenador

PT2 **Análisis de las alternativas disponibles para llevar a cabo el proyecto**

T2.1 Búsqueda de información sobre las alternativas para automatización

Duración: 10 días
Descripción: Tras hacer un estudio del estado del arte, se valoran las distintas opciónes para facilitar la automatización
Recursos humanos: Ingeniero Junior
Recursos técnicos: Ordenador

T2.2 Busqueda de información sobre alternativas para el acceso remoto a los recursos

Duración: 10 días
Descripción: Tras hacer un estudio del estado del arte, se valoran las distintas opciónes para facilitar el acceso a los recursos internos
Recursos humanos: Ingeniero Junior
Recursos técnicos: Ordenador

PT3 **Diseño de la solución; arquitectura, política de seguridad y automatización**

T3.1 Diseño de la arquitectura

Duración: 15 días
Descripción: Considerando las necesidades y objetivos del proyecto, se elabora un diseño de alto nivel que recoge la solución a implementar
Recursos humanos: Ingeniero Junior
Recursos técnicos: Ordenador

T3.2 Diseño de pólitica de seguridad

Duración: 10 días
Descripción: Se realiza un diseño en el que se indican las restricciones de seguridad a necesarias, asi como las medidas concretas a aplicar para cumpliementralas
Recursos humanos: Ingeniero Junior
Recursos técnicos: Ordenador, material fungible

PT4 **Implementación y codificación**

T4.1 Instalación del equipamiento físico

Duración: 5 días
Descripción: Se instalan los equipos adquiridos para el proyecto en la localización determinada, con su correspondiente cableado
Recursos humanos: Ingeniero Junior
Recursos técnicos: Ordenador

T**4.2** Configuración de la red

Duración: 25 días

Descripción: Se procede a configurar los equipos instalados y los previamente existentes para implementar el diseño, incluyendo la politica de seguridad

Recursos humanos: Ingeniero Junior

Recursos técnicos: Ordenador, equipos físicos, tuercas enjauladas, tornillos, destornillador

T**4.3** Codificación de los scripts de automatización

Duración: 25 días

Descripción: Se realiza el proceso de codificación de los scripts que permiten automatizar el acceso remoto a los recursos internos

Recursos humanos: Ingeniero Junior

Recursos técnicos: Ordenador

PT**5** **Pruebas y validación**

T**5.1** Pruebas de navegación

Duración: 10 días

Descripción: Para validar el diseño y configuración de la red, se procede a comprobar la navegación y acceso a recursos externos desde la red interna

Recursos humanos: Ingeniero Junior, Director de proyecto

Recursos técnicos: Ordenador

T**5.2** Pruebas de automatización

Duración: 10 días

Descripción: Para validar el funcionamiento de los scripts de automatización, se realizan pruebas de validación en un entorno de laboratorio

Recursos humanos: Ingeniero Junior, Director de proyecto

Recursos técnicos: Ordenador

PT**6** **Gestión del proyecto**

T**6.1** Seguimiento del trabajo

Duración: 150 días

Descripción: A lo largo de la duración del proyecto completo, se realiza un seguimiento de los avances realizados en el mismo, para controlar el estado y la consecución de los hitos

Recursos humanos: Director del proyecto

Recursos técnicos: Ordenador

T**6.2** Elaboración de memoria final

Duración: 30 días

Descripción: Se genera a la finalización del proyecto un documento en el que se incluye la metodología aplicada en el proyecto, el diseño, las configuraciones, los resultados y las conclusiones.

Recursos humanos: Ingeniero Junior, Director de proyecto

Recursos técnicos: Ordenador, material fungible

8.3 Hitos y entregables

Para realizar un control de los avances realizados a lo largo del proyecto se han establecido una serie de hitos, a los que se asocian unos entregables. En las fechas señaladas, se espera que estén realizados los entregables fijados. Los hitos y los entregables pueden verse en la tabla 8.1

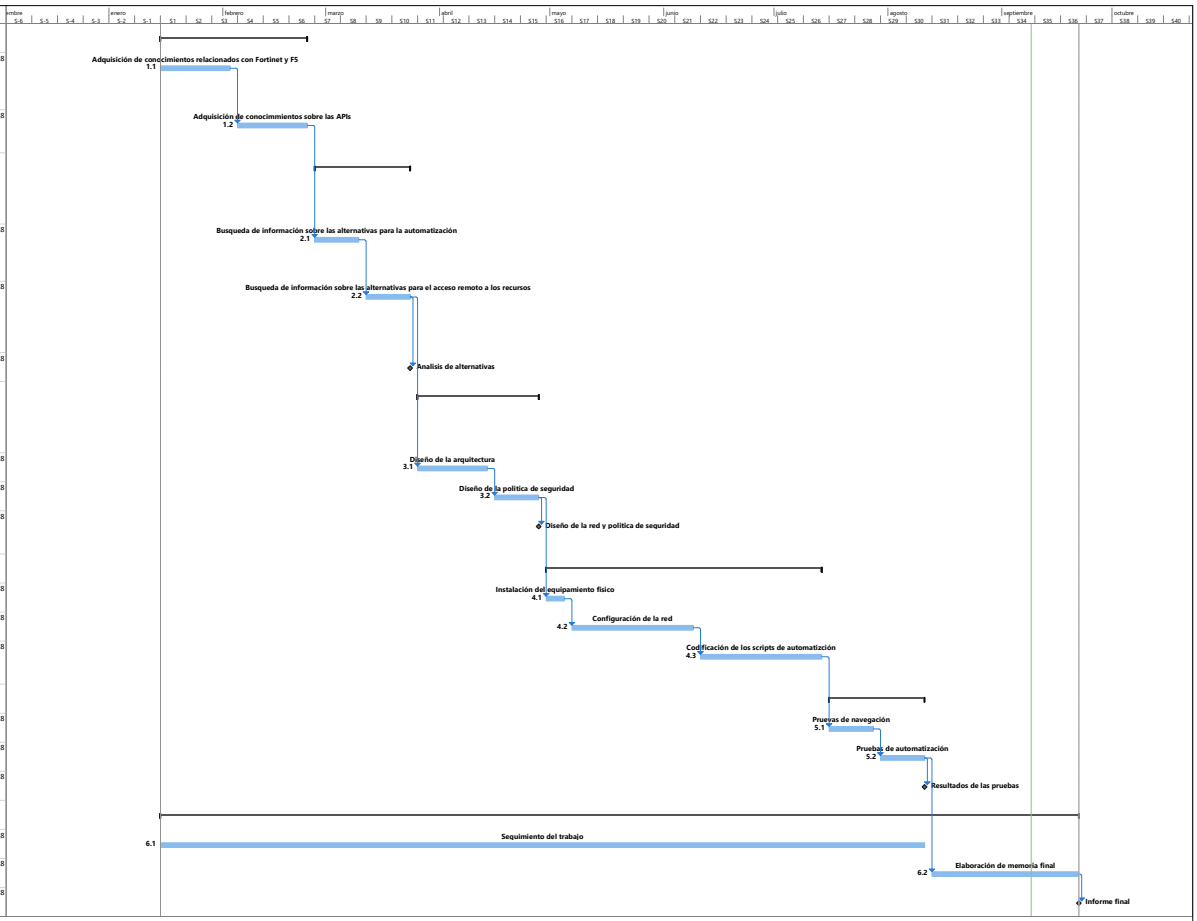
Hito	Entregable	Fecha
Fin de análisis de alternativas	Análisis de alternativas	23/03/18
Fin del diseño	Diseño de la red y política de seguridad	27/04/18
Fin de las pruebasb	Resultados de las pruebas	10/08/18
Fin del proyecto	Informe final	21/09/2018

Tabla 8.1: Hitos y entregables del proyecto.

8.4 Gantt

A continuación se muestran las tareas e hitos definos en un Gantt.

Id	EDT	Nombre de tarea	Duración	Comienzo	Fin
1	**1**	**Contextualización del proyecto**	**30 días**	**lun 15/01/18**	**vie 23/02/18**
2	1.1	Adquisición de conocimientos relacionados con Fortinet y F5	15 días	lun 15/01/18	vie 02/02/18
3	1.2	Adquisición de conocimmientos sobre las APIs	15 días	lun 05/02/18	vie 23/02/18
4	**2**	**Analisis de las alternativas disponibles para llevar a cabo el proyecto**	**20 días**	**lun 26/02/18**	**vie 23/03/18**
5	2.1	Busqueda de información sobre las alternativas para la automatización	10 días	lun 26/02/18	vie 09/03/18
6	2.2	Busqueda de información sobre las alternativas para el acceso remoto a los recursos	10 días	lun 12/03/18	vie 23/03/18
7	2.3	Analisis de alternativas	0 días	vie 23/03/18	vie 23/03/18
8	**3**	**Diseño de la solución: arquitectura, plitica de seguridad y automatización**	**25 días**	**lun 26/03/18**	**vie 27/04/18**
9	3.1	Diseño de la arquitectura	15 días	lun 26/03/18	vie 13/04/18
10	3.2	Diseño de la política de seguridad	10 días	lun 16/04/18	vie 27/04/18
11	3.3	Diseño de la red y política de seguridad	0 días	vie 27/04/18	vie 27/04/18
12	**4**	**Implementación y codificación**	**55 días**	**lun 30/04/18**	**vie 13/07/18**
13	4.1	Instalación del equipamiento físico	5 días	lun 30/04/18	vie 04/05/18
14	4.2	Configuración de la red	25 días	lun 07/05/18	vie 08/06/18
15	4.3	Codificación de los scripts de automatizición	25 días	lun 11/06/18	vie 13/07/18
16	**5**	**Pruebas y validación**	**20 días**	**lun 16/07/18**	**vie 10/08/18**
17	5.1	Pruevas de navegación	10 días	lun 16/07/18	vie 27/07/18
18	5.2	Pruebas de automatización	10 días	lun 30/07/18	vie 10/08/18
19	5.3	Resultados de las pruebas	0 días	vie 10/08/18	vie 10/08/18
20	**6**	**Gestión del proyecto**	**180 días**	**lun 15/01/18**	**vie 21/09/18**
21	6.1	Seguimiento del trabajo	150 días	lun 15/01/18	vie 10/08/18
22	6.2	Elaboración de memoria final	30 días	lun 13/08/18	vie 21/09/18
23	6.3	Informe final	0 días	vie 21/09/18	vie 21/09/18



Proyecto: Proyecto1
Fecha: dom 09/09/18

Tarea · Resumen · Hito inactivo · solo duración · solo el comienzo · Hito externo · Progreso manual
División · Resumen del proyecto · Resumen inactivo · Informe de resumen manual · solo fin · Fecha límite
Hito · Tarea inactiva · Tarea manual · Resumen manual · Tareas externas · Progreso

9 Presupuesto

El presupuesto de este proyecto está compuesto por la suma total de los costes asociados al mismo, ya sean humanos como materiales. Cada uno de ellos se ve representado en distintas partidas.

En este apartado se calculan, por un lado los costes humanos ypor otro el precio total de cada elemento material con su correspondiente periodo de amortización.

9.1 Horas internas

En esta partida se incluyen las horas dedicadas por todos aquellos trabajadores del proyecto para llevar a cabo las tareas asociadas al mismo. Estas horas conllevan un coste, que se puede calcular conocido el coste horario y el número de horas invertidas. De esta forma es posible obtener el coste total de las horas internas.

El desglose de las horas internas puede verse en la tabla 9.1. En el se puede ver el coste total referido a las horas internas, que asciende a 14.000€

Nombre	Rol	Tasa Horaria	Horas	Coste Total
Juan Jose Unzilla	Ingeniero Senior (Director del proyecto)	50€/h	40	2.000€
Endika Zuazo	Ingeniero Junior	20€/h	720	14.400€
TOTAL				16.400€

Tabla 9.1: Subtotal de horas internas

A continuación, el la tabla 9.2 se puede ver el total de horas internas como se ha repartido entre los distintos paquetes de trabajo identificados en el apartado 8.2. Cada día de trabajo se supone de 4h.

Paquete de trabajo	Responsable	Nº de horas
Contextualización del proyecto	Ingeniero Junior	120
Análisis de alternativas	Ingeniero Junior	80
Diseño de la solución	Ingeniero Junior	100
Implementación y codificación	Ingeniero Junior	220
Pruebas y validación	Ingeniero Junior Director del proyecto	40 10
Gestión del proyecto	Ingeniero Junior Director del proyecto	120 30
TOTAL		760

Tabla 9.2: Desglose de horas por paquetes de trabajo

9.2 Gastos

La partida de gastos incluyen todo aquello que se haya usado para llevar a cabo el proyecto y que debido a su uso, no pueda volver a ser empleado en proyectos posteriores.

El desglose de gastos puede verse en la tabla 9.3

Concepto	Unidades	Coste
FortiGate 1000D	3	27.520€
Electricidad	-	20€
Internet	-	100€
Material fungible	-	10€
TOTAL	-	27.650€

Tabla 9.3: Subtotal de gastos

9.3 Amortizaciones

En esta partida se recoge el coste de aquellos materiales que ya se encuentran disponibles previamente al inicio del proyecto, como es el caso de los balanceadores F5 empleados. Por tanto, las amortizaciones son el coste de los activos fijos que se utilizan para el proyecto. La parte amortizable del material a imputar en el proyecto es proporcional al tiempo que este se destina al proyecto dentro de su vida útil.

Se hará uso de un ordenador portátil de precio 1000 €. Para un equipo de esas cáracteristicas se considera una vida útil de unos 5 años.

Para la documentación y seguimiento del proyecto se utiliza el software propietario Microsoft Office, con una duración igual a la del portatil y un coste de 80€.

Además, los balanceadores F5, cuyo coste asciende a 17.995€ y del que se espera que disponga de una vida útil de 8 años se utilizarán para este proyecto y otro de forma simultanea.

El desglose de las amortizaciones puede verse en la tabla 9.4

Concepto	Valor Inicial	Valor Residual	Vida útil	Tiempo de uso	Coste final
Ordenador	1000€	250€	5 años	5 meses	62,5€
Microsoft Office	80€	0€	1 años	5 meses	33,3€
F5 Big IP	18.000€	2000€	8 años	4 años	8.000€
TOTAL					8.095,8€

Tabla 9.4: Subtotal de amortizaciones

9.4 Coste total

Finalmente, para obtener el coste total, se realiza la suma de los costes parciales de los apartados anteriores, además de un porcentaje de los anteriores, para considerar los gastos indirectos que no se pueden imputar al proyecto.

Concepto	**Subtotal**
Horas Internas	16.400€
Amortizaciones	8.095,8€
Gastos	27.650€
SUBTOTAL	**52.145,8€**
Costes Indirectos(%10)	5.214,58€
TOTAL	**57.360,38€**

Tabla 9.5: Coste total

10 Conclusiones

A lo largo de la realización de este proyecto se ha llevado a cabo el diseño e implementación de una red securizada que permite el acceso a ciertos recursos de una red privada, como son las cámaras de videovigilancia repartidas a lo largo de la red. Para dicho diseño se ha tenido muy presente la seguridad en el acceso a los servicios publicados y en paralelo al diseño de la red se ha realizado el diseño de una política de seguridad acorde a los requisitos del proyecto. Una vez realizado el diseño, se han instalado los equipos y una vez realizada la configuración se ha validado el correcto funcionamiento del mismo. Además, se han desarrollado las herramientas necesarias para automatizar la publicación de dichos servicios, que incluyen varios scripts que hacen uso de las APIs propias de los fabricantes de los equipos instalados.

Con el creciente número de dispositivos conectados a Internet, con el consiguiente riesgo, resulta cada vez más importante concienciar a los administradores y usuarios de la relevancia de una buena política de seguridad y de su cumplimiento. En este proyecto puede verse como se securiza el acceso, utilizando las reglas de los firewalls y los perfiles de seguridad, que gracias a sus funcionalidades de capa de aplicación ofrecen una seguridad sensiblemente mayor que la que había antes del proyecto.

Otra de las cosas que nos permite ver el proyecto es la importancia que están adquiriendo poco a poco las técnicas y herramientas de automatización. Actualmente, casi todos los fabricantes del ámbito de las telecomunicaciones se han sumado a desarrollar APIs y algunas otras herramientas similares para facilitar y posibilitar a los administradores de red métodos para simplificar la gestión simultanea de múltiples equipos (por ejemplo mediante el FortiManager) y los procedimientos habituales (automatizando vía scripts que hagan uso de las APIs).

Finalmente, este proyecto sirve como punto de referencia para posibles proyectos similares de migración de firewalls, así como guía y apoyo para proyectos cuyo objetivo sea automatizar nuevas funciones, ya que es posible reutilizar los scripts adjuntados en los anexos y añadir nuevas funcionalidades o crear nuevas.

Anexos

En este apartado se incluye información adicional al proyecto que puede ser de interés para profundizar en su comprensión. Con este objetivo en mente se incluyen el código de los script de python comentados en el apartado 6.

I Código del script para VPN

```
import requests
import argparse
import sys
import json
from pprint import pprint
import getopt
import csv
import getpass
from itertools import chain

requests.packages.urllib3.disable_warnings()

class FGT(object):

    def __init__(self, host):
        self.host = host
        self.url_prefix = "https://" + self.host

    def update_csrf(self):
        for cookie in self.session.cookies:
            if cookie.name == "ccsrftoken":
                csrftoken = cookie.value[1:-1]  # token stored as a list
                self.session.headers.update({"X-CSRFTOKEN": csrftoken})

    def login(self, name="admin", key="", csrf=True):
        self.logout()

        self.session = requests.session()
        url = self.url_prefix + "/logincheck"
        try:
            res = self.session.post(
                url,
                data="username=" + name + "&secretkey=" + key,
                verify=False)
        except requests.exceptions.RequestException as e:
            print(e)
            print "LOGIN failed"
            exit()

        if res.text.find("error") != -1:
```

```
            print "LOGIN failed"
            return False

        if csrf:
            self.update_csrf()
        return True

    def logout(self):
        if hasattr(self, "session"):
            url = self.url_prefix + "/logout"
            self.session.post(url)

    def get(self, url, **options):
        url = self.url_prefix + url
        try:
            res = self.session.get(
                url,
                params=options.get("params"))
        except requests.exceptions.RequestException as e:
            print(e)
            exit()
        return res

    def post(self, url, override=None, **options):
        url = self.url_prefix + url
        data = options.get("data") if options.get("data") else None

        if override:
            self.session.headers.update({"X-HTTP-Method-Override": override
                })
        try:
            res = self.session.post(
                url,
                params=options.get("params"),
                data=json.dumps(data),
                files=options.get("files"))
        except requests.exceptions.RequestException as e:
            print(e)
            exit()

        if override:
            del self.session.headers["X-HTTP-Method-Override"]
        return res

    def put(self, url, **options):
        url = self.url_prefix + url
        data = options.get("data") if options.get("data") else None
        try:
            res = self.session.put(
                url,
                params=options.get("params"),
```

```
                data=json.dumps(data),
                files=options.get("files"))
        except requests.exceptions.RequestException as e:
            print(e)
            exit()
        return res

    def delete(self, url, **options):
        url = self.url_prefix + url
        try:
            res = self.session.delete(
                url,
                params=options.get("params"))
        except requests.exceptions.RequestException as e:
            print(e)
            exit()
        return res

def get_json(response):
    try:
        rjson = response.json()
    except UnicodeDecodeError as e:
        print "Cannot decode json data in HTTP response"
        return False
    except:
        e = sys.exc_info()[0]
        print(e)
        return False
    else:
        return rjson

def check_response(res):
        rjson = get_json(res)
        status = rjson["http_status"]
        if status == 200:
                print "200 successful request"
        elif status == 400:
                print "400 Invalid request format"
        elif status == 403:
                print "403 Permission denied"
        elif status == 404:
                print "404 None existing resource"
        elif status == 405:
                print "405 Unsupported method"
        elif status == 424:
                print "424 Dependency error"
        elif status == 500:
                print "500 Internal server error"
        else:
                print status, "Unknown error"
```

```
def makepost(fgt,path,name,data,vdom):
        res = fgt.post(
    url="/api/v2/cmdb/" + path + "/" + name,
    params={"vdom": vdom},
    data=data)
        print "/api/v2/cmdb/" + path + "/" + name
        print(res.text)

def makeput(fgt,path,name,mkey,data,vdom):
        res = fgt.put(
            url="/api/v2/cmdb/" + path + "/" + name + "/" + mkey,
            params={"vdom": vdom},
            data=data)
        print(res.text)

def makeget(fgt,path,name,mkey,vdom):
        res = fgt.get(
            url="/api/v2/cmdb/" + path + "/" + name + "/" + mkey,
            params={"vdom": vdom})
        print(res.text)

def makegetwithfilter(fgt,path,name,mkey,vdom,filter):
        res = fgt.get(
            url="/api/v2/cmdb/" + path + "/" + name + "/" + mkey,
            params={"vdom": vdom,"filter":filter})
        print(res.text)

def makegetwithformat(fgt,path,name,mkey,vdom,format):
        res = fgt.get(
            url="/api/v2/cmdb/" + path + "/" + name + "/" + mkey,
            params={"vdom": vdom,"format":format})
        print(res.text)

def makegetwithformatandfilter(fgt,path,name,mkey,vdom,params,filter):
        res = fgt.get(
            url="/api/v2/cmdb/" + path + "/" + name + "/" + mkey,
            params={"vdom": vdom,"format":params,"filter":filter})
        print(res.text)

# Leemos opciones
#print '**argv:', sys.argv[1:]
opts, args = getopt.getopt(sys.argv[1:],"hi:v:u:p:f:",["help"])
#print "**options:", opts

# Fijamos valores por defecto
hostname = ""
vdom = ""
username = ""
password = ""
filename = ""
```

```
# Parseamos los argumentos
for opt, arg in opts:
    #print "*arg* " + opt + " : " + arg
    if opt in ('-h', '--help'):
                print "
                    ******************************************************"
                print "Uso: anotherone.py -i <hostname ip[:port]> -v <vdom>
                    -u <username> -p <password> -f <archivo.csv>"
                print ""
                print "  En caso de no especificar hostname, username y/o
                    password se solicitaran "
                print ""
                print "  -i: hostname del equipo, puerto por defecto: 443.
                    "
                print ""
                print "  -v: vdom a utilizar , por defecto vdom=root "
                print ""
                print "  -p: dejando la opcion en blanco se solicitara el
                    password sin mostrarse en pantalla"
                print ""
                print "  -f: dejando la opcion en blanco se solicitara el
                    nombre de fichero"
                print "      El formato de fichero tiene que ser: User,Pass
                    ,Resource,DstInt"
                print "      Las columnas han de tener encabezado con los
                    nombres mencionados"
                print "
                    ******************************************************"
                print "Ejemplos:"
                print "  vpnconfig.exe    (prompts para requeridos)"
                print "  vpnconfig.exe -i 192.168.1.99 -v Production -d 360
                    -u admin -f configusers.csv"
                print "  vpnconfig.exe -i 192.168.1.99 -u admin -p password
                    "
                print "  vpnconfig.exe -i 192.168.1.99:8443 -u admin -v
                    myVdom"
                print "
                    ******************************************************"
                sys.exit(2)
    elif opt == '-i':
                hostname = arg
    elif opt == '-v':
        vdom = arg
    elif opt == '-u':
        username = arg
    elif opt == '-p':
        password = arg
    elif opt == '-f':
        filename = arg
```

```
# En caso de faltar algun argumento lo solicitamos
if hostname == "":
    hostname = raw_input('Introduzca␣IP:␣')
if username == "":
    username = raw_input('Introduzca␣usuario:␣')
if password == "":
    password = getpass.getpass('Introduzca␣password:␣')
if filename == "":
        filename = raw_input('Introduzca␣nombre␣de␣fichero:␣')
if vdom == "":
        vdom = "root"


####Trabajando con csv ####

with open(filename) as f:
        reader = csv.DictReader(f,delimiter=';')
        dataread = [r for r in reader]


#Creamos el Objeto Firewall y nos logeamos para la sesion
fgt = FGT(hostname)
fgt.login(username,password)


#Crear grupo
data = {"name":"grupocsv"}
makepost(fgt,"user","group",data,vdom)


for line in dataread:
        #Crear usuario
        data = {"name":line["User"],"type":"password","passwd":line["Pass"
            ]}
        makepost(fgt,"user","local",data,vdom)
        #Meter usuarios
        data = {"name":line["User"]}
        makepost(fgt,"user","group/grupocsv/member",data,vdom)

#Crear portal
data = {"name":"csvportal","tunnel-mode":"enable","web-mode":"enable","host
    -check":"av", "ip-pools":"VPN_IPs_Prueba","split-tunneling":"disable","
    heading":"From␣API"}
makepost(fgt,"vpn.ssl.web","portal",data,vdom)

#Configurar vpn-ssl
data = {"groups":[{"name":"grupocsv"}],"portal":"csvportal"}
makepost(fgt,"vpn.ssl","settings/authentication-rule",data,vdom)

#Crear politicas
```

```
for line in dataread:
        data = {'json':{'name':"Politica␣para␣"+line["User"],'srcintf':[{"
            name":"ssl.root"}],'dstintf':[{"name":line["DstInt"]}],
                'srcaddr':[{"name":"all"}],'dstaddr':[{"name":line["
                    Resource"]}],'action':"accept",'status':"enable",'
                    schedule':"always",
                'service':[{'name':"ALL"}],'users':[{'name':line["User"]}],
                    'nat':"enable"}}
        makepost(fgt,"firewall","policy",data,vdom)

#Nos vamos
fgt.logout()

####De aqui en adelante son ejemplos de cosas hechas #########


#makegetwithformat(fgt,"user","group","grupofromapi",vdom,"member")

# Crea usuario
#data = {"name":"userfromapi","type":"password","passwd":"apipass"}
#makepost(fgt,"user","local",data,vdom)

# Crea grupo
#data = {"name":"grupofromapi"}
#makepost(fgt,"user","group",data,vdom)



# Aniade usuario a grupo
#data = {"member":[{"name":"userfromapi"}]}
#makeput(fgt,"user","group","grupofromapi",data,vdom)

# Ver solo algunos parametros de la policy
#makegetwithparams(fgt,"firewall","policy","2",vdom,"name|service|groups")


#Crea portal vpn-ssl
#data = {"name":"portalfromapi","tunnel-mode":"enable","web-mode":"enable",
    "host-check":"av", "ip-pools":"VPN_IPs_Prueba","split-tunneling":"
    disable","heading":"From␣API"}
#makepost(fgt,"vpn.ssl.web","portal",data,vdom)

#Crear politica, problema a la hora de anadir un grupo--> si el portal
    asignado a esos users tiene split tunneling
#no se puede poner ANY como destination address
#data = {'json':{'name':"apipolicy",'srcintf':[{"name":"ssl.root"}],'
    dstintf':[{"name":"port3"}],
#       'srcaddr':[{"name":"all"}],'dstaddr':[{"name":"google-play"}],'
    action':"accept",'status':"enable",'schedule':"always",
#               'service':[{'name':"ALL"}],'groups':[{'name':"grupofromapi"
    }],'nat':"enable"}}
```

II Código del script para Virtual Servers

```
import requests
import csv
import sys
import getopt
import argparse
import getpass
from f5.bigip import ManagementRoot

requests.packages.urllib3.disable_warnings()

# Leemos opciones
#print '**argv: ', sys.argv[1:]
opts, args = getopt.getopt(sys.argv[1:],"hi:p:u:p:f:",["help"])
#print "**options: ", opts

# Fijamos valores por defecto
hostname = ""
partition = ""
username = ""
password = ""
filename = ""


# Parseamos los argumentos
for opt, arg in opts:
    #print "*arg* " + opt + " : " + arg
    if opt in ('-h', '--help'):
                print "*****************************************"
                print "Uso: anotherone.py -i <hostname ip[:port]> -p <
                    partition> -u <username> -p <password> -f <archivo.csv>"
                print ""
                print "  En caso de no especificar hostname, username y/o
                    password se solicitaran "
                print ""
                print "  -i: hostname del equipo, puerto por defecto: 443.
                    "
                print ""
                print "  -p: partition a utilizar, por defecto partition=
                    Common "
                print ""
                print "  -p: dejando la opcion en blanco se solicitara el
                    password sin mostrarse en pantalla"
                print ""
                print "  -f: dejando la opcion en blanco se solicitara el
                    nombre de fichero"
                print "      El formato de fichero tiene que ser: NodeName,
                    Address,PoolName,VirtualServerName"
```

```
                print "      Las columnas han de tener encabezado con los
                    nombres mencionados"
                print "****************************************"
                print "Ejemplos:"
                print "  vsconfig.exe    (prompts para requeridos)"
                print "  vsconfig.exe -i 192.168.1.99 -p Production -u
                    admin -f configurevirtualservers.csv"
                print "  vsconfig.exe -i 192.168.1.99 -u admin -p password"
                print "  vsconfig.exe -i 192.168.1.99:8443 -u admin -p
                    myPartition"
                print "****************************************"
                sys.exit(2)
    elif opt == '-i':
                hostname = arg
    elif opt == '-c':
        vdom = arg
    elif opt == '-u':
        username = arg
    elif opt == '-p':
        password = arg
    elif opt == '-f':
        filename = arg


# En caso de faltar algun argumento, lo solicitamos
if hostname == "":
    hostname = raw_input('Introduzca IP: ')
if username == "":
    username = raw_input('Introduzca usuario: ')
if password == "":
    password = getpass.getpass('Introduzca password: ')
if filename == "":
        filename = raw_input('Introduzca nombre de fichero: ')
if partition == "":
        partition = "Common"

#Imprimimos la version
b = ManagementRoot(hostname, username, password)
print 'Version actual:' + b.tmos_version

with open(filename) as f:
        reader = csv.DictReader(f,delimiter=';')
        dataread = [r for r in reader]

port = 10100
for line in dataread:
#Crear nodos (name, address,[description])
        n1 = b.tm.ltm.nodes.node.create(name=line["NodeName"], address=line
            ["Address"],partition='Common')
#Crear pool (name,[description],health monitor, load balancing method)
```

```
        p1 = b.tm.ltm.pools.pool.create(name=line["PoolName"], partition='
            Common')
        p1.monitor = '/Common/http'
        p1.update()
#Incluir nodo(s) en el pool
        m1 = p1.members_s.members.create(partition='Common', name=line["
            NodeName"]+':80')
#Crear virtual server (name,destination addr/mask, service port, http
    profile?, pool)
        vs1 = b.tm.ltm.virtuals.virtual.create(name=line["VirtualServerName
            "],partition='Common',destination='213.164.51.46:'+str(port),
            pool=line["PoolName"])
#Aumentar puerto
        port = port + 1
```

Bibliografía

[1] K. Claveria, "13 stunning stats on the internet of things." URL: https://www.visioncritical.com/internet-of-things-stats/, April 28, 2017.

[2] T. Micro, "Persirai: New internet of things (iot) botnet targets ip cameras." URL: https://blog.trendmicro.com/trendlabs-security-intelligence/persirai-new-internet-things-iot-botnet-targets-ip-cameras, May 9, 2017.

[3] Fortinet, "Fortigate 1000d." URL:https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/FortiGate_1000D.pdf .

[4] F5, "F5 big ip." URL: https://www.f5.com/pdf/products/big-ip-platforms-datasheet.pdf .

[5] Fortinet, "Fortimanager." URL:https://www.fortinet.com/content/dam/fortinet/assets/data-sheets/FortiManager.pdf .

[6] Fortinet-Support, "Fqdn addresses." URL:http://help.fortinet.com/fos50hlp/54/Content/FortiOS/fortigate-firewall-52/Firewall %20Objects/Addresses/FQDN %20Addresses.html, 2017.

[7] VVAA, "Simple mail transfer protocol." URL:https://en.wikipedia.org/wiki/ Simple_Mail_Transfer_Protocol.

[8] Fortinet-Support, "Phising." URL:http://help.fortinet.com/fos50hlp/52data/Content/FortiOS/fortigate-security-profiles-52/Antivirus/Phishing.html , 2016.

[9] C. Finseth, "Rfc 1492: An access control protocol, sometimes called tacacs." URL: https://tools.ietf.org/html/rfc1492 , June 2000.

[10] C. Rigney and S. Willens, "Rfc 2865: Remote authentication dial in user service (radius)." URL: https://tools.ietf.org/html/rfc2865 , June 2000.

[11] F5-Support, "The snat automap and self ip address selection." URL: https://support.f5.com/csp/article/K7336, October 16, 2017.

[12] Fortinet, "Fortios handbook." URL: http://help.fortinet.com/fos50hlp/54/Content/FortiOS/fortiOS-HTML5-v2/Home.htm , August 2018.