UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

" SERVIDOR PROXY WEB EN UN ENTORNO CLOUD "

TRABAJO FIN DE GRADO

Marzo-2024

AUTOR: Eva Kornilova Nesterova

DIRECTOR/ES: Salvador Alcaraz Carrasco

RESUMEN

Este proyecto se enfoca en la implementación de un servidor proxy Squid en Azure, destacando la seguridad web y la conectividad remota. A través de reglas de bloqueo y una VPN Point-to-Site con autenticación de certificados, se busca asegurar un acceso controlado a internet para usuarios remotos. El servidor Squid gestionará tanto solicitudes HTTP como conexiones seguras HTTPS mediante SSL bumping, reforzando la seguridad al inspeccionar a fondo el tráfico cifrado. Además, se empleará la automatización para la eficiencia operativa, asegurando una administración efectiva del entorno. El proyecto tiene como objetivo principal fortalecer la seguridad de la red en la nube de Azure, proporcionando un entorno robusto y eficiente.

Palabras clave: Proxy, Azure, SSL bumping, HTTP, VPN, Reglas de bloqueo, automatización

ABSTRACT

This project focuses on the implementation of a Squid proxy server in Azure, highlighting web security and remote connectivity. Through blocking rules and a Point-to-Site VPN with certificate authentication, it seeks to ensure controlled access to the internet for remote users. The Squid server will handle both HTTP requests and secure HTTPS connections via SSL bumping, enhancing security by thoroughly inspecting encrypted traffic. In addition, automation will be employed for operational efficiency, ensuring effective management of the environment. The project's main objective is to strengthen the security of the Azure cloud network, providing a robust and efficient environment.

Keywords: Proxy, Azure, SSL bumping, HTTP, VPN, Blocking rules, automation.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

0. AGRADECIMIENTOS

Agradezco sinceramente a mi familia, amigos y profesores por su continuo respaldo durante mi trayectoria académica. Este Trabajo de Fin de Grado marca un hito importante, y su apoyo ha sido fundamental para llegar a este punto.

Quiero expresar mi reconocimiento especial a mi tutor, cuya orientación ha sido esencial para llevar a cabo este proyecto de manera exitosa. Su guía y apoyo han sido inestimables.

A todos quienes han formado parte de este viaje, mi más sincero agradecimiento por contribuir a mi desarrollo académico y por ser parte de este logro.

1. INTRODUCCIÓN

1.1. MOTIVACIÓN

En un mundo donde la ciberseguridad y la privacidad en línea son fundamentales, gestionar redes de manera eficiente es esencial. Enfrentamos un escenario digital dinámico, marcado por desafíos continuos y una interconexión creciente. Este proyecto se centra en configurar un proxy alojado en Azure, aprovechando así las ventajas y la relevancia que la computación en la nube nos ofrece en la actualidad. Con el propósito de abordar la creciente demanda de soluciones prácticas, se busca no solo satisfacer, sino anticipar las necesidades cambiantes en ciberseguridad y gestión de redes.

1.2. ESTADO DEL ARTE

En el mercado actual se han popularizado los sistemas de filtrado y seguridad de red dentro de las empresas, ya que protege a sus empleados y a su infraestructura de red. Esta tendencia se refleja en soluciones como firewalls avanzados y proxies en la nube. A continuación, se presentan algunos ejemplos.

1.2.1. NETSKOPE

Netskope es una empresa de seguridad en la nube que proporciona soluciones para proteger los datos y operaciones en entornos de nube, web y móviles. Su plataforma, basada en la nube, ofrece visibilidad y control en tiempo real sobre las actividades de los usuarios y el uso de datos sensibles.

Figura 1 – Logo Netskope

Actúa como un intermediario que supervisa y controla el tráfico entre los usuarios y las aplicaciones en la nube. Permite a los administradores crear políticas específicas que regulan quién puede acceder a qué recursos y en qué condiciones. Al mismo tiempo, inspecciona y analiza las URLs que los usuarios intentan acceder para identificar y controlar el uso de aplicaciones en la nube no autorizadas.

Para aplicaciones de confianza o para mejorar el rendimiento de la red, Netskope permite configurar excepciones, conocidas como "bypass", que hacen que cierto tráfico ignore las reglas de seguridad establecidas.

Además, mantiene registros detallados de todas las transacciones de los usuarios, proporcionando una trazabilidad completa sobre las decisiones tomadas en base a las políticas implementadas.

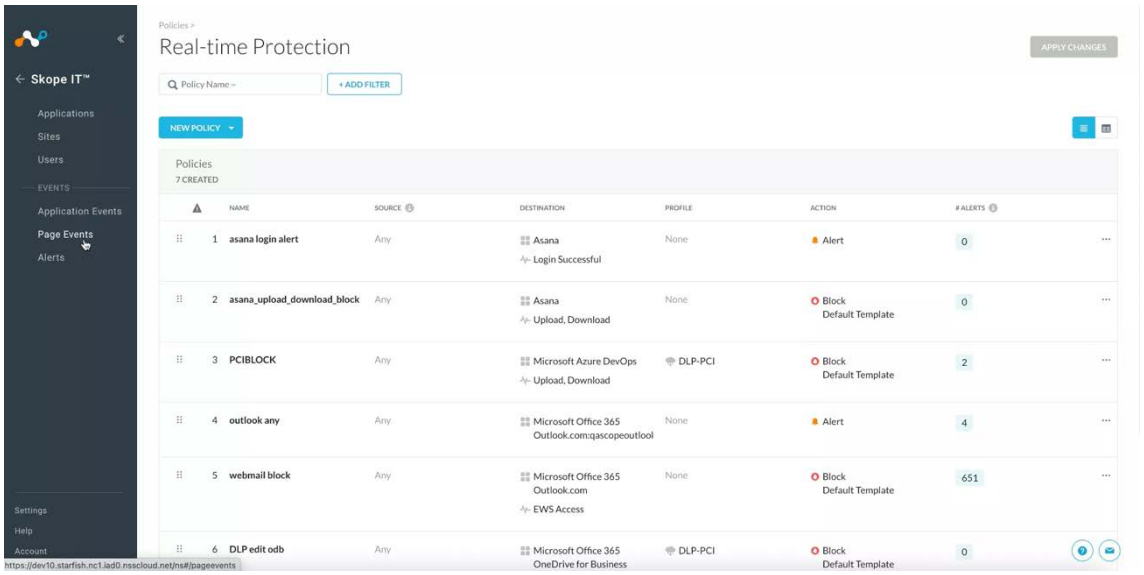

Figura 2 – Interfaz gráfica Netskope

Netskope cuenta con una API que facilita la automatización e integración de sus funcionalidades de seguridad con otras herramientas y sistemas, tales como sistemas de gestión de eventos e información de seguridad (SIEMs), plataformas de orquestación de seguridad (SOARs), y sistemas de gestión de identidades y accesos (IAM). Esta API permite a los desarrolladores gestionar políticas de seguridad, consultar registros y alertas.

1.2.2. NGINX PROXY MANAGER

Nginx Proxy Manager es una solución que consiste en una imagen docker pre-construida que permite reenviar fácilmente a los sitios web que se ejecuten desde la oficina o casa. Incluye SSL gratuito.

Este proyecto se creó ante la necesidad de proporcionar a los usuarios una manera sencilla de implementar un proxy inverso de hosts con terminación SSL. Pretende simplificar al máximo la configuración.

Figura 3 – Logo NGINX

Sus principales características son: interfaz de administración simple y segura basada en Tabler, permite crear fácilmente dominios de reenvío, redirecciones, secuencias y hosts 404, SSL gratuito mediante Let's Encrypt o uso de certificados propios, gestión de usuarios, permisos y registro de auditoría.

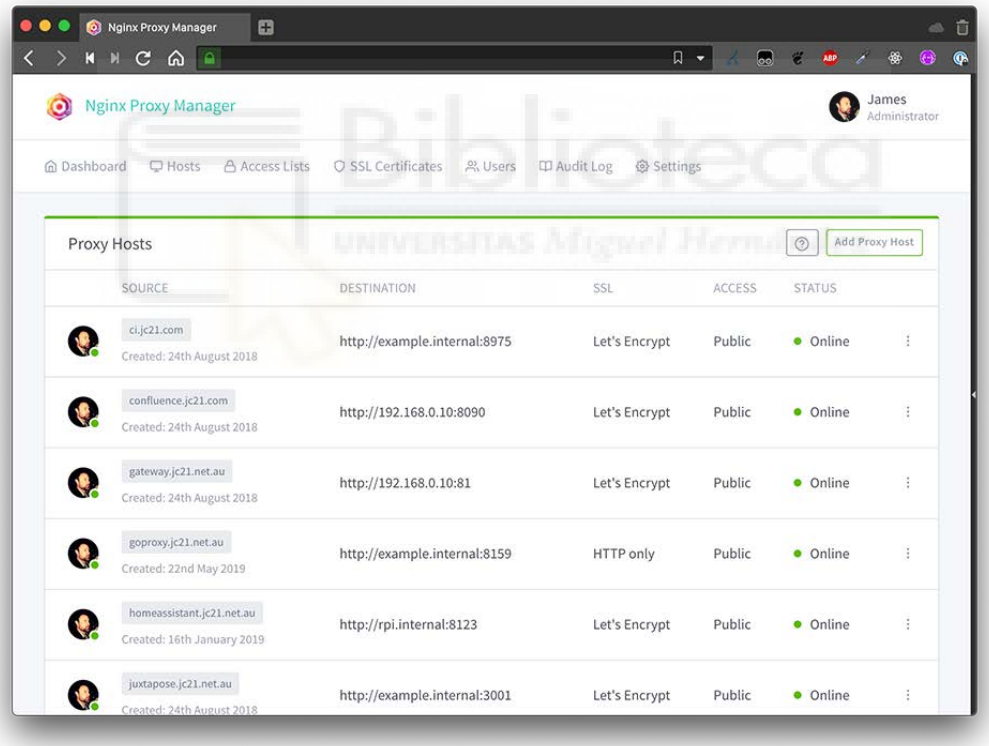

Figura 4 – Interfaz NGINX

Su principal característica es que, utilizando Nginx Proxy Manager como la puerta de enlace en el router, es capaz de reenviar el tráfico a otros servicios basados en web.

1.2.3. APACHE TRAFFIC SERVER

El software Apache Traffic Server es un servidor proxy de caché compatible con HTTP/1.1 y HTTP/2, rápido, escalable y extensible. Anteriormente un producto comercial, Yahoo! lo donó a la Fundación Apache, y actualmente es utilizado por varias de las principales CDN y propietarios de contenidos.

Figura 5 – Logo Apache traffic server

Mejora el tiempo de respuesta y reduce la carga del servidor y las necesidades de ancho de banda almacenando en caché y reutilizando páginas web, imágenes y llamadas a servicios web solicitados con frecuencia.

Sus principales ventajas consisten en un buen escalado en hardware SMP moderno, gestionando decenas de miles de peticiones por segundo y la inclusión de APIs para escribir plug-ins propios, desde modificar las cabeceras HTTP hasta gestionar peticiones ESI o escribir algoritmo de caché propio.

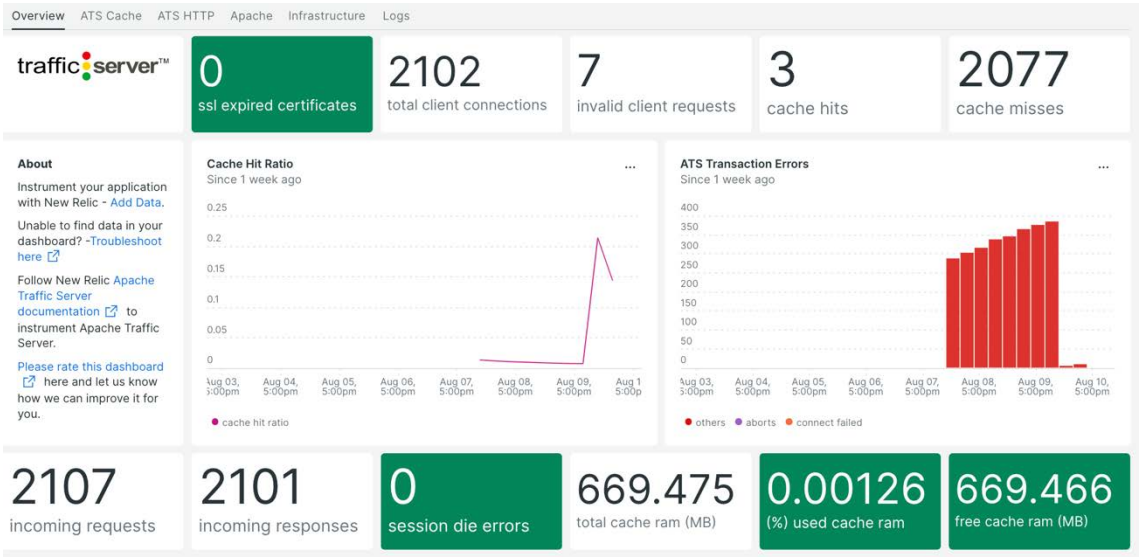

Figura 6 – Interfaz Apache traffic server

Es uno de los proxys más populares y seguros ya que se demuestran sus habilidades gestionando alrededor 400 TB diarios en Yahoo!.

1.3. OBJETIVOS

Este proyecto tiene como objetivo establecer un entorno sólido y eficiente en Azure, optimizando el control del tráfico web mediante Squid y facilitando el acceso remoto seguro mediante VPN Point-to-Site.

Más allá de garantizar la seguridad y el cumplimiento normativo a través de la cuidadosa configuración y certificados OpenSSL, se busca impulsar la automatización como un elemento clave.

La auditoría y registro detallados refuerzan la capacidad de detección y respuesta, mientras que la optimización de recursos en Azure busca equilibrar la eficiencia operativa y reducir costos. En resumen, este proyecto persigue la construcción de un entorno tecnológico efectivo en la nube de Azure.

1.4. CONCEPTOS BÁSICOS

1.4.1. ¿QUÉ ES UN PROXY?

Un servidor proxy es una tecnología que se utiliza como puente entre el origen (un ordenador) y el destino de una solicitud (Internet). Generalmente se trata de un dispositivo u ordenador intermedio que nos permite conectarnos a Internet de manera indirecta.

Cuando utilizamos un servidor proxy, toda la información pasa primero por él, este es el encargado de enviarlo al lugar de destino, impidiendo toda comunicación directa entre nuestro ordenador destino e Internet (u otro ordenador).

Figura 7 – Diagrama Proxy

Las funciones básicas de los proxys son:

- Control de acceso: los administradores del servidor proxy permiten o no que sus usuarios se conecten a ciertos sitios.

- Filtros de contenido: los administradores pueden bloquear sitios web específicos, así como categorías concretas.
- Caché: tras acceder a una página, el proxy guarda toda la información de la web en su sistema. Para futuras solicitudes a la misma página no tendrá que volver a conectarse a Internet.

Existe una amplia variedad de proxys, entre los que destacan:

1. Proxy web:

   Este es el tipo de proxy más habitual, al que todos los usuarios podemos acceder a través de una página web. De hecho, la web actúa como proxy. Se basa en HTTP y HTTPS actuando como intermediario para acceder a otros servicios en Internet. Toda nuestra navegación pasará por el proxy web que estemos utilizando.

2. Proxy caché:

   Este servidor es un intermediario entre la red a la que nos conectamos e Internet, para registrar el contenido antes: HTML, CSS e imágenes. Es muy utilizado para acelerar el contenido de un sitio al navegar. Los datos de una web quedan almacenados en la primera visita y si hay una segunda no necesita revisarlos todos de nuevo. Así el acceso es mucho más rápido.

3. Proxy inverso:

   Un proxy inverso es un servidor que acepta todo el tráfico y lo reenvía a un recurso específico. Este tipo de proxy aporta seguridad al servidor principal, restringiendo el acceso a rutas definidas. Esto permite, entre otras funciones, evitar ataques.

4. Proxy transparente:

Para utilizar este servidor proxy el usuario no tendrá que configurar nada antes de comenzar la navegación. Actúa como un intermediario entre nuestro equipo e Internet pero sin modificar nada.

5. Proxy NAT:

Este proxy es capaz de ocultar la identidad de los usuarios, enmascarando la auténtica dirección IP. Cuenta con una amplia variedad de configuraciones.

Squid [1] es un proxy caché para la web que admite HTTP, HTTPS, FTP y más. Reduce el ancho de banda y mejora los tiempos de respuesta al almacenar en caché y reutilizar páginas web solicitadas con frecuencia. Squid cuenta con controles de acceso extensos y funciona como un excelente acelerador de servidores. Se ejecuta en la mayoría de los sistemas operativos disponibles, incluido Windows, y está licenciado bajo la GPL de GNU.

Figura 8 – Logo Squid

Sus principales funciones son:

- Caché:

Almacena en su memoria caché el contenido web de acceso frecuente. Cuando un cliente solicita un recurso específico, Squid verifica si ya tiene una copia del contenido en su caché. Si es así, sirve el contenido directamente desde la caché,

lo que resulta en tiempos de respuesta más rápidos y un menor uso del ancho de banda de la red.

- Reenvío:

Puede reenviar solicitudes de clientes a servidores web en nombre de los clientes. Recupera el contenido solicitado del servidor y lo entrega de nuevo al cliente. Esto permite que el servidor proxy mejore la seguridad y brinde servicios adicionales como control de acceso, filtrado de contenido y optimización del tráfico.

- Control de Acceso:

Ofrece mecanismos extensos de control de acceso, lo que permite a los administradores definir políticas para restringir o permitir clientes específicos, direcciones IP o tipos de contenido. Esto permite que las organizaciones apliquen políticas de uso de internet, bloqueen sitios web maliciosos, eviten el acceso no autorizado y filtren contenido según reglas predefinidas.

- Autenticación:

Admite varios métodos de autenticación, incluidos nombre de usuario/contraseña, NTLM, LDAP y más. Esto permite que los administradores implementen controles de acceso basados en usuarios y rastreen las actividades individuales de los usuarios. La autenticación también permite que el servidor proxy brinde servicios personalizados y aplique diferentes niveles de acceso según los privilegios del usuario.

- Filtrado de Contenido:

Se puede configurar para filtrar contenido web según reglas o políticas predefinidas. Esta función permite que las organizaciones bloqueen o restrinjan el acceso a ciertos sitios web, controlen el tipo de contenido que se puede descargar y protejan a los usuarios de contenido malicioso o inapropiado.

- Optimización del Tráfico:

Incluye funciones como la modificación de solicitudes y respuestas, la optimización de protocolos y la compresión. Estas capacidades ayudan a optimizar el tráfico de red, reducir la latencia y mejorar el rendimiento general de la red.

1.4.2. ¿QUÉ ES AZURE?

Azure (Microsoft Azure) [2] es una plataforma de computación en la nube creada por Microsoft para construir, probar, desplegar y administrar aplicaciones y servicios mediante el uso de sus centros de datos. Es compatible con muchos lenguajes, herramientas y marcos de programación diferentes, incluidos software y sistemas específicos de Microsoft y de terceros.

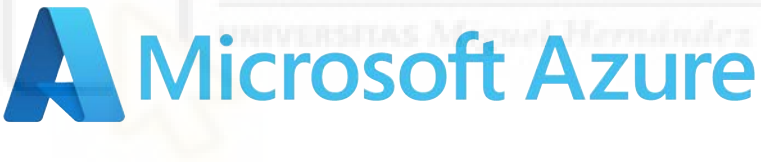

Figura 9 – Logo Azure

En la nube, los servicios se clasifican en varios modelos [3], entre ellos:

- **Infraestructura como Servicio (IaaS):** Permite evitar el coste y la complejidad de comprar y administrar servidores físicos e infraestructura de centro de datos. Cada recurso se ofrece como un componente de servicio aparte, y solo pagas por el tiempo que necesites un recurso concreto. El proveedor de servicios informáticos en la nube administra la infraestructura, mientras que tú compras, instalas, configuras y administras tu propio software (sistemas operativos, middleware y aplicaciones). En Azure, un ejemplo de IaaS es la oferta de Máquinas

Virtuales (VM). Los usuarios pueden aprovisionar y administrar VMs según sus necesidades, controlando aspectos como el sistema operativo, la configuración de red y la capacidad de almacenamiento.

- **Plataforma como Servicio (PaaS):** Hace referencia a los servicios de informática en la nube que suministran un entorno a petición para desarrollar, probar, entregar y administrar aplicaciones de software. PaaS está diseñado para facilitar a los desarrolladores la creación rápida de aplicaciones web o móviles, sin necesidad de preocuparse por la configuración o administración de la infraestructura de servidores subyacente, el almacenamiento, la red y las bases de datos necesarias para el desarrollo. Azure App Service es un ejemplo de PaaS en Azure.

- **Software como Servicio (SaaS):** Es un método de entrega de aplicaciones a través de Internet a petición y, normalmente, con una suscripción. Con SaaS, los proveedores de nube hospedan y administran las aplicaciones y la infraestructura subyacente. Estos proveedores también controlan el mantenimiento, como las actualizaciones del software y las revisiones de seguridad. Los usuarios se conectan a la aplicación a través de Internet, normalmente con un explorador web en su teléfono, tableta o PC. Microsoft 365 es un servicio SaaS de productividad en la nube en Azure. Incluye aplicaciones como Word, Excel y Outlook, junto con servicios de colaboración como SharePoint y Teams, entregados directamente a través de la nube.

- **Funciones como Servicio (FaaS):** Permite a los clientes ejecutar código en respuesta a sucesos, sin gestionar la compleja infraestructura asociada normalmente a la creación y lanzamiento de aplicaciones de microservicios. En Azure, Functions es un servicio sin servidor que permite ejecutar código en respuesta a eventos específicos sin preocuparse por la infraestructura. Los desarrolladores pueden centrarse en escribir funciones individuales, y Azure maneja la ejecución y escalabilidad automáticamente.

Microsoft Azure utiliza un sistema operativo especializado, llamado de la misma forma, para correr sus "capas", un cluster localizado en los servidores de datos de Microsoft que se encargan de manejar los recursos almacenados y procesamiento para proveer los recursos(o una parte de ellos) para las aplicaciones que se ejecutan sobre Microsoft Azure.

Azure se describe como una “capa en la nube” funcionando sobre un número de sistemas que utilizan Windows Server, estos funcionan bajo la versión 2008 de Windows Server y una versión personalizada de Hyper-V, conocido como el Hipervisor de Microsoft Azure que provee la virtualización de los servicios. La capa controladora de Microsoft Azure se encarga de escalar y de manejar la confiabilidad del sistema evitando así que los servicios se detengan si alguno de los servidores de datos de Microsoft tiene problemas y a su vez maneja la información de la aplicación web del usuario dando como ejemplo los recursos de la memoria o el balanceo del uso de esta.

Sus características principales son:

- Proceso:

  El servicio de proceso de Microsoft Azure ejecuta aplicaciones basadas en Windows Server. Estas aplicaciones se pueden crear mediante .NET Framework en lenguajes como C# y Visual Basic, o implementar sin .NET en C++, Java y otros lenguajes.

- Almacenamiento:

  Objetos binarios grandes (blobs) proporcionan colas para la comunicación entre los componentes de las aplicaciones de Windows Azure y ofrece un tipo de tablas con un lenguaje de consulta simple.

- Servicios de infraestructura:

  Posibilidad de desplegar de una forma sencilla máquinas virtuales con Windows Server o con distribuciones de Linux.

- Controlador de tejido:

Microsoft Azure se ejecuta en un gran número de máquinas. El trabajo del controlador de tejido es combinar las máquinas en un solo centro de datos de Microsoft Azure formando un conjunto armónico. Los servicios de proceso y almacenamiento de Microsoft Azure se implementan encima de toda esta eficacia de procesamiento.

- Red de entrega de contenido (CDN): el almacenamiento en caché de los datos a los que se accede frecuentemente cerca de sus usuarios agiliza el acceso a esos datos.

- Connect: organizaciones interactúan con aplicaciones en la nube como si estuvieran dentro del propio firewall de la organización.

- Administración de identidad y acceso: La solución Active directory permite gestionar de forma centralizada y sencilla el control de acceso y la identidad. Esta solución es perfecta para la administración de cuentas y la sincronización con directorios locales.

1.4.3. PRODUCTOS EN AZURE

Azure cuenta con más de 200 productos y servicios en la nube. Los servicios fundamentales empleados en este proyecto son de Infraestructura como Servicio (IaaS), así como Plataforma como Servicio (PaaS). Se van a dar una definición de los principales productos utilizados:

- **Azure Virtual Network:**

Azure Virtual Network (VNet) proporciona el fundamento para construir una red privada en Azure (clasificado como IaaS), permitiendo la comunicación segura entre varios recursos, incluyendo máquinas virtuales, tanto entre ellos como con Internet y redes locales. Similar a una red tradicional, ofrece beneficios adicionales de la infraestructura de Azure, como escalabilidad, disponibilidad y aislamiento.

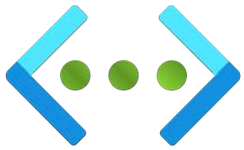

Figura 10 – Logo Virtual Network

Entre los escenarios clave que puede lograr con una red virtual se incluyen:

- Comunicación de los recursos Azure con Internet.
- Comunicación entre los recursos de Azure.
- Comunicación con recursos locales.
- Filtrado del tráfico de red.
- Enrutamiento del tráfico de red.
- Integración con servicios de Azure.

De manera predeterminada, todos los recursos de una red virtual pueden comunicarse con Internet. También puede utilizar una dirección IP pública, una puerta de enlace NAT o un equilibrador de carga pública para administrar sus conexiones salientes. Puede comunicarse de entrada con un recurso asignándole una dirección IP pública o un equilibrador de carga pública.

Para enrutar el tráfico puede implementar tanto tablas de enrutamiento o rutas del protocolo de puerta de enlace de borde (BGP).

- **VPN Point-to-Site:**

Una conexión de puerta de enlace de VPN de punto a sitio (P2S) permite crear una conexión segura a la red virtual desde un equipo cliente individual (clasificado como IaaS). Se establece una conexión de punto a sitio al iniciarla desde el equipo cliente. Esta solución resulta útil para los teletrabajadores que

deseen conectarse a redes virtuales de Azure desde una ubicación remota, por ejemplo, desde casa o un congreso. La conexión VPN de punto a sitio también es una solución útil en comparación con la conexión VPN de sitio a sitio cuando solo necesitan conectarse a la red virtual algunos clientes.

Figura 11 – Logo VPN Gateway

La conexión VPN de punto a sitio usa uno de los siguientes protocolos:

- Protocolo OpenVPN, un protocolo VPN basado en SSL/TLS. Una solución de VPN basada en TLS puede penetrar firewalls, puesto que la mayoría de ellos abre el puerto TCP 443 saliente, que usa TLS. OpenVPN puede utilizarse para la conexión desde dispositivos Android, iOS (versión 11.0 y posteriores), Windows, Linux y Mac (macOS 10.13 y versiones posteriores).

- El protocolo de túnel de sockets seguro (SSTP), que es un protocolo VPN propio basado en TLS. Una solución de VPN basada en TLS puede penetrar firewalls, puesto que la mayoría de ellos abre el puerto TCP 443 saliente, que usa TLS. El protocolo SSTP solo se admite en dispositivos Windows. Azure es compatible con todas las versiones de Windows que tienen SSTP y admiten TLS 1.2 (Windows 8.1 y versiones posteriores).

- VPN IKEv2, una solución de VPN con protocolo de seguridad de Internet basada en estándares. La conexión VPN IKEv2 puede utilizarse para la conexión desde dispositivos Mac (versión macOS 10.11 y posteriores).

Para que Azure acepte una conexión VPN de punto a sitio, el usuario primero debe autenticarse. Azure ofrece diferentes mecanismos para autenticar los

usuarios que se conectan. Con la autenticación con certificados nativos de Azure, para autenticar el usuario que se conecta, se usa un certificado de cliente presente en el dispositivo. Los certificados de cliente se generan a partir de un certificado raíz de confianza y se instalan en cada equipo cliente. Puede, o bien, usar un certificado raíz generado mediante una solución empresarial, o bien, generar un certificado autofirmado.

La validación del certificado de cliente se realiza mediante la puerta de enlace de VPN durante el establecimiento de la conexión VPN P2S. El certificado raíz es necesario para la validación y se debe cargar en Azure.

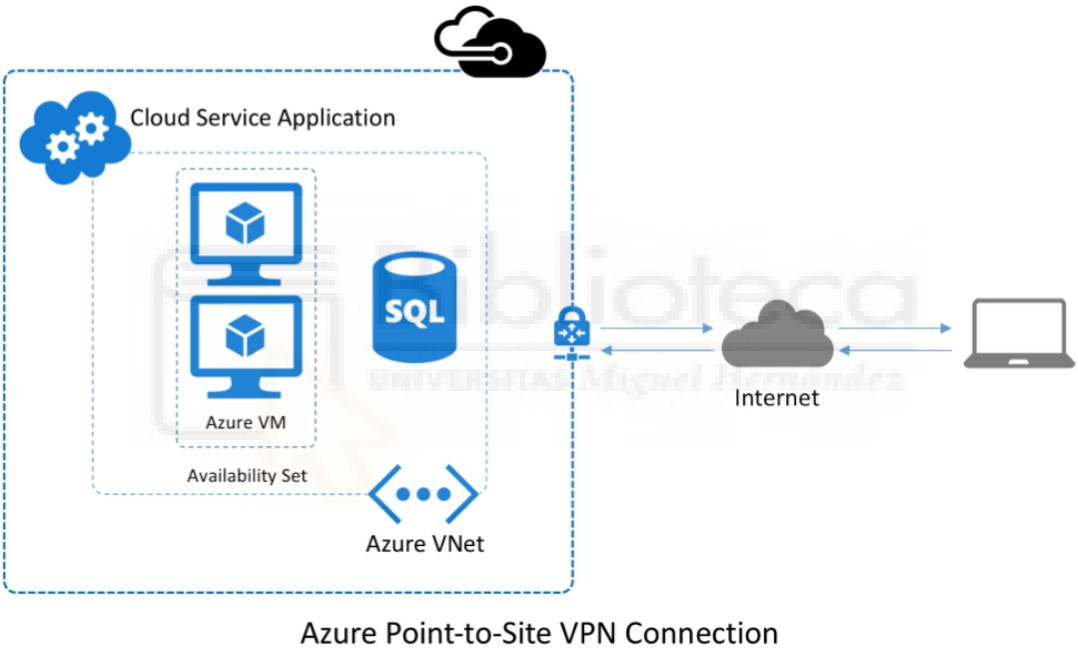

Figura 12 – Diagrama VPN Poit-to-Site

- **Azure Virtual Machine (VM):**

Una máquina virtual (VM) consiste en la creación de una versión basada en software o "virtual" de un equipo, con cantidades dedicadas de CPU, memoria y almacenamiento que se "toman prestadas" de un equipo host físico, como un PC, o un servidor remoto, como un servidor en el centro de datos de un proveedor de nube (clasificada como IaaS). Una máquina virtual es un archivo

de PC, que suele denominarse imagen, que se comporta igual que un equipo real.

En Azure podemos encontrar tanto VM de sistemas operativos Windows como de Linux.

Figura 13 – Logo VM

Ofrecen ventajas como:

- Ahorro de costes: la ejecución de varios entornos virtuales en una única infraestructura significa que puedes reducir drásticamente la superficie física de la infraestructura. Esto aumenta los beneficios, ya que reduce la necesidad de mantener tantos servidores y los costes de mantenimiento y electricidad.

- Agilidad y velocidad: la puesta en marcha de una máquina virtual es relativamente fácil y rápida, y es mucho más sencilla para tus desarrolladores que el aprovisionamiento de un entorno nuevo completo. La virtualización hace que el proceso de ejecución de escenarios de desarrollo y pruebas sea mucho más rápido.

- Tiempo de inactividad reducido: las máquinas virtuales son muy portátiles y fáciles de migrar de un hipervisor a otro en un equipo diferente, por lo que son una solución excelente para copias de seguridad, en el caso de que el host deje de funcionar de forma inesperada.

- Escalabilidad: las máquinas virtuales permiten escalar más fácilmente las aplicaciones agregando más servidores virtuales o físicos para distribuir la carga de trabajo entre varias máquinas virtuales. Como resultado, puedes aumentar la disponibilidad y el rendimiento de las aplicaciones.

- Ventajas de seguridad: dado que las máquinas virtuales se ejecutan en varios sistemas operativos, el uso de un sistema operativo invitado en una máquina virtual permite ejecutar aplicaciones de una seguridad dudosa y proteger el sistema operativo host. Las máquinas virtuales también permiten un mejor análisis forense de la seguridad y suelen usarse para estudiar virus informáticos de forma segura, aislándolos para evitar riesgos en el equipo host.

- **Azure Automation**:

Azure Automation ofrece un servicio basado en la nube de automatización, actualización de sistemas operativos y configuración, que posibilita una administración coherente en sus entornos, sean o no de Azure (clasificado como PaaS). Incluye automatización de procesos, administración de configuración, administración de actualizaciones, funcionalidades compartidas y características heterogéneas sin requerir la gestión directa de la infraestructura subyacente. La automatización de procesos de Azure Automation permite automatizar tareas de administración frecuentes, laboriosas y propensas a errores.

Figura 14 -Logo Automation

Se utilizan mayoritariamente **RunBooks**, que son objetos que permiten la ejecución de tareas dentro de Azure Automation. Existen dos tipos: RunBooks de PowerShell o de Phyton.

Los runbooks deben incluir lógica para tratar con recursos, por ejemplo, las máquinas virtuales, la red y los recursos de la red. Los recursos están vinculados a una suscripción de Azure y los runbooks requieren credenciales adecuadas para acceder a cualquier recurso.

Un runbook requiere credenciales adecuadas para acceder a cualquier recurso, ya sea en sistemas de Azure o de terceros. Estas credenciales se almacenan en Azure Automation, Key Vault, etc.

Automation admite un entorno para ejecutar trabajos de la misma cuenta de automatización. Un runbook individual puede tener muchos trabajos que se ejecutan al mismo tiempo. Cuantos más trabajos se ejecuten al mismo tiempo, más a menudo se podrán enviar al mismo espacio aislado. Un máximo de 10 trabajos se puede ejecutar en un espacio aislado. Se quitará un espacio aislado cuando no se ejecute ningún trabajo en él; Por lo tanto, no se debe usar para guardar archivos.

Un concepto importante son los **webhook**, los cuales permiten que un servicio externo inicie un runbook determinado en Azure Automation mediante una sola solicitud HTTP. Estos servicios pueden usar un webhook para iniciar un runbook sin tener que implementar la API completa de Azure Automation.

Azure Automation usa Microsoft Defender for Cloud para proporcionar seguridad a los recursos y detectar riesgos en los sistemas Linux. La seguridad se proporciona en todas las cargas de trabajo, independientemente de si los recursos se encuentran en Azure o no.

2. MATERIAL Y MÉTODOS

2.1. ESPECIFICACIONES TÉCNICAS

La implementación exitosa de este proyecto requiere la definición de especificaciones técnicas específicas [4], meticulosamente diseñadas teniendo en cuenta las necesidades de una pequeña empresa con aproximadamente 20 empleados. Estas especificaciones están dirigidas a garantizar un rendimiento óptimo y una gestión eficiente del tráfico web a través del servidor proxy Squid y la infraestructura de red en Azure.

Es importante destacar que, al considerar la expansión de este escenario para acomodar un mayor volumen de tráfico, será esencial revisar y ajustar las especificaciones tanto de la Máquina Virtual como de la Red Virtual. Esta revisión se vuelve imperativa para aumentar el límite de tráfico máximo de entrada/salida y así prevenir posibles desbordamientos o cuellos de botella que podrían afectar el rendimiento del sistema.

La capacidad de detectar de manera proactiva cualquier indicio de saturación o problemas de rendimiento se logrará gracias a las avanzadas herramientas de monitoreo proporcionadas por la plataforma Azure. Estas herramientas permitirán una supervisión continua y detallada del tráfico de red, facilitando la identificación temprana de cualquier anomalía y proporcionando la base para la toma de decisiones informadas en términos de ajustes y mejoras en la configuración.

2.1.1. ESPECIFICACIONES PROXY SQUID

Para la realización de este proyecto se ha escogido la versión Squid-6.2 [1], la versión más reciente, la cual ha sido seleccionada debido a sus características avanzadas y mejoras significativas. Entre las novedades destacadas:

- TLS ServerHello: Squid ahora es más indulgente con la configuración mal realizada de tls-cert=. Ajusta automáticamente la cadena de CA y el orden de los certificados para cumplir con TLS ServerHello.
- Registro de Secretos TLS: Squid registra secretos pre-maestros y detalles de cifrado relacionados con conexiones TLS, proporcionando una visión detallada de las comunicaciones seguras.
- Control de Acceso Mejorado: Se ha implementado una prohibición activa de cambios de clave ACL específicos, mejorando la seguridad y la coherencia en la configuración.
- Bloqueo de Tráfico to-local: Squid ahora incorpora un ACL predefinido para bloquear solicitudes desde 169.254.0.0/16 y fe80::/10, fortaleciendo la seguridad y el control de tráfico.
- Soporte RFC 9211: Estado de Caché HTTP: Introduce un nuevo encabezado HTTP que reemplaza X-Cache y X-Cache-Lookup, siguiendo el estándar RFC 9211 [15] para mejorar la interoperabilidad.
- Eliminación del Soporte del Protocolo Gopher: Squid-6.2 elimina el soporte para el protocolo Gopher, simplificando el manejo de solicitudes desconocidas.
- Eliminación de Herramientas Obsoletas: Se han eliminado herramientas desactualizadas, mejorando la eficiencia y mantenibilidad del sistema.

Estas mejoras y más hacen de Squid-6.2 la elección ideal para construir un entorno robusto y eficiente en Azure, asegurando un control de tráfico avanzado y una experiencia de usuario segura.

2.1.2. ESPECIFICACIONES VM

Especificación	Detalles
Región	Europa Occidental
Sistema operativo	Linux (Ubuntu 22.04)
Arquitectura	X64

Espacio de disco	B1s Standard
vCPUs	1
RAM	1 GB
SSD	4 GB
Zona de disponibilidad	No

Tabla 1 – Especificaciones VM

Para garantizar un rendimiento óptimo y costos eficientes, se ha seleccionado una Máquina Virtual con especificaciones cuidadosamente ajustadas para las necesidades de una pequeña empresa. Ubicada en la región de Europa Occidental, esta VM opera bajo el sistema operativo Linux (Ubuntu 22.04) con arquitectura x64, proporcionando una base estable y segura.

Con 1 vCPU y 1 GB de RAM, esta configuración es ideal para operaciones empresariales cotidianas y garantiza la capacidad necesaria para ejecutar aplicaciones esenciales. El espacio de disco de 4 GB en formato SSD mejora la velocidad de acceso a datos críticos, mientras que la omisión de una zona de disponibilidad simplifica la gestión sin comprometer la confiabilidad.

Estas especificaciones han sido elegidas considerando la eficiencia operativa y el equilibrio entre rendimiento y costos, ofreciendo a la pequeña empresa una solución robusta y rentable para satisfacer sus necesidades informáticas.

2.1.3. ESPECIFICACIONES RED VIRTUAL

Especificación	Detalles
Transferencia de datos de salida	100 GB
Región	Europa Occidental (2 regiones de redundancia)

Tabla 2 – Especificaciones Red Virtual

La Virtual Network (VNet) asociada a este proyecto presenta características clave diseñadas para optimizar la conectividad y la eficiencia en el entorno cloud. A continuación, se detallan sus especificaciones:

- Capacidad de Transferencia: La VNet cuenta con una robusta capacidad de transferencia de datos de salida de 100 GB, asegurando un rendimiento óptimo para las operaciones y servicios alojados en la red.
- Región: La red está desplegada en la región Europa Occidental, beneficiándose de la redundancia proporcionada por la presencia en dos regiones. Esta redundancia contribuye a garantizar la disponibilidad y la resistencia ante posibles fallas o interrupciones.
- Rangos de Direcciones IP Privadas: La VNet permite la definición de rangos de direcciones IP privadas, proporcionando flexibilidad en la asignación de direcciones para máquinas virtuales y otros recursos dentro de la red. Esta capacidad es fundamental para la organización y el control de la infraestructura.
- Conectividad Eficiente: Facilita la conectividad entre máquinas virtuales alojadas dentro de la misma red virtual, creando un entorno de comunicación eficiente y seguro. Además, posibilita la conexión con redes locales mediante opciones como VPN o ExpressRoute, brindando flexibilidad en la interconexión de entornos on-premise y en la nube.
- Integración de DNS Azure: La VNet se integra con el servicio de DNS de Azure, simplificando la administración de nombres de dominio y mejorando la resolución de nombres dentro de la red. Esta integración contribuye a una gestión más eficaz de los recursos y servicios desplegados en la infraestructura.

2.1.4. ESPECIFICACIONES VPN GATEWAY

Especificación	Detalles
SKU	VpnGw1
Túneles de S2S/VNet a VNet	máx. 30
P2S IKEv2	máx. 250
Agregado de rendimiento	650 Mbps
BGP	Compatible
Con redundancia	No
Núm. de VM admitidas	450

Tabla 3 – Especificaciones VPN Gateway

La VPN Gateway seleccionada para este proyecto opera bajo el SKU VpnGw1, ofreciendo un equilibrio efectivo entre rendimiento y capacidad. Con un límite máximo de 30 túneles de Site-to-Site (S2S) o de VNet a VNet, garantiza conectividad segura y escalable entre distintos entornos de red.

En términos de acceso remoto, la VPN Gateway admite hasta 250 conexiones Point-to-Site (P2S) utilizando IKEv2, proporcionando una solución robusta para conexiones seguras desde dispositivos individuales. Con un rendimiento agregado de hasta 650 Mbps, la VPN Gateway asegura un ancho de banda suficiente para operaciones y transferencia de datos dentro de la red virtual.

Compatibilidad con el protocolo BGP (Border Gateway Protocol) brinda flexibilidad en la gestión dinámica de rutas, mejorando la eficiencia de la red. Aunque no incluye redundancia, sus características y capacidades la hacen idónea para cumplir con los requisitos específicos del proyecto.

Con la capacidad de admitir hasta 450 máquinas virtuales, la VPN Gateway es una elección pertinente para entornos que demandan escalabilidad.

2.1.5. ESPECIFICACIONES AUTOMATION

Especificación	Detalles
Región	Europa Occidental
Funcionalidad	Automatización de procesos
Tiempo de ejecución de trabajos	500 minutos
Ispectores	1 x 750 horas

Tabla 4 – Especificaciones Automation

Azure Automation se utiliza para automatizar procesos y tareas en la nube de Azure. Esto puede incluir la creación, implementación y administración automatizada de recursos, así como la ejecución de flujos de trabajo y scripts.

Se dispone de un límite de 500 minutos para la ejecución de trabajos en Azure Automation. Este límite puede afectar la duración total de los trabajos que se pueden ejecutar dentro de un período específico.

Se dispone de un inspector con una asignación de 750 horas. Una tarea de inspector te permite estar atento a eventos y activar acciones. Está compuesta por un runbook de observador y un runbook de acción. El inspector busca un evento y activa la acción cuando ocurre un evento. Por ejemplo, puedes observar una carpeta en busca de archivos nuevos y activar una acción que respalde esos archivos cuando se crean.

2.2. CREACIÓN MÁQUINA VIRTUAL

El primer paso será la creación de una Máquina Virtual en el entorno de Azure [5], para ello debemos crear una cuenta que contará con una Suscripción de Azure. Necesitamos crear un nuevo grupo de recursos para organizar los recursos del proyecto, para ello desde el Marketplace de Azure buscamos la opción y elegimos la zona geográfica donde estarán alojados los recursos dentro del mismo, en nuestro caso la región de Europa Occidental, esto se refiere a dónde estará situado el centro de datos de Azure. El precio de los recursos dependerá de la región donde los despleguemos ya que hay que tener en cuenta latencia, rendimiento, disponibilidad, redundancia y cumplimiento normativo.

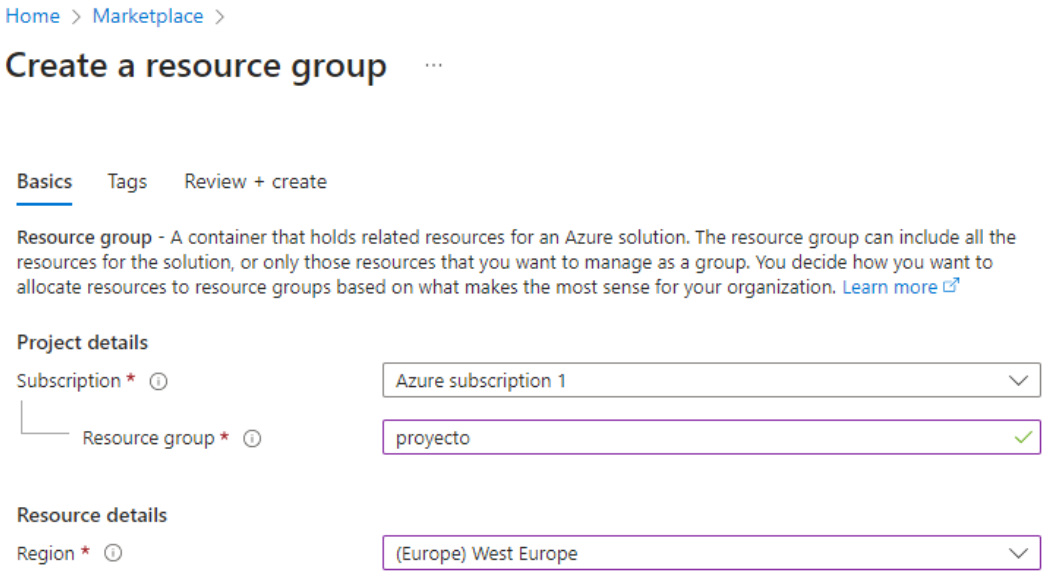

Figura 15 – Creación Grupo de recursos

Cuando tengamos nuestro grupo de recursos creado, seleccionaremos "Agregar" y buscaremos "Máquina Virtual" en el Marketplace de Azure. Las especificaciones técnicas de la máquina virtual para este proyecto son las ya mencionadas. Debemos añadir el tipo de autenticación que utilizaremos al acceder por SSH a la máquina, en este caso escogeremos contraseña ya que se trata de la red privada, pero podríamos indicar autenticación por llave pública de SSH.

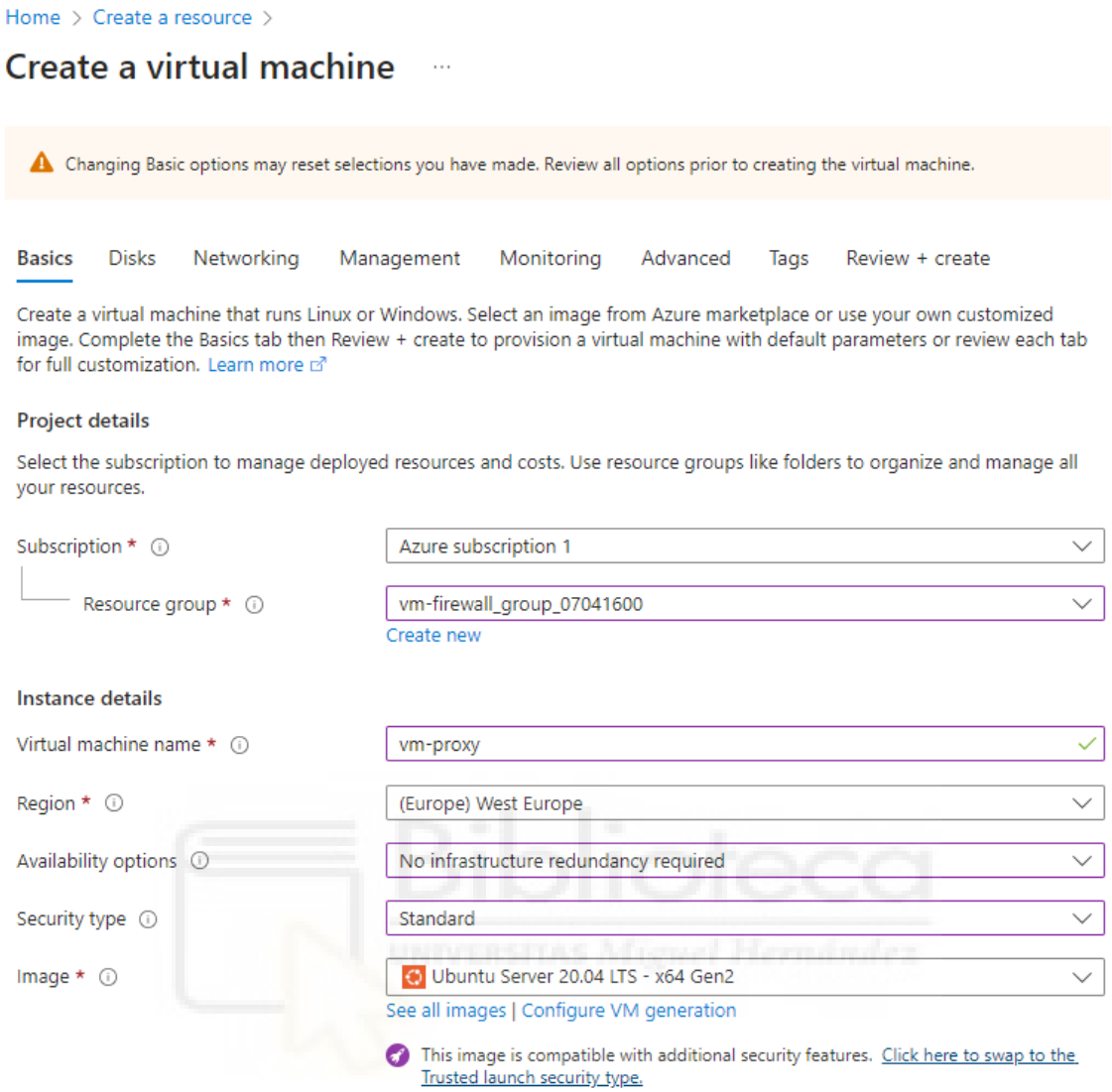

Figura 16 – Creación Máquina Virtual 1

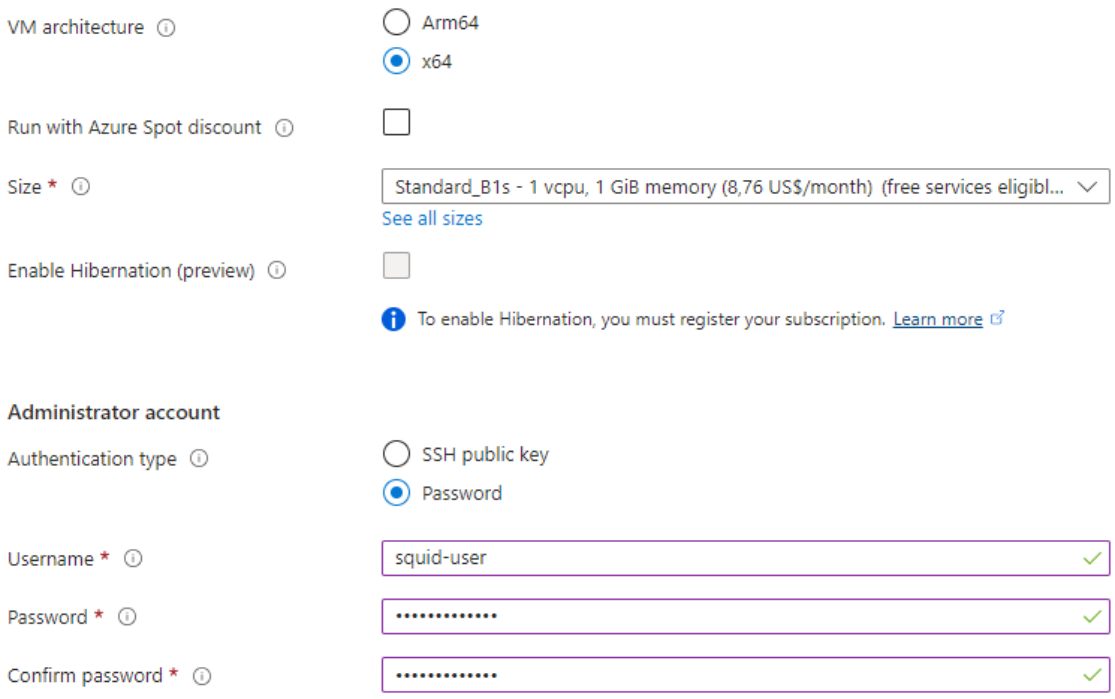

Figura 17 – Creación Máquina Virtual 2

Al seleccionar la pestaña de “Networking” lo que nos va a pedir es una Red Virtual que administrará el direccionamiento de los recursos [6], nos permitirá asignar IPs y crear rutas.

Para el Espacio de Direccionamiento de nuestra nueva Red Virtual escogeremos una clase A: 10.0.0.0/16 (65536 direcciones disponibles dentro de la Red Virtual) y para el apartado de Subredes, indicaremos que cuando dividir nuestro Espacio de direcciones usaremos un /24 (256 direcciones disponibles dentro de la misma subred).

Cuando tenemos creada nuestra Red Virtual, asignamos una subred a nuestra máquina virtual, por ejemplo 10.1.1.0/24. Necesitamos asignarle una IP pública, por lo que escogeremos que se nos asigne una nueva a partir del Espacio de Direccionamiento público de Azure. Por último, escogeremos los puertos de entrada que tendrá permitidos la máquina, que serán HTTPS (puerto 443), HTTP (puerto 80) y SSH (puerto 22).

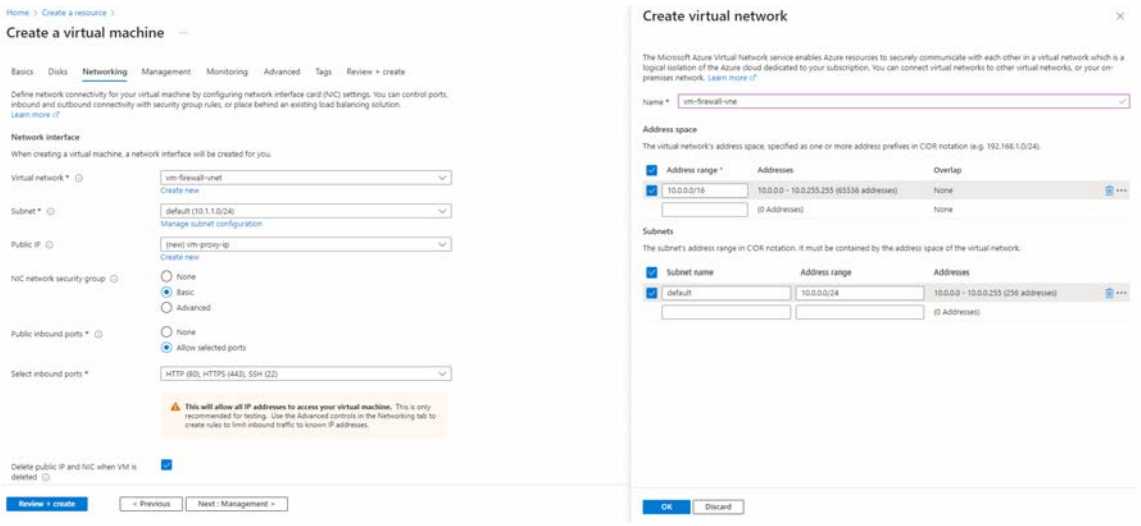

Figura 18 – Creación Virtual Network

Las demás pestañas se dejarán por defecto.

Podremos comprobar el funcionamiento de la máquina virtual accediendo por SSH, por ejemplo, desde una consola de Windows con la siguiente llamada:

ssh usuario@ip_máquina.

2.3. VPN GATEWAY

El próximo paso es la creación de la puerta de enlace de VPN [7], que nos servirá tanto para la administración del sistema como el acceso por parte de los usuarios a la red privada y uso del proxy.

Dentro de la configuración de la red virtual donde se ubica la Máquina Virtual, seleccionaremos "Gateway de VPN". Aquí indicaremos los parámetros necesarios.

La SKU (Stock Keeping Unit) se refiere a la unidad de mantenimiento de inventario utilizada para identificar y gestionar los diferentes tipos y configuraciones de servicios o productos en la plataforma de nube. En el caso específico de una VPN Gateway en Azure, la SKU se refiere al tipo y nivel de servicio que se elige para el Gateway de VPN. En nuestro caso escogeremos

VpnGW1, la más básica de las opciones, ya que no será necesario el acceso de un gran número de usuarios.

Debemos, además, asignar una IP pública al Gateway, que de nuevo será del Espacio de Direccionamiento Público de Azure. La IP pública de la VPN Gateway permite que los clientes externos, como usuarios remotos o sucursales, se conecten a la red virtual en Azure a través de Internet de manera segura. Esta dirección IP es la puerta de entrada a la red virtual desde ubicaciones externas.

No activaremos las opciones “Enable Active-Active Mode” y “Configure BGP” ya que no necesitamos redundancia y crearemos las rutas de manera estática por lo que no será necesario un protocolo de enrutamiento dinámico.

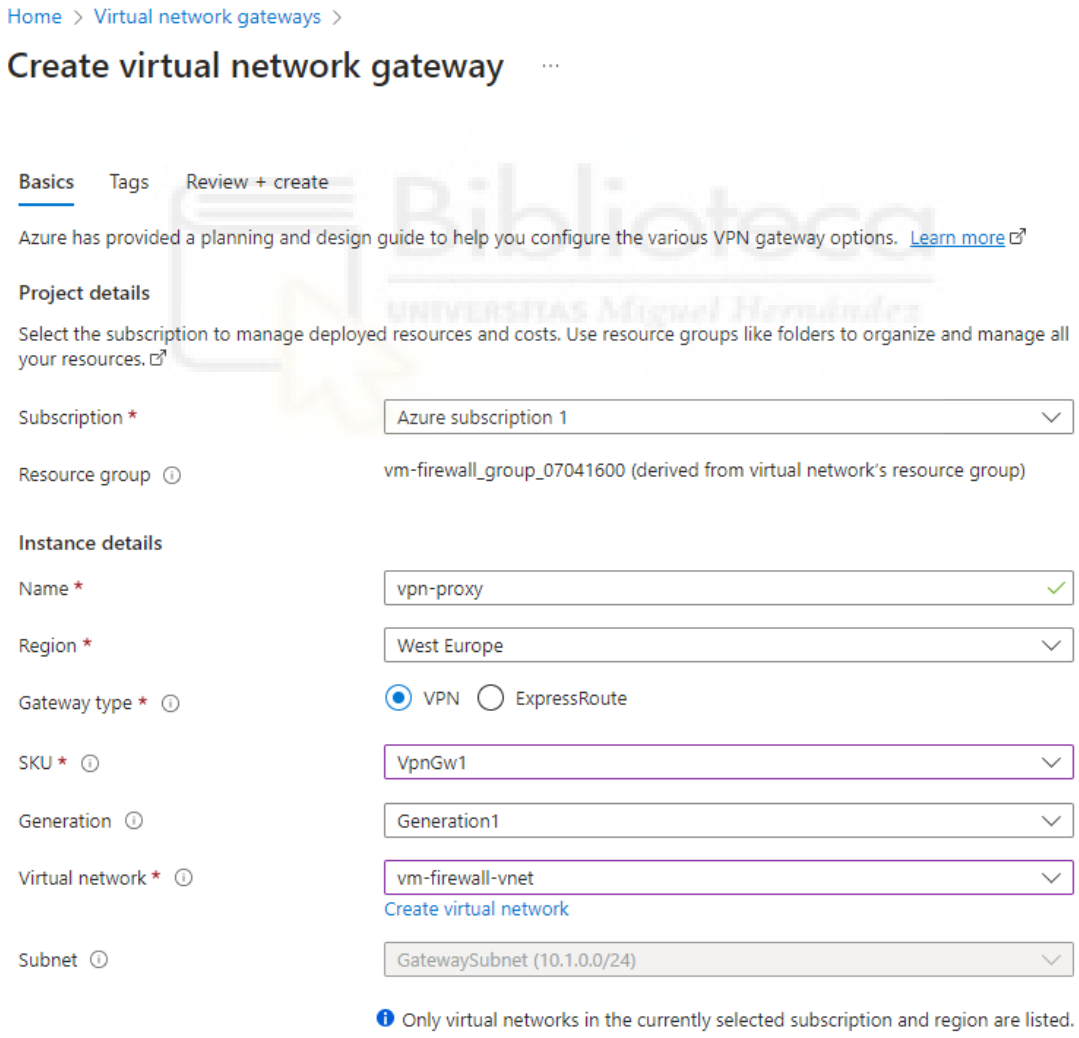

Figura 19 – Creación VPN Gateway 1

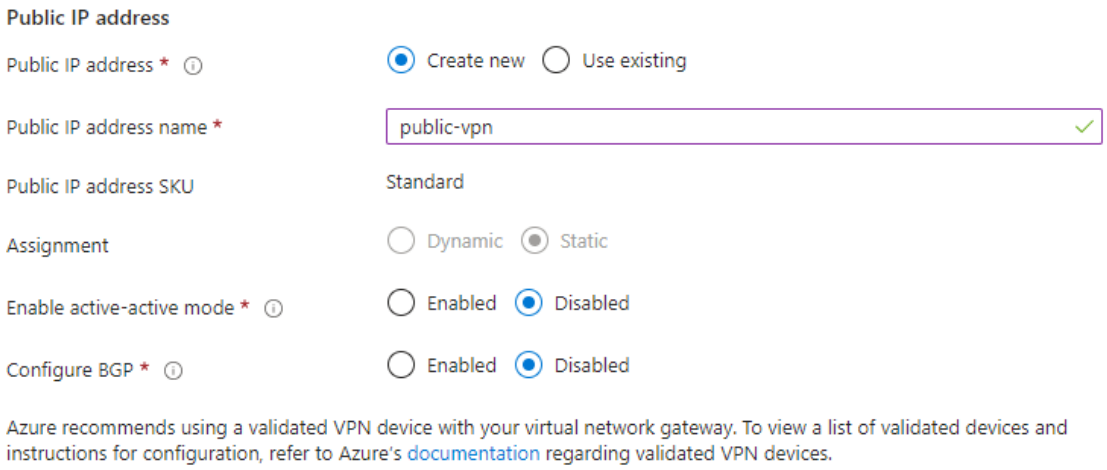

Figura 20 – Creación VPN Gateway 2

Ya creada la VPN Gateway, podremos crear una VPN Point-to-Site (P2S), que consiste en un tipo de conexión de red virtual en la nube que permite a los usuarios individuales o dispositivos remotos conectarse de forma segura a una red virtual en Microsoft Azure. A diferencia de una VPN Site-to-Site, que conecta redes enteras, una VPN Point-to-Site está diseñada para conectar dispositivos individuales como ordenadores o móviles, a la red virtual de Azure.

2.3.1. GENERACIÓN CERTIFICADO OPENSSL

Para la creación de nuestra Point-to-Site será necesario la generación de un certificado autofirmado, utilizaremos en nuestro caso OpenSSL [8]. Este certificado será utilizado para la autenticación de los usuarios para establecer la conexión. La firma del certificado raíz debe coincidir en ambos puntos para que la autenticación sea exitosa.

Para la generación del mismo utilizaremos desde un ordenador Mac la terminal de comandos del propio ordenador (la terminal de macOS utiliza una interfaz de línea de comandos que se basa en el sistema operativo Unix), desde un equipo Windows necesitaríamos utilizar Windows PowerShell. Los comandos necesarios se describen a continuación:

- Generamos la clave privada de la CA Ráiz:

```
openssl genpkey -algorithm RSA -out rootCA.key
```

open ssl genpkey	Comando para generar una clave privada.
-algorithm RSA	Especifica el algoritmo de cifrado RSA para la generación de la clave.
-out client-key.pem	Nombre del archivo donde se guardará la clave privada

Tabla 5 – Clave privada de la CA de Raíz

- Creación del Certificado Autofirmado de la CA Raíz:

```
openssl req -x509 -new -key rootCA.key -out rootCA.crt
```

open ssl req	Comando para solicitudes de certificados
-x509	Indica que se está generando un certificado X.509 autofirmado.
-new	Crea una nueva solicitud de certificado.
-key rootCA.key	Especifica la clave privada a utilizar.
-out rootCA.crt	Nombre del archivo donde se guardará el certificado

Tabla 6 – Certificado Autofirmado de la CA de Raíz

- Generamos la clave privada del cliente:

```
openssl genpkey -algorithm RSA -out client.key
```

open ssl genpkey	Comando para generar una clave privada.
-algorithm RSA	Especifica el algoritmo RSA para la generación de la clave.
-out client.key	Nombre del archivo donde se guardará la clave privada

Tabla 7 – Clave privada del cliente

- Creación de una solicitud de certificado (CSR) para cliente:

```
openssl req -new -key client.key -out client.csr
```

openssl req	Comando para generar una solicitud de certificado.
-new	Indica que se está generando una nueva solicitud.
-key client.key	Especifica la clave privada asociada.
-out client.csr	Nombre del archivo donde se guardará la solicitud de certificado.

Tabla 8 – Certificado de cliente

Durante este paso, se solicitará información adicional, como el país, la organización, el nombre común, etc.

- Firmar el Certificado con la Clave Privada (Self-Signed): En este paso, se utiliza la clave privada para firmar la solicitud de certificado, generando así el certificado autofirmado.

```
openssl x509 -req -in client.csr -CA rootCA.crt -CAkey
rootCA.key -CAcreateserial -out client.crt
```

openssl x509	Comando para realizar operaciones en certificados x509
-req	Indica que se está procesando una solicitud de certificado.
-in client.csr	Especifica la solicitud de certificado del cliente.
-CA rootCA.crt	Indica el certificado de la CA raíz.
-CAkey rootCA.key	Indica la clave privada de la CA raíz.
-CAcreateserial	Crea un número de serie para el nuevo certificado.
-out client.crt	Nombre del archivo donde se guardará el certificado autofirmado.

Tabla 9 – Firma del certificado de cliente con la clave privada

Este proceso nos devolverá los certificados de raíz y de cliente. El certificado raíz es el que deberemos de utilizar para la creación de la VPN P2S. El certificado de cliente es el que instalaremos en el llavero de certificados de cada equipo de los usuarios. Debemos de disponer de la contraseña que se especificó a la hora de generar el certificado, de esta manera podremos exportarlo sin complicaciones.

2.3.2. VPN POINT-TO-SITE

Para la creación de la VPN P2S [9], dentro del VPN Gateway seleccionaremos “Configuración Point-to-site”, aquí nos pedirá un pool de direcciones, seleccionamos una subred dentro del direccionamiento de nuestra Virtual Network, los clientes VPN reciben de forma dinámica una dirección IP del intervalo especificado.

En la opción “Tipo de túnel” se especifica el tipo de túnel que se requiere y el software cliente VPN que se usará para realizar la conexión desde el sistema operativo del usuario.

IKEv2 es la siguiente versión del protocolo Internet Key Exchange que se utiliza para negociar una Asociación de Seguridad al principio de una sesión IPsec. El cliente VPN nativo para iOS y macOS solo puede usar el tipo de túnel IKEv2 para conectarse a Azure por lo que seleccionaremos esta opción. En un sistema de Windows podríamos usar OpenVPN, que ofrece conectividad VPN con validación jerárquica de usuarios y host conectados remotamente.

En la pestaña “tipo de autenticación” podremos seleccionar autenticación contra servidor de Radius o contra Azure AD. En nuestro caso especificaremos el tipo “certificado de Azure” que nos permitirá indicar el certificado autofirmado creado anteriormente.

En la pestaña “Certificados de raíz” es donde especificaremos la firma del certificado autofirmado. Indicaremos un nombre y en la opción “datos del certificado público” extraemos esta información del propio .crt del certificado. Para ello abriremos el .crt en un bloc de notas, copiaremos la sección entre “Begin” y “End Certificate” (esta es la firma que debe coincidir entre el cliente y el Gateway).

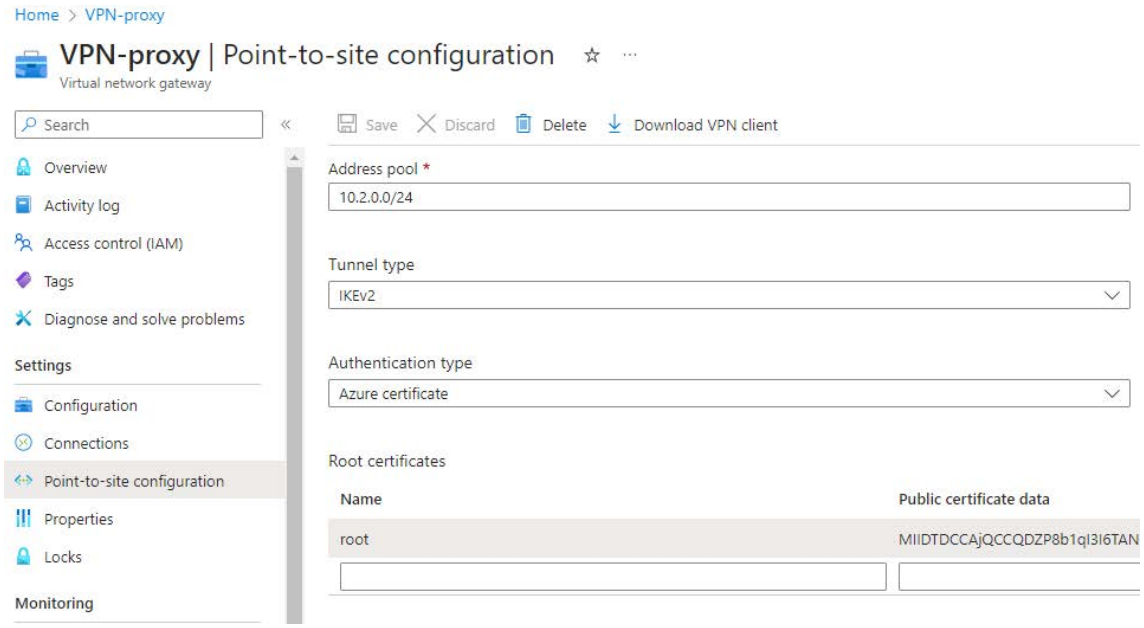

Figura 21 – Configuración VPN Point-to-Site

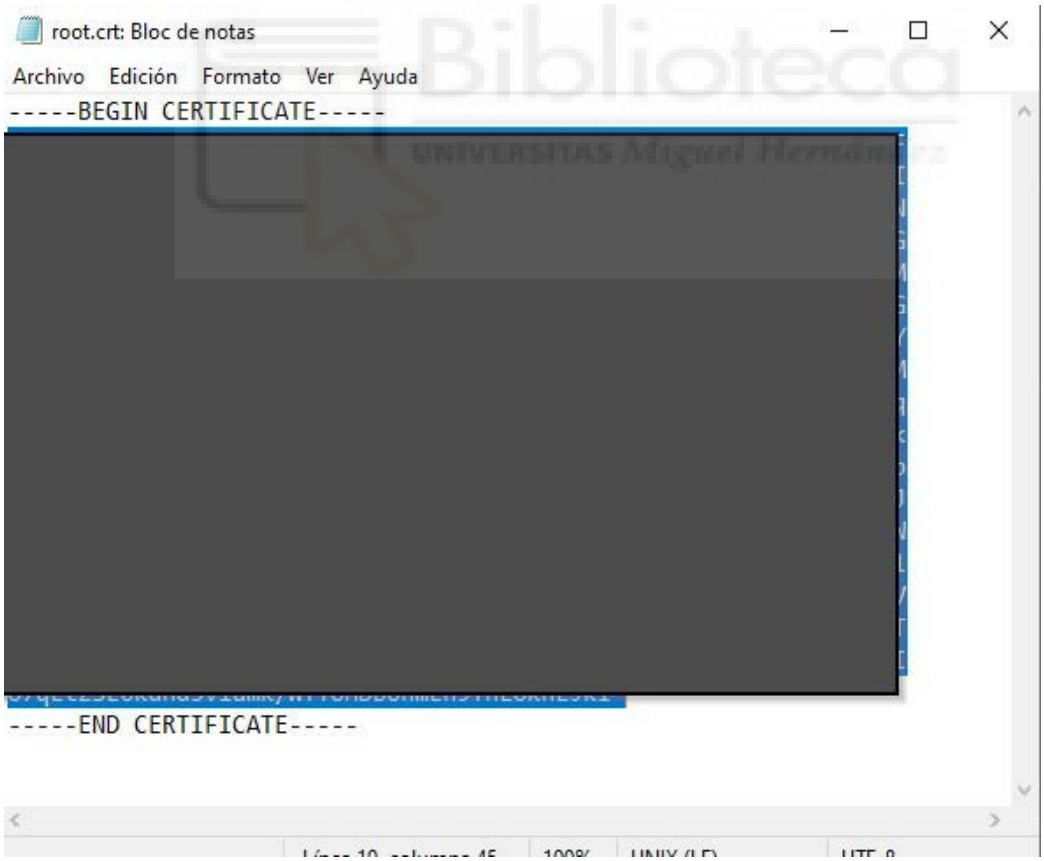

Figura 22 – Firma certificado raíz

El próximo paso será configurar el certificado en la parte de cliente [10,11]. En la pestaña de Point-to-site le daremos click en “Descargar cliente VPN”, nos descargará una carpeta comprimida donde podremos encontrar dos carpetas, una para configuración en Windows y otra denominada “Generic” para demás sistemas operativos. Dentro de Generic existirá un documento .xml que abriremos con nuestro bloc de notas:

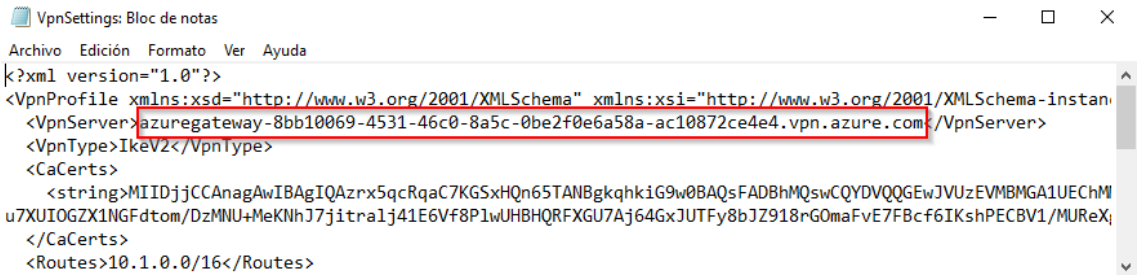

Figura 23 – Archivo VpnSettings

La URI comprendida entre los delimitadores “VpnServer” es la que debemos de copiar.

En Ajustes -> Red seleccionamos servicio VPN:

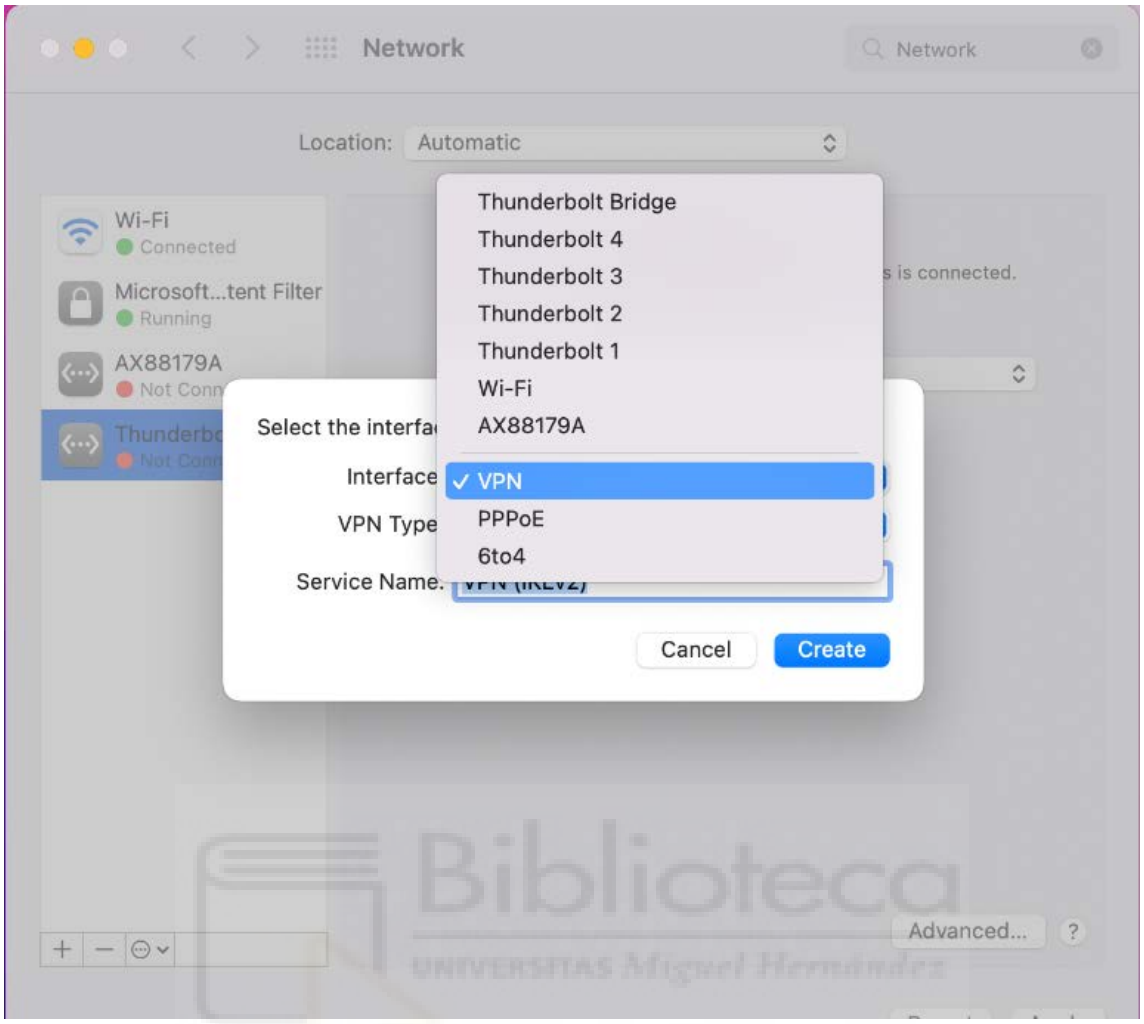

Figura 24 – Configuración VPN cliente 1

Para Tipo de VPN, haremos click en IKEv2. En Nombre de servicio, especificamos un nombre para el perfil:

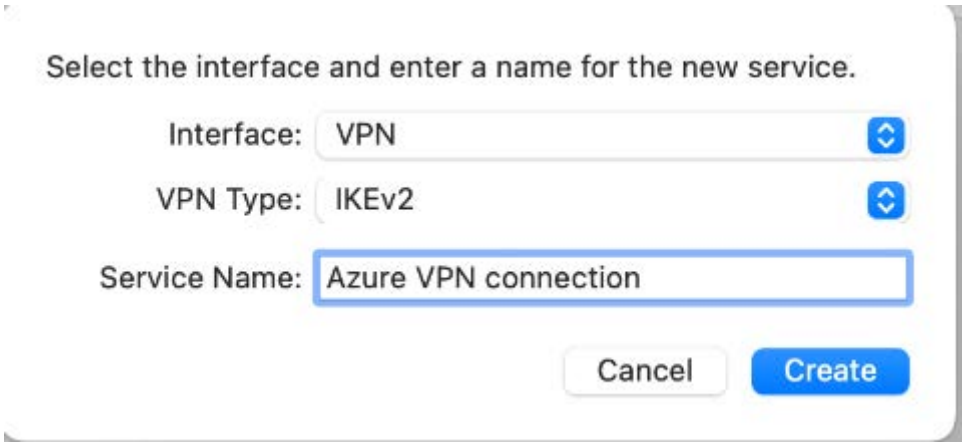

Figura 25 – Configuración VPN cliente 2

Pegamos el valor de la etiqueta VpnServer en los campos Dirección de servidor e Id. remoto del perfil:

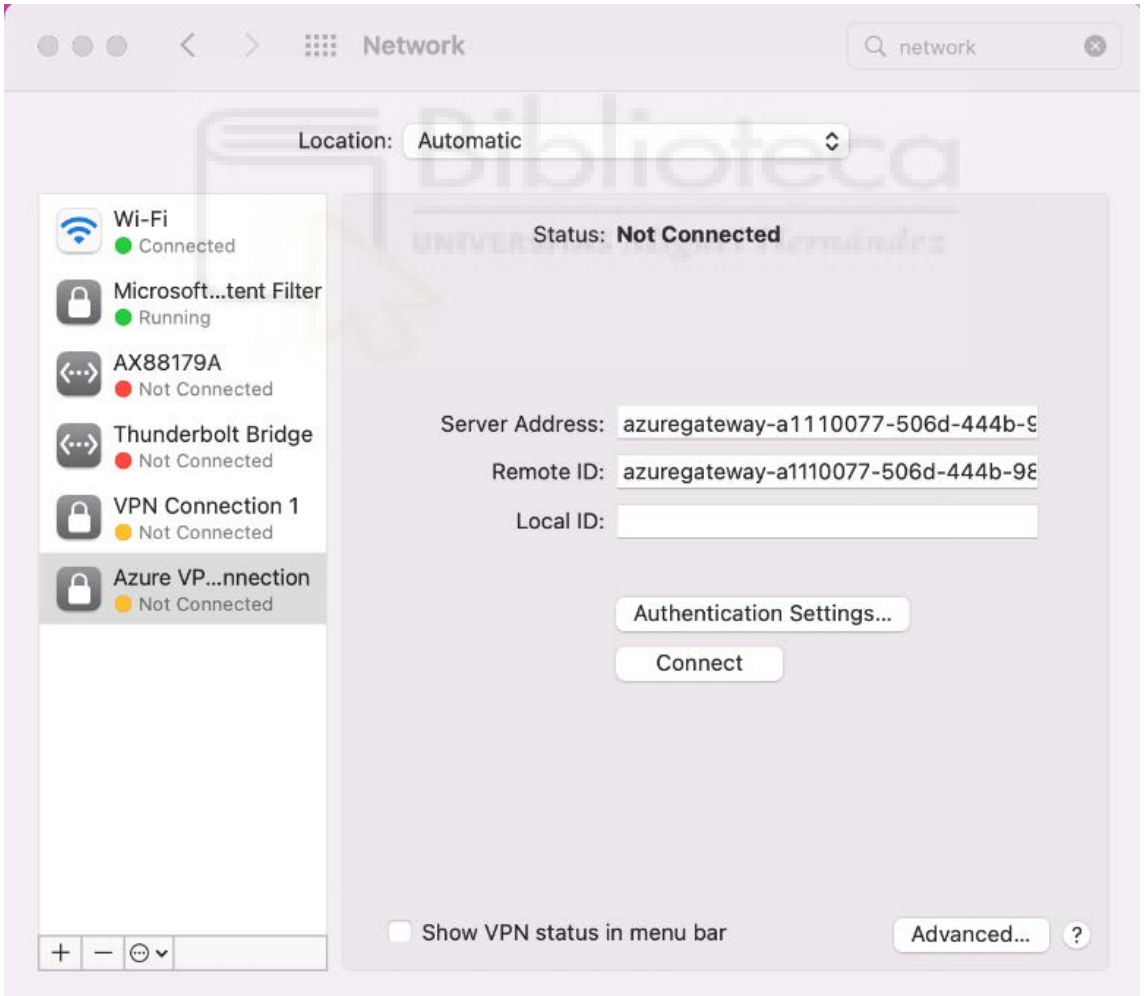

Figura 26 – Configuración VPN cliente 3

En “Authentication Settings” cargamos el .crt del certificado cliente que hemos creado anteriormente con OpenSSL:

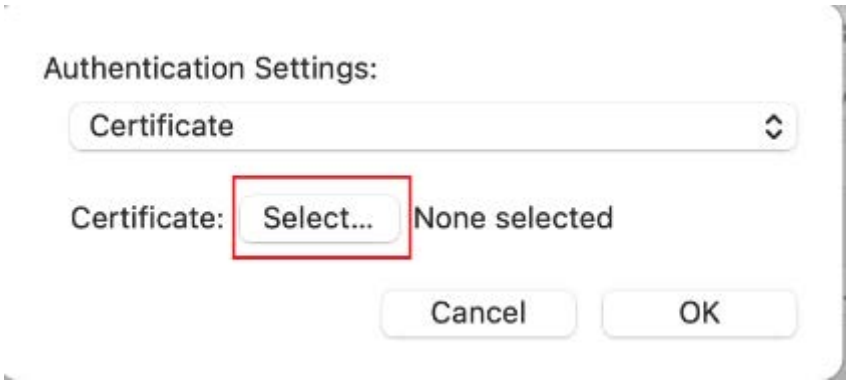

Figura 27 – Método de autenticación VPN

Este certificado cliente es el que contendrá la misma firma que el certificado raíz y nos permitirá autenticarnos contra Azure.

2.4. REGLAS NSG VM

El grupo de seguridad de red (NSG) de Azure se utiliza para filtrar el tráfico de red entre los recursos de Azure de una red virtual. Un grupo de seguridad de red contiene reglas de seguridad que permiten o deniegan el tráfico de red entrante o el tráfico de red saliente de varios tipos de recursos de Azure. Para cada regla, podemos especificar un origen y destino, un puerto y un protocolo.

Un grupo de seguridad de red contiene tantas reglas como desee, siempre que esté dentro de los límites de la suscripción de Azure. Cada regla especifica las siguientes propiedades:

- Nombre: Un nombre único dentro del grupo de seguridad de red. El nombre puede tener hasta 80 caracteres.

- Prioridad: Un número entre 100 y 4096. Las reglas se procesan en orden de prioridad. Se procesan primero las reglas con los números más bajos ya que estos tienen más prioridad. Si el tráfico coincide con una regla, se detiene el procesamiento.
- Origen o destino: Cualquiera, una dirección IP individual, un bloque CIDR de enrutamiento entre dominios sin clases (10.0.0.0/24, por ejemplo), una etiqueta de servicio o un grupo de seguridad de aplicaciones.
- Protocolo: TCP, UDP, ICMP, RDP, SSH o cualquiera.
- Dirección: Si la regla se aplica al tráfico entrante o al saliente.
- Intervalo de puertos: Podemos especificar un puerto individual o un intervalo de puertos. Por ejemplo, especificar 80 o 10000-10005.
- Acción: Permitir o denegar.

Desde nuestra VM, en la pestaña Networking podremos editar la tabla NSG. Vamos a permitir el acceso desde la VPN (rango 10.2.0.0/24) a la ip privada de la máquina (10.1.1.4), los servicios de SSH (puerto 22) para acceder a la máquina por comandos, los servicios de HTTP (puerto 80) y HTTPS (puerto 443), el servicio de File Transfer Protocol FTP (puertos 21 y 22) que nos permitirá enviar a la VM el certificado para securizar Squid.

Todos estos servicios serán entonces internos, lo cual garantiza la completa seguridad de los mismos.

Respecto al servicio de Squid, cuyo puerto de escucha por defecto es 3128, aplicaremos orígenes posibles solamente la vpn, pero dejaremos any en destino, ya que queremos acceder al proxy por la IP pública .

La regla AllowVnet permite conectarse a los elementos de la misma red virtual, esto nos permitirá que la cuenta de Automation se pueda comunicar con la máquina para lanzar comandos.

Prioridad	Nombre	Puerto	Protocolo	Origen	Destino	Acción
300	SSH	22	TCP	10.2.0.0/24	10.1.1.4	Permitir

320	HTTPS	443	TCP	10.2.0.0/24	10.1.1.4	Permitir
340	HTTP	80	TCP	10.2.0.0/24	10.1.1.4	Permitir
350	FTP	20-21	Any	10.2.0.0/24	10.1.1.4	Permitir
360	Squid	3128	TCP	10.2.0.0/24	Any	Permitir
65000	AllowVnet	Any	Any	Vnet	Vnet	Permitir
65001	AllowLB	Any	Any	LB	Any	Permitir
65500	DenyAll	Any	Any	Any	Any	Denegar

Tabla 10 – NSG Máquina Virtual

Vemos que la última regla bloquea todo el resto de tráfico entrante que no cumpla las reglas superiores, lo cual nos otorga un sistema securizado.

Inbound port rules | Outbound port rules | Application security groups | Load balancing

Network security group vm-bck-nsg (attached to subnet: default)
Impacts 1 subnets, 1 network interfaces

Add inbound port rule

Priority	Name	Port	Protocol	Source	Destination	Action
300	SSH	22	TCP	10.2.0.0/24	10.1.1.4	Allow
320	HTTPS	443	TCP	10.2.0.0/24	10.1.1.4	Allow
340	HTTP	80	TCP	10.2.0.0/24	10.1.1.4	Allow
350	allow-ftp	21-22	Any	10.2.0.0/24	10.1.1.4	Allow
360	AllowSquid	3128	TCP	10.2.0.0	Any	Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInBound	Any	Any	Any	Any	Deny

2.5. INSTALACIÓN Y CONFIGURACIÓN PROXY SQUID

2.5.1. INSTALACIÓN PROXY SQUID

Compuesto ya nuestra red en Azure, podremos configurar el proxy de Squid en la máquina virtual. Nos conectaremos desde la VPN por SSH mediante una terminal:

ssh usuario@ip_privada_vm

Como se ha comentado en anteriores apartados, se ha escogido la versión 6.2 del proxy Squid. Para poder intalarlo en nuestra máquina debemos de seguir los siguientes pasos:

- Actualizamos el Repositorio de Paquetes de Ubuntu:

```
sudo apt-get update
```

- Instalamos dependencias necesarias para compilar el software de squid:

```
sudo apt install build-essential libssl-dev libdb-dev libdb++-dev libpam0g-dev libcap2-dev libnetfilter-conntrack-dev libexpat1-dev libexpat1
```

- Descargamos Squid desde el repositorio web:

```
wget http://www.squid-cache.org/Versions/v6/squid-6.2.tar.gz
```

- Descomprimimos el archivo:

```
tar -xzf squid-6.2.tar.gz
```

- Ingresamos al directorio de Squid:

```
cd squid-6.2
```

- Configuramos Squid mediante un script de configuración que analiza el sistema y ajusta la compilación según las características del entorno.

```
./configure --with-openssl --enable-ssl --enable-ssl-crtd
```

- Compilamos Squid, leyendo un archivo llamado Makefile y ejecutando las acciones especificadas.

```
make
```

- Instalamos Squid en el sistema:

```
sudo make install
```

- Iniciamos el servicio de Squid:

```
sudo systemctl start squid
```

- Habilitamos el Arranque Automático de Squid:

```
sudo systemctl enable squid
```

- Verificamos el Estado del Servicio (debe aparecer Active) :

```
sudo systemctl status squid
```

El resultado debe ser el siguiente:

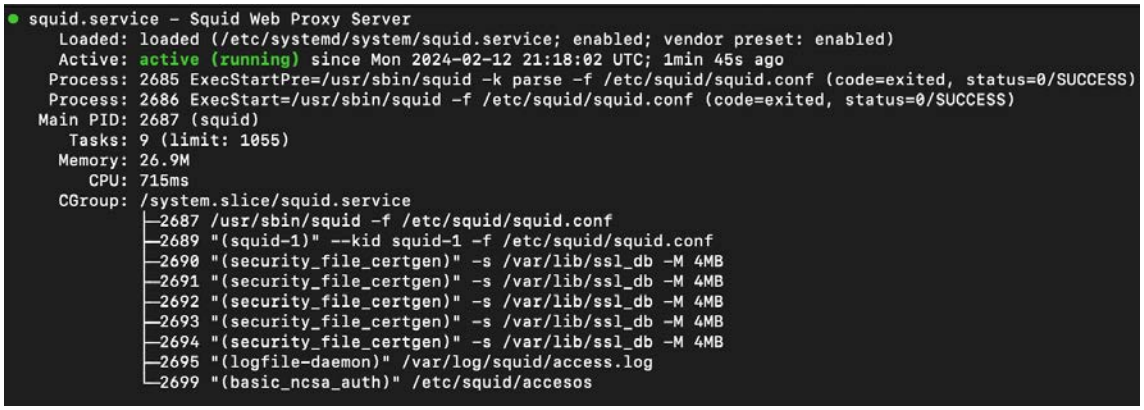

Figura 28 – Estado Squid

2.5.2. CONFIGURACIÓN PROXY SQUID

Para la configuración de Squid debemos entrar al directorio /etc/squid/squid.conf. Abrimos el archivo con un editor de texto como nano o vi.

Debemos buscar la línea donde se encuentren los “http_access”, la línea por defecto será http_access deny all, ésta denegará todas las solicitudes http que pasen por el proxy. Debemos cambiarla por http_acess allow all. Arriba de esta línea añadiremos las listas de control de acceso (acl) [12]. Vamos a crear dos listas personalizables para el bloqueo de lista dominios y el bloqueo de lista de palabras clave:

```
acl blockedurls dstdomain “etc/squid/blockedurls.acl”

https_access deny blockedurls

acl blockedwords url_regex “/etc/squid/blockedwords.acl”

https_access deny blocked words

https_acess allow all
```

“blockedurls” es una ACL basada en destinos (dstdomain), lo que significa que se aplicará a URLs o dominios específicos, mientras que blockedwords utiliza una expresión regular (url_regex) para hacer coincidir URLs que contienen ciertas palabras clave.

Es importante colocar estás reglas arriba de la especificada allow all, ya que las reglas se aplican de arriba hacia abajo.

Guardamos el archivo de configuración de squid y creamos las acls en el directorio /etc/squid/ que hemos indicado. Simplemente abrimos un editor de texto:

```
nano blockedurls.acl

nano blockedwords.acl
```

Indicamos las urls en el primer caso y las palabras que queremos bloquear, squid analiza el contenido de la acl mediante un separador de salto de línea por lo que es importante realizarlo de esta manera.

Por último, debemos reiniciar el servicio de squid para que los cambios se apliquen:

```
sudo systemctl restart squid
```

Vamos a utilizar un repositorio de listas de bloqueo de dominios público denominado **Blackweb** [13] es un proyecto que recopila y unifica listas públicas de bloqueo de dominios (porno, descargas, drogas, malware, spyware, trackers, bots, redes sociales, warez, armas, etc.) para hacerlas compatibles con el proxy Squid. Contiene un total de 4857278 que se van actualizando periódicamente.
Para descargar el repositorio introducimos el siguiente comando:

```
wget -q -N
https://raw.githubusercontent.com/maravento/blackweb/master
/blackweb.tar.gz && cat blackweb.tar.gz* | tar xzf -
```

Descargamos los valores de checksum calculados con el algoritmo de hash MD5.

```
wget -q -N
https://raw.githubusercontent.com/maravento/blackweb/master
/checksum.md5
```

Comando para comparar la suma de verificación MD5 calculada localmente para **blackweb.txt** con la suma de verificación MD5 esperada almacenada en **checksum.md5**. Si ambos valores coinciden, indica que el archivo se ha descargado correctamente y no ha sido alterado.

```
md5sum blackweb.txt | awk '{print $1}' && cat checksum.md5
| awk '{print $1}'
```

Vemos que nos devuelve dos líneas con el mismo checksum, lo cual quiere decir que el archivo no ha sido corrompido:

4f4c090834e270e274f30a75d586a1e2

4f4c090834e270e274f30a75d586a1e2

Debemos añadir entonces en /etc/squid/squid.conf las siguientes líneas para aplicar el bloqueo a los dominios del repositorio Blackweb:

```
acl blackweb dstdomain "/etc/squid/blackweb/blackweb.txt"

http_access deny blackweb
```

Volvemos a reiniciar el servicio de squid para guardar los cambios.

2.5.3. CREACIÓN USUARIOS

Para activar una autenticación en el servidor Squid, en primer lugar, se ha de instalar el módulo apache2-utils para que la utilidad htpasswd, que genera un usuario y una contraseña para la autenticación, pueda funcionar.

```
apt-get install apache2-utils
```

Creamos a continuación, los usuarios que tendrán permiso para navegar, nos solicitarán contraseña para cada uno de ellos:

```
htpasswd -c /etc/squid/accesos usuario1

htpasswd /etc/squid/accesos usuario2
```

En el archivo de configuración de squid añadiremos las siguientes líneas

`auth_param basic program /usr/lib/squid/basic_ncsa_auth /etc/squid/accesos`	Especifica el programa de autenticación que Squid utilizará para autenticar a los usuarios. En este caso, se está utilizando el programa basic_ncsa_auth, que es una implementación básica de autenticación mediante el archivo /etc/squid/accesos. Este archivo contiene pares de nombre de usuario y contraseña.
`auth_param basic children 5`	Define el número máximo de procesos hijos que Squid puede iniciar para gestionar las solicitudes de autenticación básica
`auth_param basic realm Squid proxy-caching web server`	Especifica el dominio de autenticación que se presenta a los usuarios cuando se les solicita la autenticación.
`auth_param basic credentialsttl 2 hours`	Define el TTL (Time To Live) de las credenciales de autenticación en caché. En este caso, las credenciales se almacenarán en caché durante 2 horas.
`auth_param basic casesensitive off`	Indica si la autenticación básica debe ser sensible a mayúsculas y minúsculas.

Tabla 11 – Configuración usuarios Squid

Por último, crearemos la acl de tipo “proxy_auth”, para permitir la navegación de los usuarios autenticados:

```
acl authenticated_users proxy_auth usuario1 usuario2

http_access allow authenticated_users
```

Desde el equipo de usuario deberemos seleccionar “El servidor proxy require contraseña” para autenticarnos:

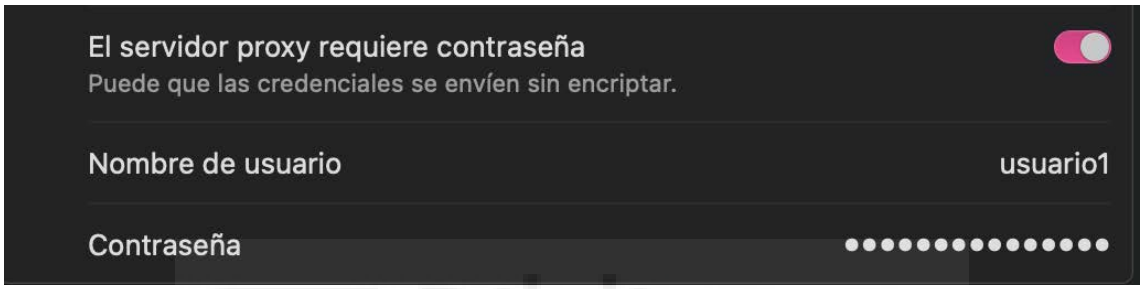

Figura 29 – Usuario Squid

2.5.4. SSL BUMPING

Squid, funciona principalmente para el tráfico HTTP. Sin embargo, cuando se trata de tráfico HTTPS, Squid, en su configuración básica, actúa principalmente como un túnel de paso. El tráfico HTTPS está cifrado de extremo a extremo, y Squid simplemente permite que este tráfico pase sin inspeccionar ni modificar el contenido cifrado.

El concepto de "SSL Bumping" [14] en Squid introduce la capacidad de realizar inspección SSL/TLS en el tráfico cifrado. Esto significa que Squid, al estar configurado con SSL Bumping, puede abrir y examinar el contenido cifrado de las conexiones HTTPS. Esto es útil para aplicar políticas de seguridad, control de acceso y filtrado de contenido en conexiones HTTPS.

El proceso de SSL Bumping en Squid generalmente sigue estos pasos:

1. **Terminación SSL/TLS en Squid**: Squid actúa como un servidor SSL/TLS para el cliente y como cliente SSL/TLS para el servidor web. Esto implica que Squid debe presentar un certificado SSL al cliente, que el cliente confiará como si fuera el certificado del servidor original.

2. **Inspección del tráfico cifrado**: Squid descifra el tráfico SSL/TLS, inspecciona el contenido y luego vuelve a cifrar el tráfico antes de enviarlo al servidor web de destino. Este proceso permite a Squid analizar el contenido de las conexiones HTTPS y aplicar políticas de filtrado y control de acceso.

3. **Manejo de certificados**: Squid puede generar certificados SSL en tiempo real para los sitios web de destino, y estos certificados son firmados por una Autoridad de Certificación (CA) raíz de Squid. Estos certificados son presentados al cliente en lugar de los certificados originales del servidor web.

Para realizar este paso, es importante que, antes de compilar Squid, agreguemos las configuraciones de soporte SSL/TLS, sin estas no sería posible habilitar ssl-bumping:

```
./configure --with-openssl --enable-ssl --enable-ssl-crtd
```

Necesitaremos generar un certificado autofirmado vía OpenSSL (explicado en punto 2.3.1) que se utilizará para negociar con el cliente durante la interceptación SSL. Exportamos el mismo a formato .pem.

Debemos añadirlo al llavero del sistema de los equipos de cliente ya que, para que el cliente confíe en estos certificados de servidor generados por Squid, debe confiar en la entidad de certificación raíz (CA) que emitió el certificado de usado por Squid.

En la máquina virtual creamos el directorio **/etc/squid/ssl_cert** donde almacenaremos el certificado.

Como tenemos acceso por ftp a la máquina virtual desde la VPN, enviaremos el certificado generado desde el equipo de cliente mediante el siguiente comando de scp (secure copy) para sistemas MacOS:

```
scp /Escritorio/myCA.pem
usuario@direccion_ip:/etc/squid/ssl_cert/
```

Ahora podremos modificar el archivo de configuración /etc/squid/squid.conf:

`http_port 3128 ssl-bump \`	Puerto en el que Squid escuchará las conexiones HTTP y habilita el manejo de SSL
`cert=/etc/squid/ssl_cert/myCA.pem \`	Especifica la ubicación del certificado SSL que Squid utilizará para establecer una conexión segura.
`  generate-host-certificates=on dynamic_cert_mem_cache_size=4MB`	Genera certificados de host dinámicamente y especificar el tamaño de la memoria caché para estos certificados.
`sslcrtd_program /usr/lib/squid/security_file_certgen -s /var/lib/ssl_db -M 4MB`	Configura el programa que generará y almacenará los certificados de host.
`acl step1 at_step SslBump1`	Crea una acl que se activará en el paso **SslBump1** del proceso de bumping de SSL.

`ssl_bump peek step1`	Configura Squid para examinar el tráfico SSL en el paso **SslBump1**.
`ssl_bump bump all`	Configura Squid para interceptar y modificar todo el tráfico SSL que pasa a través de él.

Tabla 12 – SSL Bumping Squid

Finalmente reiniciamos Squid para aplicar los cambios. La configuración final de Squid debe quedar de la siguiente manera:

```
#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#

http_port 3128 ssl-bump \
  cert=/etc/squid/ssl_cert/myCA.pem \
  generate-host-certificates=on dynamic_cert_mem_cache_size=4MB

sslcrtd_program /usr/lib/squid/security_file_certgen -s /var/lib/ssl_db -M 4MB

acl step1 at_step SslBump1

ssl_bump peek step1
ssl_bump bump all

# For example, to allow access from your local networks, you may uncomment the
# following rule (and/or add rules that match your definition of "local"):
http_access allow localnet

auth_param basic program /usr/lib/squid/basic_ncsa_auth /etc/squid/accesos
auth_param basic children 5
auth_param basic realm Squid proxy-caching web server
auth_param basic credentialsttl 2 hours
auth_param basic casesensitive off

#acl usuario1 proxy_auth usuario1
acl authenticated_users proxy_auth usuario1 usuario2

#reglas específicas

acl blackweb dstdomain "/etc/squid/blackweb/blackweb.txt"
http_access deny blackweb

acl blockedurls dstdomain "/etc/squid/blockedurls.acl"
http_access deny blockedurls

acl blockedwords url_regex "/etc/squid/blockedwords.acl"
http_access deny blockedwords

# And finally allow all other access to this proxy
http_access allow authenticated_users
http_access allow all

# Squid normally listens to port 3128
http_port 3128
```

Figura 30 – Configuración final Squid

2.5.5. GENERADOR DE INFORMES DE ANÁLISIS DE SQUID

Sarg (Squid Analysis Report Generator por sus siglas en inglés) [19] es un generador de reportes de análisis de tráfico que funciona con el Proxy Squid en Linux. Algunas de sus funcionalidades son las siguientes:

- Generador de informes
- Estadísticas de los usuarios
- Sitios más visitados

- Autenticaciones fallidas de los usuarios
- Gráficas de los usuarios

Necesitaremos instalar en nuestra máquina virtual un servidor web como Apache2, de código abierto:

```
sudo apt install apache2
```

Apache2 se integra comúnmente con Sarg para proporcionar una interfaz web amigable para visualizar y analizar los informes generados por Sarg basados en los registros de Squid.

Instalaremos entonces Sarg en nuestra vm:

```
sudo apt install sarg
```

Para configurar Sarg debemos abrir con un editor de texto el archivo en el directorio /etc/sarg/sarg.conf.

`access_log /var/log/squid/Access.log`	Especifica la ubicación del archivo de registro de acceso de Squid.
`output_dir /var/www/html/squid-reports`	Indica el directorio donde SARG almacenará los informes generados.
`date_format e`	Establece el formato de fecha europeo para los informes.
`resolve-ip yes`	Indica si se deben resolver las direcciones IP en nombres de host.

Tabla 13 – Configuración SARG

Cuando hayamos generado tráfico conectados al proxy de Squid, generamos un informe de Sarg con el comando:

```
sarg -x
```

Como en la directiva output_dir teníamos configurado el subdirectorio /squid-reports, por defecto accederemos a los informes de Sarg introduciendo en el navegador la IP_privada/squid-reports.

Figura 31 – Página inicio SARG

2.6. AZURE AUTOMATION

2.6.1. CUENTA AUTOMATION

El primer paso para automatizar procesos en nuestro entorno de Azure, es la creación del recurso “Cuenta Azure Automation” [16], que es desde donde administraremos todos los elementos de automatización que ofrece Azure.

Managed Identities (Identidades Administradas) se refiere a una característica de Azure Active Directory (Azure AD) que proporciona identidades de servicio seguras para los recursos de Azure.

Escogemos la opción System Assigned Managed Identity (Identidad Administrada Asignada por el Sistema),esta identidad administrada se asocia directamente con el recurso de Azure Automation en el que está habilitada.

Al utilizar Managed Identities, no es necesario almacenar ni gestionar credenciales en el código o en la configuración. Esto mejora la seguridad y reduce el riesgo de exposición de credenciales.

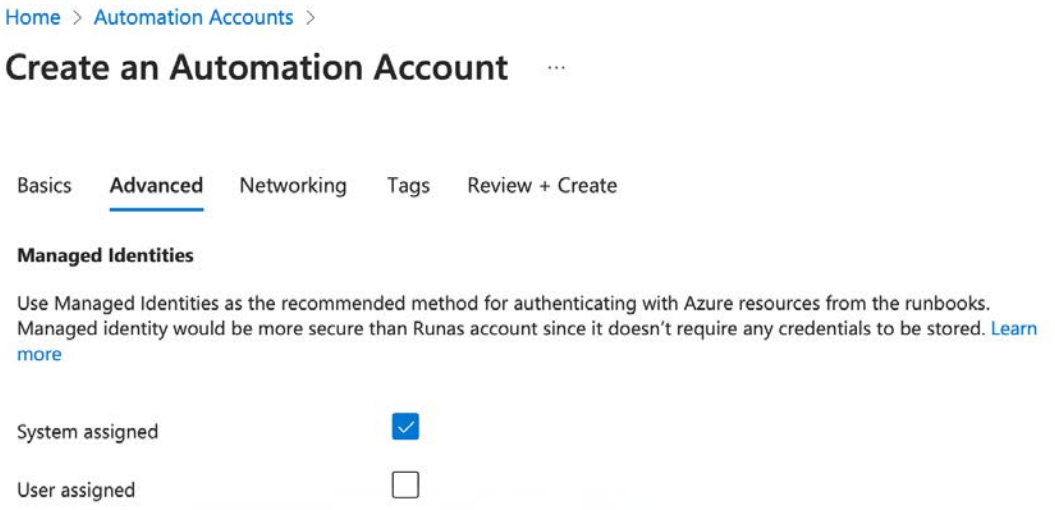

Figura 32 – Creación cuenta Automation

Para la conectividad de red, vamos a utilizar exclusivamente Acceso privado, securizando los procesos de automatización. Para ello debemos crear una utilidad denominada "Private Endpoint". Un punto de conexión privado es una interfaz de red que utiliza una dirección IP privada de la red virtual.

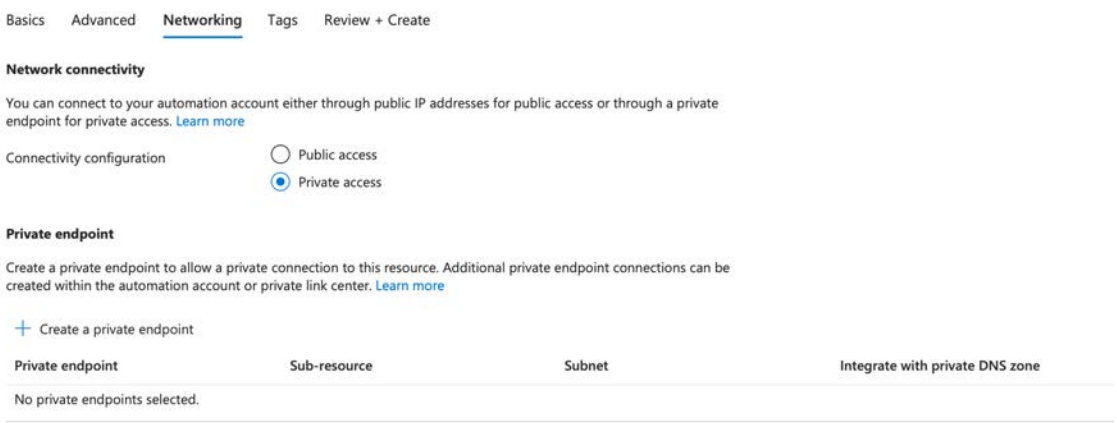

Figura 33 – Configuración de red Automation

Se nos abrirá entonces un desplegable para la creación del mismo.

Seleccionar "Webhook" como subrecurso de destino en la creación de un Private Endpoint indica que estamos estableciendo una conexión segura desde la red virtual a un servicio externo que utiliza webhooks para recibir notificaciones automáticas.

Seleccionamos nuestra red virtual, donde se sitúa la VM, para que se le asigne una ip privada de la subred. Además necesitamos activar la integración de DNS ya que la llamada al Webhook se realizará vía URI.

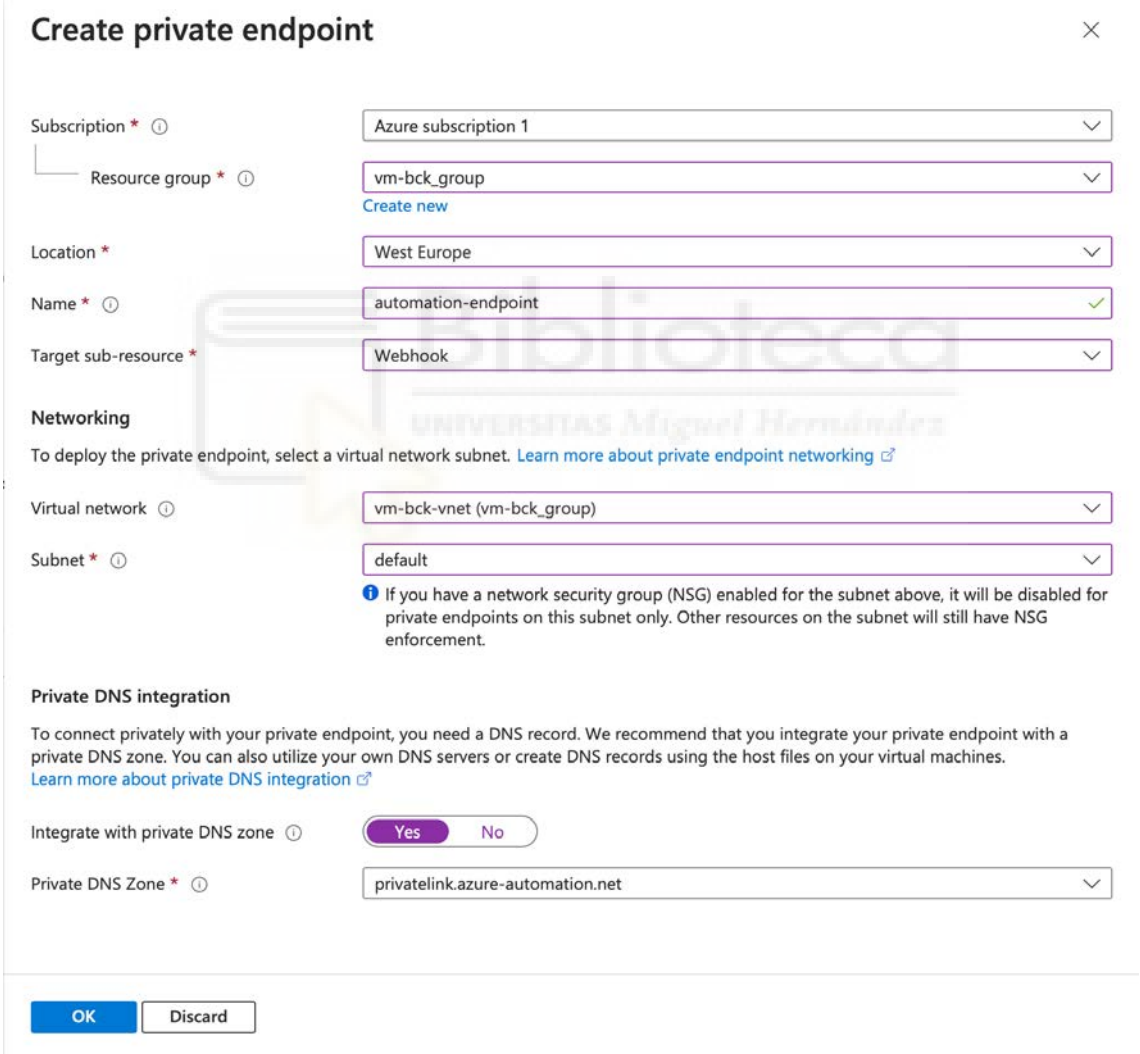

Figura 34 – Creación private endpoint

Finalizado el despliegue, podemos ver que se ha creado el punto de conexión privado con la ip 10.1.1.5, de rango de direccionamiento de la red virtual.

Figura 35 – Private endpoint final

2.6.2. RUNBOOK

Dentro de nuestra cuenta Automation podremos crear la automatización de procesos conocida como “Runbooks”.

Los runbooks de PowerShell están basados en Windows PowerShell [17]. Podremos modificar directamente el código del runbook con el editor de texto en el Portal de Azure. La versión de PowerShell viene determinada por la versión del entorno de ejecución especificada.

Algunas ventajas de los Runbook de PowerShell son:

- Podemos implementar toda la lógica compleja con código de PowerShell sin otras complejidades del flujo de trabajo de PowerShell.
- Se inician con más rapidez que los runbooks de flujo de trabajo de PowerShell, ya que no necesitan compilarse antes de la ejecución.
- Podemos ejecutar en Azure para sistemas operativos Windows y Linux.

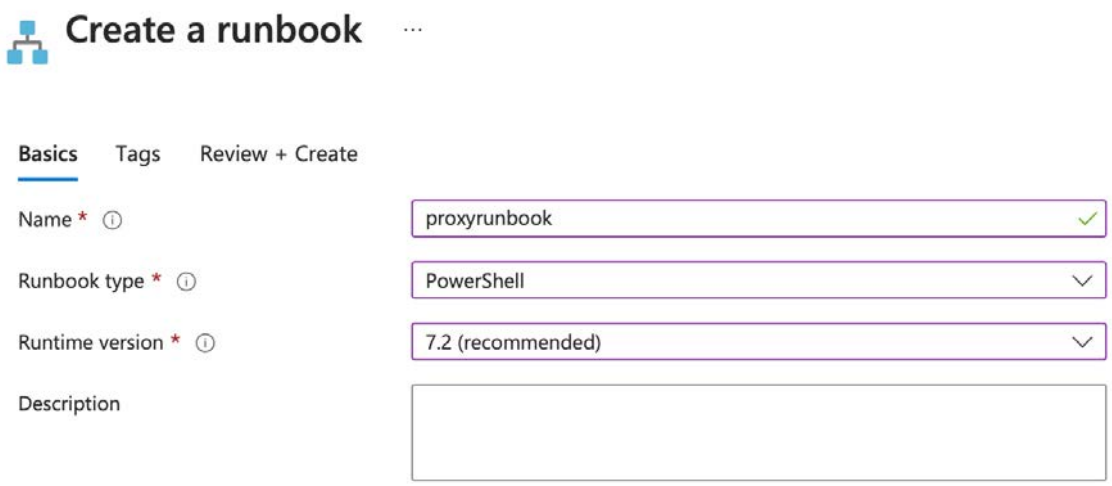

Figura 36 – Creación Runbook

Previamente, debemos añadir en nuestra cuenta de Automation Posh-SSH [18], que es un módulo de PowerShell que proporciona cmdlets para interactuar con servicios SSH directamente desde un script o sesión de PowerShell.

Podremos crear entonces, un script de Powershell desde Edit -> Edit in portal:

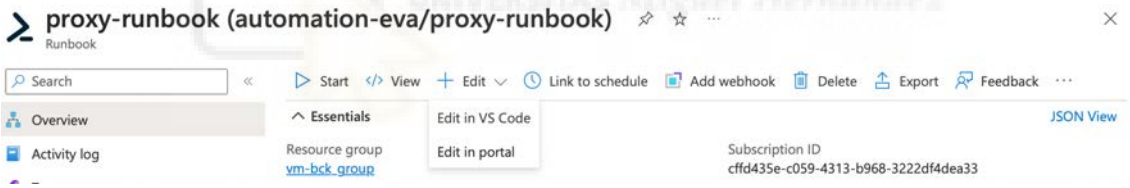

Figura 37 – Editar Runbook

El script automatizado que se ha creado para este proyecto consiste en recoger, mediante un Webhook dos parámetros: una nueva URL a bloquear y una nueva palabra clave a bloquear. Si el parámetro no es nulo, se añade, mediante el comando “echo”, el string contenido en el parámetro a la ACL correspondiente accediendo a través de SSH a la máquina virtual.

```powershell
param (
    [Parameter(Mandatory=$false)]
    [object] $WebhookData
)

Write-Output "Start"
Write-Output ("Object type: {0}" -f $WebhookData.GetType())
Write-Output $WebhookData
Write-Output "`n`n"
Write-Output $WebhookData.WebhookName
Write-Output $WebhookData.RequestBody
Write-Output $WebhookData.RequestHeader
Write-Output "End"

if ($WebhookData.RequestBody -eq $null) {
    Write-Output "Error: El cuerpo de la solicitud es nulo."
    exit
}

$webhookData = ConvertFrom-Json -InputObject $WebhookData.RequestBody

$newBlockedUrl = $webhookData.newBlockedUrl
$newBlockedWord = $webhookData.newBlockedWord

if ($newBlockedUrl -ne $null) {
    Write-Output "Recibido el valor de newBlockedUrl: $newBlockedUrl"
    $blockedCommand = "echo '$newBlockedUrl' | sudo tee -a /etc/squid/blockedurls.acl"
} elseif ($newBlockedWord -ne $null) {
    Write-Output "Recibido el valor de newBlockedWord: $newBlockedWord"
    $blockedCommand = "echo '$newBlockedWord' | sudo tee -a /etc/squid/blockedwords.acl"
} else {
    Write-Output "Error: No se proporcionaron parámetros válidos."
    exit
}

$vmIp = "10.1.1.4"
$vmUsername = "eva"
$vmPassword = ConvertTo-SecureString -String "" -AsPlainText -Force
$sshPort = 22

Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

if (-not (Get-Module -Name Posh-SSH -ListAvailable)) {
    Install-Module -Name Posh-SSH -Force -SkipPublisherCheck
}

$session = New-SSHSession -ComputerName $vmIp -Credential (New-Object PSCredential -ArgumentList $vmUsername, $vmPassword) -Port $sshPort -AcceptKey

Invoke-SSHCommand -Index $session.SessionId -Command $blockedCommand

Invoke-SSHCommand -Index $session.SessionId -Command "sudo systemctl restart squid"

Remove-SSHSession -Index $session.SessionId
```

Figura 38 – Script PowerShell

A continuación, se explicará en detalle:

`param (`
 	
`[Parameter(Mandatory=$false)]`
 	
`[object] $WebhookData`
 	
`)`
	Establece un parámetro **$WebhookData** que representa los datos proporcionados por en la llamada al webhook.

`Write-Output "Start"`
 	
`# ...`
 	
`Write-Output "End"`
	Simples mensajes de salida para registrar el inicio y el final del script.

`if ($WebhookData.RequestBody -eq $null) {`	Verifica si el cuerpo de la solicitud es nulo. Si es nulo, registra un error y termina el script.

`    Write-Output "Error: El cuerpo de la solicitud es nulo."`
 	
`    exit`
 	
`}`	

`$webhookData = ConvertFrom-Json -InputObject $WebhookData.RequestBody`	Convierte el cuerpo de la solicitud JSON en un objeto PowerShell para facilitar el manejo de datos.

`$newBlockedUrl = $webhookData.newBlockedUrl`
 	
`$newBlockedWord = $webhookData.newBlockedWord`
	Extrae los valores de **newBlockedUrl** y **newBlockedWord** del objeto JSON.

`if ($newBlockedUrl -ne $null) {} elseif ($newBlockedWord -ne $null) {} else {`	Verifica qué parámetro se proporcionó y construye un comando correspondiente para ejecutar en el servidor remoto.
`$vmIp, $vmUsername, $vmPassword, $sshPort`	Establece la información de la máquina virtual y la configuración SSH.
`Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass`	Configura la política de ejecución
`$session = New-SSHSession -ComputerName $vmIp -Credential (New-Object PSCredential -ArgumentList $vmUsername, $vmPassword) -Port $sshPort -AcceptKey`	Crea una sesión SSH utilizando la información de la máquina virtual.

`Invoke-SSHCommand -Index $session.SessionId -Command $blockedCommand`
 	
`Invoke-SSHCommand -Index $session.SessionId -Command "sudo systemctl restart squid"`
	Ejecuta el comando Linux construido previamente y reinicia el servicio Squid para aplicar los cambios.

`Remove-SSHSession -Index $session.SessionId`	Cierra la sesión SSH.

Tabla 14 – Script automatización PowerShell

2.6.3. WEBHOOK

Webhook es un recurso disponible en nuestra cuenta de Automation, que se define como, una URL única asociada a un Runbook que permite la invocación externa del Runbook mediante una solicitud HTTP. Cuando se realiza una solicitud HTTP a esa URL, Azure Automation ejecuta automáticamente el Runbook asociado, pasando cualquier dato necesario que se haya incluido en la solicitud.

Desde la pestaña de nuestro Runbook, podremos crear un Webhook asociado:

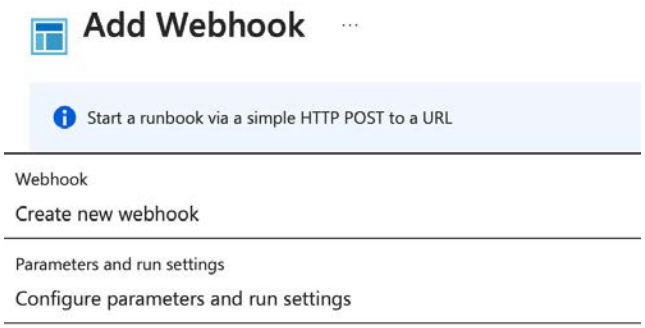

Figura 39 – Añadir Webhook

Al establecer una fecha de expiración para un webhook, se limita el período durante el cual la URL del webhook es válida y puede recibir solicitudes. Esta medida de seguridad ayuda a prevenir el uso malintencionado o no autorizado del webhook.

Nos proporcionarán una URL que, por razones de seguridad, después de crear el Webhook no se podrá volver a visualizar, por lo que deberemos guardarla.

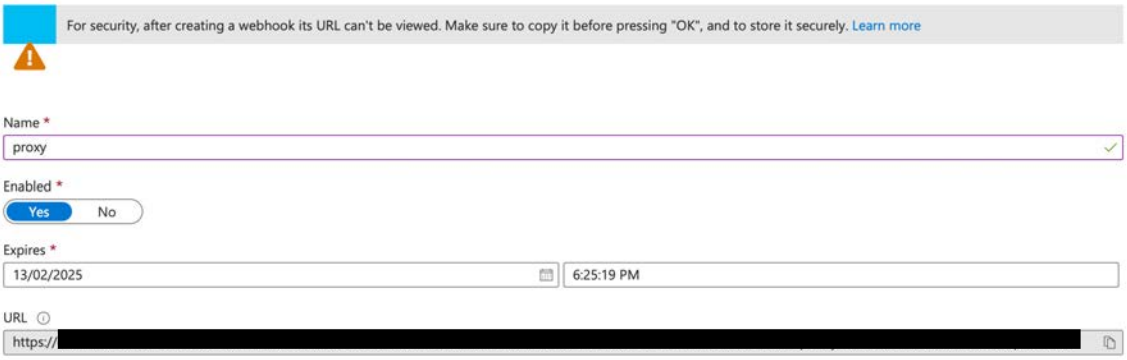

Figura 40 – Creación Webhook

No proporcionaremos parámetros desde la configuración, ya que estos parámetros serán proporcionados por el usuario que realice la llamada al Webhook.

Para iniciar un Runbook, simplemente, desde la máquina virtual (contenida en la misma red virtual que el punto de conexión privado de Automation), lanzaremos el siguiente comando curl:

```
curl -X POST -H "Content-Type: application/json" -d
'{"newBlockedUrl": "zara.es"}' https://--.webhook.we.azure-
automation.net/webhooks?token=--
```

En este curl, que es una herramienta de línea de comandos que permite realizar solicitudes HTTP, especificamos el método de solicitud HTTP a utilizar, en este caso, POST. Esto significa que estamos enviando datos al servidor. El tipo de contenido que estamos enviando en el cuerpo de la solicitud es de tipo JSON. Después, especificamos los datos que se enviarán en el cuerpo de la solicitud, el parámetro "newBlockedUrl" con el valor "zara.es", por ejemplo.

El componente que va a continuación es la URL del Webhook a la que se envía la solicitud POST.

Podremos ver que se ha añadido el string al final del archivo blockedurls.acl:

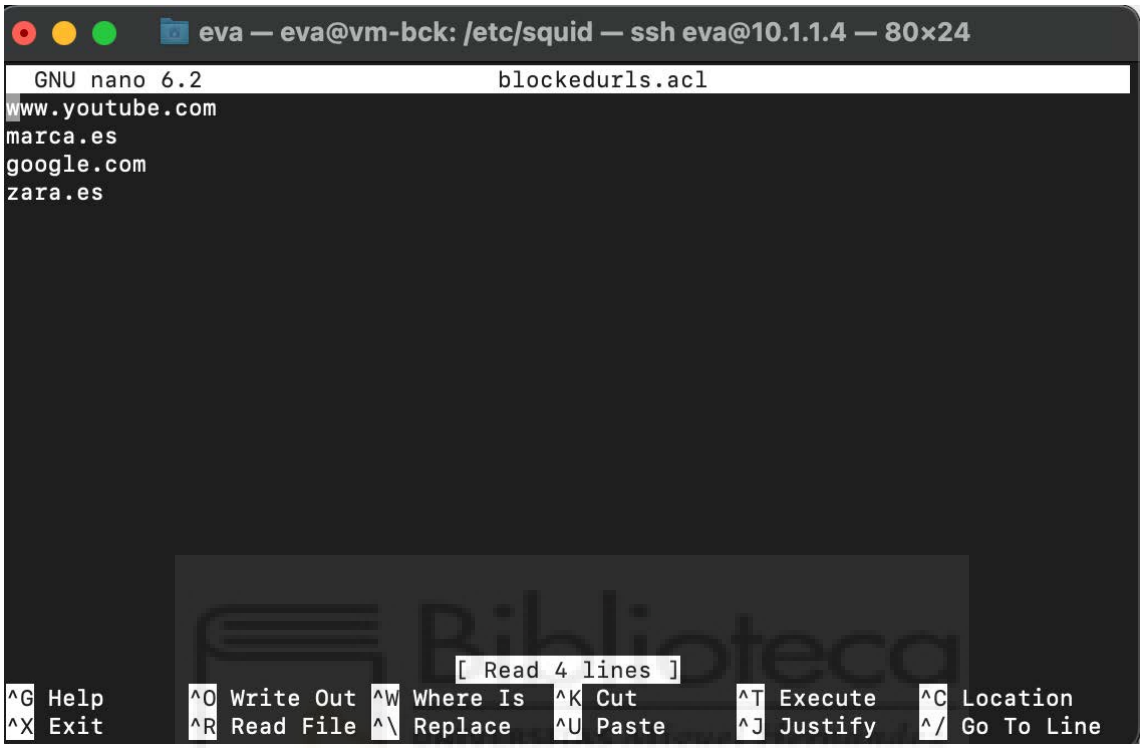

Figura 41 – Nueva entrada en acl de bloqueo

3. RESULTADOS Y DISCUSIÓN

Las pruebas de funcionamiento consisten en la activación y navegación del proxy Squid en dos equipos cliente, uno con sistema operativo MacOS y otro con sistema operativo Windows. Se conectarán autenticándose como usuario1 y usuario2 respectivamente.

Se recogerán trazas mediante la herramienta de Wireshark y se analizarán. Además, se generará un informe de Squid Sarg y se discutirán los resultados.

3.1. PRUEBAS DE FUNCIONAMIENTO

3.1.1. MACOS

Activamos la VPN creada anteriormente:

Figura 42 – VPN conectada MacOS

Debemos configurar el proxy en el equipo cliente, para ello desde los ajustes del equipo, en Red buscamos la opción “Proxies”.

Debemos configurar tanto el proxy web de HTTP como el HTTPS, para ello usamos la ip del servidor virtual que aloja el proxy y el puerto de escucha 3128 asociado a Squid. Además en los sistemas MacOS nos indicará desde los propios ajustes si queremos autenticarnos contra el servidor, en este caso vamos a autenticarnos como usuario1:

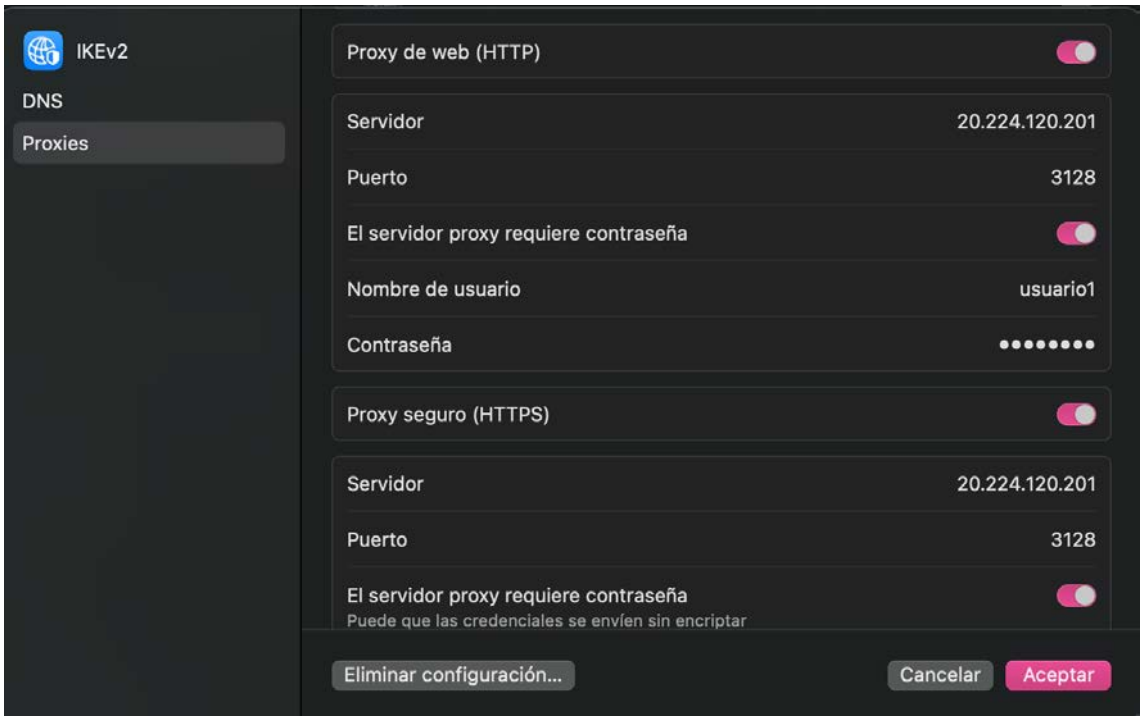

Figura 43 – Configuración proxy MacOS

La lista de control de acceso “blockedurls.acl” contiene las siguientes urls:

www.youtube.com

marca.es

google.com

zara.es

Probamos algunas de ellas para lograr ver si se produce el bloqueo. Vemos como se nos redirige a una página personalizada de Squid de bloqueo y el motivo del mismo:

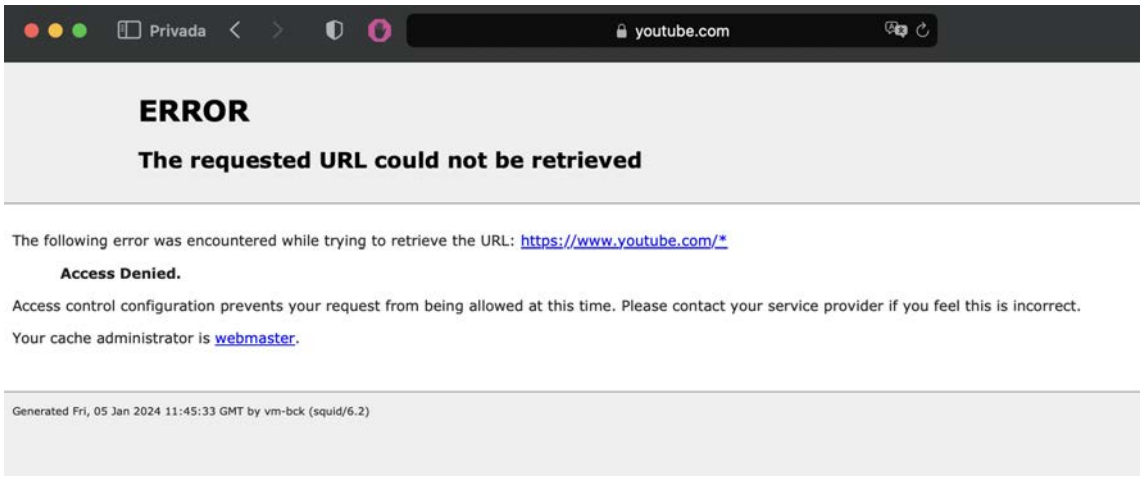

Figura 44 – Bloqueo youtube.com

Tanto para conexiones HTTP (no seguras) como HTTPS bloquea una de las URLs de la lista:

Figura 45 – Bloqueo HTTP marca.es

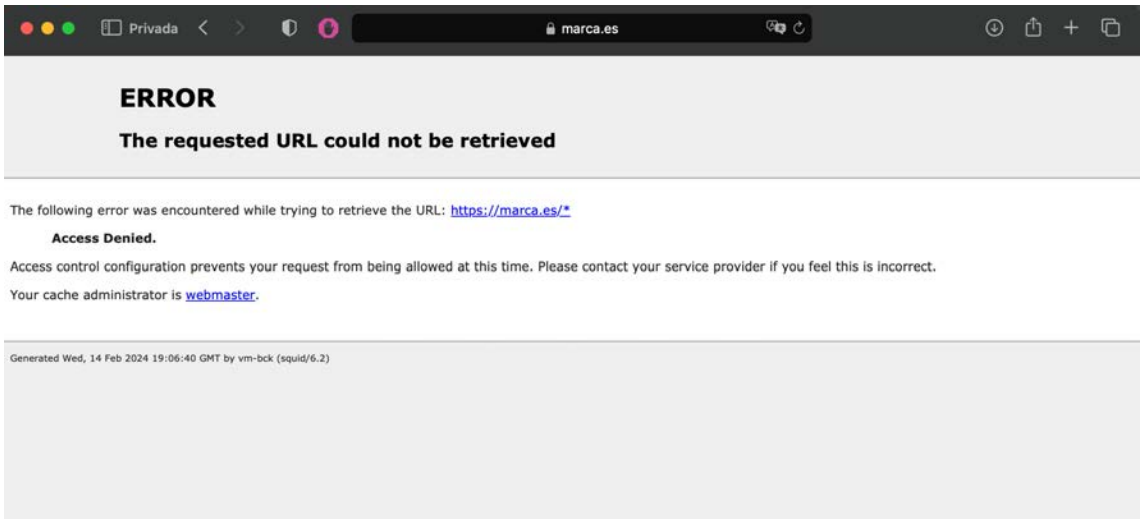

Figura 46 – Bloqueo HTTPS marca.es

Vemos que si accedemos a una web que no está dentro de la lista funciona correctamente:

Figura 47 – Acceso mundo.es

Podemos ver que, por ejemplo, la web de la UMH contiene un banner de youtube, el proxy Squid solamente bloquea esta parte de la web:

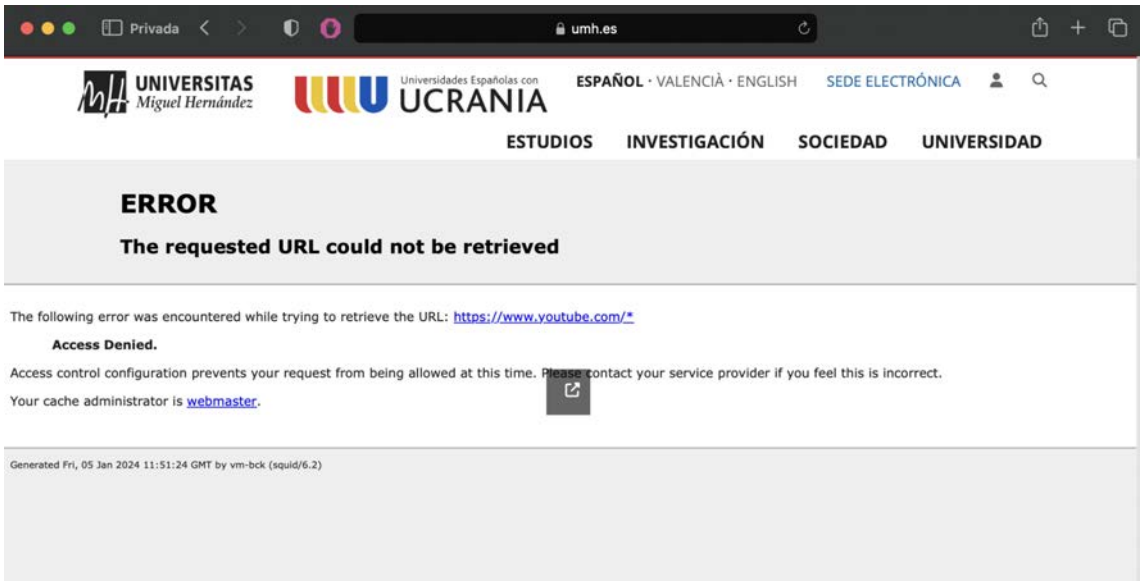

Figura 48 – Acceso umh.es

Por otro lado, la lista de control de acceso de palabras bloqueadas contiene las siguientes palabras:

droga

armas

porno

Si probamos a entrar directamente al dominio “droga”, podemos ver que lo bloquea ya que contiene una de las palabras clave:

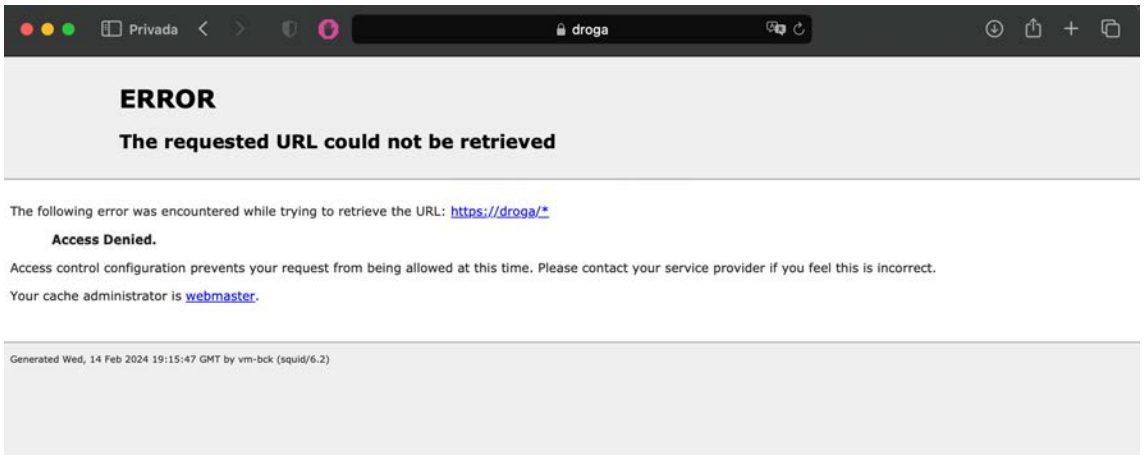

Figura 49 – Bloqueo dominio droga

Si lo buscamos en una simple búsqueda de Google también lo bloquea, ya que analiza la página completa encontrando dicha palabra clave:

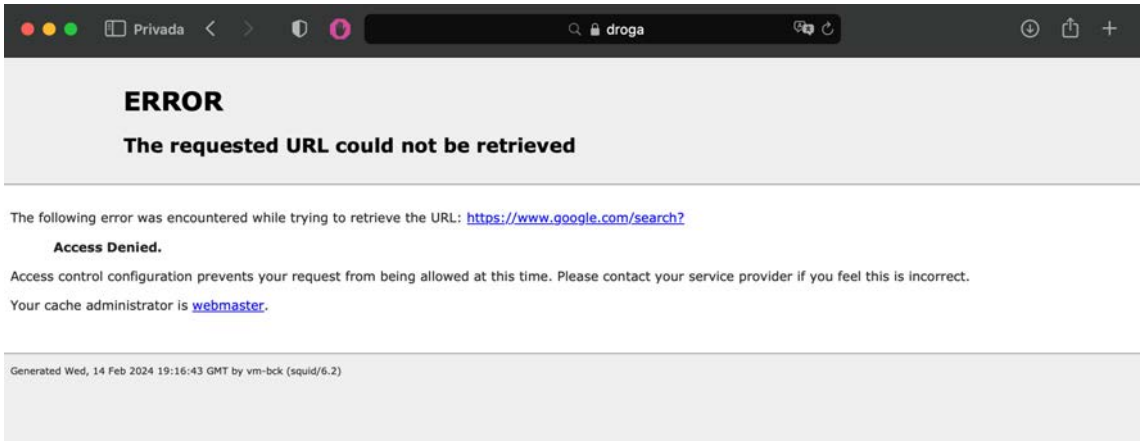

Figura 50 – Bloqueo búsqueda droga

Lanzamos también una consulta a una de las líneas incluidas en el repositorio de **Blackweb**, siendo una de las URLs bloqueada también:

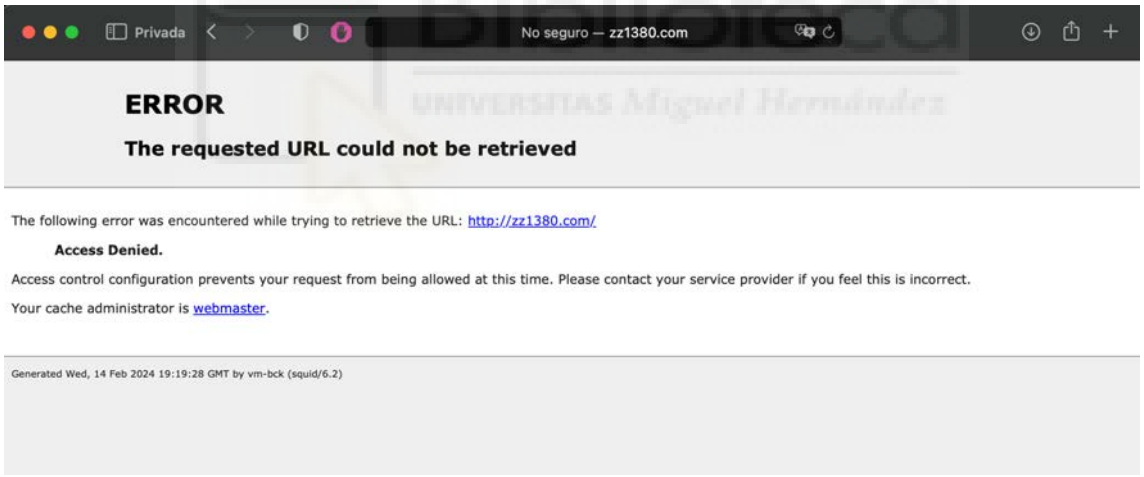

Figura 51 – Bloqueo repositorio Blackweb

3.1.2. WINDOWS

Activamos la VPN en el equipo de Windows:

Figura 52 – Conexión VPN Windows

Configuramos el proxy desde Configuración -> Red e Internet -> Proxy. Desde configuración manual indicamos la ip del servidor y puerto. Vemos que, en este caso, en comparación a MacOS, la configuración engloba HTTP y HTTPS. Tampoco nos pedirá autenticación en este momento.

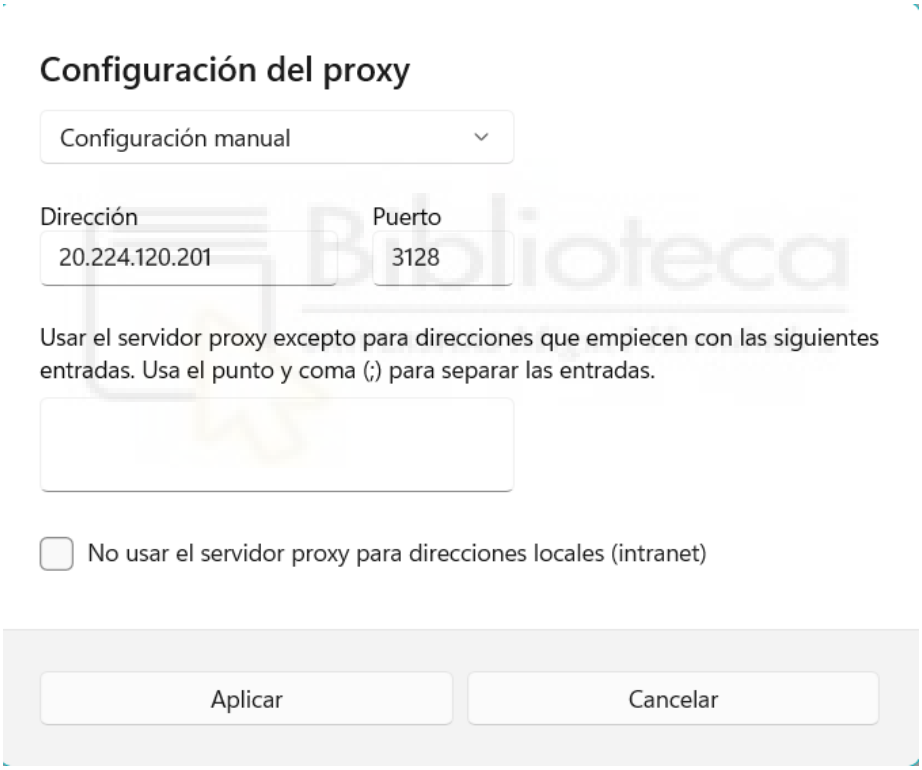

Figura 53 – Configuración proxy Windows

Al acceder a nuestro navegador web, es cuando veremos que nos pide autenticarnos contra el proxy. Para este caso, indicamos el usuario2:

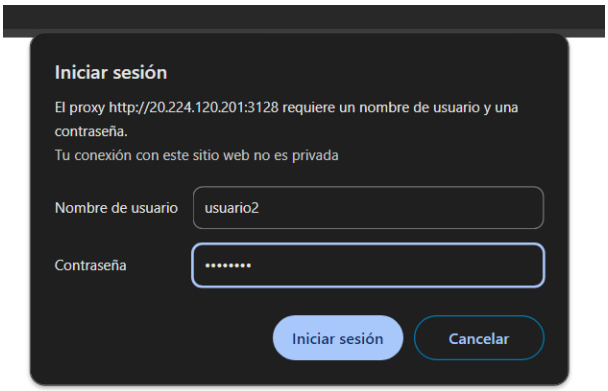

Figura 54 – Configuración usuario proxy Windows

Vamos a probar a acceder a una de las URLs de la acl de bloqueo. Vemos que para el caso de marca.es, al igual que en el primer caso, nos devuelve la página de bloqueo de Squid:

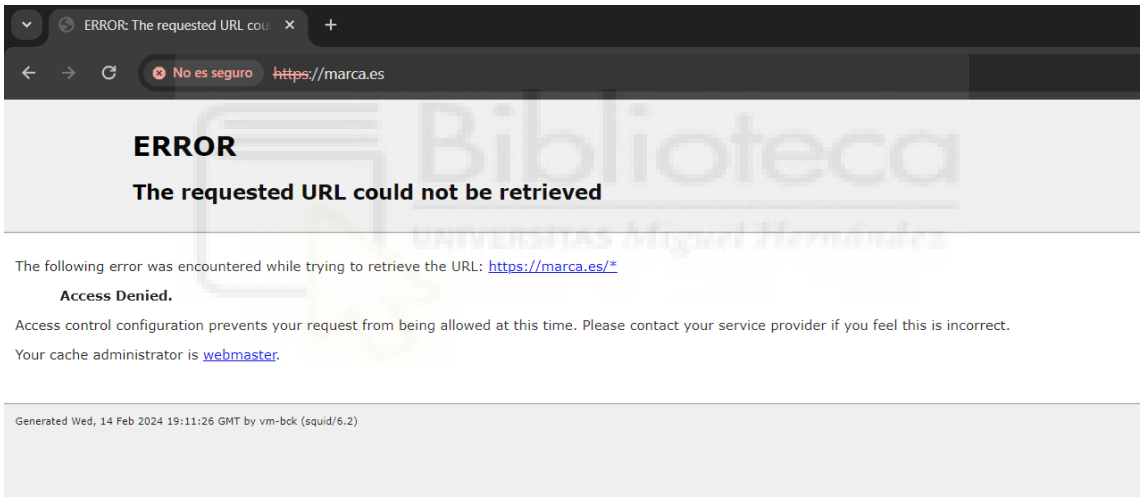

Figura 55 – Bloqueo marca.es Windows

Si accedemos a una página que no hemos bloqueado, vemos que funciona correctamente:

Figura 56 – Acceso chess.com Windows

3.2. ANÁLISIS CON WIRESHARK

Se han capturado trazas en Wireshark [20] para ambos casos de funcionamiento.

Desde el equipo de cliente con sistema operativo MacOS lo primero que vemos es que el tráfico está pasando exitosamente por la ip del servidor proxy Squid.

La traza 29 de Wireshark muestra una solicitud HTTP CONNECT para un dominio en el puerto 443, lo que indica un intento de conexión segura (HTTPS) a través del proxy Squid. Sin embargo, la solicitud es respondida con un código de estado 407, indicando que se requiere autenticación proxy:

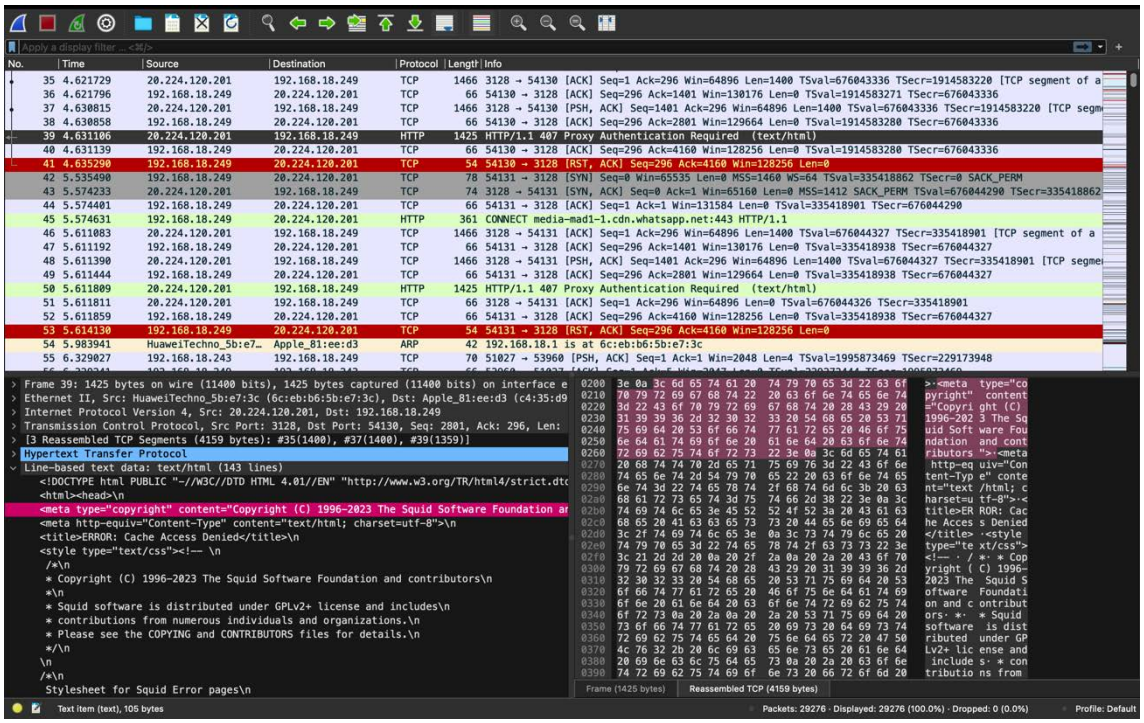

Figura 57 – Traza Wireshark MacOS (Autenticación proxy)

Desde el equipo de Windows, también podemos ver solicitud de autenticación desde la ip del servidor del proxy:

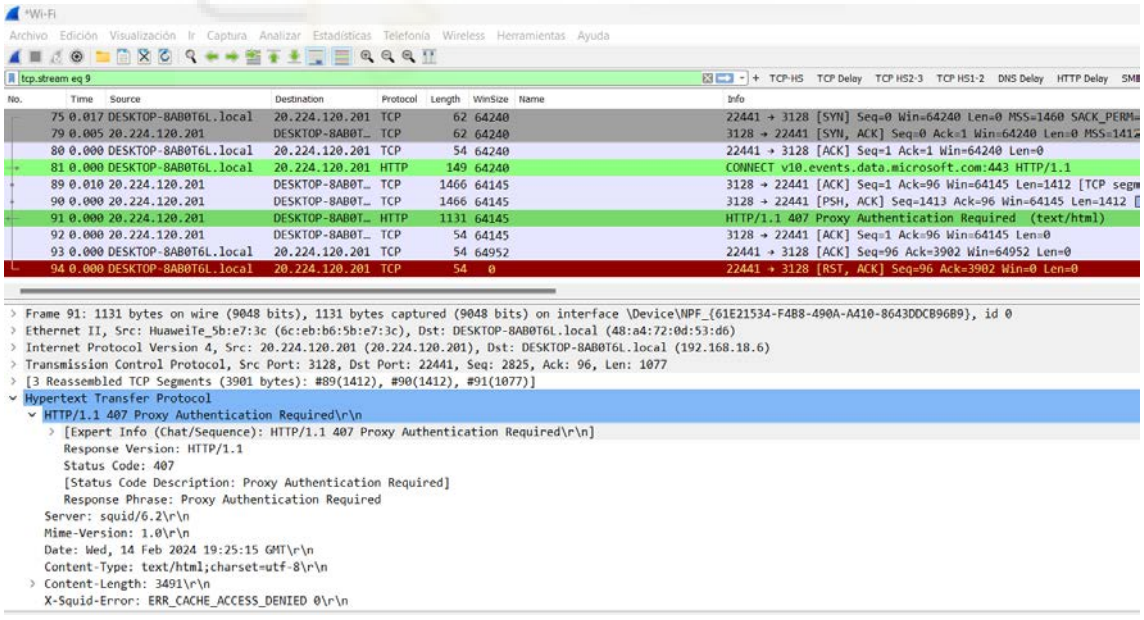

Figura 58 – Traza Wireshark Windows (Autenticación proxy)

La traza 28989 muestra una solicitud HTTP CONNECT para el dominio "marca.es" en el puerto 443. La respuesta es un código de estado 200.

En el contexto de una conexión CONNECT, un código de estado 200 significa que el proxy ha aceptado y establecido la conexión con éxito. La frase "Connection established" confirma que la conexión ha sido establecida con éxito entre el cliente y el servidor de destino (marca.es en este caso) a través del proxy.

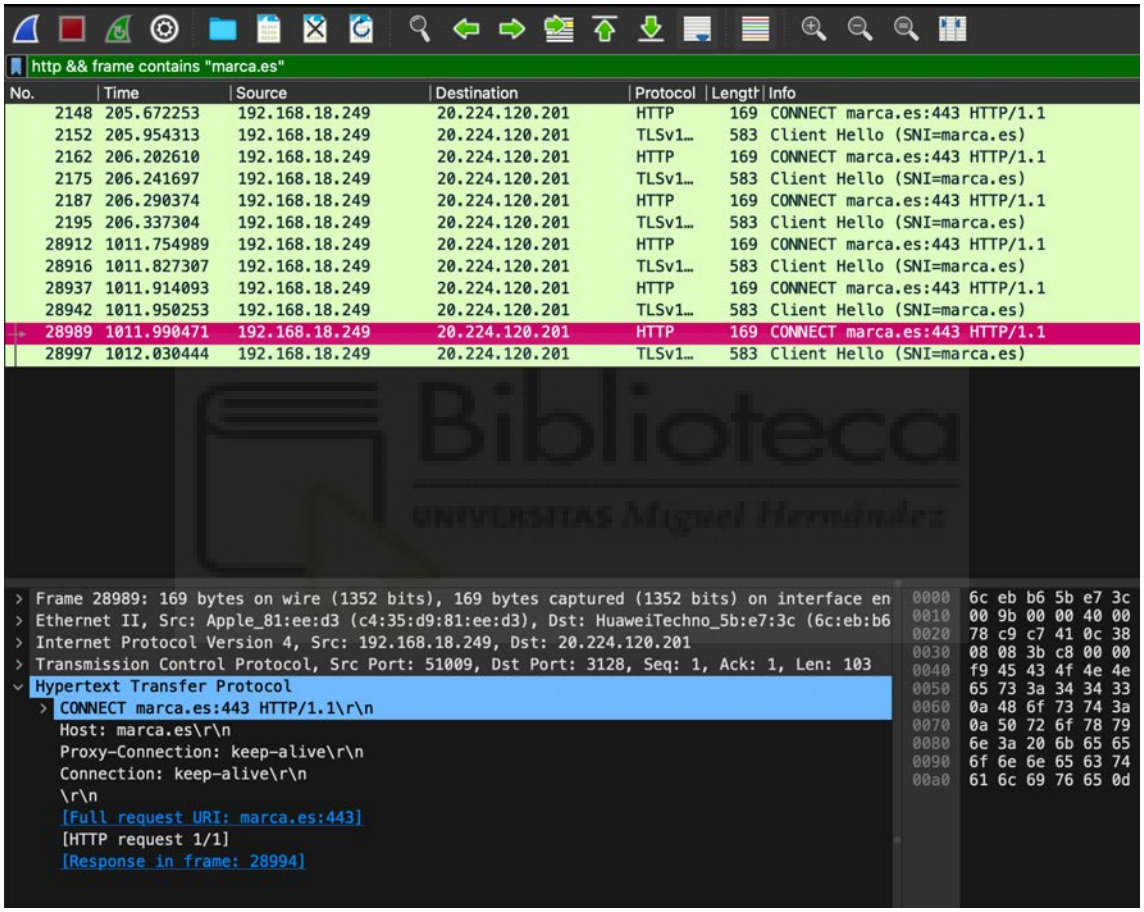

Figura 59 – Traza Wireshark MacOS (Connect)

Desde el equipo de Windows también podemos ver esto. En este caso hemos seguido el flujo de una petición:

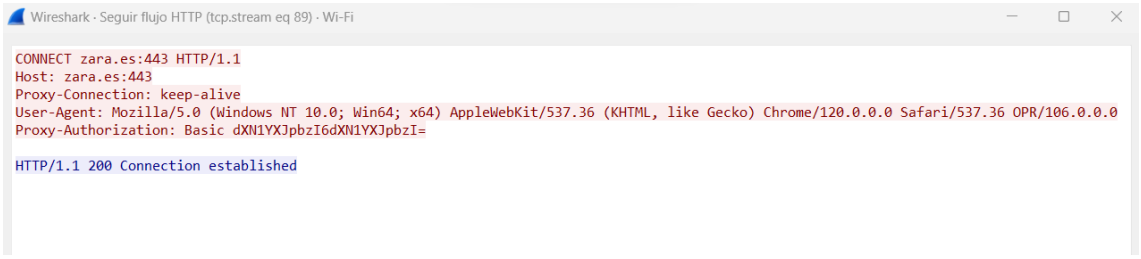

Figura 60 – Traza Wireshark Windows (Connect)

En una de las trazas podemos ver la implementación de ssl-bumping. En esta traza se muestra una negociación exitosa de TLS, lo que indica que se ha establecido una conexión segura entre el cliente y el servidor. La presencia de "Server Hello" y "Change Cipher Spec" indica que el servidor ha respondido con éxito a la solicitud de conexión segura del cliente:

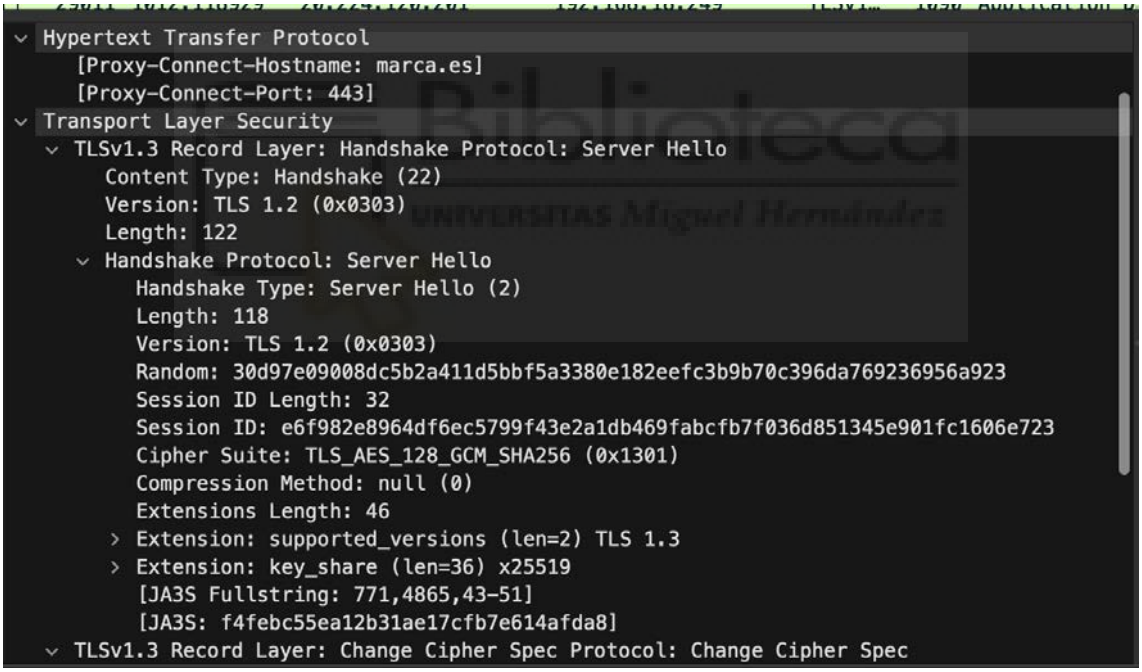

Figura 61 – Traza Wireshark MacOS (TLS)

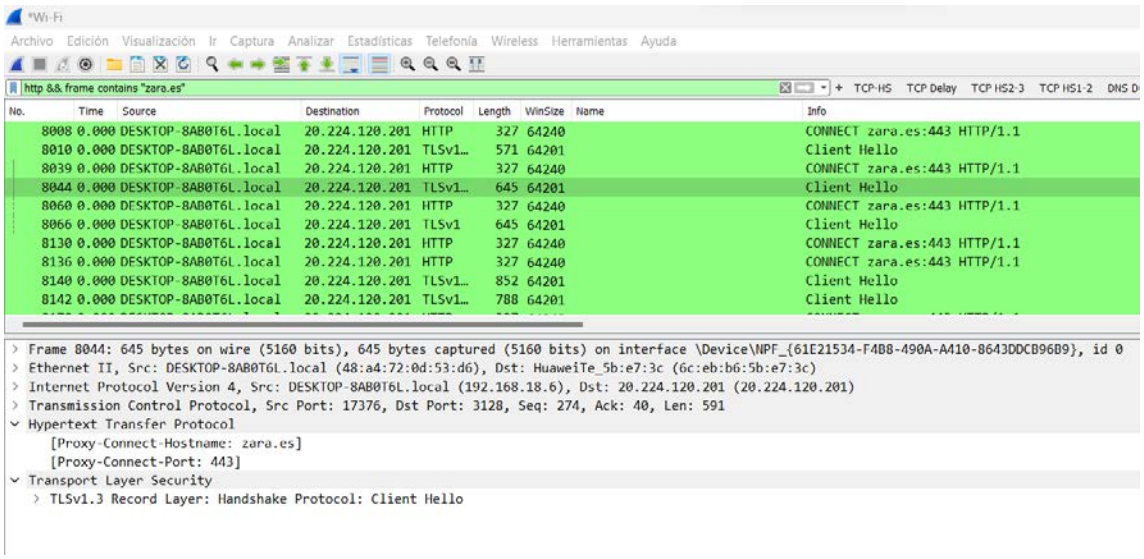

Figura 62 – Traza Wireshark Windows (TLS)

3.3. INFORME DE ANÁLISIS DE SQUID

Para generar un informe detallado, desde la máquina virtual seguimos el procedimiento explicado anteriormente para generar un informe Sage. Seleccionamos el informe generado:

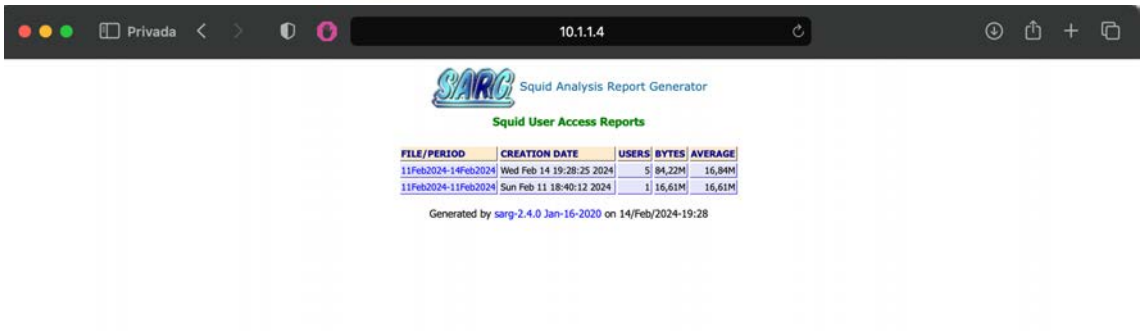

Figura 63 – Selección informe Sarg

Podemos ver que nos aparecen las conexiones de ambos usuarios al proxy además de varios parámetros como tiempo de uso del proxy, porcentaje de cache utilizado, número de bytes,etc.

NUM		USERID	CONNECT	BYTES	%BYTES	IN-CACHE-OUT		ELAPSED TIME	MILLISEC	%TIME
1		usuario2	923	32,29M	38.35%	66.72%	33.28%	00:04:43	283,454	35.68%
2		usuario1	1,07K	27,59M	32.77%	0.95%	99.05%	00:03:45	225,518	28.38%

Figura 64 – Reporte de usuarios Sarg

Vemos también las webs más visitadas (con más peticiones HTTP Connect):

Squid Analysis Report Generator

Squid User Access Reports
Period: 11 Feb 2024—14 Feb 2024
Top 100 sites

NUM	ACCESSED SITE	CONNECT	BYTES	TIME	USERS
1	umh.es	354	43,42M	0:00:43	3
2	www.google.com	255	9,71M	0:00:14	3
3	www.chess.com	191	3,06M	0:00:04	1
4	static.zara.net	96	5,30M	0:00:03	2
5	gateway.icloud.com:443	92	0	0:00:09	2
6	www.notion.so	87	3,65M	0:00:08	1
7	umh.es:443	86	0	0:00:33	3
8	www.google.com:443	61	0	0:00:07	3
9	www.google.es	59	1,29M	0:00:12	2
10	www.chess.com:443	59	0	0:00:13	1
11	www.youtube.com:443	54	0	0:00:10	2
12	af.opera.com:443	51	0	0:00:39	1
13	marca.es:443	44	0	0:01:07	2
14	www.gstatic.com	40	1,80M	0	3
15	www.zara.com	38	676,53K	0:00:05	2

Figura 65 – Webs más visitadas Sarg

También, nos aparecen las webs bloqueadas, que podemos corroborar con nuestras acls:

Squid User Access Reports
Period: 11 Feb 2024—14 Feb 2024
Denied

DATE/TIME	ACCESSED SITE
02/14/24-19:16:28	http://google.es/droga
02/11/24-18:38:37	http://miau/
02/11/24-18:38:37	http://miau/favicon.ico
02/11/24-18:38:37	http://vm-bck:3128/squid-internal-static/icons/SN.png
02/14/24-19:16:29	http://vm-bck:3128/squid-internal-static/icons/SN.png
02/14/24-19:19:28	http://vm-bck:3128/squid-internal-static/icons/SN.png
02/14/24-19:22:29	http://vm-bck:3128/squid-internal-static/icons/SN.png
02/14/24-19:23:38	http://vm-bck:3128/squid-internal-static/icons/SN.png
02/14/24-19:22:29	http://zara.es/
02/14/24-19:23:38	http://zara.es/

Figura 66 – Webs denegadas Sarg

Para usuario1, sacamos un reporte completo de cada página visitada:

Squid User Access Reports
Period: 11 Feb 2024—14 Feb 2024
User: usuario1
Sort: bytes, reverse
User report

ACCESSED SITE	CONNECT	BYTES	%BYTES	IN-CACHE-OUT		ELAPSED TIME	MILLISEC	%TIME
www.google.com	208	8,01M	29.04%	0.41%	99.59%	00:00:09	9,913	4.40%
umh.es	62	7,33M	26.57%	0.46%	99.54%	00:00:04	4,317	1.91%
static.zara.net	48	2,65M	9.61%	0.00%	100.00%	00:00:01	1,963	0.87%
universite.umh.es	5	1,98M	7.19%	0.00%	100.00%	00:00:02	2,224	0.99%
www.gstatic.com	29	1,72M	6.25%	0.00%	100.00%	00:00:00	515	0.23%
www.youtube.com	9	1,08M	3.93%	0.00%	100.00%	00:00:00	330	0.15%
www.google.es	33	696,11K	2.52%	0.00%	100.00%	00:00:11	11,885	5.27%
phantom-elmundo.unidadeditorial.es	16	633,84K	2.30%	0.00%	100.00%	00:00:00	178	0.08%

Figura 67 – Reporte usuario1 Sarg

Por último, generamos una gráfica para poder comparar datos de los diferentes reportes para un usuario, donde medimos el uso de bytes:

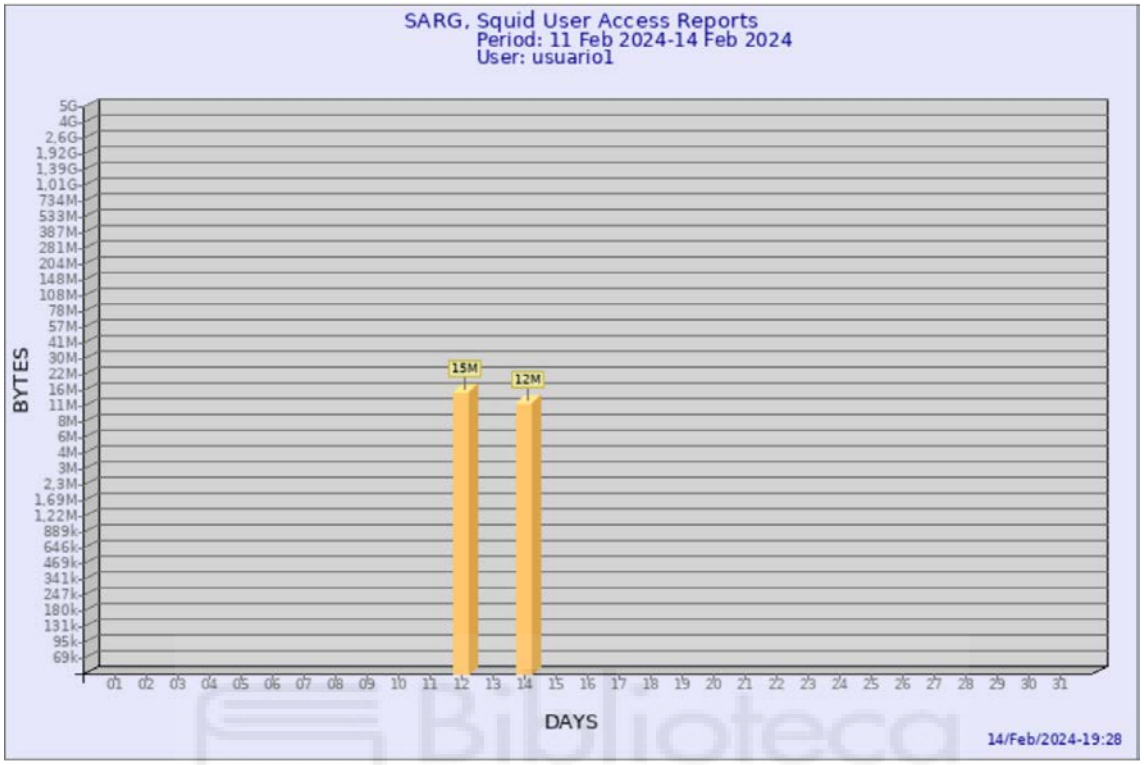

Figura 68 – Gráfica uso de bytes Sarg

3.4. DISCUSIÓN DE LOS RESULTADOS

La implementación del proyecto ha demostrado un despliegue efectivo y funcional del proxy Squid en un entorno de red con máquinas cliente que ejecutan sistemas operativos MacOS y Windows.

Las pruebas de funcionamiento han validado la capacidad del sistema para gestionar la autenticación de usuarios y aplicar restricciones de acceso según las listas de control de acceso (ACL). La configuración y el bloqueo de sitios web específicos, así como palabras clave, se han ejecutado correctamente.

El análisis con Wireshark ha proporcionado una visión detallada de la interacción entre los clientes y el servidor proxy, confirmando la autenticación exitosa y la conexión establecida a través de solicitudes HTTP CONNECT. Se han identificado los códigos de estado 407 para autenticación proxy y 200 para conexión establecida, respaldando la efectividad del proceso.

El informe de análisis de Squid Sage ha ofrecido una comprensión profunda de las actividades de los usuarios, destacando webs más visitadas, bloqueadas y generando reportes detallados por usuario. Este enfoque analítico proporciona una herramienta valiosa para evaluar el rendimiento y la eficacia del proxy.

En resumen, los resultados obtenidos validan la implementación exitosa del proyecto, destacando la capacidad de Squid para gestionar el tráfico web de manera efectiva y proporcionar un control detallado sobre el acceso a recursos en línea.

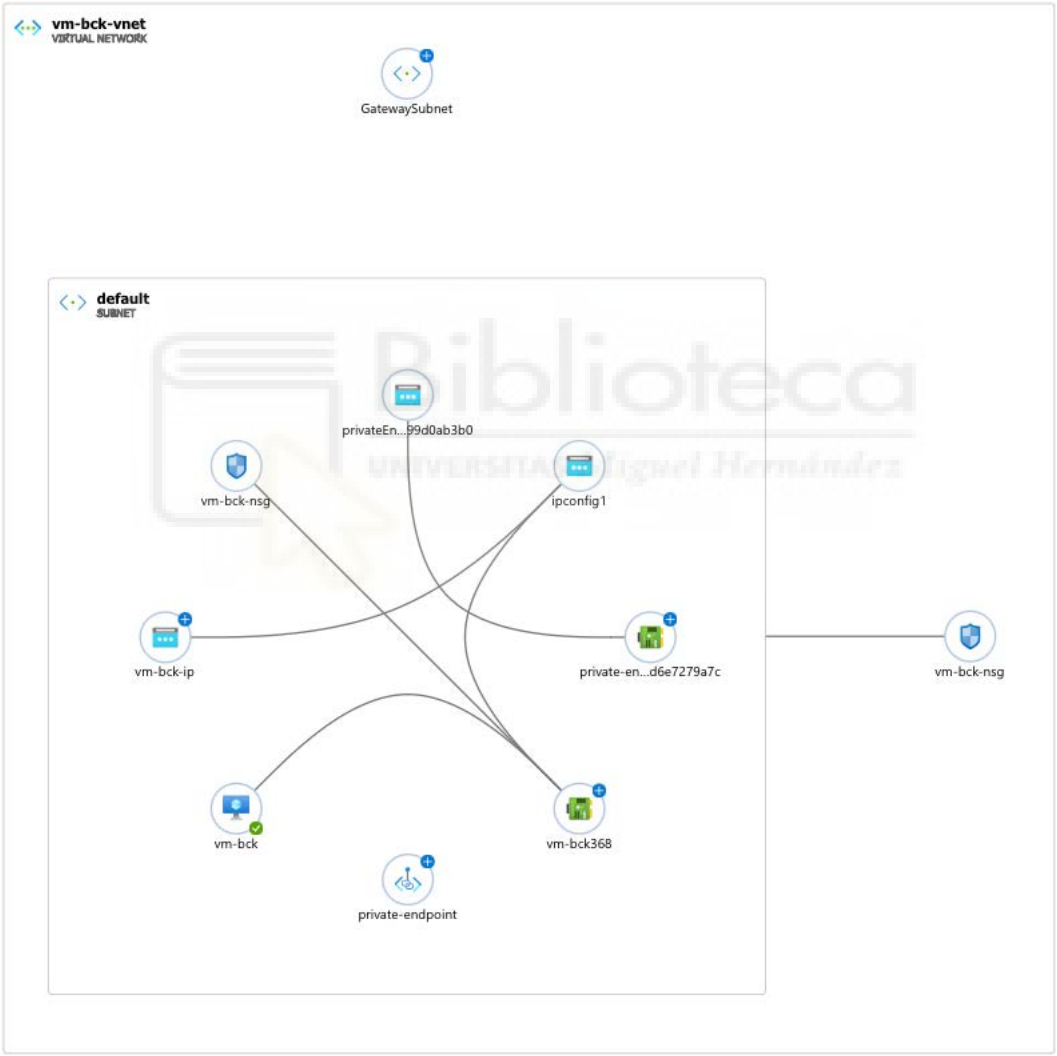

Figura 69 – Topología final red virtual

3.5. LÍNEAS DE MEJORA

Entre las posibles áreas de mejora, se pueden considerar las siguientes:

- Explorar opciones para una interfaz de usuario más intuitiva.
- Automatizar tareas administrativas recurrentes.
- Continuar refinando las políticas de seguridad.
- Integrar el proxy Squid con sistemas de autenticación existentes.
- Explorar opciones para implementar políticas de control de acceso más detalladas y específicas.
- Establecer un sistema de monitoreo continuo del rendimiento.
- Explorar la integración con herramientas de seguridad adicionales.

Estas mejoras buscan optimizar la administración, seguridad y rendimiento del proxy Squid, adaptándolo a las cambiantes necesidades del entorno de red.

4. CONCLUSIONES

A lo largo de este proyecto, se han logrado implementar exitosamente diversas tecnologías y herramientas, demostrando una comprensión profunda de las especificaciones técnicas y la aplicación práctica de conceptos clave en el ámbito de la administración y seguridad de redes. Desde la creación de una máquina virtual en Azure hasta la implementación de un proxy Squid con SSL bumping, cada paso ha sido detalladamente abordado.

La configuración de una VPN Point-to-Site y la generación de certificados OpenSSL para establecer conexiones seguras han demostrado la capacidad para diseñar e implementar soluciones robustas de red. Además, se han establecido reglas en el Grupo de Seguridad de Red (NSG) para fortalecer la seguridad de la máquina virtual, añadiendo un componente esencial en la administración de accesos.

La integración de Azure Automation y la creación de un runbook han llevado la automatización a un nivel superior, permitiendo ejecutar tareas específicas de manera eficiente y programada. La implementación de webhooks ha proporcionado una interfaz para la interacción externa, brindando una mayor flexibilidad y potencial de integración.

Finalmente, los resultados obtenidos fueron validados mediante el análisis detallado de registros de Squid y trazas capturadas con Wireshark, proporcionando una visión clara del rendimiento y la seguridad del sistema implementado. La discusión de los resultados resalta los logros alcanzados y sugiere posibles áreas de mejora, brindando oportunidades para futuras optimizaciones y refinamientos en base a la experiencia práctica obtenida durante la implementación.

Este proyecto no solo ha cumplido con las expectativas técnicas y funcionales, sino que también ha demostrado la capacidad de abordar desafíos complejos en el ámbito de la administración de redes y la seguridad, consolidando así un sólido trabajo de investigación y desarrollo.

5. ANEXOS

5.1. ANEXO DE ACRÓNIMOS

Acrónimo	Significado
SSL	Secure Socket Layer
TLS	Transport Layer Security
TCP	Transmission Control Protocol
IP	Internet Protocol
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
API	Application Programming Interface
HTTP/HTTPS	Hypertext Transfer Protocol/Secure
CDN	Content Delivery Network
ESI	Edge Server Includes
VPN	Virtual Private Network
FTP	File Transfer Protocol
GPL	General Public License
NAT	Network Address Translation
BGP	Border Gateway Protocol
RFC	Request for Comment
ACL	Access Control List

Tabla 15 – Tabla de acrónimos

6. BIBLIOGRAFÍA

1. Documentación Squid

[http://www.squid-cache.org/Versions/v6/cfgman/]

2. Portal Azure

[https://portal.azure.com/]

3. Tipos de servicios en la nube

[https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-cloud-computing/]

4. Calculadora de Azure

[https://azure.microsoft.com/es-es/pricing/calculator/]

5. Crear VM en Azure

[https://azure.microsoft.com/es-es/products/virtual-machines/linux/]

6. Configuración VNET

[https://learn.microsoft.com/es-es/azure/virtual-network/virtual-networks-overview]

7. Creación VPN Gateway

[https://learn.microsoft.com/es-es/azure/vpn-gateway/tutorial-create-gateway-portal]

8. Certificado OpenSSL

[https://www.ibm.com/docs/es/rstfsq/9.1.0?topic=overview-creating-digital-certificate-openssl- Creación VPN Gateway]

9. Configuración VPN Point-to-Site

[https://learn.microsoft.com/es-es/azure/vpn-gateway/vpn-gateway-howto-point-to-site-resource-manager-portal- Crear VM en Azure]

10. Configuración VPN en cliente Mac

[https://learn.microsoft.com/es-es/azure/vpn-gateway/point-to-site-vpn-client-cert-mac]

11. Configuración VPN en cliente Windows

[https://learn.microsoft.com/es-es/azure/vpn-gateway/point-to-site-vpn-client-cert-windows]

12. Acls en Squid

[https://wiki.squid-cache.org/SquidFaq/SquidAc]

13. Repositorio Blackweb

[https://github.com/maravento/blackweb]

14. SSL Bump Squid

[https://wiki.squid-cache.org/Features/SslBump]

15. Documentación RFC

[https://www.rfc-editor.org/rfc-index-100a.html]

16. Configuración Azure Automation

[https://learn.microsoft.com/es-es/azure/automation/overview]

17. Runbook Powershell

[https://learn.microsoft.com/es-es/azure/automation/learn/powershell-runbook-managed-identity]

18. Módulo Posh-SSH

[https://www.powershellgallery.com/packages/Posh-SSH/3.0.8]

19. Configuración Sarg

[https://books.google.es/books?hl=es&lr=&id=WwJwDQAAQBAJ&oi=fnd&pg=PP1&dq=squid+sarg&ots=m12ANGn5op&sig=wMFJj1Re7-9CXrdfp0yDpKcyjUc#v=onepage&q=squid%20sarg&f=false]

20. Instalación Wireshark

[https://www.wireshark.org]