Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

PAI 4 - VULNAWEB. AUDITORÍA DE SEGURIDAD EN SISTEMAS INFORMÁTICOS Y ANÁLISIS DE VULNERABILIDADES EN APLICACIONES WEB PARA EMPRESA DE COMERCIO ELECTRÓNICO

Grado en Ingeniería Informática – Ingeniería del Software

Seguridad en Sistemas Informáticos e Internet

Curso 2023 – 2024

Participantes:
Juan Luis Ruano Muriedas

José Joaquin Rojas Romero

Antonio José Suárez García

Índice

Resumen ejecutivo

Tarea 1 - Se ha realizado un proceso de hardening del sistema empleando la herramienta de código abierto ***Lynis***, pasando de una puntuación inicial de 67 puntos a **74 puntos** sobre un sistema *Arch Linux*. Por otro lado se han establecido unas pautas y acciones a seguir para garantizar más seguridad en las comunicaciones de comercio electrónico a través del browser ***Mozilla Firefox***.

Tarea 2 - Usando la aplicación **CONAN mobile** se ha comprobado la seguridad de los dispositivos móviles de cada uno de los miembros del Security Team. Tras esto, se ha procedido a comprobar que se cumplen con los puntos de la **Política de Seguridad** de la empresa y a realizar las **modificaciones necesarias** a la configuración del dispositivo para cumplirlos.

Tarea 3 - A través del uso de la herramienta **OWASP ZAP (Zed Attack Proxy)** se pretende configurar un **proxy de ataque** para realizar pruebas de seguridad de aplicaciones dinámicas con el objetivo de **comprobar la seguridad** de estas.

Tarea 4 - Se procederá a realizar ataques de prueba a una **aplicación propia** de la organización INSEGUS 12 para mostrar el **funcionamiento del OWASP ZAP** y se generará un informe con los fallos y **recomendaciones** que usted podrá leer.

Tarea 1

Auditoría de Seguridad y Bastionado de un determinado sistema informático de sobremesa.

Se ha instalado la herramienta de *hardening* ***Lynis*** en un host con sistema operativo ***Arch Linux***. Dicho sistema operativo cuenta con un ***kernel Linux-zen*** en su última versión. También cuenta con ***GRUB***. Por otro lado el host es controlado con solo dos usuarios ***sudoers*** (totales privilegios sobre el sistema) los cuales son ***root* y el usuario creado** para gestionar el sistema en general, pues ambos usuarios son los únicos que pertenecen al grupo ***wheel***.

El **primer análisis** realizado con la herramienta *Lynis* mediante el comando: ***sudo lynis -Q*** (análisis rápido) ha finalizado con éxito generando un log en la ruta ***/etc/log/lynis.log***. Dicho log muestra un **índice de hardening de 67 puntos** (Ver **Figura 1**), distando de solo un punto de la puntuación requerida por la empresa cliente. Hemos llegado a la conclusión de que el principal motivo por el que la puntuación supera los 50 puntos, sin hacer previas acciones de hardening, se debe a que el sistema operativo *Arch Linux* es desarrollado por la comunidad de manera abierta, lo cual permite revelar y corregir vulnerabilidades en cada nueva versión de manera transparente, a diferencia de sistemas operativos como *macOS*, donde el kernel es privado y nadie externo a la empresa que lo

controla puede analizar y/o participar en el proyecto de mejora. El motivo que se comenta junto con el hecho de que en el proceso de instalación de *Arch* el usuario puede realizar la mínima instalación de recursos de manera libre para que el host pueda realizar tareas básicas sin ningún tipo de problema, hacen de este sistema un sistema más robusto que otros más privativos como los que pertenecen a *Microsoft* o a *Apple*, e incluso más robusto que otras distribuciones de *Linux*, pues cuenta con menor volumen de aplicaciones preinstaladas las cuales pueden ser el origen principal de las vulnerabilidades.

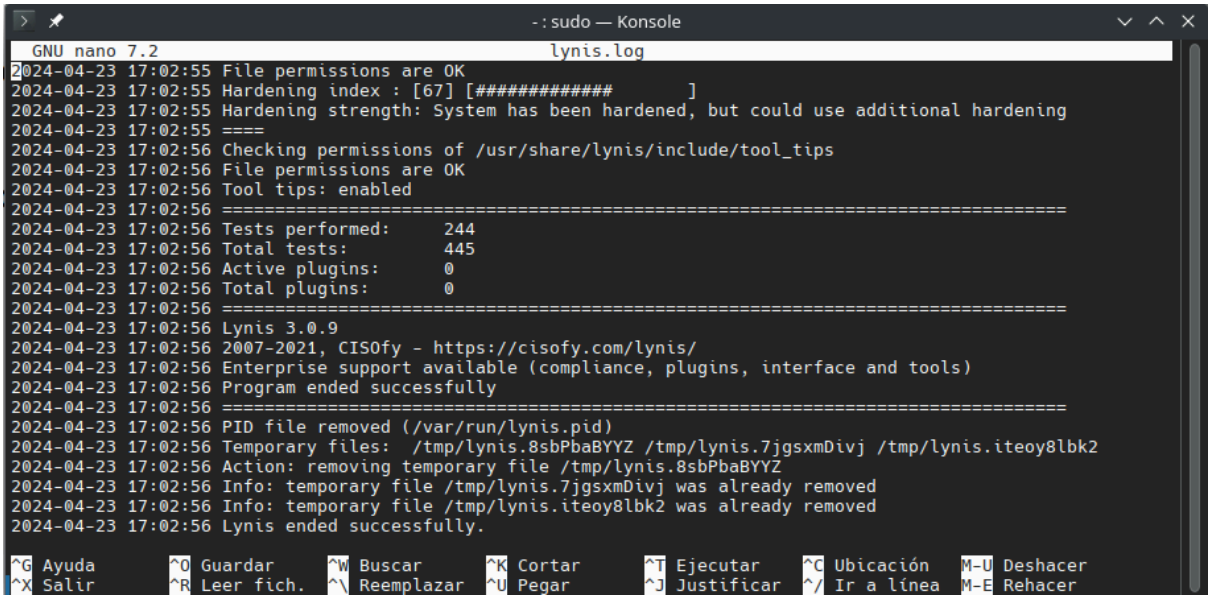

Figura 1. Log generado tras realizar el primer análisis con Lynis, mostrando una puntuación de 67.

Para mejorar la puntuación tras el primer análisis, se ha revisado el log en busca de sugerencias sobre los elementos conocidos poco protegidos, o en ocasiones, sin nivel de protección. Se han realizado las siguientes modificaciones en el sistema para garantizar las correcciones sobre las sugerencias y llevar a cabo el proceso de hardening. Para cada modificación realizada se mostrará la captura de la sugerencia y las acciones realizadas sobre el sistema.

Sugerencia 1

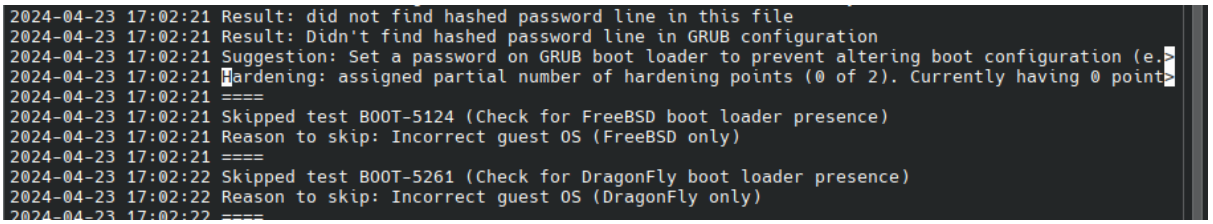

Figura 2. Sugerencia sobre establecer contraseña en el *GRUB* para evitar alteraciones en la configuración de arranque del sistema.

GRUB (*GNU GRand Unified Bootloader*) es un cargador de arranque múltiple, desarrollado por el proyecto *GNU* que nos permite elegir qué Sistema Operativo arrancar de los instalados en nuestro sistema. Es necesario en el proceso de instalación de *Arch* y empleado en la mayoría de distribuciones de *Linux*. Por defecto se puede acceder a la configuración del *GRUB* cuando este nos

aparece al iniciar el host, y si no lo tenemos protegido con contraseña, podría ser un vector de ataque bastante contundente permitiéndonos hacer escalada de privilegios, entre otras posibles acciones malintencionadas. En *Arch* este problema se soluciona relativamente fácil siguiendo la documentación oficial en ***wiki.archlinux.org/title/GRUB/Tips_and_tricks*** en el apartado ***6 Password protection of GRUB menu***. La documentación nos recomienda establecer ususario y contraseña para acceder al archivo de configuración de *GRUB*, mediante el comando: ***grub-mkpasswd-pbkdf2***, el cual nos pedirá ingresar una contraseña la cual será hasheada y devuelta para copiarla y pegarla en el archivo de configuración de *GRUB* (ver **Figura 3**), el cual deberá ser previamente modificado con los permisos correspondientes para que solo el usuario root pueda leerlo, mediante el comando: ***chmod o-r /etc/grub.d/40_custom***. Después modificamos el archivo de configuración de *GRUB* en la ruta ***/etc/grub.d/40_custom***, donde añadiremos la líneas: **set superusers="username"** y ***password_pbkdf2 = username contraseñaHasheadaCopiada***, donde "*username*" será nuestro usuario del sistema con el que gestionaremos el *GRUB* (ver **Figura 4**).

Se ha decidido resolver esta sugerencia al plantearse un supuesto caso en el que alguien ajeno al equipo de la empresa, por ejemplo un personal de limpieza o un técnico que se decide a reparar ordenadores, tenga acceso físico al host y decida manipularlo a través del *GRUB*.

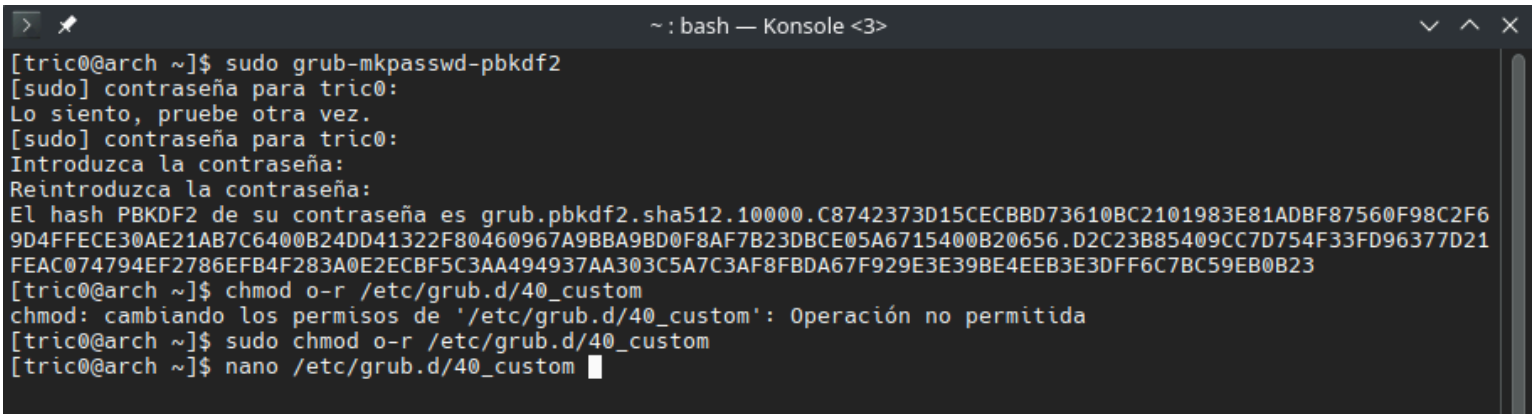

Figura 3. Comando *sudo grub-mkpasswd-pbkdf2* para generar hash dada una clave por el usuario.

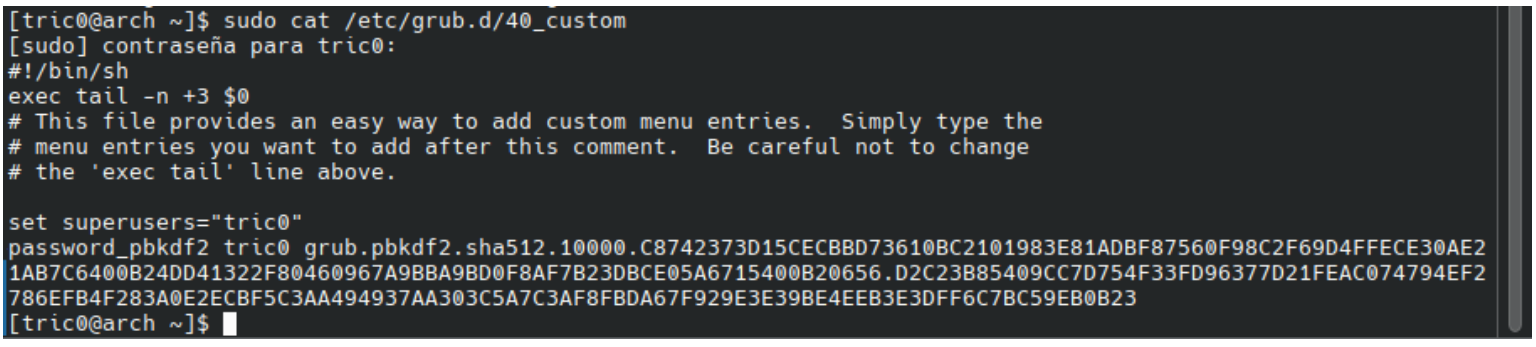

Figura 4. Archivo de configuración de *GRUB* editado con un *superuser* y la clave generada.

Sugerencia 2

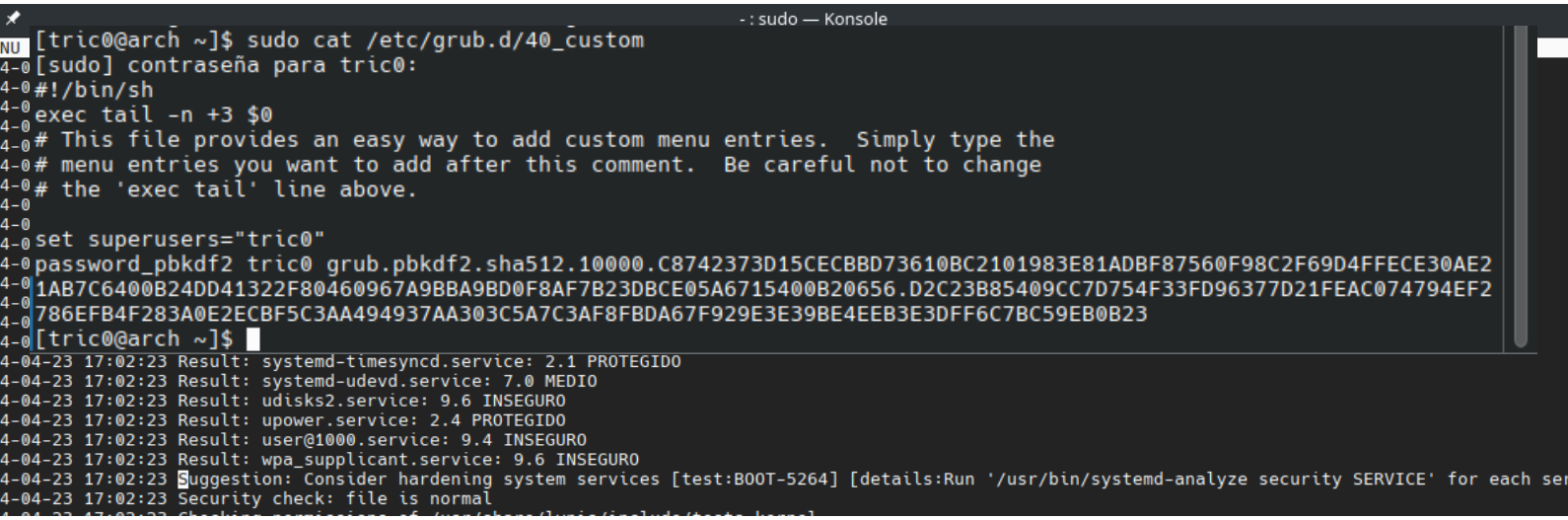

Figura 5. Sugerencia sobre hacer bastionado de todos los servicios que se listan en esta captura.
La siguiente sugerencia del log muestra los **servicios del sistema** con una de las siguientes etiquetas: ***PROTEGIDO, MEDIO, INSEGURO*** y ***EXPUESTO***. La sugerencia nos recomienda emplear el comando ***/usr/bin/systemd-analyze security "SERVICIO"*** donde "SERVICIO" es el servicio del listado del log que queremos analizar, cuya etiqueta es **INSEGURO** o **EXPUESTO** para obtener posibles mejoras de seguridad. Se ha empleado dicho comando para mejorar la seguridad en aquellos servicios relacionados con el **nivel de red** y/o **contraseñas del sistema**. Uno de los servicios examinados ha sido **httpd.service**, el cual se encarga de controlar el servidor web **HTTP Apache**.

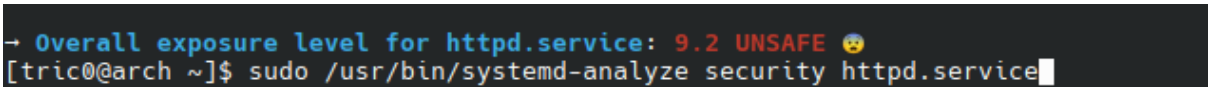

Figura 6. Resultado del comando */usr/bin/systemd-analyze security httpd.service* con 9.2 puntos.

El análisis de dicho servicio dio una puntuación de **9.2 bastante insegura**, por lo que se tomaron varias medidas al respecto. Primero se limitó el acceso a los archivos de configuración de Apache con los siguientes comandos: ***sudo chmod 750 /etc/httpd/conf*** y ***sudo chown http:http /etc/httpd/conf***.

El motivo por el que se decidió tomar esta acción sobre dicha sugerencia fue debido a que suele ser habitual que los ordenadores de una empresa empleen el servidor **HTTP Apache** para la prueba o despliegue de alguna aplicación, por lo que es buena práctica securizar el acceso al archivo de configuración de *Apache* para que solo puedan acceder usuarios permitidos.

Sugerencia 3

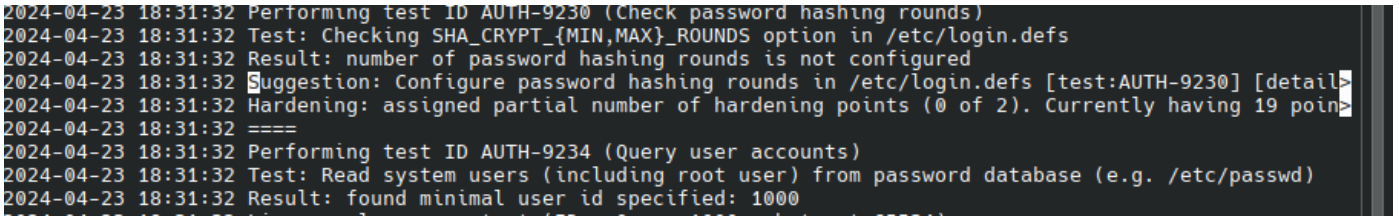

Figura 7.Sugerencia de aumentar las *rounds* del archivo de configuración ***login.defs***.

Otra sugerencia del análisis fue aumentar el número de hashing rounds del archivo ***/etc/login.defs***, lo cual nos permitirá ser más resistentes frente a ataques de fuerza bruta en nuestras contraseñas del sistema. Para ello se accedió a dicho fichero y se puso un valor ***SHA_CRYPT_MIN_ROUNDS 10000***, en vez de 5000 por defecto.

Sugerencia 4

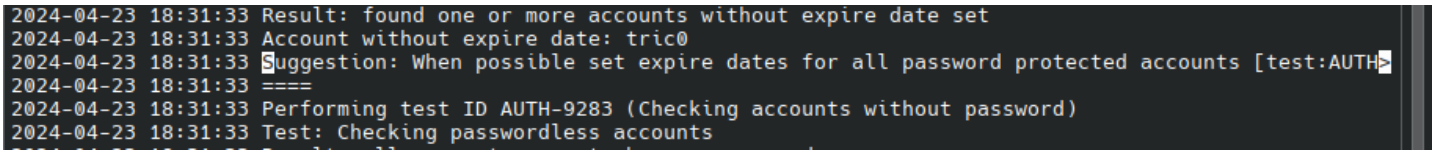

Figura 8.Sugerencia de establecer fechas de expiración de contraseñas en archivo ***login.defs***.

Siguiendo la línea de securizar contraseñas, el log nos advierte de añadir una fecha de expiración de contrasñas en el fichero ***/etc/longing.defs***, en nuestro caso hemos decidido seguir las recomendaciones del portal RedHat:

https://www.redhat.com/sysadmin/password-expiration-date-linux,

indicando las siguientes modificaciones: ***PASS_MAX_DAYS 90, PASS_MIN_DAYS 7, PASS_WARN_AGE 5***. Estas modificaciones tendrán validez para nuevas contraseñas, por lo que tendremos que hacer uso del comando: ***chage***.

Sugerencia 5

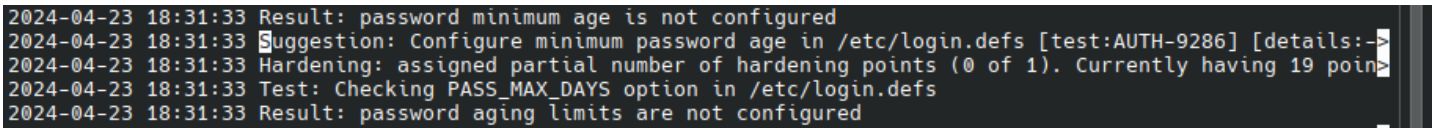

Figura 9.Sugerencia de modificación de la edad mínima de las contraseñas en el archivo ***login.defs***.

Otra sugerencia indicada por el log de Lynis está estrechamente relacionada con la anterior, pues en el mismo portal de RedHat se encuentra la solución a dicho problema, el cual se trata de establecer

un mínimo y máximo 'age' o edad de las contraseñas. Se han seguido los pasos y se han agregado las siguientes modificaciones mediante el comando: ***sudo chage --list user1***

Minimum number of days between password change : 7
Maximum number of days between password change : 90
Number of days of warning before password expires : 5

Sugerencia 6

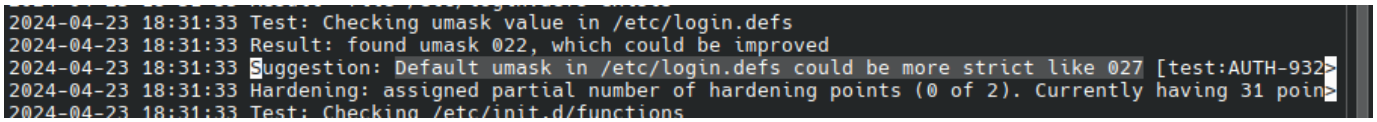

Figura 10.Sugerencia de modificación de ***umask*** en el archivo ***login.defs***.

La siguiente sugerencia que se ha corregido ha sido aumentar el valor de **022** a **027** en el parámetro ***umask*** del fichero ***/etc/loging.defs***. ***umask*** es un comando que establece los permisos predeterminados para archivos y directorios recién creados. El valor de *umask* se resta de los permisos predeterminados (777 para directorios y 666 para archivos) para determinar los permisos reales para nuevos archivos y directorios. El valor de umask predeterminado en la mayoría de las distribuciones de *Linux* es 022, lo que significa que se crean nuevos archivos con los permisos 644 (rw-r--r--) y nuevos directorios con los permisos 755 (rwxr-xr-x).

Sugerencia 7

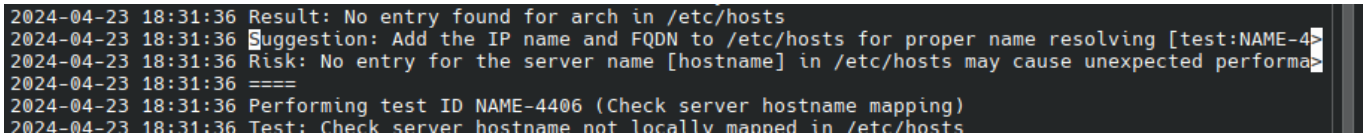

Figura 11.Sugerencia de añadir **IP** y **FQDN** (*Full Qualified Domain Name*) al archivo ***/etc/hosts***.

Por otro lado se ha añadido **IP** y **FQDN** al fichero ***/etc/hosts*** para un correcto ***name resolving***. Para ello hemos añadido la línea siguiente en dicho fichero:
192.168.1.106 myname.homelinux.org myname donde *myname* sería el nombre identificativo del host en particular.

Sugerencia 8

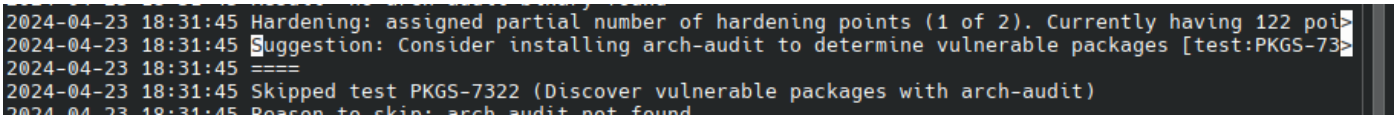

Figura 12.Sugerencia de instalación de herramienta ***arch-audit***.

La siguiente sugerencia tras el análisis nos indicaba instalar una herramienta de auditoría para el sistema operativo *Arch* para detectar **paquetes vulnerables**. Dicha herramienta es ***arch-audit*** y se ha

instalado con los valores por defecto. Solo ha sido necesario instalar dicha herramienta para mejorar la puntuación tras el análisis, pero se recomienda su uso y así poder detectar posibles irregularidades en los paquetes de nuestro sistema.

Sugerencia 9

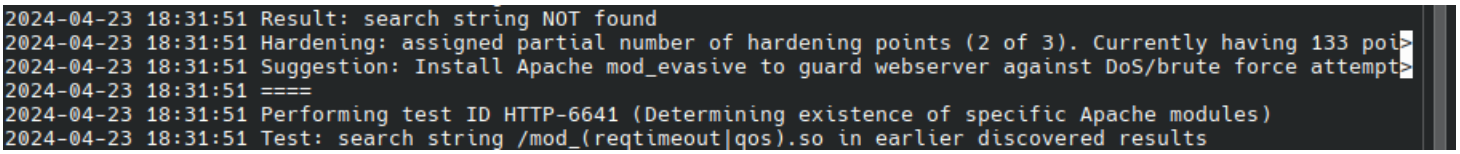

Figura 13.Sugerencia de instalar la herramienta mod_evasive.

Relacionado con Apache o cualquier servidor web instalado en nuestra máquina, *Lynis* nos recomendó instalar ***mod_evasive*** para prevenir intentos de ataques ***DoS*** y ***fuerza bruta*** en nuestro servidor web. Para ello instalamos dicha herramienta con la configuración por defecto y mejorar el índice de puntuación. De la misma manera que en la sugerencia 8, recomendamos su uso y estudio.

Sugerencia 10

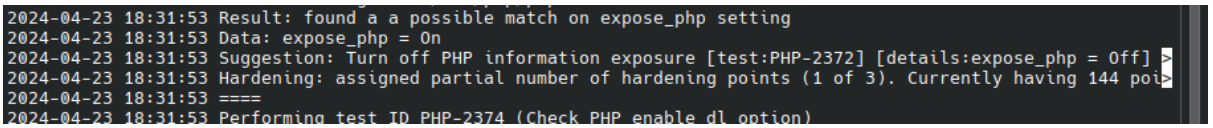

Figura 14.Sugerencia de desactivación de la exposición de información de ***PHP***.

En caso de que la web del cliente funcionase con ***PHP***, hemos seguido la recomendación de *Lynis* de desactivar la exposición de información de *PHP*. Para ello accedemos al fichero de configuración de php: ***sudo nano /etc/php/<version>/apache2/php.ini*** y reescribimos la siguiente línea tal y como se indica: ***expose_php = Off***.

Resultados tras aplicar las sugerencias de *Lynis*

Con todas estas modificaciones del sistema se ha vuelto a lanzar un análisis de *Lynis* el cual ha retornado una puntuación de hardening de 74 puntos, lo cual nos indica que hemos superado el umbral previsto para el cliente (ver **Figura 15**). No obstante se podría aumentar más dicho valor, con la diferencia de que el resto de sugerencias recomendadas por el análisis están relacionadas con el hardening mediante la instalación de algún sistema FIM, como el empleado en el *PAI1* (*Wazuh*) u otras aplicaciones similares, por lo que no se han tenido en cuenta ya que no se ha visto necesario.

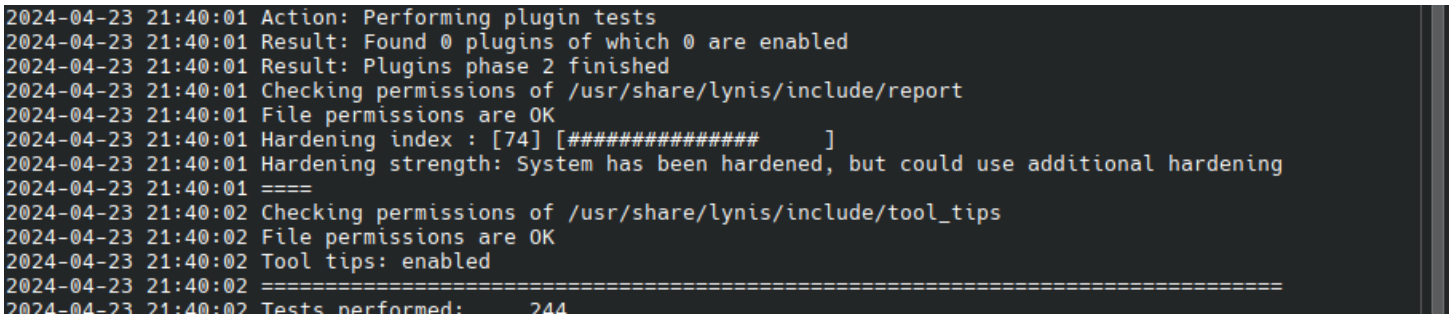

Figura 15.Análisis final con todas las sugerencias llevadas a cabo, obteniendo la puntuación de **74**.

Finalmente comentar que tras el proceso de bastionado, se ha hecho especial hincapié en la seguridad de los usuarios y contraseñas a través de su correspondientes archivos de configuración, para evitar en caso de hackeo la posible escalada de privilegios y sufrir riesgos mayores.

Configuración segura del browser para realizar comunicaciones.

Para realizar una configuración segura del *browser*, se ha elegido ***Mozilla Firefox*** como browser para los equipos informáticos de la empresa. Esta elección se debe a que *Firefox* cuenta con **utilidades nativas** para mejorar la seguridad web según nuestras necesidades, esto en una mayor proporción a otros buscadores actuales como *Chrome* o *Edge*.

Otro buscador interesante para este caso hubiese sido ***DuckDuckGo***, el cual es el buscador que emplea la red de ***Tor*** y nos garantiza que no se guardará ningún tipo de dato entre sesiones de búsquedas, dándonos la opción de “quemar” los datos al final de nuestra sesión, pero se dejará como mención honorífica ya que *Firefox* es mucho más conocido y estable y su set de utilidades y configuraciones nos vale para este caso.

En primer lugar recomendamos desinstalar cualquier extensión en nuestra sesión de navegación, pues esto mejorará nuestra huella digital y evitará muchos problemas relacionados con la privacidad de nuestros datos. Después hacemos clic en las tres barras de la izquierda y después clic en ***ajustes*** en la parte superior.

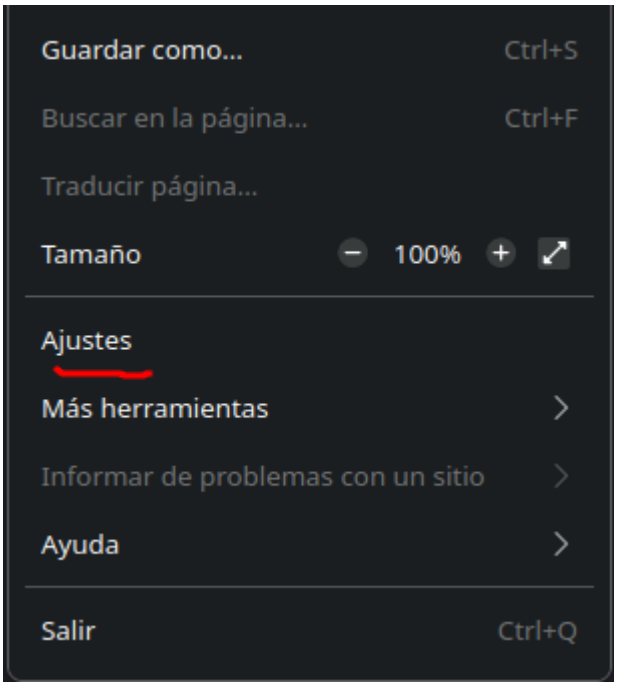

Figura 16.Análisis final con todas las sugerencias llevadas a cabo, obteniendo la puntuación de **74**.

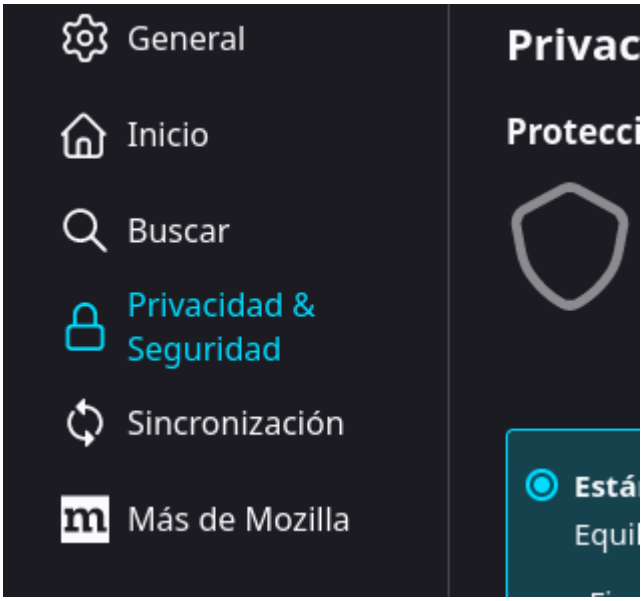

Figura 17.Privacidad & Seguridad.

En la ventana que se nos abre buscamos la opción del menú lateral izquierdo **Privacidad & Seguridad.** Una vez dentro tendremos a nuestra disposición una cantidad de ajustes razonable para mejorar la seguridad de nuestro browser. Dichos ajustes se verán a continuación mediante capturas y una breve descripción de su funcionamiento.

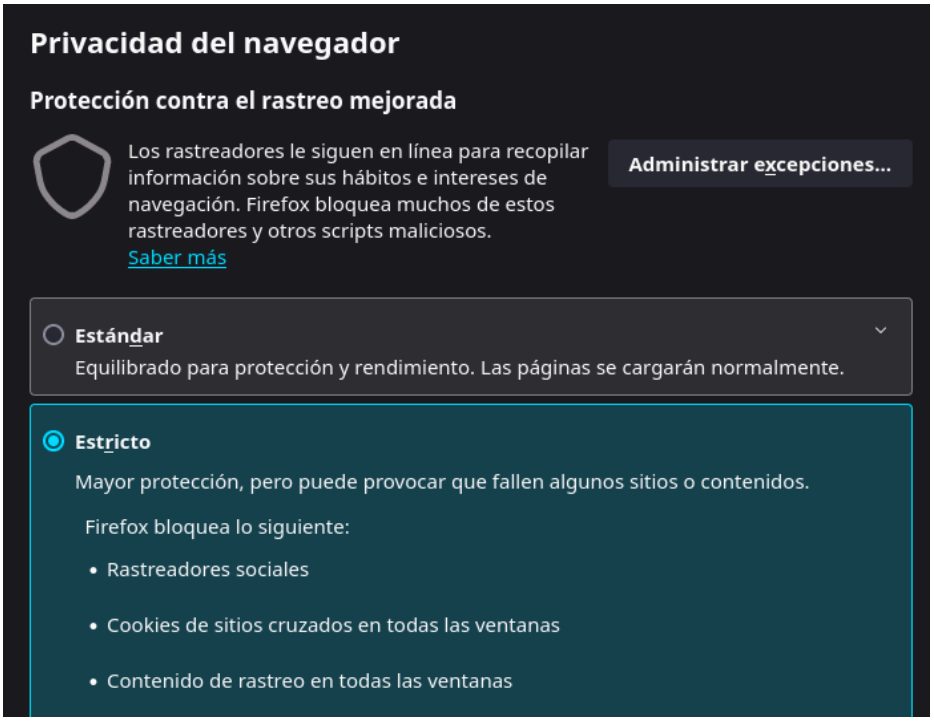

Figura 18.Protección contra el rastreo.

La **Figura 18** nos muestra la opción más segura para evitar rastreos o *trackers*. Estos rastreadores tratan de registrar nuestra huella digital, en muchas ocasiones, para generar un perfil concreto con nuestras preferencias y enfocar un *advertising* particular para nosotros y así tener una publicidad con mayor afinidad, buscando nuestra atención. Para aplicaciones seguras y con pocos anuncios se puede emplear el modo estricto, aunque si diese pie a malas visualizaciones de ciertas páginas, se podría establecer el modo *personalizado* indicando las webs a las que queremos dejar rastrearnos

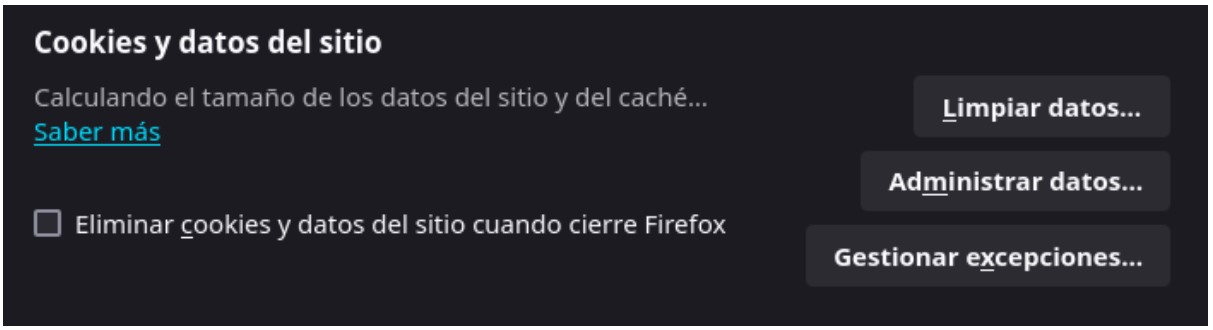

Figura 19.Cookies y datos del sitio.

Otro aspecto fundamental a la hora de realizar comunicaciones a través de una aplicación web es el permitir o no el uso de ***cookies*** del sitio. Para una comunicación segura, recomendamos no solo limpiar todas las posibles *cookies* almacenadas en nuestro cliente, sino prohibir en la medida de lo posible que las webs tomen nuestros a través de las *cookies*, pues no es nada nuevo el gran mercado de los conocidos *data brokers* que hacen empleo de los datos de estas cookies para venderlos en el

mercado negro; o en otros casos, hacer un secuestro de sesión mediante esta y hacerse pasar por nosotros, vulnerando por completo la seguridad de la comunicación.

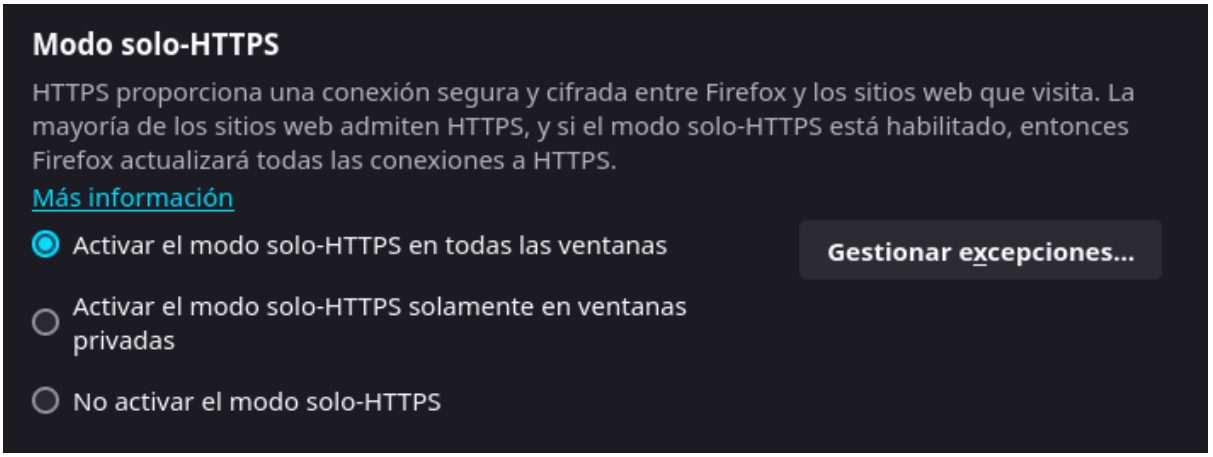

Figura 20.Modo solo HTTPS.

En la **Figura 20** podemos marcar la opción de activar el modo HTTPS en todas las ventanas. No es una mala práctica, pero tampoco es la mejor, pues si nuestro equipo ya se encuentra infectado con algún *malware,* el cibercriminal podría falsificar el *Certificado de Autoridad* y escuchar nuestra conversación, pues él tendría el cifrado correspondiente de los paquetes que se envían en el proceso de comunicación y podría descifrarlos.

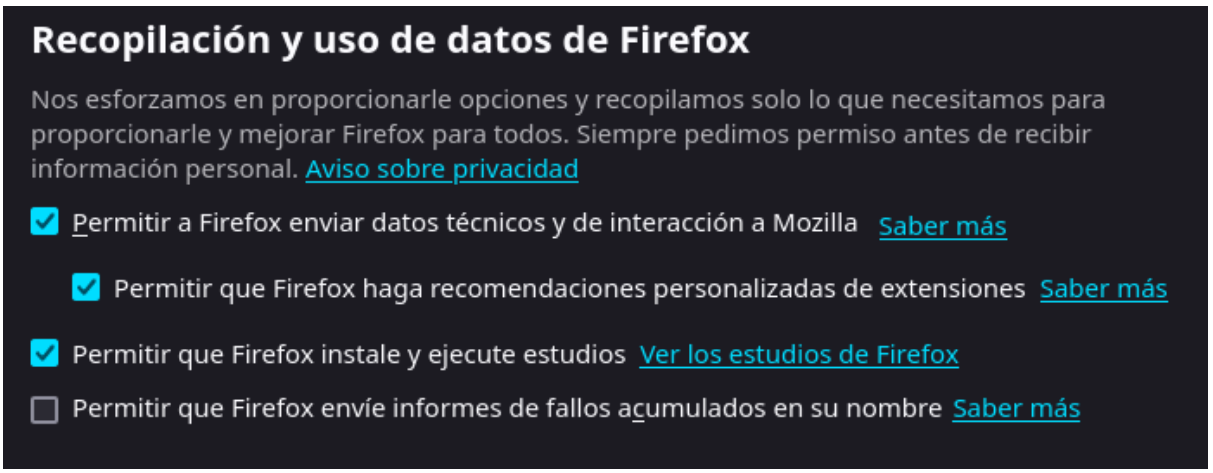

Figura 21. Recopilación de datos de Firefox.

Al igual que muchas otras aplicaciones, Firefox recopila datos de nuestras sesiones para mejorar el funcionamiento de la aplicación. Esto puede ser un problema cuando empleamos una aplicación que aparentemente inspira fiabilidad pero que en la práctica hace mal uso de los datos recopilados, como por ejemplo vender dichos datos a terceros. Una buena práctica sería desmarcar todas las opciones de este menú evitando que se recopilen datos de nuestra navegabilidad.

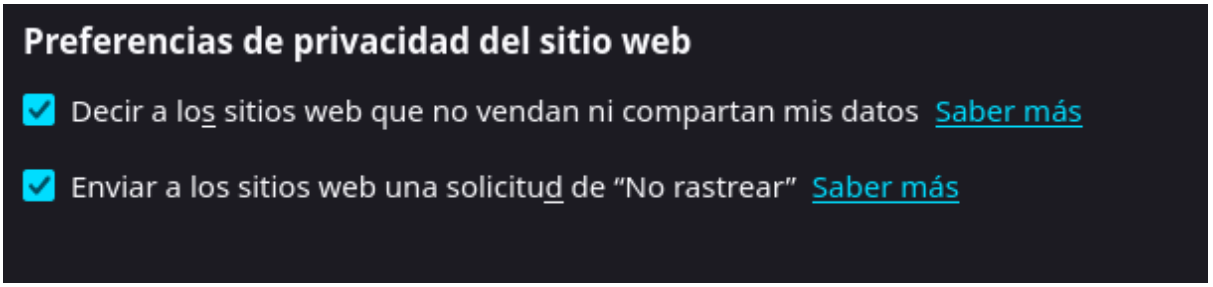

Figura 22.Privacidad del sitio web.

Para terminar las acciones que nos permiten securizar nuestras conversaciones a través del *browser* de *Firefox,* podemos marcar estas dos casillas que aparecen en la **Figura 22**. Evidentemente no se garantiza que no nos rastreen por marcar dicha casilla, ni tampoco que se compartan mis datos, pero sí que es cierto que muchas webs actuales cuentan con una moral y ética bastante férrea en cuanto a la privacidad de sus webs, por lo que marcar estas casillas ayuda a indicar a determinadas webs que no nos *trackeen* ni tampoco compartan mis datos. Estas dos opciones se han dejado para el final al ser mucho menos importantes que todas las anteriores mencionadas.

Con todos los pasos indicados debería de disponer de un navegador con las funciones mínimas a la hora de cargar ciertos elementos webs y garantizar una buena privacidad y seguridad de los datos frente a la mayoría de webs por las que realice comunicaciones.

Tarea 2

Para la **tarea 2** se han comprobado uno por uno los dispositivos móviles del Security Team para asegurarse de que cumplen con la **Política de Seguridad** de la empresa. Para alcanzar el hardening necesario se ha hecho uso de la herramienta ***CONAN mobile***. A continuación se detalla el estado de cada uno de los teléfonos de los miembros del Security Team.

Teléfono 1

Para comenzar, se usó la herramienta *CONAN mobile* para comprobar la seguridad del dispositivo, obteniendo **una** única advertencia. Esta se trataba de la necesidad de habilitar el administrador de dispositivos del móvil. Siguiendo las indicaciones de la aplicación se ha activado, permitiendo así la geolocalización del dispositivo en caso de robo o extravío. A continuación, se han comprobado uno por uno los puntos de la Política de Seguridad de la empresa.

1. La instalación de software de orígenes desconocidos se encontraba ya **deshabilitada**, tanto para el sistema como para todas las aplicaciones.
2. El dispositivo **no está rooteado ni jailbreak**, ya que habría sido detectado por *CONAN mobile*.
3. El teléfono se encontraba en una versión software anticuada(**MIUI 13**). Se ha procedido a actualizarlo a la versión más reciente(**MIUI 14.0.6 con Android 13**).
4. El smartphone no está conectado a ninguna red Wifi que tenga baja seguridad ni SSID oculta, siendo la **única** red Wifi a la que se conecta la del domicilio del miembro del Security Team.
5. El Bluetooth se encuentra **desactivado siempre**, sin excepción.

6. Ninguna de las aplicaciones instaladas en el dispositivo son maliciosas, habiendo sido todas instaladas desde la plataforma de distribución digital de aplicaciones móviles **Play Store**.
7. **No hay** ninguna conexión abierta de aplicaciones consideradas sospechosas, puesto que **no hay** aplicación sospechosas instaladas en el dispositivo.
8. **Ninguna aplicación** tiene habilitados permisos para la lectura de información de lo que envía y descarga el browser.

Teléfono 2

Al igual que en el caso anterior, se ha usado CONAN mobile para comprobar la seguridad del dispositivo, aunque esta vez no se ha recibido ningún aviso. También se ha procedido a comprobar que el dispositivo cumpla con los puntos de la Política de Seguridad.

1. La instalación de software de orígenes desconocidos se encontraba ya **deshabilitada**, tanto para el sistema como para todas las aplicaciones.
2. El dispositivo **no está rooteado ni jailbreak**, ya que habría sido detectado por *CONAN mobile*.
3. El teléfono se encontraba en la última versión disponible del software(**Android 10**).
4. El smartphone no está conectado a ninguna red Wifi que tenga baja seguridad ni SSID oculta, siendo la **única** red Wifi a la que se conecta la del domicilio del miembro del Security Team.
5. El Bluetooth se encuentra **desactivado siempre**, sin excepción.
6. Ninguna de las aplicaciones instaladas en el dispositivo son maliciosas, habiendo sido todas instaladas desde la plataforma de distribución digital de aplicaciones móviles **Play Store**.
7. **No hay** ninguna conexión abierta de aplicaciones consideradas sospechosas, puesto que **no hay** aplicación sospechosas instaladas en el dispositivo.
8. **Ninguna aplicación** tiene habilitados permisos para la lectura de información de lo que envía y descarga el browser.

Teléfono 3

Por último, se ha repetido el mismo procedimiento con el tercer dispositivo. Al igual que en el caso anterior, CONAN mobile no ha lanzado ninguna alerta.

1. La instalación de software de orígenes desconocidos se encontraba ya **deshabilitada**, tanto para el sistema como para todas las aplicaciones.
2. El dispositivo **no está rooteado ni jailbreak**, ya que habría sido detectado por *CONAN mobile*.
3. El teléfono se encontraba en la última versión disponible del software(**Android 8.0.0**).
4. El smartphone no está conectado a ninguna red Wifi que tenga baja seguridad ni SSID oculta, siendo la **única** red Wifi a la que se conecta la del domicilio del miembro del Security Team.
5. El Bluetooth se encuentra **desactivado casi siempre**, siendo la única excepción para conectar unos auriculares Bluetooth y escuchar música desde una aplicación segura(**Spotify**), pero nunca para conectarse a una web.
6. Ninguna de las aplicaciones instaladas en el dispositivo son maliciosas, habiendo sido todas instaladas desde la plataforma de distribución digital de aplicaciones móviles **Play Store**.
7. **No hay** ninguna conexión abierta de aplicaciones consideradas sospechosas, puesto que **no hay** aplicación sospechosas instaladas en el dispositivo.
8. **Ninguna aplicación** tiene habilitados permisos para la lectura de información de lo que envía y descarga el browser.

Así queda demostrado que la seguridad de los teléfonos móviles de los miembros del Security Team **cumple con la Política de Seguridad** de la empresa, pudiendo así acceder a sus servicios de comercio electrónico.

Tarea 3

Para la realización de esta tarea se usará la herramienta ZAP (**Zed Attack Proxy**) para comprobar la seguridad de aplicaciones web. En este documento se comentará brevemente sobre la configuración de la misma y como el cliente puede realizar los análisis propios (**DAST**), cómo interpretarlos y por último se procederá a enseñar las pruebas realizadas.

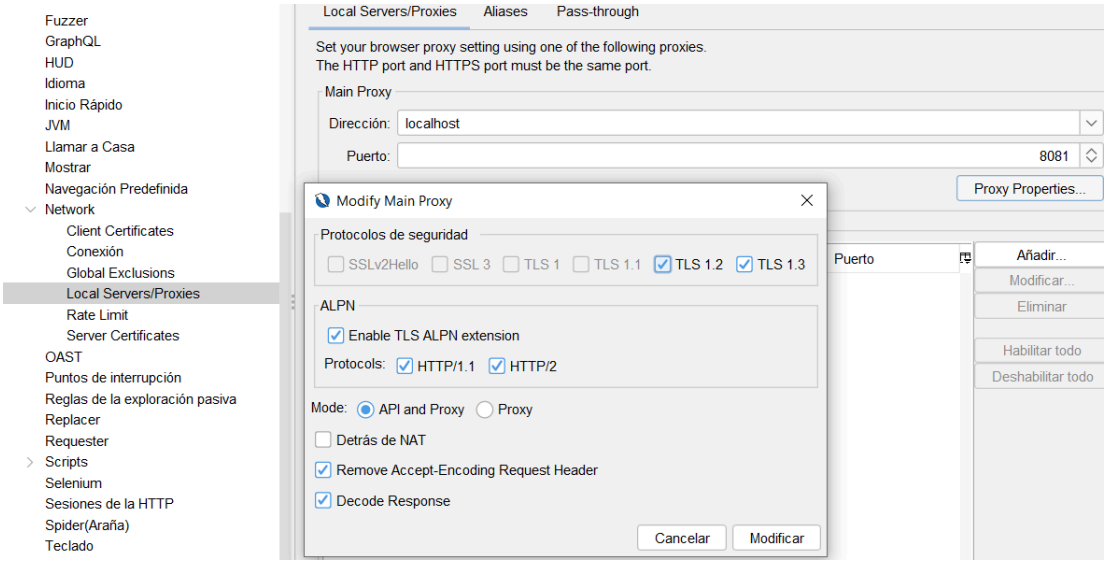

Figura 23. Configuración del proxy en ZAP

Para comenzar, se tratará brevemente la **configuración** del proxy de ZAP. Para comenzar configuraremos el proxy en *localhost:8081* con **ZAP en modo ataque** y marcaremos que acceda al tráfico TLS disponible.

Después configuraremos el navegador que usamos para que utilice el proxy de ZAP, introduciendo el certificado digital que podemos descargar de la opción *"server certificates"* de ZAP e introduciendo los datos del servidor proxy en el navegador. Para esta prueba se usará el **navegador Firefox.**

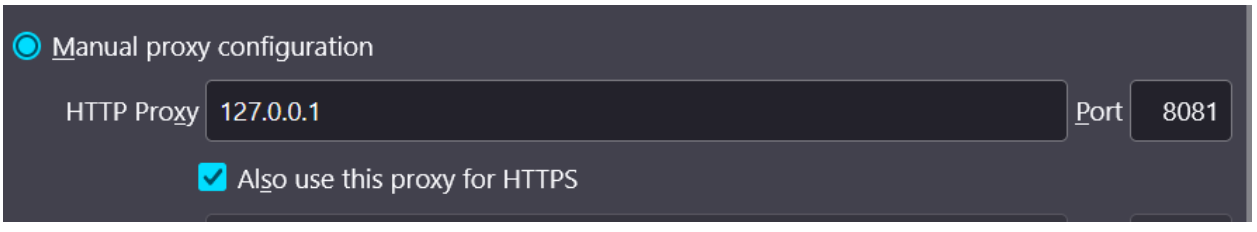

Figura 24. Configuración del proxy en Firefox

Con esto nuestro servidor proxy ya funcionando, podremos ver que ZAP empieza a analizar y enseñar las distintas peticiones y respuestas que hagamos con el navegador.

Figura 25. Ejemplo de petición web realizada desde una aplicación de desarrollo local e interceptada por el proxy de ZAP.

Ahora procederemos a realizar las **pruebas dinámicas**, para ello, puede elegir la opción de realizar las pruebas de manera automática o manual, nosotros le recomendamos que usará el **escáner automático** que será capaz de realizarlo más rápidamente y es eficaz para encontrar las rutas.

Al realizar el escáner podrá comprobar como ZAP hace peticiones a la aplicación y busca alertas y vulnerabilidades adicionales. En la pestaña de historial podrá ver las peticiones realizadas, en alertas podrá ver las distintas vulnerabilidades encontradas en el escaneo. A continuación le mostramos un **ejemplo de una vulnerabilidad encontrada** en una aplicación de pruebas la cual no está soportando token CSRF para las peticiones.

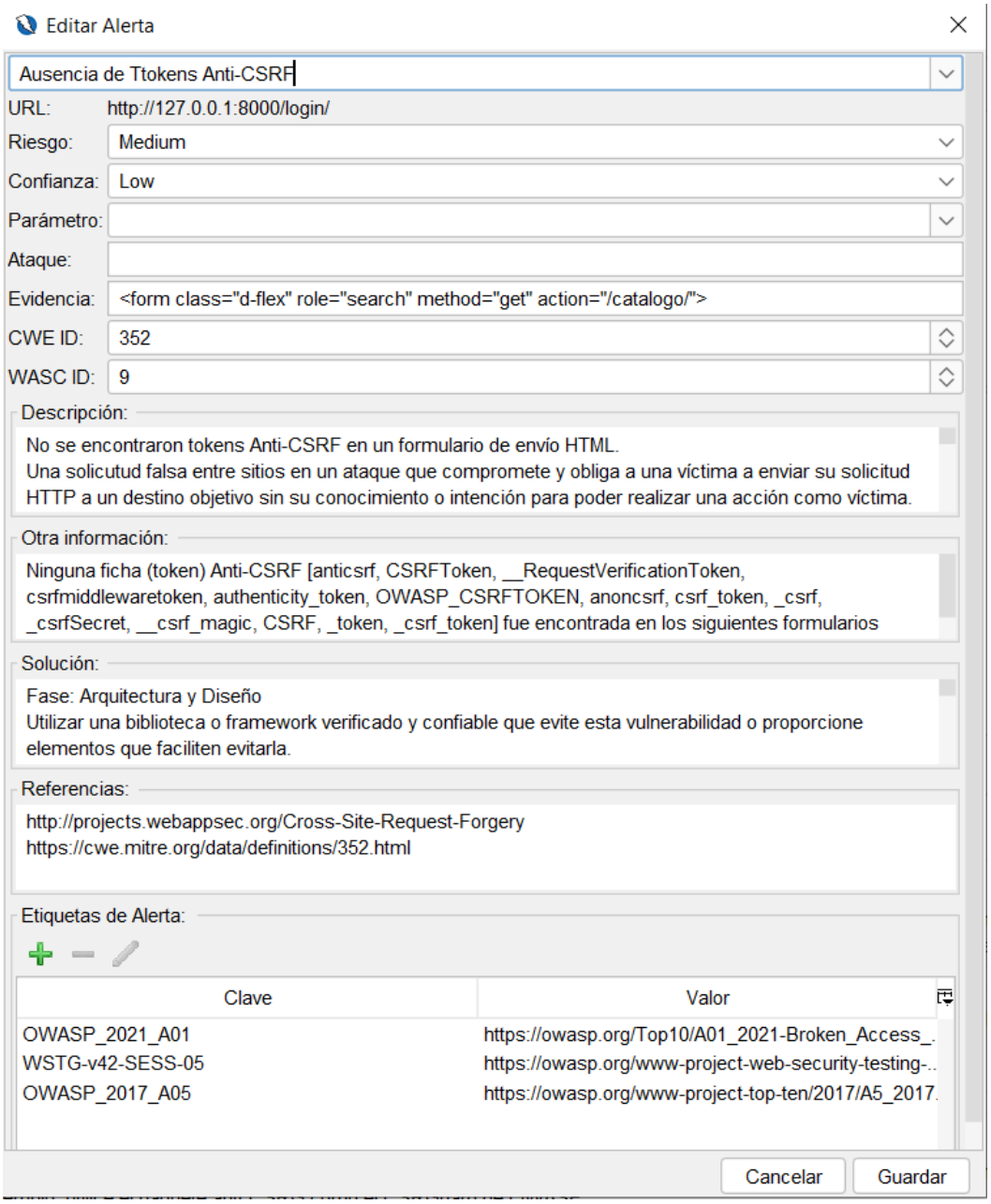

Figura 26. Ejemplo de alerta media de ZAP, te entrega campos de descripción, te sugiere soluciones, y en la línea "evidencia" es la línea donde encuentra la vulnerabilidad en la aplicación.

Por cada petición podrá comprobar su respuesta a través de los campos **petición/respuesta de ZAP** y así confirmar que no se ha modificado ninguna de las entradas y también podrá observar en la figura 27 un ejemplo de petición con el añadido de **cookies de sesión**.

A continuación se mostrará un análisis realizado a una aplicación propia de INSEGUS para las pruebas con escaneo de ZAP en el apartado tarea 4.

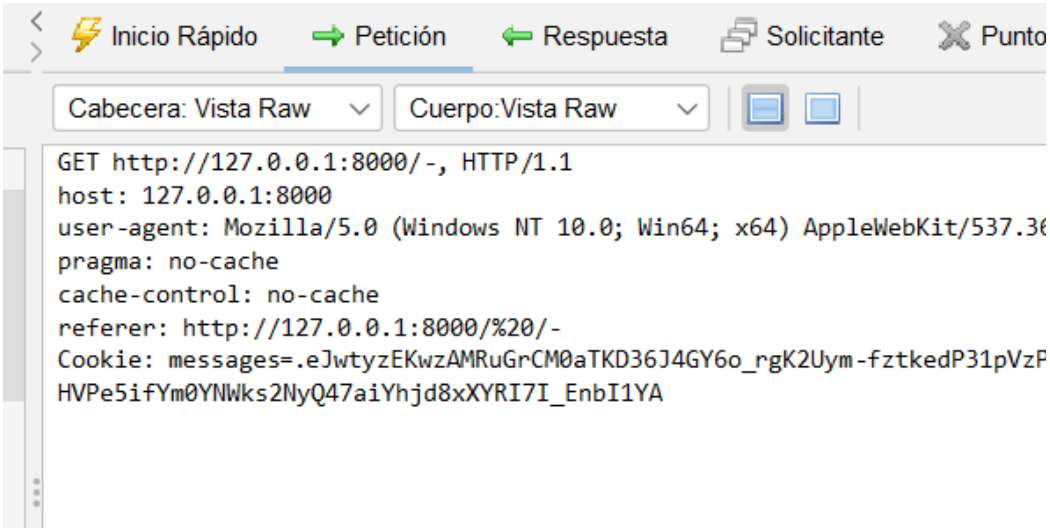

Figura 27. Ejemplo de petición HTTP en ZAP, podrá ver su respuesta a través de la pestaña respuesta.

Tarea 4

Análisis de vulnerabilidades Web para Servidor de Pruebas

Para completar esta tarea se ha hecho uso de la herramienta de ***OWASP ZAP*** sobre ***ETSIIMarkt***, una web creada en anteriores asignaturas por parte de uno de los integrantes del Security Team.

En concreto se ha usado un ***Fuzzer*** el cual se ha configurado para explorar todas las posibles vulnerabilidades del sistema**.**

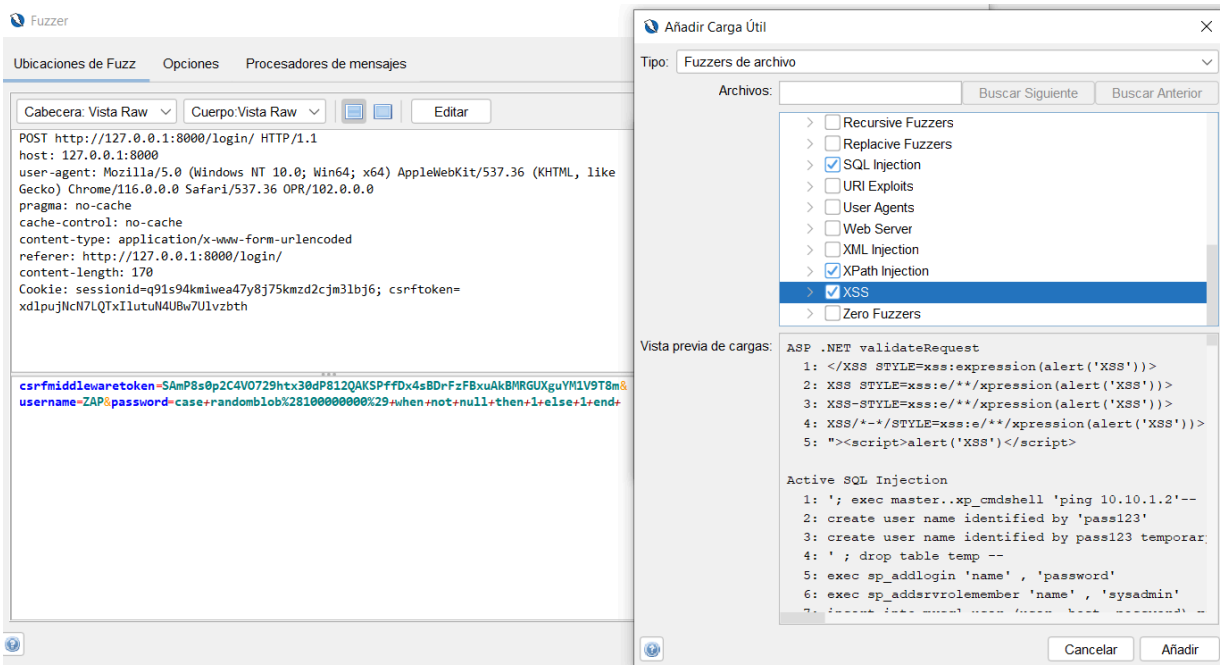

Figura 28. Configuración del Fuzzer.

Se han detectado **varias vulnerabilidades**. Podrá encontrar el informe proporcionado por ***OWASP ZAP*** en el ***Anexo 1***. Comentaremos las vulnerabilidades encontradas en materia de inyecciones SQL, Path/Directory Traversal y XSS reflejado y almacenado.

Inyecciones SQL

OWASP ZAP ha encontrado **4 vulnerabilidades** de inyecciones SQL, teniendo todas ellas un **riesgo alto** y una **confianza media**. Han sido encontradas en las páginas de *login*, *producto/detalle* y *seguimiento*.

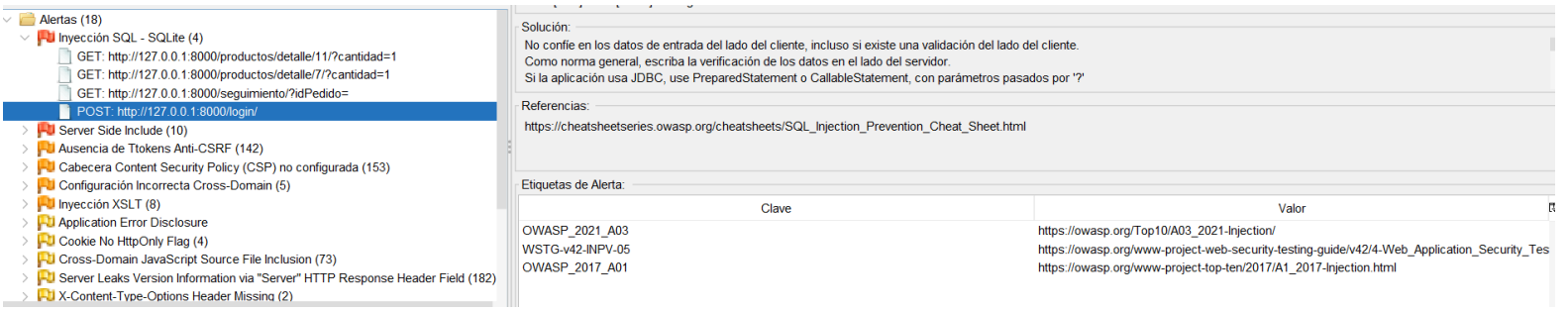

Figura 29. Inyección SQL encontrada por OWASP ZAP.

En los 4 casos se incumple la **posición A1 de OWASP 2017**. Siguiendo las direcciones de OWASP ZAP, la forma de solucionar estos fallos sería añadir **validación por parte del servidor**.

Path/Directory Traversal

OWASP ZAP no ha encontrado ninguna vulnerabilidad de tipo Path/Directory Traversal en la aplicación.

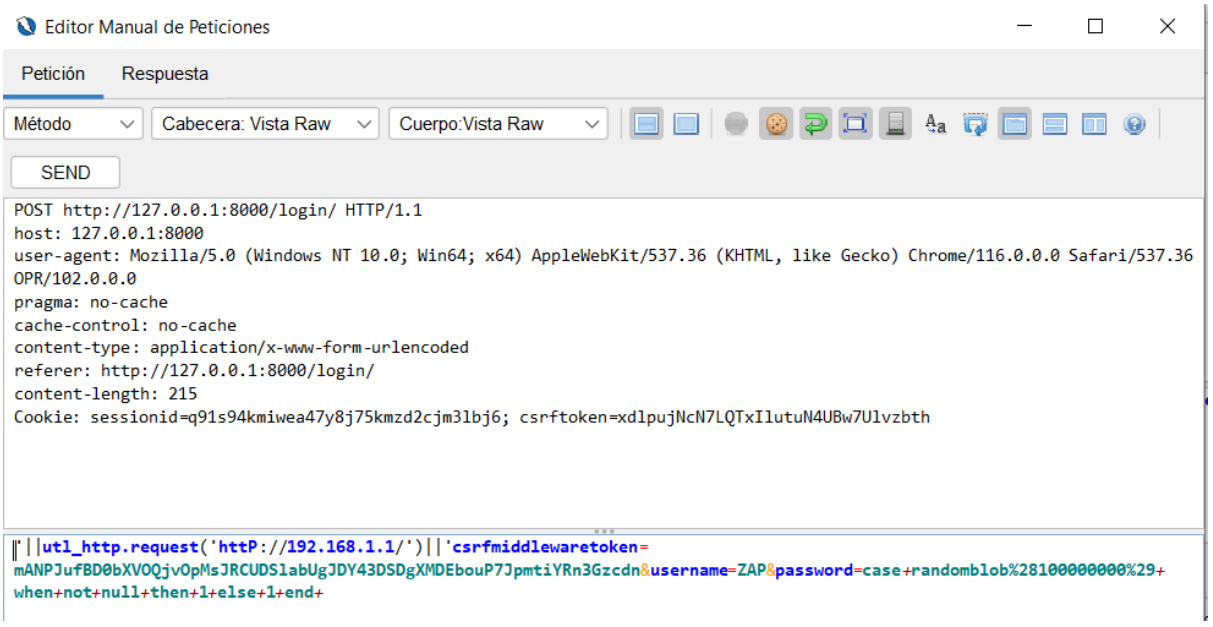

Figura 30. Intento de *OWASP ZAP* de encontrar vulnerabilidades de tipo **Path/Directory Traversal**.

XSS reflejado y almacenado

OWASP ZAP no ha encontrado ninguna vulnerabilidad de tipo XSS reflejado y almacenado en la aplicación.

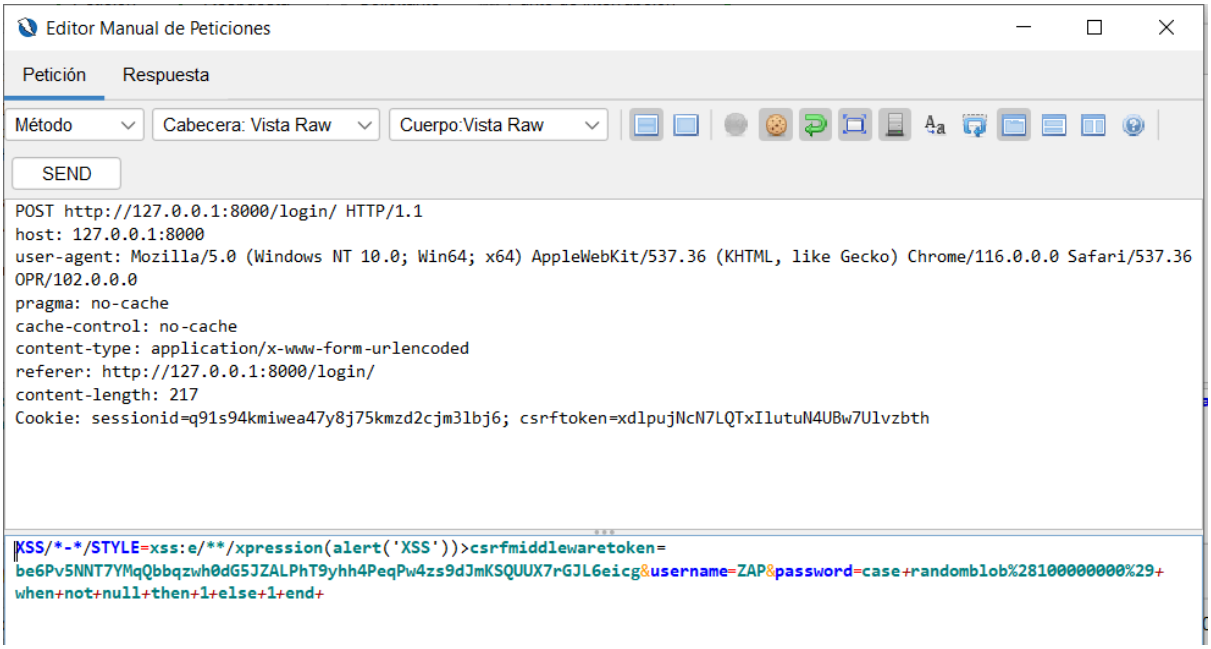

Figura 31. Intento de *OWASP ZAP* de encontrar vulnerabilidades de tipo XSS reflejado y almacenado.