Revista Ciencia UNEMI
Vol. 15, Nº 40, Septiembre-Diciembre 2022, pp. 01 - 15
ISSN 1390-4272 Impreso
ISSN 2528-7737 Electrónico
https://doi.org/10.29076/issn.2528-7737vol15iss40.2022pp1-15p

Plan de seguridad para plataformas web empleando normas ISO-27001 y considerando el OWASP top 10-2017

Álvaro Pinango-Bayas[1]; Pablo Méndez-Naranjo[2]*;
Diego Caiza-Méndez[3]; Danilo Barreno-Naranjo[4]

Resumen

La presente investigación se realizó sobre la plataforma tecnológica del Cuerpo de Bomberos del GAD Municipal de Santo Domingo con el objetivo de identificar amenazas, vulnerabilidades y definir un plan de seguridad empleando normas ISO 27001 y OWASP Top 10-2017. El proceso para las pruebas de penetración y explotación de vulnerabilidades web se lo realizó en 4 fases: recopilación de información, análisis de vulnerabilidades, explotación y generación de informes. Para las pruebas se utilizó las herramientas: Nessus, Vega, BurpSuite, BeEF, Metasploit, Synflood, Hydra y Zenmap. Como resultado se determinó que la plataforma web era vulnerable a: inyección (A1:2017), pérdida de autenticación (A2:2017), exposición de datos sensibles (A3:2017), pérdida de control de acceso (A5:2017), configuración de seguridad incorrecta (A6:2017), uso de componentes con vulnerabilidades conocidas (A9:2017), registro y monitoreo insuficientes (A10:2017); en base a esta evaluación se creó e implementó políticas y procedimientos internos de seguridad. Posteriormente se ejecutó pruebas de vulnerabilidades sobre 2 prototipos (Prototipo I: sin plan de seguridad, Prototipo II: considerando el plan de seguridad) de lo que se obtuvo un incremento en la seguridad de la plataforma tecnológica de un 75%.

Palabras clave: ISO 27001, OWASP Top 10-2017, plan de seguridad, plataforma web, vulnerabilidades.

Security plan for web platforms using ISO 27001 standards and considering the OWASP top 10-2017

Abstract

The present investigation was carried out on the technological platform of the Fire Department of the Municipal GAD of Santo Domingo with the objective of identifying threats, vulnerabilities and defining a security plan using ISO 27001 and OWASP Top 10-2017 standards. The process for penetration testing and exploitation of web vulnerabilities was carried out in 4 phases: information gathering, vulnerability analysis, exploitation and report generation. For the tests, the following tools were used: Nessus, Vega, BurpSuite, BeEF, Metasploit, Synflood, Hydra and Zenmap. As a result, it was determined that the web platform was vulnerable to: injection (A1:2017), loss of authentication (A2:2017), exposure of sensitive data (A3:2017), loss of access control (A5:2017), configuration incorrect security (A6:2017), use of components with known vulnerabilities (A9:2017), insufficient logging and monitoring (A10:2017); Based on this evaluation, internal security policies and procedures were created and implemented. Subsequently, vulnerability tests were carried out on 2 prototypes (Prototype I: without a security plan, Prototype II: considering the security plan) from which an increase in the security of the technological platform of 75% was obtained.

Keywords: ISO 27001, OWASP Top 10-2017, security plan, web platform, vulnerabilities.

Recibido: 17 de febrero de 2022
Aceptado: 15 de junio de 2022

[1] Ingeniero en Sistemas y Computación – Magíster en Seguridad Telemática. Escuela Superior Politécnica de Chimborazo. apinango@lalytto.com; https://orcid.org/0000-0002-9281-3046

[2] Ingeniero en Sistemas Informáticos – Magíster en Seguridad Telemática. Universidad Nacional de Chimborazo; pmendez@unach.edu.ec ; https://orcid.org/0000-0002-3967-3718

[3] Ingeniero en Electrónica Telecomunicaciones y Redes - Magíster en Seguridad Telemática; Universidad Nacional de Chimborazo; gustavo.caiza@unach.edu.ec ; https://orcid.org/0000-0002-9970-8193

[4] Ingeniero en Sistemas – Magíster en Informática Aplicada; Universidad Estatal de Bolívar; danilobarreno@gmail.com ; https://orcid.org/0000-0001-7557-4453

Autor de correspondencia: * pmendez@unach.edu.ec

I. INTRODUCCIÓN

La dependencia de la tecnología en las organizaciones no solo trae beneficios para las empresas hay que considerar medidas de Ciberseguridad, ya que estarán expuestas a una gran cantidad de amenazas que de aprovechar sus vulnerabilidades podrían comprometer seriamente sus activos de información. (Santiago & Allende, 2017)

En vista de la constante evolución de la tecnología las organizaciones migran sus aplicaciones a la web con la finalidad de brindar un mejor servicio a los clientes, sin embargo, estos sistemas pueden estar en riesgo; debido a esto se busca implementar lineamientos de seguridad en todos los activos de información para precautelar la confidencialidad, disponibilidad e integridad de los sistemas y de la data que almacenan garantizando su valor y reputación. (AlGhamdi & Vlahu, 2020) (Ladino, Villa, & López, 2011)

La Organización Internacional de Estandarización (ISO), a través de las normas recogidas en ISO / IEC 27000, establece una implementación efectiva de la seguridad de la información empresarial desarrolladas en las normas ISO 27001 / ISO 27002. Para preservar la información, no es suficiente la implementación de controles y procedimientos de seguridad realizados frecuentemente con criterio común, en torno a la compra de productos técnicos y sin considerar toda la información esencial que se debe proteger. (Normas ISO, 2020)

OWASP (Open Web Application Security Project) es un proyecto de código abierto cuyo propósito está dedicado a la búsqueda y la lucha contra las vulnerabilidades en el software. (OWASP, 2021). El OWASP Top 10-2017 se basa principalmente en el envío de datos de más de 40 empresas que se especializan en seguridad de aplicaciones y una encuesta de la industria completada por más de 500 personas. Esta información abarca vulnerabilidades recopiladas de cientos de organizaciones y más de 100.000 aplicaciones y APIs del mundo real. Las 10 principales categorías son seleccionadas y priorizadas de acuerdo con estos datos de prevalencia, en combinación con estimaciones consensuadas de explotabilidad, detectabilidad e impacto. (OWASP, 2017)

La plataforma tecnológica del Cuerpo de Bomberos del GAD Municipal de Santo Domingo (CB-GADM-SD) no cuenta con políticas y procedimientos de seguridad estandarizados por lo que está expuesta ante amenazas y vulnerabilidades.

Por lo que, el objetivo de la presente investigación es mejorar el nivel de seguridad de la plataforma tecnológica estableciendo un plan de seguridad para la mitigación de las vulnerabilidades más potenciales que pueden afectar a la disponibilidad, confidencialidad e integridad de los servicios en línea que ofrece la Organización, utilizando las normas ISO 27001 y considerando la propuesta de OWASP Top 10-2017.

Se han realizado investigaciones relacionadas las cuales tratan parcialmente más no comprenden la temática propuesta, dentro de esos trabajos se pueden resaltar a:

- La investigación realizada por Ballen, Ayala y Sierra (2017) en la cual los autores realizan pruebas de SQL injection tomando como base el listado de OWASP Top Ten 2013 que considera a esta vulnerabilidad como la primera. Utilizan la herramienta "Zed Attack Proxy" para determinar de forma específica las vulnerabilidades de la aplicación. Esta investigación se limita a recopilar la documentación de la técnica en base a el listado del OWASP-2013, el análisis de una aplicación web y a proponer recomendaciones relacionadas únicamente con SQL injection.
- La investigación realizada por Rojas-Osorio, Medina-Cárdenas y Bautista (2016), en la cual los autores realizan pruebas de penetración utilizando técnicas de hacking ético en redes que utilizan el protocolo IP, empleando como sistema operativo BackTrack para determinar las vulnerabilidades en estos sistemas. Esta investigación se limita a pruebas de laboratorio desarrolladas en una red de área local sobre los protocolos IPv4 e IPv6 (configurada en ambientes controlados).
- La investigación de Solarte, Rosero y Benavides (2015), en la cual los autores implementan sistemas de seguridad de la

información y consideran la metodología de Magerit para el análisis y gestión de riesgos. En su propuesta aplican etapas de auditoría, análisis y evaluación de riesgos. Como resultado de la investigación proponen controles de la seguridad que puedan incrementarla en base de sus necesidades. Esta investigación se enfoca al uso de una metodología diferente a la familia de las normas ISO 27000 específicamente para el análisis y evaluación de riesgos y no se incluye el análisis de las vulnerabilidades potenciales de acuerdo con el listado OWASP-2017.

II. DESARROLLO

La metodología utilizada fue la diseñada por Toledo-Díaz (2014), la cual abarca 4 fases para realizar las pruebas de penetración y explotación de las vulnerabilidades web, las herramientas que se utilizaron en cada una de ellas se resumen en la Tabla 1.

Tabla 1. Resumen de las fases de pruebas de penetración y explotación

Fases	Descripción	Herramienta
Fase 1: Recopilación de información (rastreo y exploración)	En esta etapa se obtiene información de la plataforma web de la organización utilizando herramientas de pentesting	• Kali Linux • Zenmap
Fase 2: Análisis de vulnerabilidades (enumeración)	En esta etapa se realizan pruebas para identificar recursos y servicios específicos	• Kali Linux • Zenmap • Nessus • Vega • Chrome • Mozilla Firefox
Fase 3: Explotación (acceso, escalada de privilegios, daño, borrado de huellas)	En esta etapa se obtiene el acceso no autorizado a los recursos o servicios identificados.	• Kali Linux • Burn Site • Wireshark • Metasploit • Synflood • Hydra • SQLmap • BeEF • Chrome • Mozilla Firefox
Fase 4: Generación de informes	En esta etapa se informa las acciones y pruebas que se han realizado, las técnicas y herramientas utilizadas con su respectivo nivel de gravedad para la organización y la propuesta del plan.	• Procesador de texto

En la Tabla 2 se describe de forma resumida las herramientas utilizadas en la investigación.

Tabla 2. Resumen de las herramientas utilizadas

Sistema operativo	Escaneo y detección de vulnerabilidades	Explotación de vulnerabilidades	Sniffer	Navegadores web
Kali Linux: destinada a pruebas de penetración y auditoría. (Kali linux, 2021)	**Zenmap:** escaneo de puertos (Zenmap, 2021)	**SQLmap:** explotación de vulnerabilidades de SQL injection (Sqlmap, 2021)	**Wirewhark:** analizador de protocolos y tráfico en la red. (Wireshark, 2021)	**Google Chrome:** navegador web sin coste económico desarrollado por Google.(Google, 2021)
	Nessus: escaneo de vulnerabilidades (Tenable, 2021)	**Hydra:** pruebas con cuentas de usuario por SSH bajo fuerza bruta. (Cortez, 2017)		
	Burp Suite: pruebas de seguridad en aplicaciones (PortSwigger, 2021)	**Metasploit:** información de vulnerabilidades de seguridad. (Rapid7, 2021)		

Vega: escaneo de vulnerabilidades de aplicaciones. (Subgraph, 2021)	**Synflood con metasploit:** inundación SYN para denegación de servicios. (Comando It, 2021)
	BeEF: pruebas de penetración que se centra en el navegador web. (Alcorn, 2021)

Fase 1: Recopilación de información (rastreo y exploración)

En la fase de recopilación de información se realizó la verificación física de la infraestructura, se utilizó la herramienta Zenmap, para determinar el esquema de red local de la Organización; como se muestra en la Figura 1.

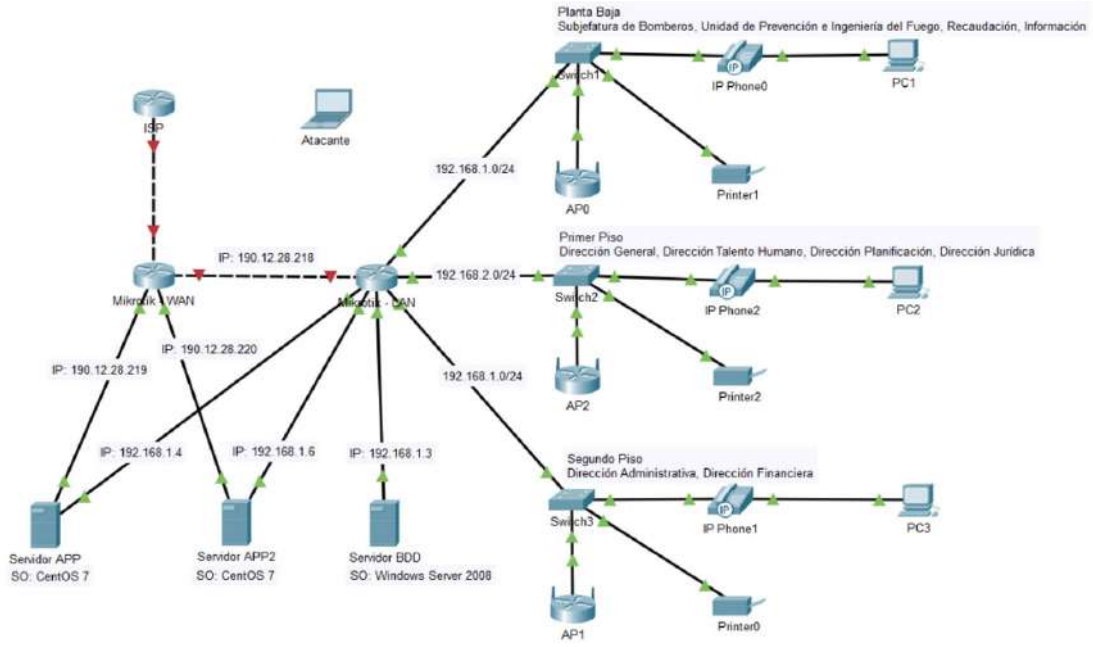

Figura 1. Infraestructura de la organización

La Tabla 3 detalla los equipos de red que forman parte de la plataforma web, la misma que se integra por los servidores de aplicaciones, servidor de base de datos y router de salida.

Tabla 3. Listado de dispositivos de la plataforma web

Dispositivo	Detalle	Función
Mikrotik WAN	Router interno Mikrotik	Administración principal de servicios de internet
Mikrotik LAN	Router de salida Mikrotik	Servidor del Firewall
Servidor de aplicaciones	Sistema operativo: CentOS 7 Apache NodeJS	Servidor principal para servicios de atención al cliente
Servidor 2 de aplicaciones	Sistema operativo CentOS 7 Apache NodeJS	Servidor de respaldo
Servidor de base de datos	Sistema operativo Windows Server 2008 PostgreSQL	Servidor de base de datos, alimenta la información a los servidores de aplicaciones

Fase 2: Análisis de vulnerabilidades (enumeración)

En la fase de análisis de vulnerabilidades se utilizaron las siguientes herramientas:

- Zenmap para el escaneo de los servicios y puertos abiertos o activos.
- Nessus, BurpSuite y Vega para el escaneo de vulnerabilidades de seguridad.

Una vez realizadas las pruebas de vulnerabilidades a la plataforma web, se determinó que es vulnerable a 8 de las 10 consideradas en el listado OWASP Top 10-2017, las cuales fueron: A1:2017 – Inyección, A2:2017 – Pérdida de autenticación, A3:2017 – Exposición de datos sensibles, A5:2017 – Pérdida de control de acceso, A6:2017 – Configuración de seguridad incorrecta, A7:2017 – Cross-Site Scripting (XSS), A9:2017 – Uso de componentes con vulnerabilidades conocidas, A10:2017 – Registro y monitoreo insuficientes.

Fase 3: Explotación (acceso, escalada de privilegios, daño, borrado de huellas)

En la fase de explotación se utilizaron las siguientes herramientas:

- Burn Suite para escaneo y detección de vulnerabilidades
- SQLmap para detección de vulnerabilidades de SQL
- Hydra para pruebas con cuentas de usuario por SSH bajo fuerza bruta
- Metasploit para explotación de vulnerabilidades
- Synflood para pruebas de denegación de servicios.
- BeEf para prueba centradas en el navegador

Pruebas realizadas para cada una de las vulnerabilidades del listado OWASP Top-2017 sobre la plataforma web:

A1:2017 - Inyección

Para la explotación de vulnerabilidades de SQL se utilizó SQLMap a través de cual se obtuvo el acceso a los registros de las tablas de la base de datos como datos de usuario, cuentas bancarias, entre otros. En la Figura 2 se muestra las propiedades de las tablas de la base de datos.

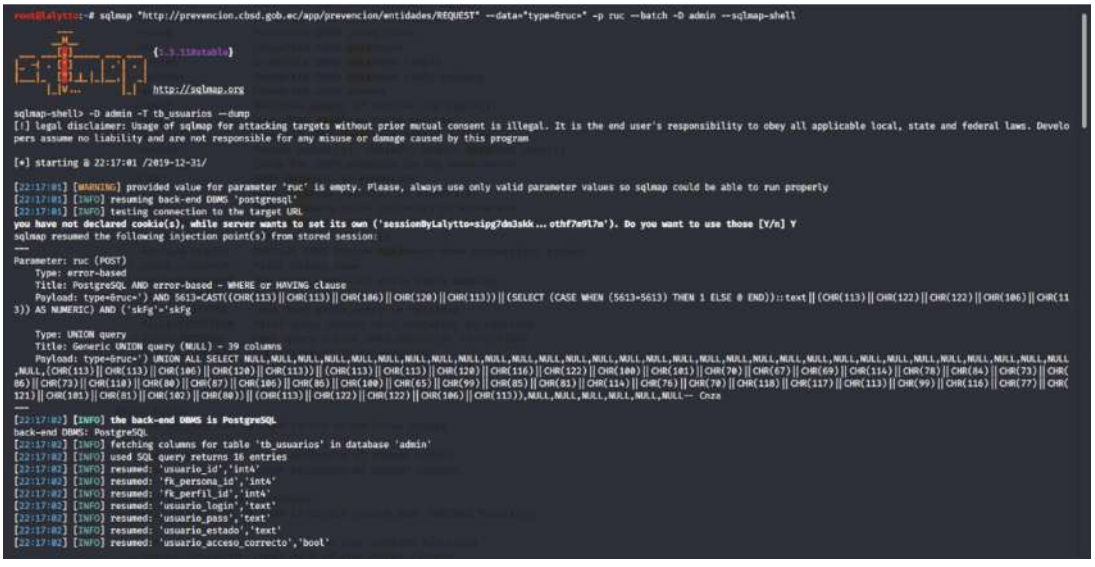

Figura 2. Propiedades de la base de datos

A2:2017 – Pérdida de autenticación

Una vez que se ha ganado el acceso a la base de datos se analizó la información de las tablas en busca de información relacionada con las credenciales de los usuarios de la Organización. Mediante SQLMap se realizaron consultas y se obtuvo la informac¬¬ión de los registros y claves descifradas que no contaban con criterios de seguridad. En la Figura 3 se muestra las claves descifradas de la tabla de usuarios.

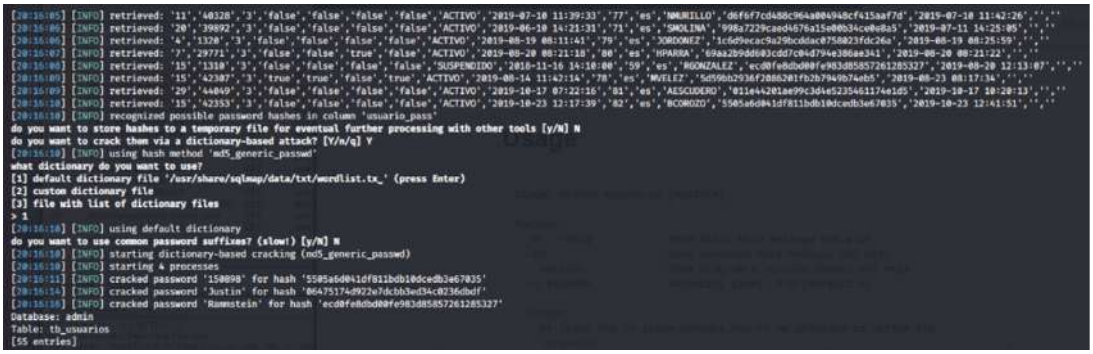

Figura 3. Descifrado de credenciales con SQLMap

Se realizaron pruebas con las cuentas de usuario por SSH utilizando fuerza bruta con ataques de diccionario de la herramienta Hydra como se muestra en la Figura 4.

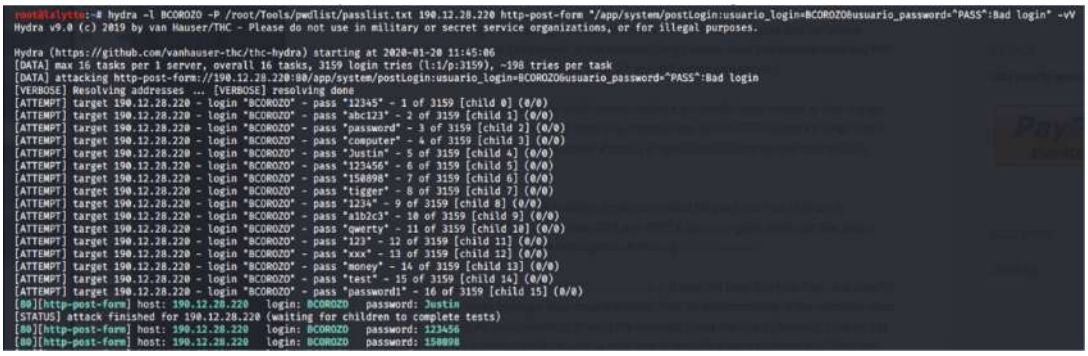

Figura 4. Obtención de contraseñas con Hydra

A3:2017- Exposición de datos sensibles

Se configuró la herramienta Burp Suite como proxy por defecto, utilizando la comunicación entre el cliente y servidor con la finalidad de obtener las credenciales de acceso a la plataforma web. En la Figura 5 se muestran los paquetes que se envían y reciben entre el cliente y el servidor.

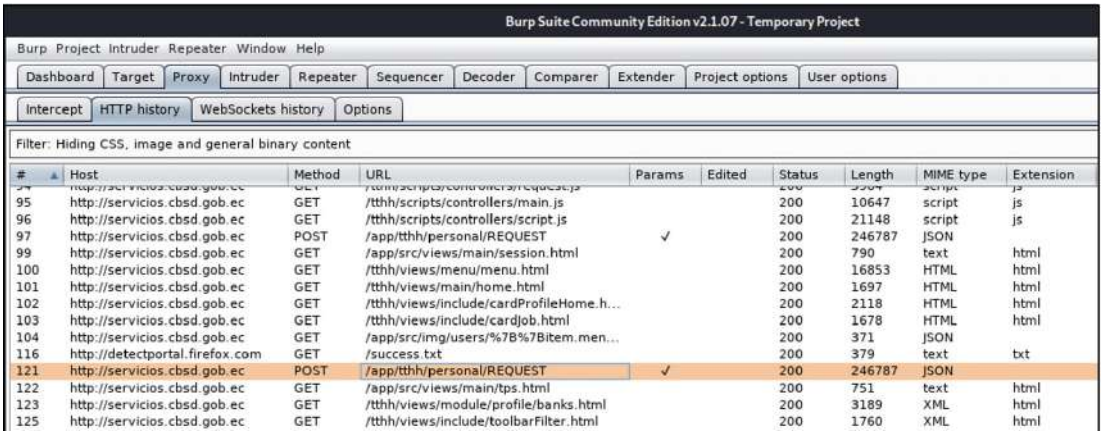

Figura 5. Paquetes interceptados con Burp Suite

En la Figura 6 se muestra la información interceptada por el atacante en texto plano, el atacante puede capturar el payload, modificarlo y enviarlo al servidor con datos alterados para un fin específico.

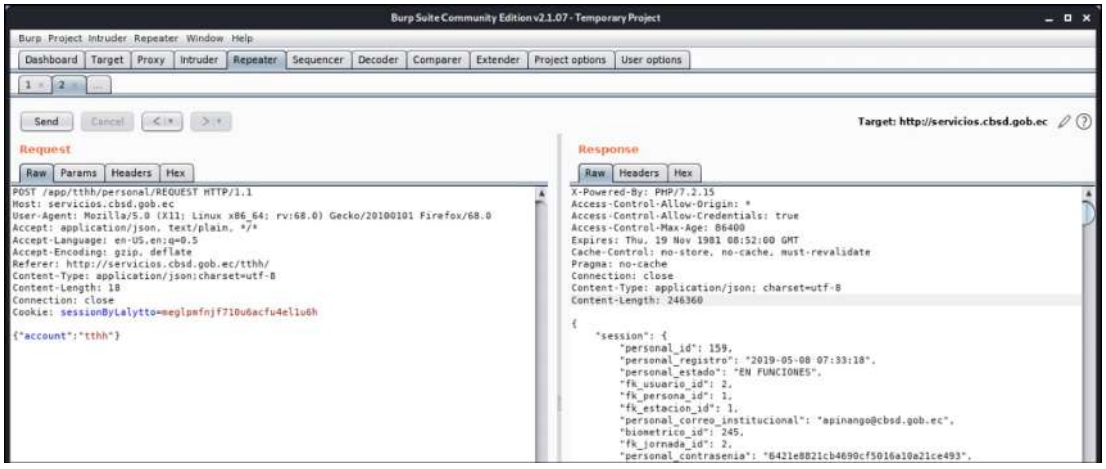

Figura 6. Alteración de sesión de usuario para obtener información sesiones creadas

A5:2017 - Pérdida de control de acceso

Utilizando Burp Suite se interceptaron los paquetes de los pedidos realizados al servidor con la finalidad de modificarlos y acceder a sitios no autorizados, debido a que la plataforma web genera consultas y navegación mediante la dirección URL. Esta herramienta permitió cambiar las direcciones URL manualmente para navegar por el sitio, de esta manera se obtuvieron las peticiones realizadas al back-end para el llenado de datos del usuario. Como se muestra en la Figura 7, se altera el URL para acceder a sitios que requieren roles de acceso específicos.

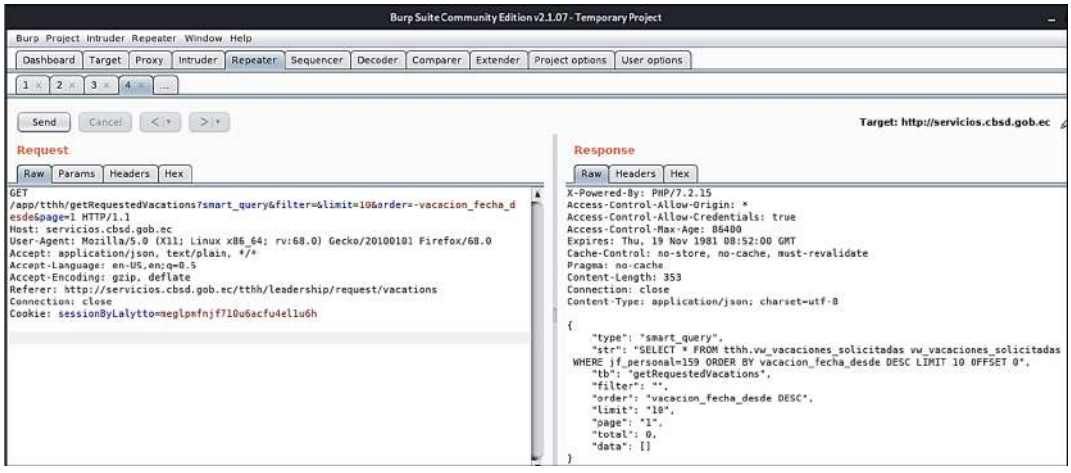

Figura 7. Paquetes interceptados con Burp Suite

A6:2017 – Configuración de seguridad incorrecta

Se generó un synflood utilizando Metasploit con el objetivo de producir una denegación de servicio (Dos), explotando la vulnerabilidad de la red al no contar con controles en su configuración como se muestra en la Figura 8 y en la Figura 9 se verifica la afectación en la disponibilidad de la plataforma web.

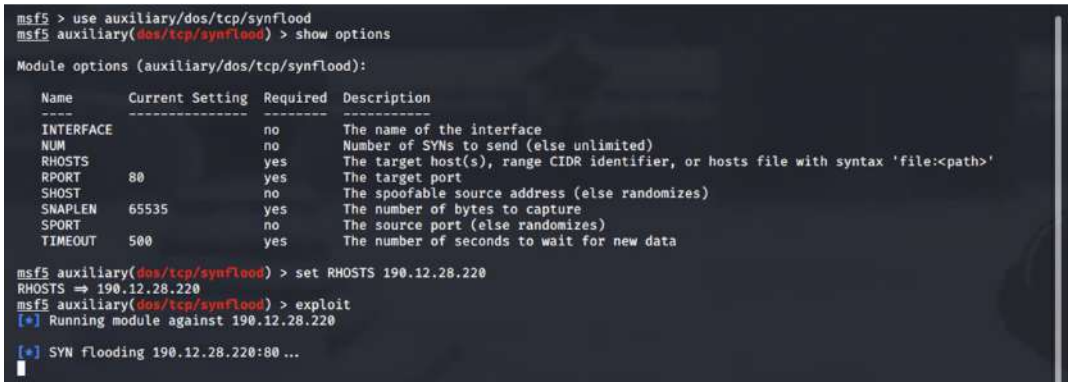

Figura 8. Ejecución de Synfood con Metasploit

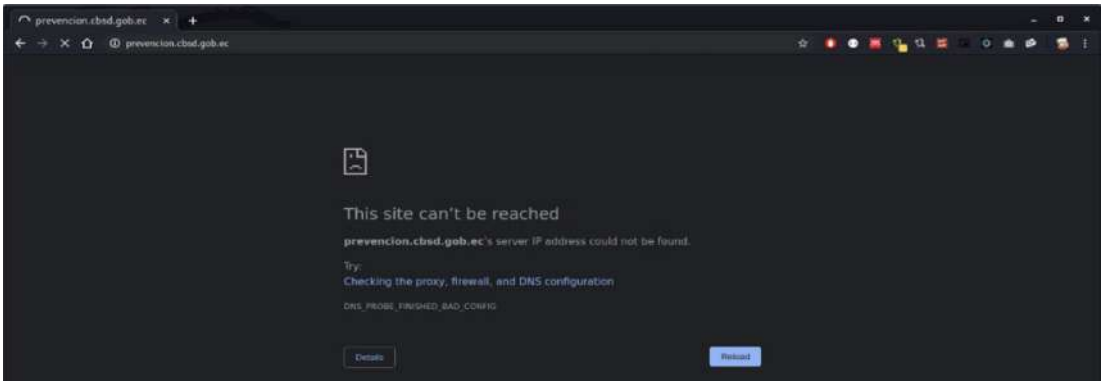

Figura 9. Falta de disponibilidad de la plataforma web

A7:2017 - Cross-Site Scripting (XSS)

Se inyectó código JavaScript, explotando la vulnerabilidad en la interfaz gráfica de un formulario con el navegador y se obtuvo información del usuario. Para verificar que se ha ingresado exitosamente el código se utilizó el inspector de elementos en el navegador visualizando la etiqueta HTML como se muestra en la Figura 10.

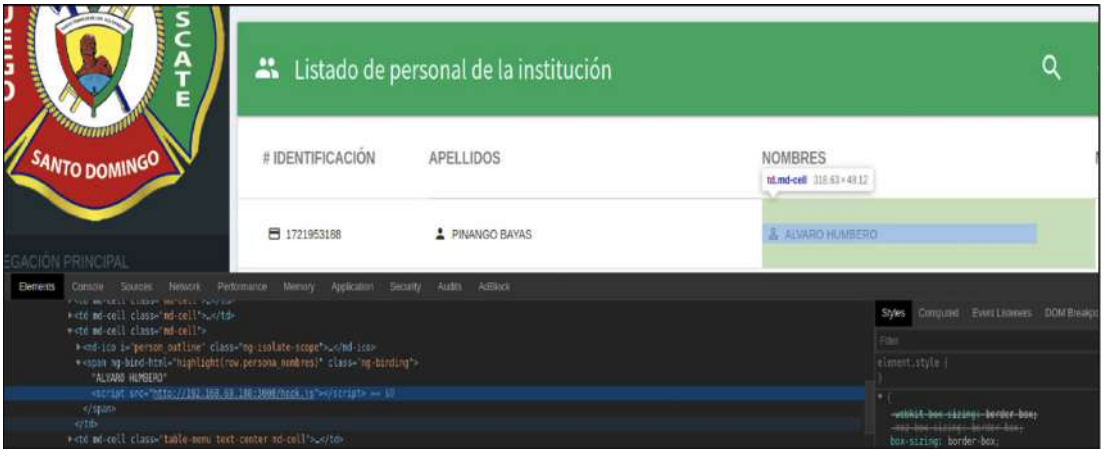

Figura 10. Verificación de código javascript ingresado

De igual forma se utilizó la herramienta BeEF para inyectar código JavaScript con la finalidad de capturar las cookies de las sesiones del navegador, como se muestra en la Figura 11.

Figura 11. Obtención de sesiones con BeEF

A9:2017 - Uso de componentes con vulnerabilidades conocidas

Bajo el servidor web Apache, se utilizó los CVE (Vulnerabilidades y exposiciones comunes) obtenidas en la etapa de escaneo y se las explotó con la herramienta Metasploit, como se muestra en la Figura 12.

```
       =[ metasploit v5.0.62-dev                          ]
+ -- --=[ 1949 exploits - 1090 auxiliary - 334 post       ]
+ -- --=[ 558 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

msf5 > use exploit/unix/webapp/twiki_history
msf5 exploit(unix/webapp/twiki_history) > show options

Module options (exploit/unix/webapp/twiki_history):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   URI      /twiki/bin       yes       TWiki bin directory path
   VHOST                     no        HTTP server virtual host
```

Figura 12. Explotación de vulnerabilidades a Apache

En la Figura 13 se realizó el ataque para acceder al servidor, se escaló privilegios para obtener más información como puertos, denegación de algún servicio, configuraciones en el servidor, además mantener una conexión backdoor para el acceso posterior.

```
msf5 exploit(unix/webapp/twiki_history) > set RHOSTS 190.12.28.219
RHOSTS => 190.12.28.219
msf5 exploit(unix/webapp/twiki_history) > set payload cmd/unix/bind_netcat
payload => cmd/unix/bind_netcat
msf5 exploit(unix/webapp/twiki_history) > show options

Module options (exploit/unix/webapp/twiki_history):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS   190.12.28.219    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   URI      /twiki/bin       yes       TWiki bin directory path
   VHOST                     no        HTTP server virtual host


Payload options (cmd/unix/bind_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LPORT  4444             yes       The listen port
   RHOST  190.12.28.219    no        The target address
```

Figura 13. Payload para explotación de vulnerabilidad en Apache

A10:2017 - Registro y monitoreo insuficientes

En el proceso de explotación de vulnerabilidades se determinó que no cuenta con políticas de seguridad ni el conocimiento específico del equipo de tecnologías por lo que no existió el registro ni el monitoreo.

Fase 4. Generación de informes

En la fase de generación de informes, se define el plan que pretende mejorar la seguridad de la plataforma web de la Organización, definiendo

políticas de seguridad mediante recomendaciones para minimizar las vulnerabilidades existentes, considerando:

- La norma ISO 27001 (INEN, 2015)
  - ◊ Adquisición, desarrollo y mantenimiento del sistema
  - ◊ Seguridad en el desarrollo y en los procesos de soporte
  - ◊ Política de desarrollo seguro
- Recomendaciones de seguridad de OWASP Top-2017

Los cuales deben ser implementados por los nuevos sistemas y por los existentes (servicios en línea, talento humano, subjefatura y administración), como se muestra en la Tabla 4.

Tabla 4. Políticas y procedimientos específicos de seguridad implementados

Política ISO 27001: Política de desarrollo seguro Control: Se deben establecer y aplicar reglas dentro de la organización para el desarrollo de aplicaciones y sistemas	
A1:2017 - Inyección	
Implicación de seguridad	Las fallas de inyección, como la inyección SQL, NoSQL, OS y LDAP, ocurren cuando se envían datos que no son de confianza a un intérprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al intérprete para que ejecute comandos no deseados o acceda a datos sin la debida autorización.
Procedimiento específico: Prevenir inyecciones	Prevenir la inyección requiere mantener los datos separados de los comandos y las consultas: • Usar una API segura, que eviten por completo el uso del intérprete. • Utilizar una interfaz parametrizada o usar herramientas de mapeo relacional de objetos (ORM). • Usar validación de entrada del lado del servidor positiva o "lista blanca" • Para consultas dinámicas, se deben saltar los caracteres especiales usando la sintaxis de escape específica. • Usar LIMIT y otros controles de SQL dentro de las consultas para evitar la divulgación masiva de registros en caso de inyección de SQL.
A2:2017 – Pérdida de autenticación	
Implicación de seguridad	Las funciones de la aplicación relacionadas con la autenticación y la gestión de sesiones a menudo se implementan incorrectamente, lo que permite a los atacantes comprometer contraseñas, tokens de sesión, o explota otras fallas de implementación para asumir las identidades de otros usuarios de forma temporal o permanente.
Procedimiento específico: Proteger los métodos de autenticación	• Siempre que sea posible, implementar la autenticación multifactor para evitar ataques automatizados, de relleno de credenciales, de fuerza bruta y de reutilización de credenciales robadas. • No enviar por medios inseguros ninguna credencial predeterminada, especialmente para usuarios administradores. • Implementar verificaciones de contraseñas débiles, como probar contraseñas nuevas o modificadas y compararlas con una lista de las 10000 peores contraseñas. • Alinear las políticas de longitud, complejidad y rotación de contraseñas. • Verificar que las rutas de registro, recuperación de credenciales y API estén protegidas contra los ataques de enumeración de cuentas mediante el uso de los mismos mensajes para todos los resultados. • Limitar o retrasar cada vez más los intentos fallidos de inicio de sesión. • Registrar todos los errores y alertar a los administradores cuando se detecten ataques de Credential Stuffing, fuerza bruta u otros. • Usar un administrador de sesión incorporado, seguro y del lado del servidor que genera una nueva ID de sesión aleatoria con alta entropía después de iniciar sesión. • Los ID de sesión no deben estar en la URL, deben almacenarse de forma segura e invalidarse después de los tiempos de espera de cierre de sesión, inactividad y absoluto.
A3:2017 – Exposición de datos sensibles	
Implicación de seguridad	Muchas aplicaciones web y API no protegen adecuadamente los datos confidenciales, como los financieros, de salud e información de identificación personal. Los atacantes pueden robar o modificar dichos datos débilmente protegidos para realizar fraudes con tarjetas de crédito, robo de identidad u otros delitos. Los datos confidenciales pueden verse comprometidos sin protección adicional, como el cifrado en reposo o en tránsito, y requieren precauciones especiales cuando se intercambian con el navegador.

Procedimiento específico: Prevenir la exposición de datos sensibles	• Clasificar los datos procesados, almacenados o transmitidos por una aplicación. Identificar los datos que son confidenciales de acuerdo con las leyes de privacidad, los requisitos reglamentarios o las necesidades comerciales. • Aplicar controles según la clasificación. • No almacenar datos confidenciales innecesariamente. • Cifrar todos los datos confidenciales en reposo. • Garantizar que se implementen algoritmos, protocolos y claves estándar sólidos y actualizados. • Utilizar una gestión de claves adecuada. • Cifrar todos los datos en tránsito con protocolos seguros como TLS con cifrados Perfect Forward Secrecy (PFS), priorización de cifrado por parte del servidor y parámetros seguros. Utilizar el cifrado mediante directivas como HTTP Strict Transport Security (HSTS). • Deshabilitar el almacenamiento en caché para la respuesta que contiene datos confidenciales. • Almacenar contraseñas utilizando funciones de hash adaptables y sólidas con un factor de trabajo (factor de demora), como Argon2, scrypt, bcrypt o PBKDF2.
A5:2017 – Pérdida de control de acceso	
Implicación de seguridad	Las restricciones sobre lo que pueden hacer los usuarios autenticados a menudo no se aplican correctamente. Los atacantes pueden explotar estas fallas para acceder a funciones y/o datos no autorizados, como acceder a las cuentas de otros usuarios, ver archivos confidenciales, modificar los datos de otros usuarios, cambiar los derechos de acceso, etc.
Procedimiento específico: Prevenir la pérdida de control de acceso	El control de acceso solo es efectivo si se aplica en un código confiable del lado del servidor o en una API sin servidor, donde el atacante no puede modificar la verificación de control de acceso o los metadatos. • Implementar mecanismos de control de acceso una vez y reutilizarlos en toda la aplicación, incluida la minimización del uso de CORS. • Los controles de acceso al modelo deben hacer cumplir la propiedad de los registros, en lugar de aceptar que el usuario pueda crear, leer, actualizar o eliminar cualquier registro. • Deshabilitar la lista de directorios del servidor web y asegurarse de que los metadatos del archivo (por ejemplo, .git) y los archivos de copia de seguridad no estén presentes en las raíces web. • Registrar fallas de control de acceso, alertar a los administradores cuando corresponda (por ejemplo, fallas repetidas). • Tasa de límite de API y acceso al controlador para minimizar el daño de las herramientas de ataque automatizado. • Los tokens JWT deben invalidarse en el servidor después de cerrar la sesión. • Los desarrolladores y el personal de control de calidad deben incluir una unidad de control de acceso funcional y pruebas de integración.
A6:2017 – Configuración de seguridad incorrecta	
Implicación de seguridad	La mala configuración de la seguridad es el problema más común. Esto suele ser el resultado de configuraciones predeterminadas inseguras, configuraciones incompletas o ad hoc, almacenamiento en la nube abierto, encabezados HTTP mal configurados y mensajes de error detallados que contienen información confidencial. No solo se deben configurar de forma segura todos los sistemas operativos, macros, bibliotecas y aplicaciones, sino que también se deben parchear/actualizar de manera oportuna.
Procedimiento específico: Evitar configuraciones de seguridad por defecto	Deben implementarse procesos de instalación seguros: • Los entornos de desarrollo, control de calidad y producción deben configurarse de manera idéntica, con diferentes credenciales utilizadas en cada entorno. Este proceso debe automatizarse para minimizar el esfuerzo requerido para configurar un nuevo entorno seguro. • Eliminar o no instalar características y macros no utilizados. • Revisar y actualizar las configuraciones apropiadas para todas las notas de seguridad, actualizaciones y parches como parte del proceso de administración de parches. • Una arquitectura de aplicación segmentada que proporciona una separación eficaz y segura entre componentes, con segmentación y grupos de seguridad en la nube (ACL). • Envío de directivas de seguridad a los clientes. • Un proceso automatizado para verificar la eficacia de las configuraciones y ajustes en todos los entornos.

A7:2017 – Cross-Site Scripting (XSS)	
Implicación de seguridad	Las fallas de XSS ocurren cada vez que una aplicación incluye datos que no son de confianza en una nueva página web sin la validación o el escape adecuados, o actualiza una página web existente con datos proporcionados por el usuario mediante una API de navegador que puede crear HTML o JavaScript. XSS permite a los atacantes ejecutar secuencias de comandos en el navegador de la víctima que pueden secuestrar sesiones de usuario, desfigurar sitios web o redirigir al usuario a sitios maliciosos.
Procedimiento específico: Prevenir los ataques XSS	La prevención de XSS requiere la separación de los datos que no son de confianza del contenido activo del navegador. Esto se puede lograr mediante: • Usar frameworks que escapen automáticamente de XSS por diseño, como Ruby on Rails, React JS. • Escapar los datos de solicitud HTTP que no son de confianza según el contexto en la salida HTML (cuerpo, atributo, JavaScript, CSS o URL) resolverá las vulnerabilidades de XSS reflejadas y almacenadas. • La aplicación de codificación sensible al contexto al modificar el documento del navegador en el lado del cliente actúa contra DOM XSS. • Habilitar una política de seguridad de contenido (CSP) como un control de mitigación de defensa en profundidad contra XSS. Es efectivo si no existen otras vulnerabilidades que permitirían colocar código malicioso a través de archivos locales incluidos (por ejemplo, sobrescrituras transversales de ruta o bibliotecas vulnerables de redes de entrega de contenido permitidas).
A9:2017 – Uso de componentes con vulnerabilidades conocidas	
Implicación de seguridad	Los componentes, como bibliotecas, macros y otros módulos de software se ejecutan con los mismos privilegios que la aplicación. Si se explota un componente vulnerable, dicho ataque puede facilitar la pérdida grave de datos o la toma de control del servidor. Las aplicaciones y las API que utilizan componentes con vulnerabilidades conocidas pueden socavar las defensas de las aplicaciones y permitir varios ataques e impactos.
Procedimiento específico: Utilizar únicamente componentes de fuentes confiables	Debe existir un proceso de administración de parches para: • Eliminar las dependencias no utilizadas, las características, los componentes, los archivos y la documentación innecesarios. • Hacer un inventario continuo de las versiones de los componentes del lado del cliente y del lado del servidor (por ejemplo, macros, bibliotecas) y sus dependencias utilizando herramientas como versiones, DependencyCheck, retire.js, etc. • Monitorear continuamente las fuentes como CVE y NVD en busca de vulnerabilidades en los componentes. Utilizar herramientas de análisis de composición de software para automatizar el proceso. • Solo obtener componentes de fuentes oficiales a través de enlaces seguros. Utilizar únicamente paquetes firmados para reducir la posibilidad de incluir un componente malicioso modificado. • Supervisar las bibliotecas y los componentes que no reciben mantenimiento o que no crean parches de seguridad para versiones anteriores. • Cada organización debe asegurarse de que haya un plan continuo para monitorear, clasificar y aplicar actualizaciones o cambios de configuración durante la vida útil de la aplicación o cartera.
A10:2017 – Registro y monitoreo insuficientes	
Implicación de seguridad	El registro y la supervisión insuficientes, junto con la integración faltante o ineficaz con la respuesta a incidentes, permite a los atacantes atacar aún más los sistemas, mantener la persistencia, pasar a más sistemas y manipular, extraer o destruir datos. La mayoría de los estudios muestran que el tiempo para detectar una brecha es de más de 200 días, generalmente detectado por partes externas en lugar de procesos internos o monitoreo.

Procedimiento específico: Registrar las incidencias y monitorear las actividades sospechosas	Según el riesgo de los datos almacenados o procesados por la aplicación: • Asegurarse de que todos los inicios de sesión, las fallas de control de acceso y las fallas de validación de entrada del lado del servidor puedan registrarse con suficiente contexto de usuario para identificar cuentas sospechosas o maliciosas, y mantenerse durante el tiempo suficiente para permitir un análisis forense retrasado. • Asegurarse que los registros se generen en un formato que las soluciones de administración de registros centralizados puedan consumir fácilmente. • Asegurarse que las transacciones de alto valor tengan un seguimiento de auditoría con controles de integridad para evitar la manipulación o la eliminación, como tablas de base de datos de solo agregar o similares. • Establecer un monitoreo y alertas efectivas para que las actividades sospechosas sean detectadas y respondidas de manera oportuna. • Establecer o adoptar un plan de recuperación y respuesta a incidentes. • Utilizar frameworks de protección de aplicaciones comerciales y de código abierto como OWASP AppSensor, firewalls de aplicaciones web como ModSecurity con el conjunto de reglas básicas de OWASP ModSecurity y software de correlación de registros con paneles y alertas personalizados.

III. RESULTADOS Y DISCUSIÓN

Las pruebas para el análisis de vulnerabilidades se realizaron sobre dos prototipos:

- **Prototipo I:** no se considera el plan de seguridad
- **Prototipo II:** sí se considera el plan de seguridad propuesto

Los resultados obtenidos se contrastaron para comprobar en nivel de mejoría de seguridad sobre la infraestructura tecnológica de la organización, como se muestra en la Tabla 5.

Tabla 5. Resultado de comparación de vulnerabilidades de cada prototipo

VULNERABILIDAD	Prototipo I (sin plan de seguridad	Prototipo II (con plan de seguridad)
A1:2017 – Inyección	SI	NO
A2:2017 – Pérdida de autenticación	SI	NO
A3:2017 – Exposición de datos sensibles	SI	SI
A5:2017 – Pérdida de control de acceso	SI	NO
A6:2017 – Configuración de seguridad incorrecta	SI	NO
A7:2017 – Cross-Site Scripting (XSS)	SI	NO
A9:2017 – Uso de componentes con vulnerabilidades conocidas	SI	SI
A10:2017 – Registro y monitoreo insuficientes	SI	NO
TOTAL	**8 (100% vulnerable)**	**2 (25% vulnerable)**

Con los resultados mostrados en cada prototipo se determina que sin plan de seguridad (Prototipo I) la plataforma informática web era vulnerable en los 8 criterios definidos lo que equivale al 100% vulnerable, sin embargo, con la implementación del plan de seguridad (Prototipo II) las vulnerabilidades se reducen a 2 criterios lo que equivale al 25% vulnerable, por lo que se mejora el 75% del nivel de seguridad en la plataforma web.

La presente investigación combina las normas de la ISO 27001 considerando la política y el control relacionado al desarrollo seguro y las vulnerabilidades más representativas para plataformas web de la lista de OWASP TOP10-2017 con sus respectivos procedimientos específicos, esto genera un plan de seguridad que se constituye en una guía para mitigar cada una de las vulnerabilidades del sistema y orientar las acciones que deben realizar el personal técnico de la Organización.

IV. CONCLUSIONES

- En la fase de recopilación de información se determinó la infraestructura de las instalaciones y los dispositivos que manejan.

- En la fase de análisis se determinó que la plataforma informática web presentó 8 vulnerabilidades de seguridad del listado de OWASP Top 10-2017: inyección, pérdida de autenticación, exposición de datos sensible, pérdida de control de acceso, configuración de seguridad incorrecta, XSS, uso de componentes con vulnerabilidades conocidas, registro y monitoreo insuficientes.
- En la fase de explotación se realizaron pruebas de penetración con herramientas especializadas a cada una de las vulnerabilidades encontradas en la plataforma web con la finalidad de analizarlas y presentar propuestas de solución.
- En la fase de generación de informes se procedió a definir el plan de seguridad para la plataforma web empleando normas ISO 27001 y considerando la propuesta de OWASP-2017.
- Una vez que se implementó del plan de seguridad propuesto se obtuvo una mejora de la seguridad de la plataforma web en un 75%.

V. REFERENCIAS BIBLIOGRÁFICAS

Alcorn, W. (2021). Obtenido de https://tools.kali.org/exploitation-tools/beef-xss

AlGhamdi, S. K., & Vlahu, E. (2020). Information security governance challenges and critical success factors: Systematic review. *Elsevier, 99*. doi:https://doi.org/10.1016/j.cose.2020.102030

Ballen, A., Ayala, C., & Sierra, A. (2017). Análisis de vulnerabilidades en aplicaciones Web desarrolladas en PHP Versión 5.6.24 con base de datos MYSQL Versión 5.0.11 a partir de ataques SQL Inyección. Bucaramanga: Universidad Cooperativa de Colombia, Facultad de Ingenierías, Ingeniería de Sistemas.

Comando It. (2021). Obtenido de https://comandoit.com/ataque-syn-flooding-con-metasploit/

Cortez, D. (2017). *Hydra - Kali Linux, una excelente herramienta de Auditoria.* . Obtenido de https://www.seguridadyfirewall.cl/2017/03/hydra-kali-linux-una-excelente.html

Cuerpo de Bomberos Santo Domingo. (2021). Obtenido de https://bomberossantodomingo.gob.ec/

Gaba, J., & Kumar, M. (2013). Implementation of steganography using CES technique. *IEEE Second International Conference on Image Information Processing (ICIIP)* (págs. 395-399). Shimla: IEEE.

Google. (2021). Obtenido de https://www.google.com/intl/es-419/chrome/

INEN. (2015). TECNOLOGIAS DE LA INFORMACION — TECNICAS DE SEGURIDAD — SISTEMAS DE GESTION DE SEGURIDAD DE LA INFORMACION - REQUISITOS.

ISOTools. (2021). Obtenido de https://www.isotools.org/normas/riesgos-y-seguridad/iso-27001/

Kali linux. (2021). *Kali*. Obtenido de https://www.kali.org/docs/introduction/what-is-kali-linux/

Ladino, M., Villa, P., & López, A. (2011). Fundamentos de iso 27001 y su aplicación en las empresas. *47*(17), 334-339.

Mozilla. (2021). Obtenido de https://www.mozilla.org/es-ES/firefox/new/

Normas ISO. (2020). Obtenido de https://www.normas-iso.com/iso-27001/

OWASP. (2017). *OWASP Top 10 - 2017*. Obtenido de https://wiki.owasp.org/images/5/5e/OWASP-Top-10-2017-es.pdf

OWASP. (2021). Obtenido de https://owasp.org/

Panday, R., & Pandey, V. (2016). Cryptography & security implementation in network computing environments. *3rd Computing for Sustainable Global Development (INDIACom)* (págs. 3136-3140). IEEE.

PortSwigger. (2021). Obtenido de https://portswigger.net/burp

Postor, H. (2020). Mapping the OWASP Top Ten to Blockchain. *Elsevier, 177*, 613-617. doi:https://doi.org/10.1016/j.procs.2020.10.087

Puime, J. (2009). El ciberespionaje y la ciberseguridad. In La violencia del siglo XXI. *Nuevas dimensiones de la guerra* (págs. 45-76). Instituto Español de Estudios Estratégicos.

Rapid7. (2021). Obtenido de https://www.metasploit.com/

Rojas, O., Medina, Y., & Bautista, D. (2016). Pentesting empleando técnicas de ethical hacking en redes IPv5. *11*, págs. 79-96. Rev. Ingenio UFPSO.

Santiago, E., & Allende, J. (2017). RIESGOS DE CIBERSEGURIDAD EN LAS EMPRESAS. Madrid: Revista Tecnología y desarrollo.

Solarte, F., Rosero, E., & Benavides, M. (2015). Metodología de análisis y evaluación de riesgos aplicados a la seguridad informática y de información bajo la norma ISO/IEC 27001. *28*. Escuela Superior Politécnica del Litoral (ESPOL).

Sqlmap. (2021). Obtenido de https://sqlmap.org/

Subgraph. (2021). Obtenido de https://subgraph.com/vega/

Tenable. (2021). Obtenido de https://es-la.tenable.com/products/nessus

Toledo, A. (2014). Test de penetración Exploración de vulnerabilidades con Metasploit-framework.

Wireshark. (2021). Obtenido de https://www.wireshark.org/

Zenmap. (2021). Obtenido de https://nmap.org/zenmap/