TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Bastionado de servidores. Desarrollo de herramienta de auditoria.

Curso 2020/2021

Alumno/a:

Juan Merlos García

Director/es:

José Antonio Álvarez Bermejo

Agradecimientos

Agradecer al tutor del presente trabajo fin de grado, José Antonio Álvarez Bermejo por su atención y apoyo técnico en el desarrollo de dicho trabajo.

También agradecer a Alfonso Bosch por su ayuda en la gestión de trámites y corrección de aspectos clave de dicho trabajo.

Por último, agradecer a todo el profesorado del grado de ingeniería informática por las labores realizadas y por intentar que todo alumno pueda graduarse con los conocimientos y actitudes adecuados.

Resumen

En el presente trabajo de fin de grado se analizará un marco teórico de la seguridad de la infraestructura a nivel empresarial, se profundizará en el bastionado de servidores y, por último, se estudiará y evaluará la necesidad y el impacto de su implementación.

Una vez hecho esto, se replicará la implementación de alguna de las líneas base de configuración ofrecidas por organizaciones prestigiosas en el ámbito de la seguridad informática evitando producir inestabilidades en el sistema. Todo esto se realizará en un laboratorio personal que replicará la infraestructura empresarial típica.

Finalmente, en este trabajo de fin de grado se procederá, de forma adicional, a desarrollar una herramienta mediante programación con Powershell, con la que se evaluará y medirá el nivel de cumplimiento implementado en el servidor objetivo. Esta herramienta permitirá realizar auditorías de seguridad a servidores Windows.

Abstract

In this Final Degree Project, I am going to analyze a theoretical framework of infrastructure security at the business level, delving into the hardening of servers and, finally, I will study and evaluate the need and the impact of its implementation.

Once this is done, I will replicate the implementation of some of the configuration baselines offered by renowned organizations in the field of computer security, avoiding producing instabilities in the system. All of this will be done in a personal lab that will replicate typical business infrastructure.

To conclude this Final Degree Project, I will proceed -in addition- to develop a tool using PowerShell's programming, which will evaluate and measure the level of compliance implemented in the target server. This tool will allow security audits performed in Windows servers.

Índice de contenidos

Tabla de figuras

Tabla de códigos

1. Introducción

Todas las empresas, sin importar el ámbito en el que se desarrollan, disponen en mayor o menor medida de una infraestructura informática de servidores que almacenan y procesan información corporativa de vital importancia para el negocio.

Frecuentemente, a nivel empresarial, se realizan avaluaciones ejecutivas del impacto de un posible incidente de seguridad, resultando la mayoría de ellos, lo suficientemente altos como para considerar imprescindible una determinada inversión en la mejora de la seguridad de la infraestructura (INCIBE-CERT, 2019).

Teniendo en cuenta que los servidores son la parte de la infraestructura informática más importante, puesto que son el núcleo de los servicios tecnológicos sobre los que se sustenta la empresa, el bastionado de dichos servidores y una correcta auditoria de estos es una parte fundamental dentro de las medidas de seguridad informática (Vacca, 2013).

El Bastionado de Sistemas (hardening) se entiende como el conjunto de tareas que tienen como objetivo la correcta implantación de políticas de seguridad, endurecimiento y delimitación clara de los privilegios de usuarios, grupos, roles y configuración de servicios (Security I. , 2018). Este proceso suele ser aplicado a los distintos servicios como Active Directory, LDAP, Fortificación de contraseñas, Firewall, DMZ.

Con el objetivo de conseguir cierto nivel de fortificación de servidores, existen ciertas organizaciones como DISA (Defense Information System Agency) o CIS (Center for Internet Security) que indican líneas base de configuración para este objetivo.

La mayoría de las empresas actualmente tienen que hacer uso de software de terceros para medir el nivel de cumplimiento de esas líneas base de configuración. El objetivo con este trabajo fin de grado, a parte del marco teórico de estos aspectos, será la creación de una herramienta que permita auditar dichas líneas base de configuración.

1.1. Objetivos

El actual Trabajo Fin de Grado consistirá principalmente en dos aspectos.

Por una parte, una explicación teórica de la necesidad de una correcta implantación de seguridad de los servidores, las distintas opciones actuales para dicho fin, y la explicación de cómo implementar distintas líneas base de seguridad de servidores.

Por otra parte, se procederá a desarrollar una herramienta que permitirá auditar y medir el nivel de cumplimiento a nivel de seguridad de dichas líneas base establecidas por los organismos competentes. Esto permitirá realizar evaluaciones de riesgo de seguridad a nivel ejecutivo, y tener un control de cumplimiento de los servidores.

1.2. Planificación del proyecto

Para la correcta ejecución de este trabajo fin de grado se han establecido una serie de fases:

1. **Búsqueda y análisis teórico de la necesidad de implementación de bastionado en servidores Windows** (30 horas): en esta fase se buscará información acerca del marco de seguridad informática adoptado en medianas y grandes empresas referente a la fortificación de servidores, se desarrollará la importancia, impacto y organización de la fortificación de dicha infraestructura de servidores.
2. **Análisis y evaluación de las distintas entidades u organismos para bastionado de servidores** (25 horas): en esta fase se realizará un análisis de las principales entidades de seguridad que aportan guías de implementación de bastionado de servidores. Se realizará una comparación de ellas midiendo el nivel de seguridad alcanzado en sus implementaciones.
3. Implementar infraestructura de servidores Windows en **laboratorio replicando entorno empresarial** (30 horas): para esta tarea se utilizará un entorno de virtualización.
4. **Implementar bastionado en el servidor** (40 horas): una vez analizadas las guías de implementación de bastionado de las diferentes entidades, se procederá a implementarlas en el laboratorio.
5. **Analizar mediante herramientas el nivel de seguridad** antes y después del bastionado del servidor (10 horas).
6. **Desarrollo y programación de herramienta de auditoria de bastionado** (90 horas): una vez implementado el bastionado de servidores, se desarrollará la herramienta que nos permitirá auditar dicho servidor y medir el nivel de cumplimiento alcanzado.
7. **Revisión, corrección y ajuste del nivel de la herramienta** (30 horas): se revisarán las directrices tenidas en cuenta para determinar el nivel de cumplimiento en la herramienta.
8. **Elaboración del informe** (45 horas): crear informe de marco teórico, implantación del bastionado en el servidor, explicación de la herramienta (comprobaciones, esquema de programación, estructuración de código, funcionamiento, etc).

El desarrollo se ha llevado a cabo en 4 meses, estableciendo por semanas la consecución de las distintas fases, quedando la división como se ve en Figura 1. Planificación del proyecto - vista general. Para una mejor visión de la planificación se han añadido las figuras Figura 2. Planificación del proyecto - marzo/abril y Figura 3. Planificación del proyecto - mayo/junio.

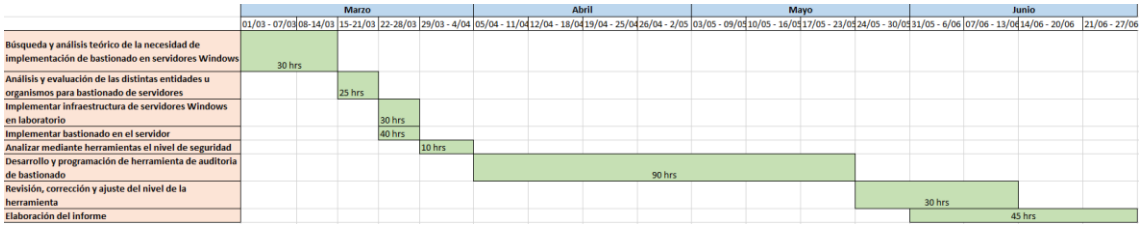


Figura 1. Planificación del proyecto - vista general

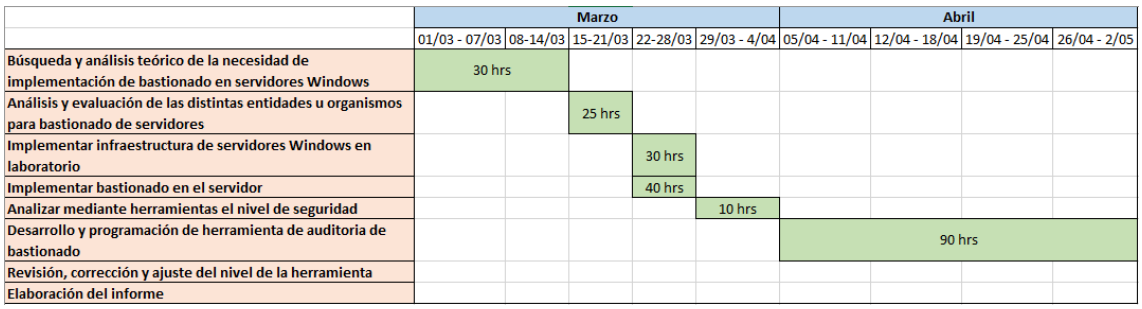


Figura 2. Planificación del proyecto - marzo/abril

	Mayo				Junio			
	03/05 - 09/05	10/05 - 16/05	17/05 - 23/05	24/05 - 30/05	31/05 - 6/06	07/06 - 13/06	14/06 - 20/06	21/06 - 27/06
Búsqueda y análisis teórico de la necesidad de implementación de bastionado en servidores Windows								
Análisis y evaluación de las distintas entidades u organismos para bastionado de servidores								
Implementar infraestructura de servidores Windows en laboratorio								
Implementar bastionado en el servidor								
Analizar mediante herramientas el nivel de seguridad								
Desarrollo y programación de herramienta de auditoria de bastionado	90 hrs							
Revisión, corrección y ajuste del nivel de la herramienta				30 hrs				
Elaboración del informe					45 hrs			

Figura 3. Planificación del proyecto - mayo/junio

Además, cabe destacar que, en paralelo durante la consecución de las fases que han mantenido reuniones con el tutor para definir los requisitos del proyecto y realizar un seguimiento de este, además de resolver posibles dudas que puedan surgir.

El proyecto estaba estimado en 300 horas, y se ha desarrollado con un margen pequeño de error en dicha estimación. Sin embargo, respecto al anteproyecto, se han encontrado diferencias en las horas de cada tarea:

- La fase 3 estaba planificada en 35 horas, pero se realizó en 30 horas finalmente.
- La fase 4 estaba planificada en 50 horas, pero se realizó en 40 horas.
- La fase 7 estaba planificada en 15h. Esta tarea ha sufrido retraso debido a multitud de correcciones en el código, por lo que se han empleado en total 30h.

2. Tecnologías y herramientas

En este apartado vamos a ver las tecnologías y herramientas necesarias para la consecución del proyecto.

2.1. Tecnologías

2.1.1. PowerShell

PowerShell es una solución de automatización de tareas multiplataforma formada por un shell de línea de comandos, un lenguaje de scripting y un marco de administración de configuración. PowerShell funciona en Windows 10, Linux y macOS (Microsoft, ¿Qué es PowerShell?, 2021).

PowerShell se basa en .NET Common Language Runtime (CLR). A diferencia de la mayoría de los shells que solo aceptan y devuelven texto, PowerShell acepta y devuelve objetos .NET. Por tanto, estamos ante un lenguaje de programación de alto nivel.

Como lenguaje de scripting, PowerShell se usa normalmente para automatizar la administración de sistemas.

2.2. Herramientas

2.2.1. VirtualBox

Oracle VM VirtualBox (conocido generalmente como VirtualBox) es un software de virtualización. Este actúa como un hipervisor con el objetivo de poner ejecutar otros sistemas operativos dentro del propio equipo.

VirtualBox soporta algunos sistemas operativos como Linux, MAC OS X, Windows, Solaris, FreeBSD, etc.

En este caso, utilizaremos VirtualBox con el objetivo de simular el entorno empresarial en nuestro propio equipo. Podríamos utilizar openstack que nos provee la universidad de Almería, sin embargo, no tenemos control total de la maquina como tendríamos con un programa que nos permita virtualizar en nuestro propio equipo.

2.2.2. Kali

Kali es una distribución GNU/Linux basada en Debian y destinada a pruebas avanzadas de penetración y auditoría de seguridad. Kali Linux contiene varios cientos de herramientas dirigidas a diversas tareas de seguridad de la información, como pruebas de penetración, investigación de seguridad, informática forense e ingeniería inversa (Kali, 2021).

Esta distribución nos permitirá en nuestro caso, comparar resultados respecto a la herramienta desarrollada propiamente. Para ello, utilizaremos software de terceros actualmente en el mercado y algunas herramientas contenidas en Kali.

2.2.3. Nessus

Nessus es una suite de escaneo de vulnerabilidades y soluciones de cumplimiento desarrollado por Tenable. Nessus permite automatizar las evaluaciones de seguridad escaneando las vulnerabilidades, incluyendo parches, defectos de software, malware, configuraciones erróneas etc.

Además, permite generar reportes de cumplimiento o auditoria, realizar escaneos de vulnerabilidades offline, crear informes basados en vistas personalizadas.

Debido a los complementos incluidos en Nessus, este permite medir el cumplimiento de distintas líneas base de seguridad.

Según HG Insights, hay 15.685 compañías que actualmente hacen uso de Nessus para la parte de auditoria (Insights, 2021). Según el datasheet de Nessus hay +30.000 (Tenable, Nessus Professional Datasheet, 2021).

A nivel empresarial, es la herramienta más utilizada actualmente en cuanto a escaneo de vulnerabilidades.

En nuestro caso, podremos utilizar esta herramienta como comparación respecto a la herramienta desarrollada propiamente.

3. Análisis de la necesidad de implementación de líneas de seguridad en servidores.

En este capítulo, analizaremos desde un marco teórico la importancia y el impacto de la seguridad de la infraestructura empresarial y plantearemos el problema a resolver.

3.1. La importancia de la seguridad de los sistemas

En 2016, Kaspersky lab, en conjunto con B2B International, realizó un estudio global en más de 4.000 empresas representativos de 25 países, observando sus presupuestos de seguridad informática, la complejidad de su infraestructura, las actitudes que toman ante las amenazas de seguridad y soluciones, el costo real de las filtraciones de información, y los incidentes de seguridad que han sufrido. Todo esto fue enfocado a la parte financiera de la ciberseguridad (Kaspersky, 2016).

A continuación, podemos ver el presupuesto medio invertido según el tipo de empresa (gran empresa, PYME o pequeños negocios).

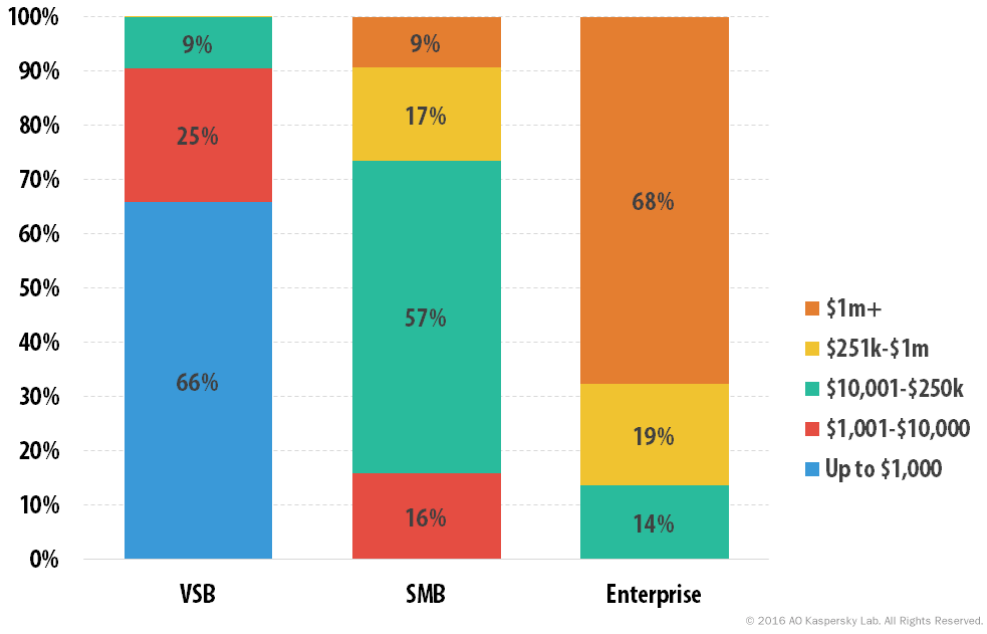


Figura 4: Presupuesto medio invertido en seguridad informática según el tipo de empresa

La fuente de la Figura 4: Presupuesto medio invertido en seguridad informática según el tipo de empresa es *"Report: Measuring the Financial Impact of IT Security on Businesses", Kaspersky, 2016*. En ella podemos observar que, en grandes empresas, en su mayoría se destina un largo presupuesto +1M en seguridad de la infraestructura.

Además, en este mismo estudio, podemos ver los tipos de eventos de seguridad experimentados durante el periodo del último mes, entre los que destaca la afectación por virus y malware causando una gran pérdida de productividad.

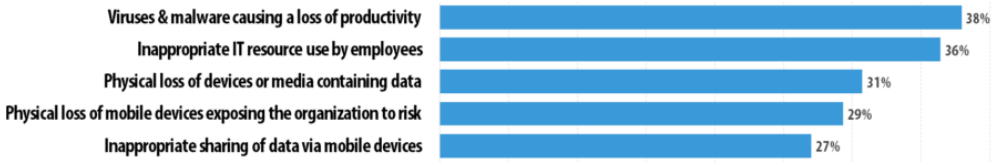


Figura 5. Eventos de seguridad experimentado por las empresas

La Fuente de la Figura 5. Eventos de seguridad experimentado por las empresas es *"Report: Measuring the Financial Impact of IT Security on Businesses", Kaspersky, 2016.* En ella, podemos ver que el costo promedio de la recuperación de un solo incidente de seguridad está estimado en $86.5 mil USD para las pequeñas y medianas empresas y $861 mil USD para las grandes.

Debido a este problema actual de seguridad y al coste asociado a incidentes, se plantea la necesidad de implementar políticas de seguridad en la infraestructura. Dado que los servidores son los encargados de ofrecer los servicios dentro de la empresa, habitualmente, la parte servidora es crítica, ya que, según las políticas de seguridad implementadas, determinará cómo y cuándo se podrá hacer uso de dichos servicios.

3.2. Auditoria de cumplimiento de seguridad

Para garantizar la seguridad de dicha parte servidora, suelen hacerse uso de guías de cumplimiento (STIGs). STIGs es un estándar de configuración que consta de requisitos de ciberseguridad para un producto específico. El uso de STIG permite una metodología para proteger protocolos dentro de redes, servidores, computadoras y diseños lógicos para mejorar la seguridad general. Estas guías, cuando se implementan, mejoran la seguridad del software, hardware, arquitecturas físicas y lógicas para reducir aún más las vulnerabilidades.

Desde un punto de vista objetivo, cualquier organización podría crear e implementar su propio estándar de bastionado de sistemas. Sin embargo, estos sistemas y servicios tienen cientos de opciones de configuración, que, además, pueden cambiar en cada versión. Esto significa que, implementar un propio estándar, sería muy costoso a nivel económico y de recursos.

Por este motivo, la mayoría de las organizaciones optan por implementar un estándar ya existente y validado. En este caso, los más populares son los ofrecidos por DISA y CIS.

La constante evaluación y auditoria de dicho cumplimiento es una parte imprescindible a nivel empresarial, ya que estas auditorias deben formar parte de un programa de gestión de riesgos que suele ser revisado recurrentemente con el objetivo de mantenerse al tanto de los posibles puntos débiles y áreas de mejora.

3.2.1. DISA

DISA (Agencia de Sistemas de Información de Defensa), conocida como Agencia de Comunicaciones de Defensa (DCA) hasta 1991, es una agencia de apoyo al combate del Departamento de Defensa de los Estados Unidos (DoD).

Figura 6. Logo de DISA

Esta provee apoyo a nivel tecnológico y de comunicaciones a todos los institutos e individuos que trabajan para el DOD. DISA supervisa los aspectos tecnológicos y de TI de la organización, entrega y gestión de información relacionada con la defensa, etc.

Desarrolladas por DISA en nombre del departamento de defensa, las STIGs (guías de configuración técnica de seguridad) son estándares aceptados por el gobierno federal de los estados unidos y contratistas, con el objetivo de garantizar la seguridad de la información gubernamental.

Actualmente DISA tiene publicadas más de 500 guías que son mantenidas por el departamento de defensa de Estados Unidos. Este provee regularmente de actualizaciones de estas guías de configuración técnica con el objetivo de que las distintas empresas que conforman el gobierno federal puedan:

- Realizar configuraciones hardware y software correctas.
- Implementar protocolos de seguridad.
- Organizar procesos de formación.

La aplicación de estas guías de configuración técnica de seguridad (en delante STIGs) y su cumplimiento ayuda a garantizar que los sistemas afiliados al Departamento de Defensa estén a salvo de ataques desde el exterior y desde el interior del sistema. La aplicación de estas guías de configuración técnica de seguridad (en delante STIGs) no son un requerimiento para empresas que no sean contratistas del gobierno con acceso al departamento de Defensa de los Estados Unidos. No obstante, ciertas empresas, por el impacto y consecuencias de una brecha de seguridad, se ven obligadas a establecer un nivel de protocolos de seguridad y monitorización alto, y asegurar un cumplimiento de estas guías, permite certificar cierto nivel de seguridad en la compañía.

Las guías técnicas de DISA están originadas debido a las leyes establecidas DODD 8500.1, DODI 8500.2, las cuales principalmente recogen los siguientes puntos:

- DODD 8500.1 (Department of Defense, 2014).
    - Todos los productos TI incorporados en los sistemas de información de DOD deberán configurarse de acuerdo con las pautas de configuración de seguridad aprobadas por el DOD.
    - DISA desarrollará y proporcionará orientación de configuración de seguridad para productos de TI en coordinación con la NSA.
- DODI 8500.2 (Department of Defense, 2003).
    - Especificación de configuración de seguridad. DISA y NSA apoyarán el programa de defensa a través del desarrollo y difusión de implementaciones de seguridad para la configuración de productos de TI. Ejemplos de tales especificaciones incluyen las pautas de implementación técnica de seguridad (STIGS) y las Guías de recomendaciones de seguridad (SRG).

- Un documento de referencia del DOD, como, por ejemplo, una guía de implementación técnica de seguridad o una guía de recomendación de seguridad, constituye la fuente principal para la configuración de seguridad o la guía de implementación para la implementación de productos de TI.

Además, las guías técnicas de DISA están respaldadas por las siguientes normas: CJCSI 6510.01, AR 25-2, and AFI 33-202 (Defense information Systems Agency, 2010).

Hay tres tipos de niveles de cumplimiento de DISA STIG que comúnmente son llamadas categorías. Estas categorías indican la severidad del riesgo de no implementar cada vulnerabilidad.

Si ordenamos de forma descendente en nivel de severidad (desde la mayor severidad a la menor) tendríamos las siguientes categorías:

- Categoría I. Referida a cualquiera vulnerabilidad que afecta directa e inmediatamente en una pérdida de confidencialidad, disponibilidad o integridad. Estas vulnerabilidades pueden dar acceso no autorizado a datos clasificados.
  - Pueden conllevar: pérdida de vidas (según en qué contexto), daños en los sistemas, o que desencadene en una misión fallida (en términos de defensa).
- Categoría II. Referida a cualquiera vulnerabilidad que puede acabar resultando en pérdida de confidencialidad, disponibilidad o integridad
  - Pueden conllevar:
    - Escalada a vulnerabilidad de categoría I.
    - Provocar daños en los sistemas o instalaciones, ataques personales.
    - Degradar la misión (en términos de defensa).
- Categoría III. Referida a cualquier vulnerabilidad que disminuya las medidas de protección contra la pérdida de confidencialidad, disponibilidad o integridad.
  - Puede conllevar:
    - Escalada a vulnerabilidad de categoría II.
    - Retraso en la recuperación de una interrupción del servicio.
    - Afectación a la precisión de los datos e información.

Para acceder a las líneas base de seguridad ofrecidas por DISA, debemos acceder a través del siguiente enlace: https://public.cyber.mil/stigs/downloads

3.2.2. CIS

El Centro para la seguridad de Internet es una entidad sin ánimo de lucro cuya misión es "identificar, desarrollar, validar, promover y mantener soluciones de prácticas recomendadas para la ciberdefensa". Se basa en la experiencia de los profesionales de ciberseguridad y IT de los gobiernos, las empresas y el mundo académico de todo el mundo para desarrollar estándares y buenas prácticas, incluyendo líneas de seguridad, controles e imágenes protegidas por CIS, siguiendo un modelo de toma de decisiones consensuado.

Figura 7: Logo de CIS

Los benchmark de CIS son líneas de base de configuración y buenas prácticas para la configuración segura de un sistema (Microsoft, Center for Internet Security (CIS) Benchmarks, 2021). Cada una de las recomendaciones de la guía hace referencia a uno o más controles CIS, que fueron desarrollados para ayudar a las organizaciones a mejorar sus capacidades de ciberdefensa. Los controles CIS se ajustan a muchas normas y marcos normativos establecidos, incluidos el NIST Cybersecurity Framework (CSF) y NIST SP 800-53, la serie de normas ISO 27000, PCI DSS, HIPAA y otros.

Cada benchmark pasa por dos fases de revisión de consenso. La primera ocurre durante el desarrollo inicial, cuando los expertos se reúnen para discutir, crear y probar los borradores de trabajo hasta que llegan a un consenso sobre el indicador. Durante la segunda fase, después de la publicación del indicador, el equipo de consenso revisa la retroalimentación de la comunidad de Internet para incorporarla al indicador.

Desde un punto de vista de seguridad, los benchmark de CIS ayudan a las organizaciones a:

- Crear y mantener un perfil de seguridad acorde con las mejores prácticas de la industria.
- Eliminar los ajustes de configuración que se tiene constancia que son inseguros.
- Proteger la organización de amenazas conocidas.
- Eliminar el riesgo cibernético innecesario reduciendo la superficie de ataque a solo lo necesario.

Por otro lado, los benchmark ofrecidos por CIS se diferencian en ciertos aspectos de otros estándares de seguridad:

- Se relacionan específicamente con la configuración de activos existentes. No cubren las defensas de seguridad como firewalls y EDR.
- Se desarrollan por consenso entre expertos que incluyen pymes, proveedores de seguridad, el equipo de evaluación comparativa de CIS e incluso la comunidad de seguridad global a través de CIS Workbench.
- Si bien no es un requisito reglamentario, los marcos de cumplimiento más destacados señalan los puntos de referencia de CIS como el estándar de la industria.

Respecto a la estructuración de los benchmark de CIS proporcionan dos niveles de configuración de seguridad:

- El nivel 1 recomienda requisitos básicos de seguridad esenciales que pueden configurarse en cualquier sistema y que deberían causar poca o ninguna interrupción del servicio o una funcionalidad reducida.

- El Nivel 2 recomienda configuraciones de seguridad para entornos que requieren una mayor seguridad que podría resultar en una reducción de la funcionalidad.

Cabe destacar que, los benchmark de CIS están desarrollados para funcionar tanto como recursos independientes, como acompañados de otros protocolos o estándares. Algunos de los protocolos o estándares en los que se apoya CIS son: PCI DSS, NIST and FISMA, HIPAA, GDPR, ISO/IEC 27001 (Security, Center of Internet, 2020).

Para acceder a las líneas base de seguridad ofrecidas por CIS, debemos acceder a través del siguiente enlace: https://www.cisecurity.org/benchmark/microsoft_windows_server/

La visualización de estas únicamente se podrá realizar mediante registro con correo electrónico.

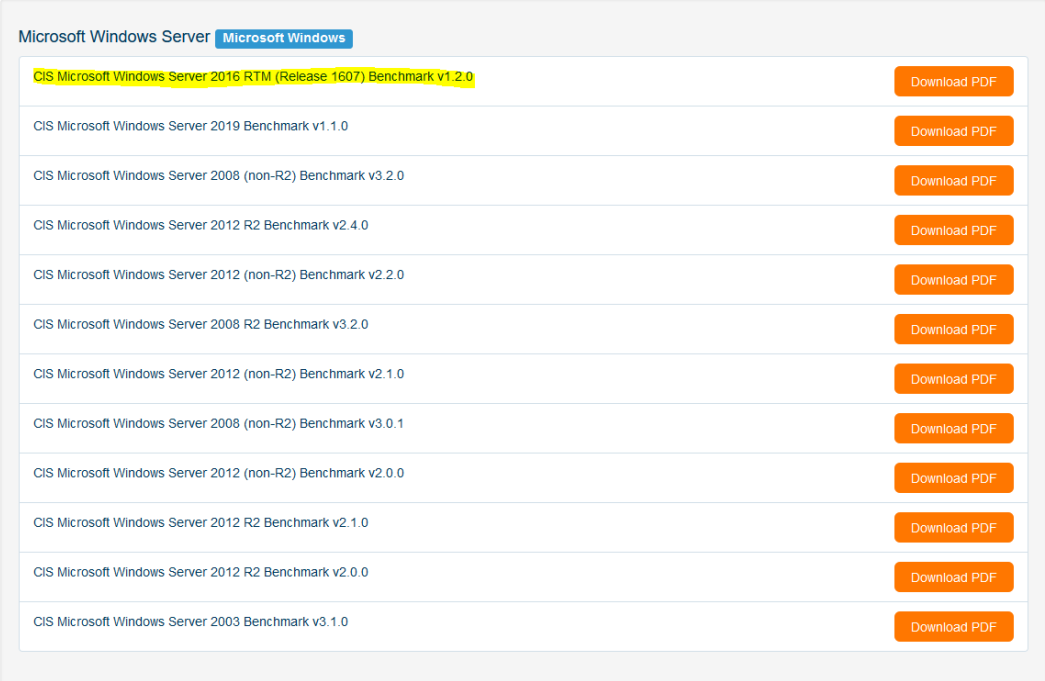


Figura 8. Guías de cumplimiento de CIS

En estos PDF se detallan los distintos ítems a configurar en el servidor para su cumplimiento.

3.3. Contextualización de problema a resolver.

Actualmente en el mercado, existen multitud de software de terceros como el citado anteriormente (Nessus, Qualys, etc) que permiten cubrir la necesidad de auditar el cumplimiento de las guías técnicas de configuración. No obstante, este software de terceros contiene ciertos aspectos negativos que repercute a la hora de su implementación en según qué entornos empresariales:

- Presupuesto. El coste asociado a este tipo de software suele ser bastante elevado, al igual que el coste de mantenimiento. Esto repercute en la posibilidad de implementación por parte de PYMES.
- Necesidad de adaptación de los ítems de cumplimiento para adaptarlos a la infraestructura.
- Estas herramientas de terceros suelen estar escritas en lenguaje propio y de difícil acceso (Nessus por ejemplo utiliza NASL - *Nessus* Attack Scripting Language).

Por tanto, actualmente, salvo proyectos abandonados, no hay proyectos de software comunitario que provean de la necesidad de una herramienta de auditoria de cumplimiento de las guías técnicas de seguridad.

En este proyecto se abordará el desarrollo de una herramienta escrita en Powershell, que permita auditar el sistema con un margen de error más pequeño que el provisto por software de terceros como Nessus, y que a su vez pueda ser escalable mediante repositorios públicos.

4. Requerimientos técnicos

El principal requerimiento técnico será la implementación de un laboratorio que nos permita replicar un entorno empresarial con ciertos servicios básicos.

Vamos a utilizar virtualización para montar un servidor que replique un entorno empresarial y realizar las pruebas necesarias.

4.1. Windows Server 2016

Windows Server hace referencia al grupo de sistemas operativos desarrollado por Microsoft enfocado a la parte de prestación de servicios dentro del entorno empresarial.

4.1.1. Justificación de elección de sistema operativo

Aunque existen guías técnicas de seguridad tanto para Windows como para Linux, en este caso vamos a utilizar Windows server debido a la dominancia de la cuota de mercado en el ámbito de sistema operativo servidor.

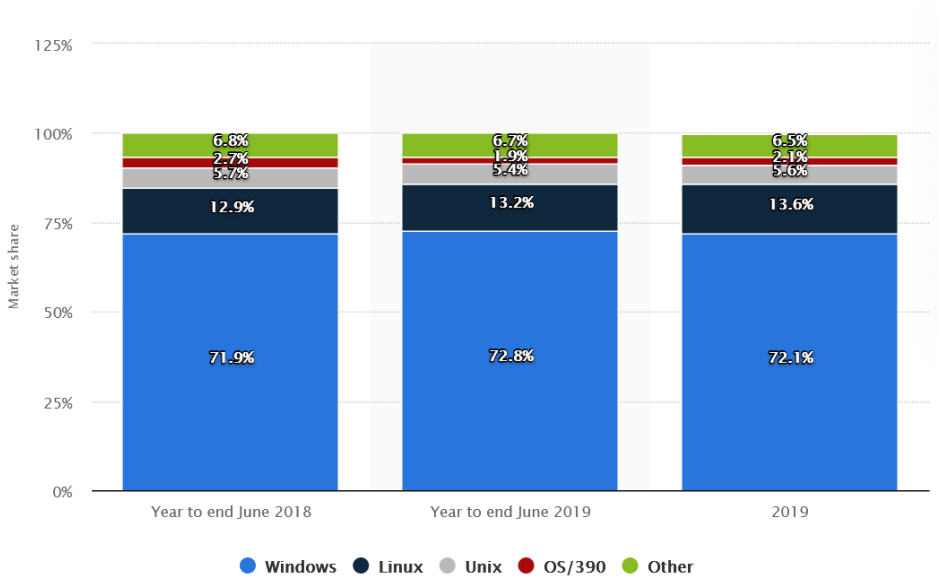


Figura 9. Cuota de mercado en el ámbito de sistema operativo servidor

Los datos de la Figura 9. Cuota de mercado en el ámbito de sistema operativo servidor han sido obtenidos de “Statista”, el cual es un portal de estadística en línea (Statista, 2020).

Esta dominancia de la cuota de mercado es en parte debida a algunos servicios populares con gran rendimiento que ofrecen los sistemas operativos de Microsoft, tales como:

- Active Directory Domain Services (AD DS).
- Active Directory Federation Services (AD FS).
- Network Policy Access Services (NPAS).

- Web & Application Servers.
- Printer and Document Services.
- Domain Name System (DNS) Server.
- Dynamic Host Configuration Protocol (DHCP) Server.
- File Services Server.
- Windows Server Update Services (WSUS) Server.

Por otra parte, aunque es cierto que la última versión disponible de Windows Server es Windows Server 2019, debido a la menor capacidad de cambio respecto a los equipos de usuario, y a la actualización en menor medida que suelen tener grandes compañías, Windows Server 2016 sigue siendo el sistema operativo que en mayor medida se encuentra implementado en el tejido empresarial (Krause, 2016).

4.1.2. Creación de máquina virtual

Por tanto, el primer paso será la creación de una máquina virtual Windows Server 2016 en VirtualBox (ver apartado 2.2.1). El sistema operativo puede ser descargado en versión de evaluación de 180 días en la página de Microsoft (Microsoft, Productos Windows Server y recursos, 2021).

Un aspecto importante a tener en cuenta en el ámbito de auditoria de seguridad es el idioma. Ciertas comprobaciones, tanto si se realiza con software de terceros, como desarrollado propiamente, es altamente probable que algunas no se realicen correctamente debido al idioma. Por compatibilidad siempre es aconsejable la descarga en inglés.

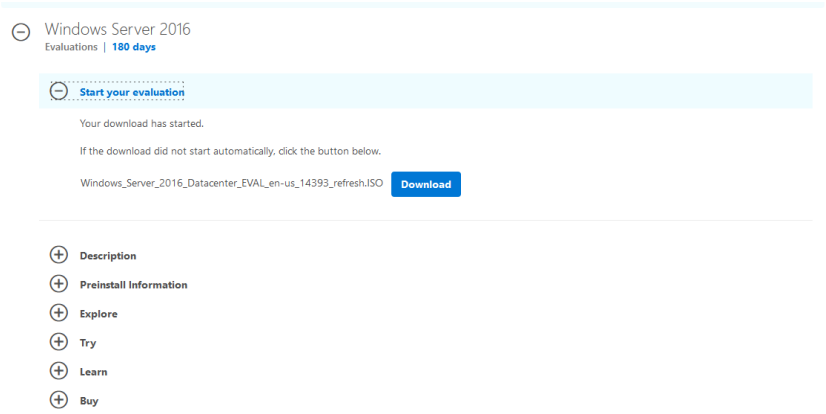


Figura 10. Descarga de Windows Server 2016

Una vez descargado, procedemos a la creación de la máquina virtual en VirtualBox. Procedemos a crear la máquina virtual.

El procedimiento puede ser consultado en (Oracle, 2020), no obstante, también puede consultarse el

Anexo I. Creación de máquina virtual Windows Server en VirtualBox.

4.1.3. Configuración interna y preparación para adaptar a entorno empresarial

Tras esto, procederemos a implementar algunos servicios básicos existentes en la mayoría de los entornos empresariales.

Algunos de los servicios principales implementados por la mayoría de las compañías son DNS y ADDS.

DNS (Sistema de nombres de dominio) es un protocolo que tiene como objetivo la resolución de nombres de dominio, traduciendo estos a IPs. Este protocolo es uno de los estándares que componen TCP/IP.

Active Directory Domain Service (AD DS) es un servicio de directorio, entendiendo como directorio una estructura jerárquica que almacena información acerca de los objetos de la red. ADDS proporciona una forma de almacenar los datos de directorio y hacer que estos estén disponibles para los distintos usuarios y administradores de la red (Microsoft, Introducción a Active Directory Domain Services, 2017).

La instalación y configuración de estos servicios en Windows Server puede ser consultado en el Anexo II. Instalación de servicios ADDS y DNS en Windows Server.

4.2. Kali

Como se cita en el apartado 2.2.2, Kali es una distribución GNU/Linux basada en Debian y destinada a pruebas avanzadas de penetración y auditoría de seguridad.

En este caso, utilizamos Kali para medir el nivel de auditoria de la máquina Windows Server.

4.2.1. Creación de máquina virtual

La distribución Kali Linux fue fundada y es mantenida por Offensive Security Ltd. En este caso, esta empresa pone a disposición pública los archivos referentes a una máquina virtual ya preparada (Security O. , 2021).

Por tanto, simplemente descargamos la máquina virtual ya preparada y procedemos a importarla en VirtualBox siguiendo el Anexo III. Importación de Kali Linux en VirtualBox.

4.2.2. Preparación para auditoria

A continuación, procederemos a instalar Nessus (ver apartado 2.2.3) en la máquina virtual Kali Linux.

Para ver el proceso detallado de instalación de Nessus y su inicialización, véase el Anexo IV. Instalación de Tenable Nessus en Kali Linux e inicialización.

Nessus será utilizado como comparación respecto a la herramienta desarrollada propiamente, ya que es uno de los softwares de terceros más usados actualmente.

5. Implementación de bastionado

En este capítulo veremos cómo implementar la guía técnica de seguridad de DISA para Windows Server 2016.

5.1. Contextualización

La guía técnica de DISA contiene 273 ítems según la última revisión. Debemos entender que cada uno de estos ítems indica una opción de configuración, y, por tanto, la aplicación y cumplimiento en alto grado de estas guías, conlleva una revisión más o menos manual de estos ítems.

Según la nomenclatura descrita en el ID y versión de cada uno de los ítems, podemos subdividir los 273:

Overview

Finding ID	Version	Rule ID	IA Controls	Severity
V-73505	WN16-CC-000070	SV-88157r1_rule		Low

Figura 11. Nomenclatura de elementos de DISA

- WN16-00. Estos ítems realizan comprobaciones a nivel de dominio y otras comprobaciones generales a nivel de sistema operativo.
- WN16-AC. Se realizan comprobaciones de Account Policies.
- WN16-AU. Estos ítems revisan el auditaje de eventos de seguridad del sistema.
- WN16-CC. Realizan comprobaciones de claves de registro.
- WN16-DC. Comprobaciones específicas de los controladores de dominio.
- WN16-MS. En estos ítems se excluyen los controladores de dominio. Se comprueban los servidores miembros de dominio, pero sin rol de controlador de dominio.
- WN16-PK. Estos ítems realizan comprobaciones de certificados.
- WN16-SO. Revisa las opciones de seguridad dentro de las políticas locales.
- WN16-UR. Verifican la asignación de permisos de usuarios/grupos.

Aunque es cierto que existen algunos métodos para la implantación de las guías STIG como Chef Automate, Powershell DSC o SCAP, lo cierto es que estos métodos únicamente permiten implementar cierto número de ítems, no llegando a un porcentaje de implementación adecuado.

En el siguiente gráfico se puede observar en la STIG versión 1 release 4 de Windows Server 2016, la cantidad de ítems que es posible implementar con cada método. Podemos observar que, podremos automatizar la implementación de cierta parte de ítems, pero otra parte tendremos que implementarlos de manera manual.

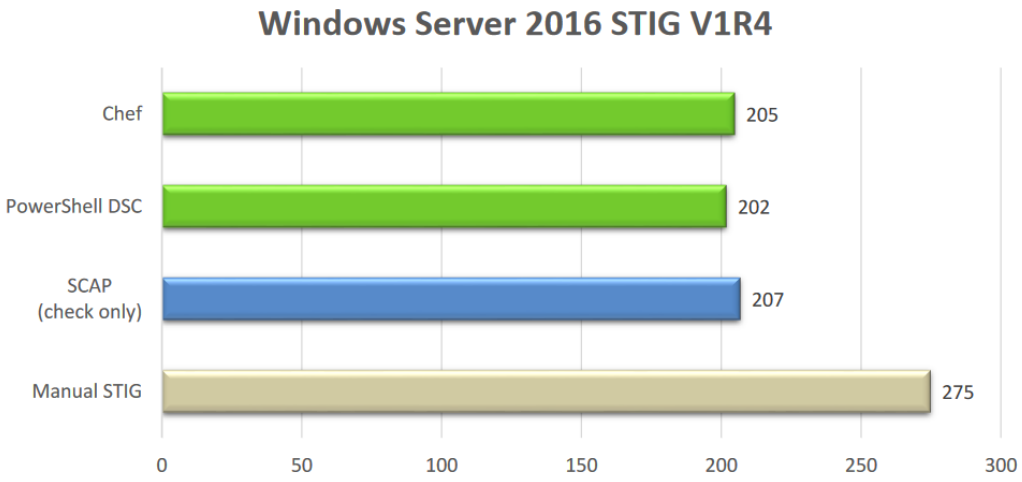


Figura 12. Tipos de implementaciones de DISA y capacidad de cumplimiento

Los datos de la Figura 12. Tipos de implementaciones de DISA y capacidad de cumplimiento han sido obtenido de *"Automated Implementation of Windows-related Security-Configuration Guides", Patrick Stöckle, Bernd Grobauer, and Alexander Pretschner, 2020*. (Stöckle, Grobauer, & Pretschner, 2020). En ella, como vemos, con los diferentes métodos, el número de ítems que se pueden aplicar son similares. La explicación de esto es que la parte de ítems que se pueden automatizar corresponden a los ítems que pueden ser implementados mediante políticas de equipo (group policy) en un entorno con servicio de directorio activo.

5.2. Creación de proceso

DISA nos provee de una automatización para importar dos políticas de grupo que nos permitirán corregir estos ítems (NIST, Microsoft Windows Server 2016 STIG Checklist Details, 2021).

SCAP 1.2 Content:

- Download SCAP 1.2 Content - Microsoft Windows Server 2016 STIG Benchmark - Ver 2, Rel 1
  - Author: Defense Information Systems Agency

Supporting Resources:

- Download Standalone XCCDF 1.1.4 - Microsoft Windows Server 2016 STIG - Ver 2, Rel 1
  - Defense Information Systems Agency
- Download Machine-Readable Format - Microsoft Windows Server 2016 STIG for Chef - Ver 1, Rel 3
  - Defense Information Systems Agency
- Download Machine-Readable Format - Microsoft Windows Server 2016 STIG for PowerShell DSC - Ver 1, Rel 3
  - Defense Information Systems Agency
- Download GPOs - Group Policy Objects (GPOs) - November 2020
  - Defense Information Systems Agency

Figura 13. Descarga de políticas de grupo de DISA

Descargaremos estas junto al XML que definirá los ítems a cumplir en esta guía técnica, y pondremos ambos en el directorio del código fuente del proyecto.

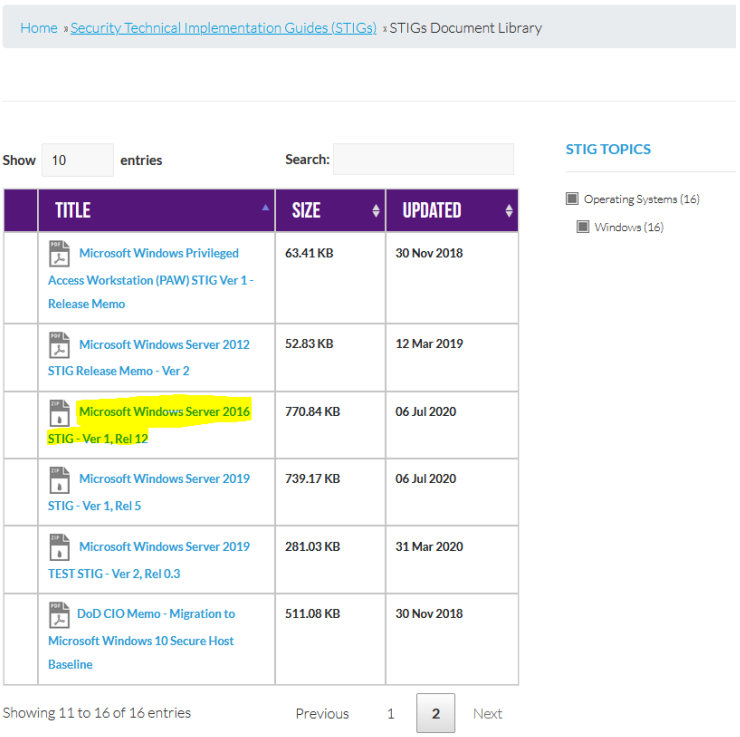


Figura 14. Descarga de guía técnica de DISA en formato XML

En este caso, en el código fuente del proyecto ya se encuentran descargadas estas políticas, aunque pueden ser actualizadas cada cierto tiempo.

Aunque la creación de herramienta de auditoria de este proyecto únicamente estaba pensada para auditar, finalmente, se ha implementado también la funcionalidad de soporte en la implementación del bastionado, aunque como citamos anteriormente, habrá ítems que habrá que implementar de forma manual.

Principalmente, con el script desarrollado, se realizarán las siguientes operaciones de implementación de bastionado:

- Instalación de utilidad ofrecida por DISA para la implementación de los certificados necesarios, importación por la línea de comandos de los certificados necesarios y posterior eliminación de la utilidad para dejar el sistema lo más intacto posible.
- Implementación de políticas de grupo ofrecidas por DISA.
- Implementación de algunos ítems que no se implementan mediante las políticas de grupo, y que, al no depender de la organización de la compañía, se pueden automatizar:
    - Modificación de atributo de máximo de conexiones LDAP.
    - Eliminación de característica de SMB1.

Para que la herramienta pueda interpretar que se pretende implementar el bastionado y no auditar, se debe indicar el parámetro “-Implement”. Para más información del código fuente se puede consultar el Anexo V. Código fuente de herramienta de auditoria. Anexo V. Código fuente de herramienta de auditoria.

Por tanto, en primera instancia, copiaremos todo el directorio de código fuente del proyecto al servidor. Una vez hecho esto, necesitaremos ejecutar el script con credenciales de administrador y pasando por parámetro “-Implement”.

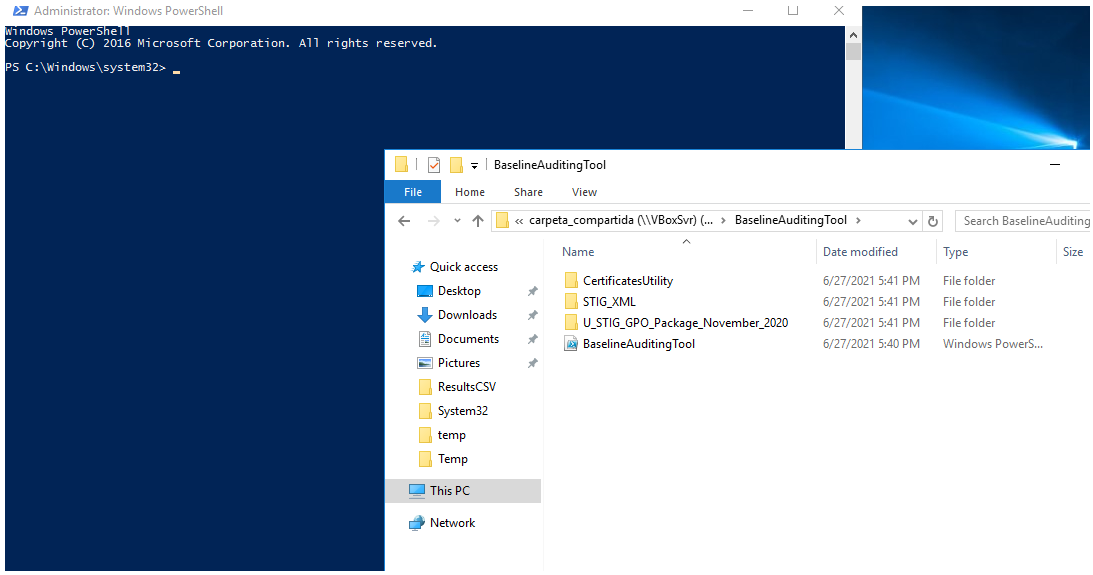


Figura 15. Estructura de directorio de código fuente

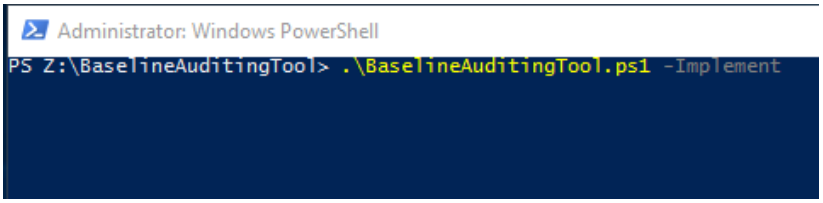


Figura 16. Ejecución de herramienta para implementación de bastionado

Tras esto, nos solicitará la selección del archivo XML de la guía técnica de seguridad. Esta se encuentra en el directorio “STIG_XML” del código fuente del proyecto. Esto es debido a que, el proyecto puede ser escalado para implementar diferentes guías técnicas de seguridad.

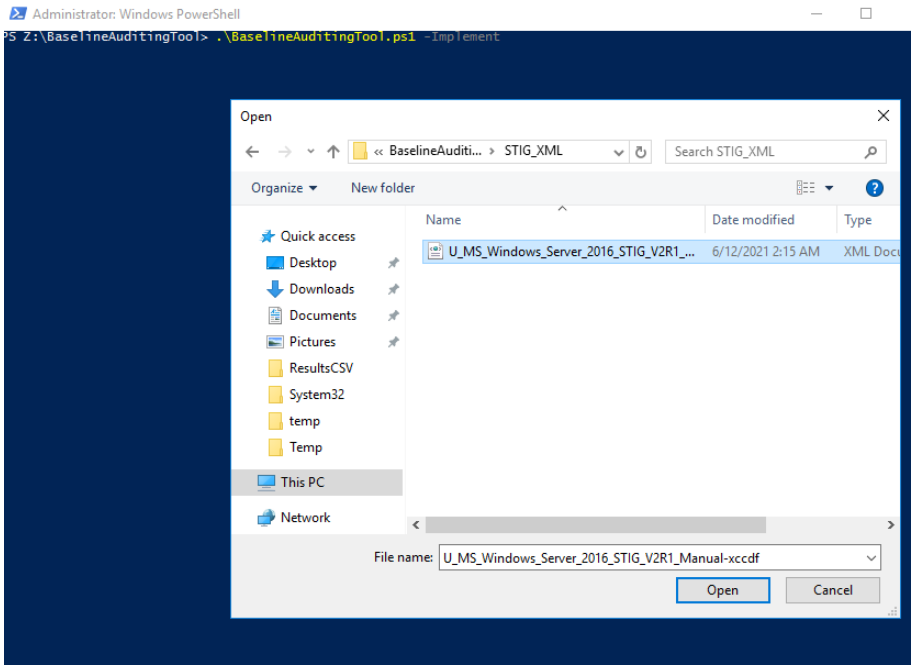


Figura 17. Selección de guía técnica de seguridad en la herramienta

Tras esto, la herramienta comenzará con la instalación de la utilidad de DISA para la importación de certificados. Simplemente avanzaremos en la instalación y finalmente cerraremos el diálogo.

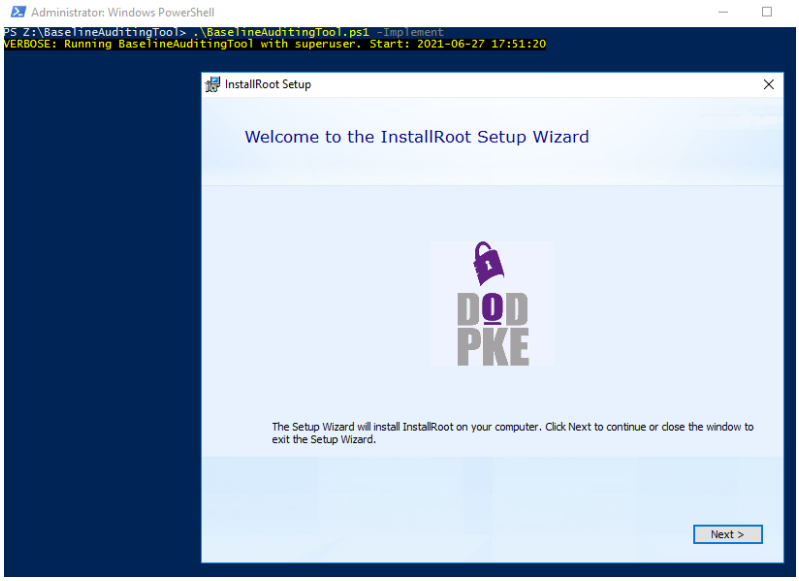


Figura 18. Instalación de utilidad de implantación de certificados de DOD

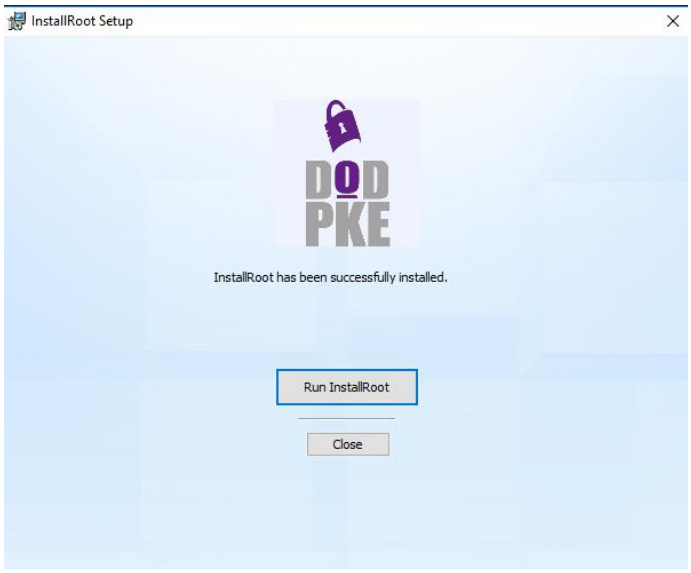


Figura 19. Dialogo final de Instalación de utilidad de certificados de DOD

A continuación, solicitará el nombre del grupo de administradores de dominio, y de Enterprise admins. Esto es necesario debido a que la implementación variará en función de estos, y debido a que en cada compañía el nombre de estos grupos será diferente.

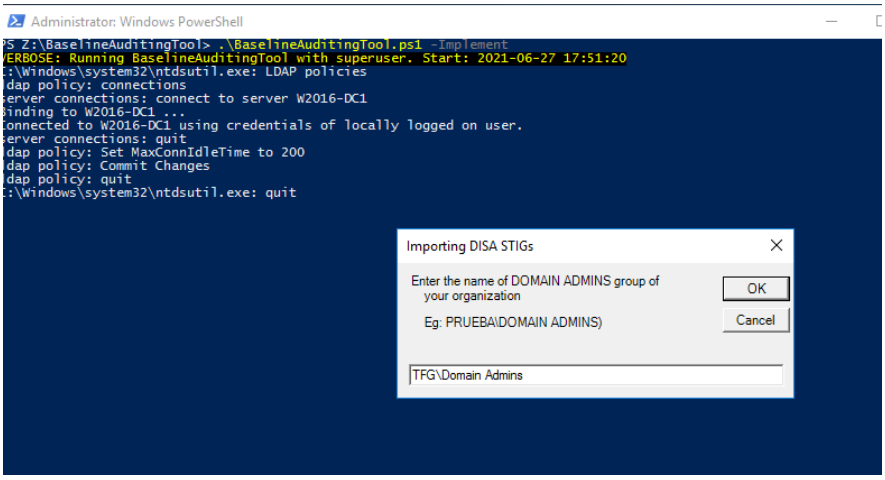


Figura 20. Introducción de información en la herramienta para el bastionado

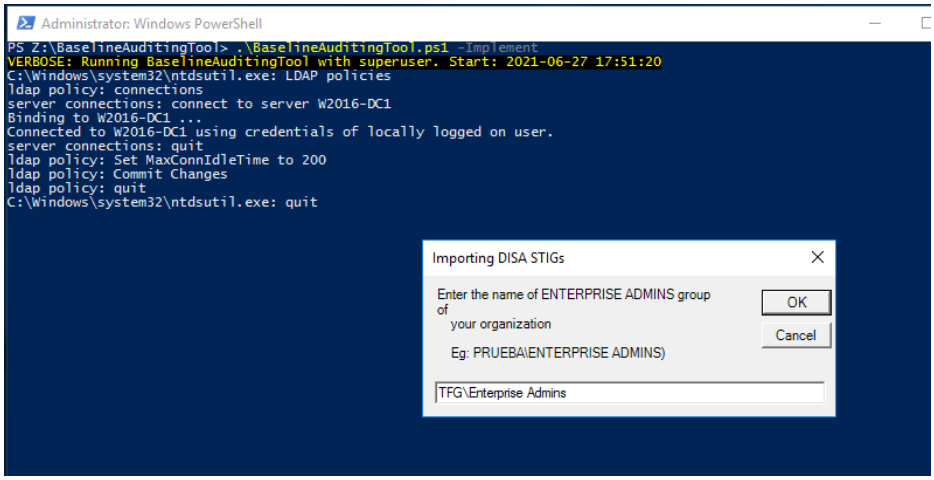


Figura 21. Introducción de información en la herramienta para el bastionado - parte 2

Tras esto, la herramienta procederá a la importación de los certificados de DISA.

```
PS C:\temp\BaselineAuditingTool> .\BaselineAuditingTool.ps1 -Implement
VERBOSE: Running BaselineAuditingTool with superuser. Start: 2021-06-27 18:32:45
C:\Windows\system32\ntdsutil.exe: LDAP policies
ldap policy: connections
server connections: connect to server W2016-DC1
Binding to W2016-DC1 ...
Connected to W2016-DC1 using credentials of locally logged on user.
server connections: quit
ldap policy: Set MaxConnIdleTime to 200
ldap policy: Commit Changes
ldap policy: quit
C:\Windows\system32\ntdsutil.exe: quit
Checking for Log file
C:\Users\superuser\Documents\GPO_Automation_Logs\STIG_GPO_Import.log was located
DoD Windows Defender Antivirus STIG Computer v2r1
No GPO named DoD Windows Defender Antivirus STIG Computer v2r1 was found. We will now create and import GPO.


DisplayName      : DoD Windows Defender Antivirus STIG Computer v2r1
DomainName       : tfg.es
Owner            : TFG\Domain Admins
Id               : 45b9d5a8-7ac2-4694-99c4-a67e4beb4f0f
GpoStatus        : UserSettingsDisabled
Description      :
CreationTime     : 6/27/2021 6:32:59 PM
ModificationTime : 6/27/2021 6:33:00 PM
UserVersion      : AD Version: 1, SysVol Version: 1
ComputerVersion  : AD Version: 1, SysVol Version: 1
WmiFilter        :

DoD Windows Server 2016 Domain Controller STIG Computer v2r1
No GPO named DoD Windows Server 2016 Domain Controller STIG Computer v2r1 was found. We will now create and import GPO.
DisplayName      : DoD Windows Server 2016 Domain Controller STIG Computer v2r1
DomainName       : tfg.es
Owner            : TFG\Domain Admins
Id               : 19e11327-b5f1-496b-b2c3-9a89bd1b0463
GpoStatus        : UserSettingsDisabled
Description      :
CreationTime     : 6/27/2021 6:33:00 PM
ModificationTime : 6/27/2021 6:33:00 PM
UserVersion      : AD Version: 1, SysVol Version: 1
ComputerVersion  : AD Version: 1, SysVol Version: 1
WmiFilter        :

DoD Windows Server 2016 Domain Controller STIG User v2r1
No GPO named DoD Windows Server 2016 Domain Controller STIG User v2r1 was found. We will now create and import GPO.
DisplayName      : DoD Windows Server 2016 Domain Controller STIG User v2r1
DomainName       : tfg.es
Owner            : TFG\Domain Admins
Id               : be835c6c-868a-43a3-88fe-e30699551a1d
GpoStatus        : ComputerSettingsDisabled
Description      :
CreationTime     : 6/27/2021 6:33:00 PM
ModificationTime : 6/27/2021 6:33:01 PM
UserVersion      : AD Version: 1, SysVol Version: 1
ComputerVersion  : AD Version: 1, SysVol Version: 1
WmiFilter        :
```

Figura 22. Importación de políticas de DISA

Una vez importadas las políticas de grupo, debemos aplicar enlace en el dominio de forma que aplique al actual sistema. Esto dependiendo de la estructura de directorio activo de la empresa será necesario aplicar el enlace en una determinada unidad organizativa u otra (McCabe, 2016).

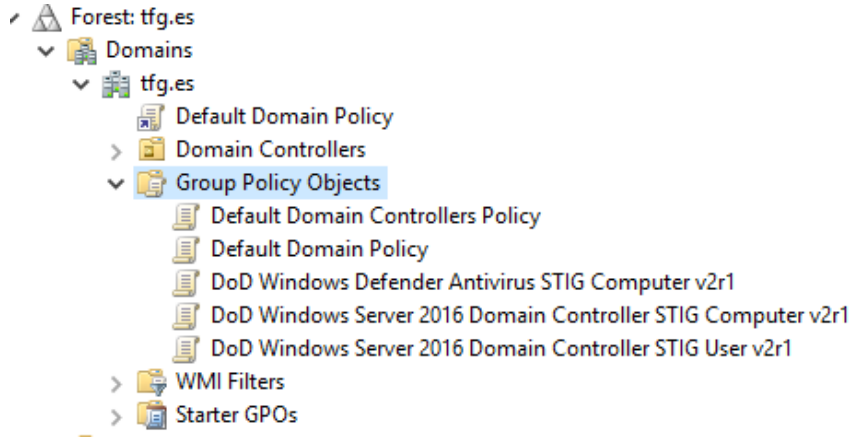


Figura 23. Estructura de políticas en dominio tras importación de políticas

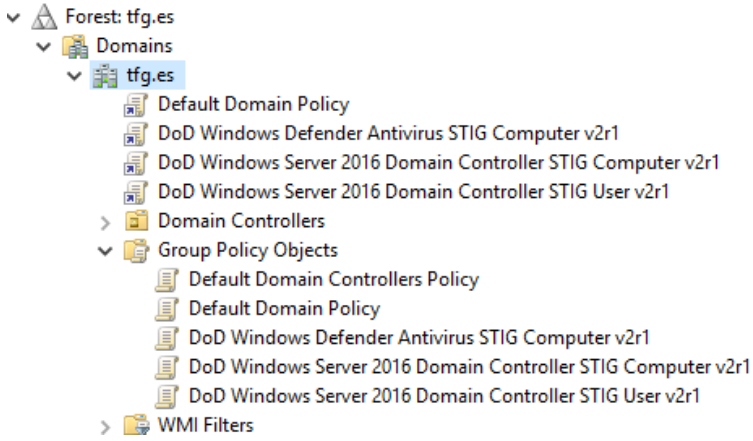


Figura 24. Estructura correcta de políticas en dominio

Además, veremos que la herramienta crea un directorio en la raíz del código fuente llamada “Logs” donde almacenará un registro de la ejecución.

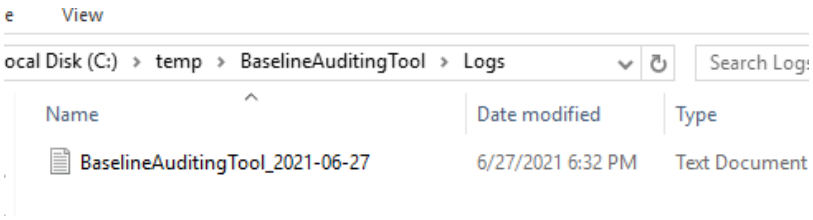


Figura 25. Registro de ejecución de la herramienta

Tras realizar esta automatización, ya tendremos cierto cumplimiento de DISA, sin embargo, no será completo.

Para continuar con el correcto bastionado, se deben realizar algunas modificaciones manuales que corrigen algunos ítems.

Por una parte, tendremos que revisar que el usuario “Administrator” integrado y “Guest” hayan sido renombrados y, además, no contengan la propiedad de que la contraseña nunca expire. Mantener ciertas cuentas de usuario por defecto como estas, constituye un problema de seguridad en la infraestructura. Podemos obtener más información acerca de estos aspectos en (Warner, 2016)

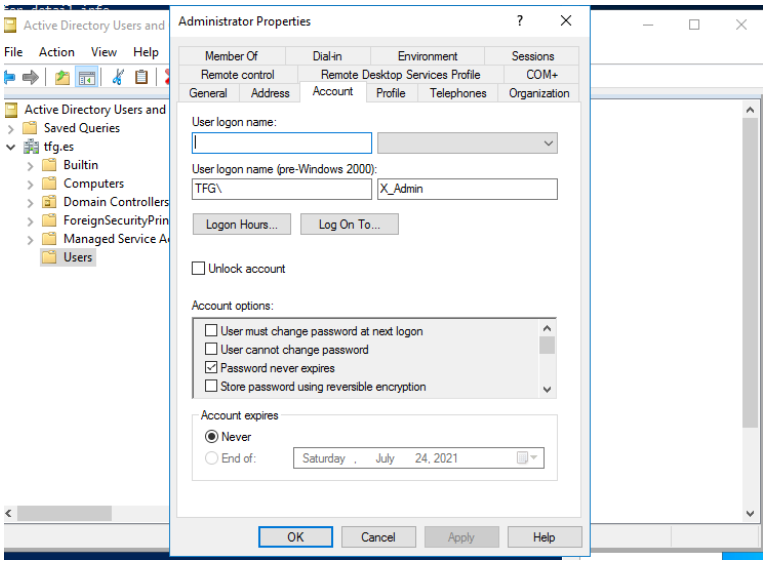


Figura 26. Modificación de usuario Administrator para cumplimiento

Además, debemos confirmar que algunos usuarios integrados como “krbtgt” hayan renovado la contraseña en un periodo inferior a un año.

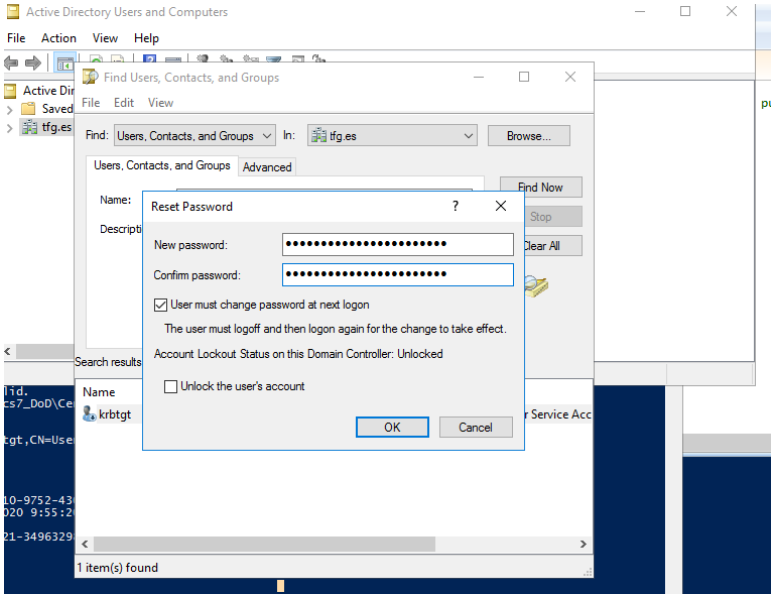


Figura 27. Modificación de usuario krbtgt para cumplimiento

Por otro lado, tendremos que configurar la auditoria de ciertos contenedores y unidades organizativas del directorio activo para recopilar eventos en caso de fallo (Scarfone, Jansen, Department of Commerce, Tracy, & Commerce, 2008).

Por tanto, tendremos que configurar la auditoria “Fail” con permiso “Full Control” en los siguientes elementos:

- En la raíz del dominio.
- Unidad organizativa de los controladores de dominio.
- Contenedor “AdminSDHolder”.
- Contenedor “RID Manager$”.
- Contenedor “Infrastructure”

Además, necesitamos configurar esto mismo en todas las políticas de grupo existentes en el directorio activo. Para ello, procedemos a realizarlo desde “ADSI Edit” sobre “CN=System,CN=Policies” (Microsoft, Windows Server Security Guide, 2017).

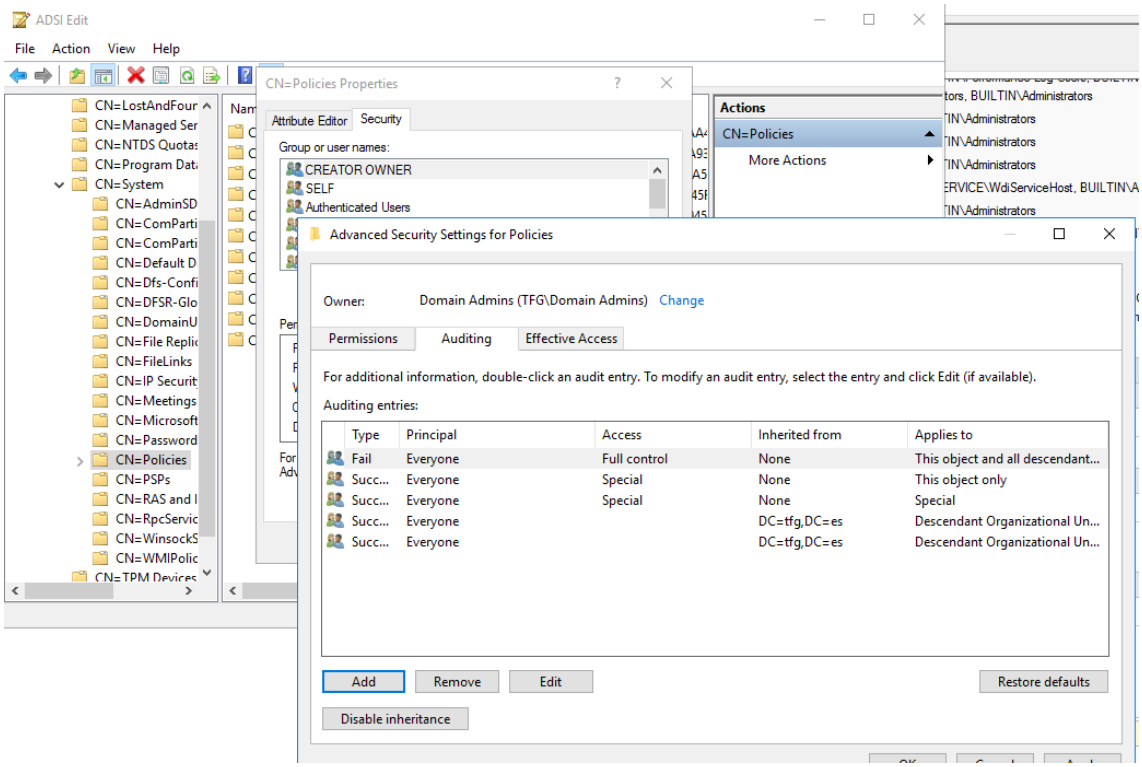


Figura 28. Modificación de auditoria de contenedores en dominio

Tras esto, habremos conseguido un correcto bastionado de Windows Server, en el que podremos ver en el apartado Análisis de seguridad posterior a la mejora de seguridad del sistema el resultado de dicha implementación a nivel de cumplimiento, habiendo conseguido un **91,58%,** teniendo en cuenta que hay ciertos ítems que no suponen un problema de seguridad, sino que simplemente deben ser revisados para confirmar que la forma en la que está implementado en la entidad no supone un problema de seguridad.

6. Análisis de nivel de seguridad antes y después del bastionado mediante Nessus

6.1. Análisis de seguridad previo a la mejora de seguridad del sistema

6.1.1. Escaneo básico

Inicialmente, realizamos un escaneo básico de la máquina controlador de dominio para ver las vulnerabilidades presentadas en la base de datos de Tenable.

En esta máquina recién instalada con la imagen de Windows Server 2016 oficial de Microsoft y añadidos los roles de DNS y DC, presenta 2 vulnerabilidades en el escaneo básico y 54 elementos a revisar.

Figura 29. Escaneo básico del servidor previo al bastionado

6.1.2. Cumplimiento de DISA

Respecto a la guía técnica de seguridad de DISA para Windows Server 2016, se detectan un total de 111 vulnerabilidades críticas, 48 advertencias y 124 ítems configurados correctamente. Esto equivale en porcentaje a un 44% de cumplimiento, 39% de fallos de seguridad y 17% de advertencias.

Por tanto, respecto a la guía técnica de seguridad de DISA para Windows Server 2016, con una instalación por defecto, obtenemos un nivel de cumplimiento de 44%.

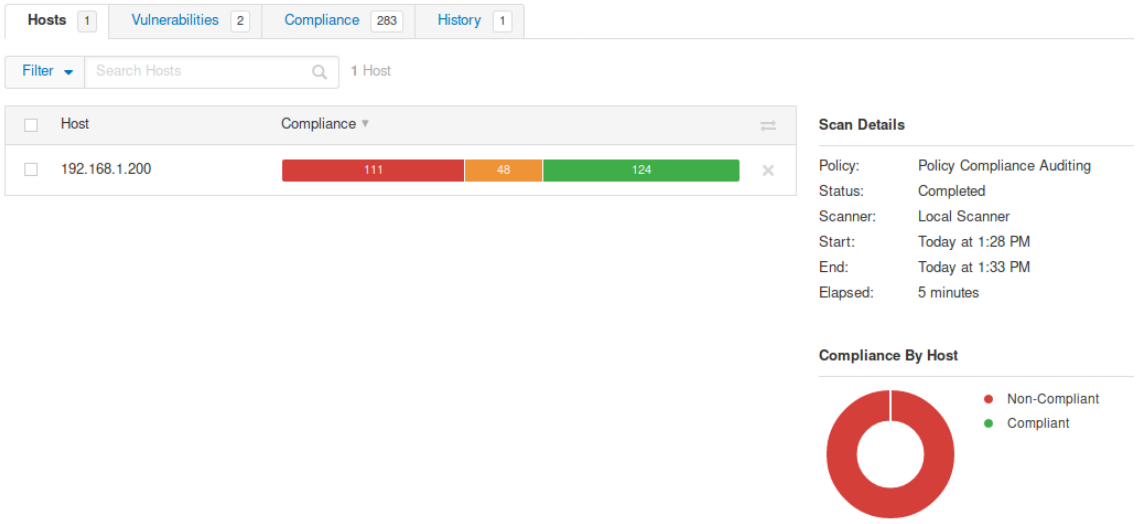


Figura 30. Medición de cumplimiento DISA mediante Nessus previo al bastionado

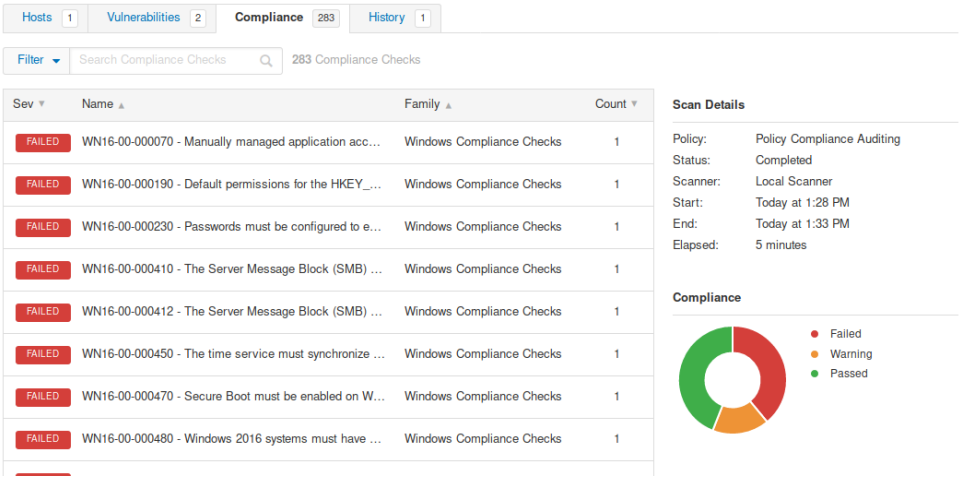


Figura 31. Medición de cumplimiento DISA mediante Nessus previo al bastionado - información

6.1.3. Cumplimiento de CIS

Respecto a la guía técnica de seguridad de CIS para Windows Server 2016 Domain Controllers L2, se detectan un total de 63 vulnerabilidades y 4 ítems configurados correctamente.

Por tanto, respecto a la guía técnica de CIS para WS 2016 Domain Controllers L2, con una instalación por defecto, obtenemos un nivel de cumplimiento de aproximadamente 6%.

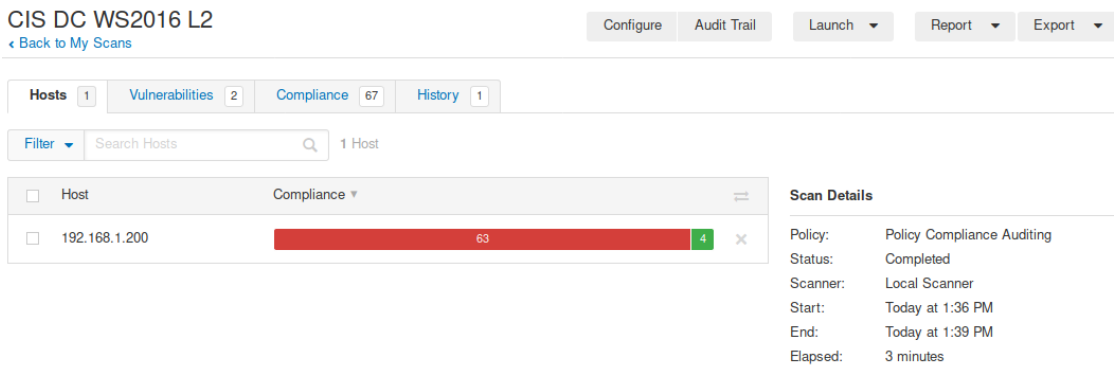


Figura 32. Medición de cumplimiento CIS mediante Nessus previo al bastionado

6.2. Análisis de seguridad posterior a la mejora de seguridad del sistema

6.2.1. Escaneo básico

Una vez realizado el correcto bastionado del servidor, repetimos los escaneos realizados con la herramienta de Nessus para observar las diferencias presentadas tras el bastionado.

Inicialmente, repetimos un escaneo básico de la máquina controlador de dominio y vemos que presenta 52 elementos a revisar. Por tanto, vemos que las 2 vulnerabilidades que presentaba anteriormente han sido mitigadas, y quedarían los elementos que deben ser revisado para confirmar que no constituya un problema de seguridad.

Figura 33. Escaneo básico del servidor tras bastionado

6.2.2. Cumplimiento de DISA

Tras el escaneo básico, procedemos a la revisión del cumplimiento de DISA, primero según Nessus y posteriormente, según la herramienta desarrollada propiamente.

Primero, realizamos el análisis de cumplimiento con la herramienta de Nessus, y observamos el siguiente resultado:

7 vulnerabilidades críticas, 40 advertencias y 238 ítems configurados correctamente. Esto equivale en porcentaje a un 87’17% de cumplimiento, 2’56% de fallos de seguridad y 14’65% de advertencias.

El motivo de las vulnerabilidades activas es debido a una parte por la virtualización mediante VirtualBox, y por otra, a interpretaciones de elementos que dependen de la entidad. Las advertencias son indicaciones que deben ser revisadas por la entidad, ya que según esté descrito en los procedimientos, puede constituir una vulnerabilidad o no.

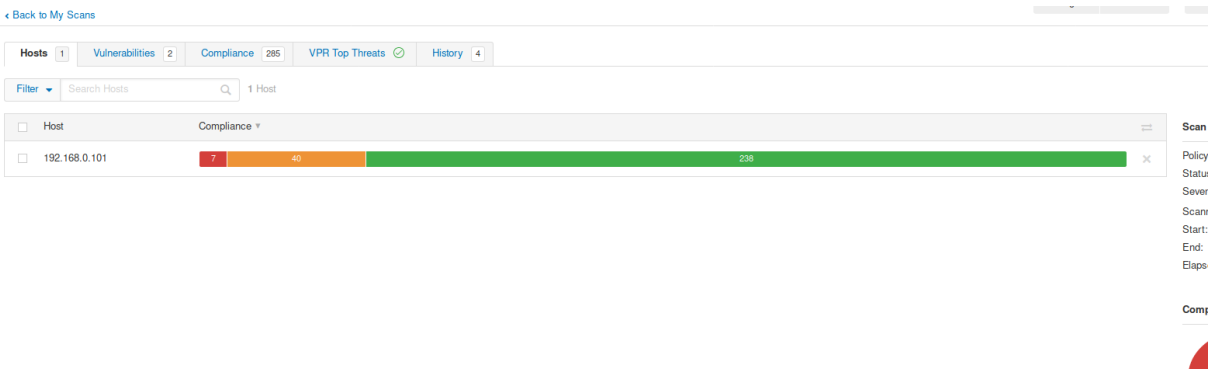


Figura 34. Medición de cumplimiento DISA mediante Nessus tras bastionado

Si auditamos el sistema mediante la herramienta desarrollada y descrita en el apartado Creación de herramienta de auditoria, obtenemos los siguientes resultados:

2 vulnerabilidades críticas, 21 advertencias y 250 ítems configurados correctamente. Esto equivale en porcentaje a un 91’58% de cumplimiento, 0’73% de fallos de seguridad y 7’69% de advertencias.

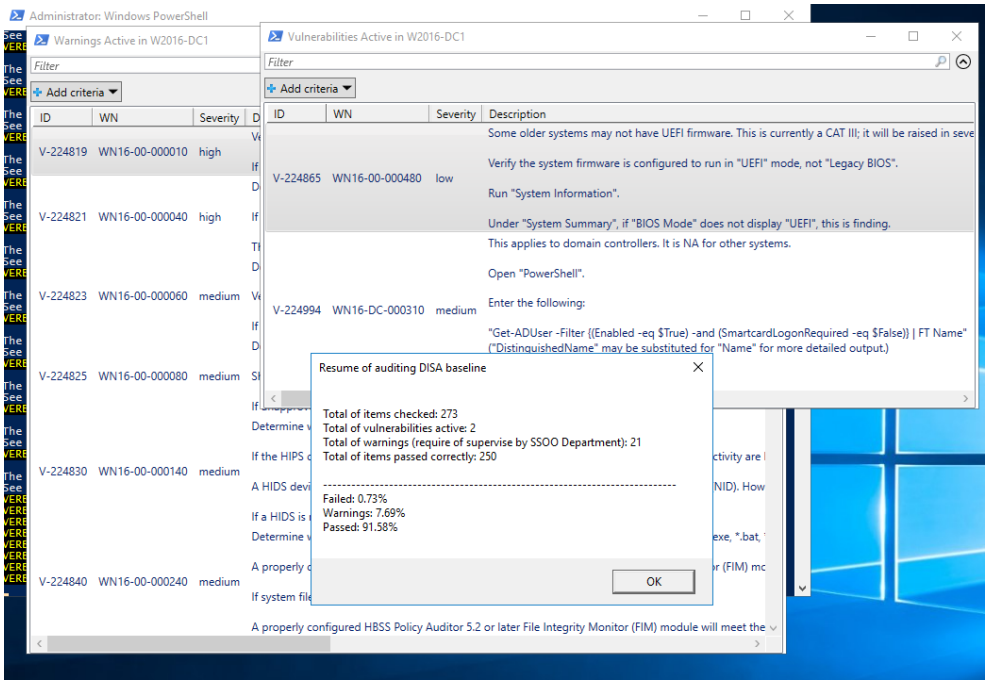


Figura 35. Medición de cumplimiento de DISA mediante herramienta propia tras bastionado

Como vemos, la herramienta de auditoria desarrollada, es capaz de mejorar la auditoria, ya que permite interpretar y analizar mayor cantidad de elementos que Nessus cataloga como advertencias y no puede analizar, y permite mejorar la eficacia en la verificación de algunos elementos.

6.2.3. Cumplimiento de CIS

Respecto a la guía técnica de seguridad de CIS para Windows Server 2016 Domain Controllers L2, tras el bastionado del servidor, se detectan un total de 35 elementos informativos. Estos elementos, aunque no suponen una vulnerabilidad, deben ser revisados, ya que en su mayoría dependen de la estructura de la organización o compañía.

Figura 36. Medición de cumplimiento CIS mediante Nessus tras bastionado

Como vemos, aun implementando el bastionado mediante DISA, se corrigen las vulnerabilidades descritas por la guía técnica de CIS, ya que comparten elementos, quedando únicamente los elementos que se deben revisar manualmente según Nessus.

7. Desarrollo de herramienta de auditoria de bastionado

Le herramienta de auditoria creada está basada en Powershell íntegramente (Lemesle & Petitjean, 2018). Esto tiene la ventaja de que, nos permitirá escalar la herramienta de auditoria para su utilización tanto en Windows como en Linux, ya que Powershell puede ser instalado en Linux, y nos permite obtener ventajas respecto a Bash en cuanto a la salida de comandos, ya que Powershell devuelve objetos y no cadenas de texto como Bash.

Aunque en un principio el objetivo únicamente era utilizar dicha herramienta para la auditoria, se ha introducido en este script la posibilidad de automatizar la parte que hay posibilidad de automatizar dentro de la aplicación del bastionado.

Cabe destacar que, en todo momento durante el desarrollo de la programación de este script, se ha tenido en cuentas las buenas prácticas recomendadas por Microsoft para el scripting en entornos compartidos (Microsoft, Buenas prácticas en scripting Powershell en entornos compartidos, 2014).

7.1. Creación de herramienta de auditoria

El desarrollo de la herramienta se ha realizado siguiendo una premisa básica: la generalización de código. Esto significa que, partiendo de la entrada de un archivo que corresponderá con cada una de las guías técnicas de seguridad, se intentará obtener cada uno de los elementos que compone dicha guía de forma automatizada.

Además, se intentará generalizar las funciones para que cumplan un objetivo común en el análisis de los elementos.

Las fases o partes del procedimiento de programación de la herramienta han sido las siguientes:

- Análisis manual de cada uno de los elementos.
- Agrupación y generalización de análisis de los elementos.
- Comparación con reporte de la herramienta de terceros Nessus, lo cual permite detectar fallos en el análisis de elementos.
- Una vez la herramienta está desarrollada y proporciona resultados similares a herramientas de terceros, se abordan elementos que estas herramientas no son capaces de verificar o que verifican de forma incorrecta, reduciendo así el número de elementos que dependen de la estructura de la organización.

La estructura interna del script tiene las siguientes partes o bloques, siguiendo las buenas prácticas de Microsoft citadas anteriormente.

- Cabecera y parámetros de entrada.
- Bloque de funciones.
- Bloque de inicializaciones.
- Bloque de ejecución.

Respecto al bloque de cabecera y parámetros, en esta se incluye una pequeña descripción de la funcionalidad de la herramienta, otras indicaciones como la localización de registros (logs) que recogerán las ejecuciones de la herramienta, y un pequeño control de cambios.

El bloque de funciones contendrá todas las funciones necesarias para la ejecución de la herramienta, con el fin de generalizar las comprobaciones de los distintos elementos. Algunas de estas funciones son:

- Comprobación de estado correcto de características de Windows.
- Comprobación de tipo de equipo (controlador de dominio, servidor miembro, equipo de escritorio).
- Obtención de políticas de restricción de permisos de usuario y tratado de las cadenas de datos para obtener el valor buscado.
- Obtención de políticas de restricción de seguridad en el equipo y tratado de las cadenas de datos para obtener el valor buscado.
- Pasado un elemento de configuración, se comprueba si este se trata de un tipo “clave de registro” y se trata la cadena de texto para obtener las rutas y valores a comprobar.

Debido a que normalmente, la implementación de las guías técnicas de seguridad se realiza de forma manual, para el análisis de los elementos, se debe tratar la información introducida en bruto con un XML mediante expresiones regulares que nos permitan automatizar los tipos de elementos que debemos analizar, y los valores correctos o incorrectos.

Por último, el bloque de ejecución simplemente recorrerá el archivo de la guía técnica de seguridad, llamará a las funciones que realizarán las comprobaciones y tratará la muestra de resultados.

La muestra de resultados se realiza mediante una simple ventana que dará el resumen de cada elemento, y el porcentaje de cumplimiento, otorgará ventanas para la correcta visualización de cada uno de los elementos (vulnerabilidades, advertencias) con el fin de que puedan ser revisadas correctamente, y finalmente dejará almacenado en el propio directorio del código fuente dos CSV donde se recogerán de forma redundante los elementos analizados y el resultado de estos.

Como se comenta anteriormente, se han seguido las buenas prácticas definidas por Microsoft para el scripting mediante Powershell, en el que se incluye la globalización del código, por lo que todo el código ha sido escrito utilizando nombres de variables, salida de resultados etc en inglés. Únicamente se han mantenido los comentarios de las distintas funciones en castellano, ya que pueden dejarse indicados en varios idiomas.

El código fuente de la herramienta puede consultarse en el Anexo V. Código fuente de herramienta de auditoria.

7.2. Muestra de ejecución y medición mediante herramienta

A continuación, se muestran varios ejemplos de ejecución de la herramienta de auditoria. Como se indicó en el apartado anterior “Creación de herramienta de auditoria”, la herramienta tiene dos funcionalidades: implementación de bastionado, y auditoria de bastionado.

En este caso, se mostrarán ejemplos sin recibir parámetros en el script, por lo que ejecutará la funcionalidad de auditoria.

En primera instancia, se observa la ejecución de la herramienta con una implementación del servidor por defecto, con los servicios indicados en el apartado 4.1 Windows Server 2016. Como se puede observar, se obtiene un 48% de cumplimiento.

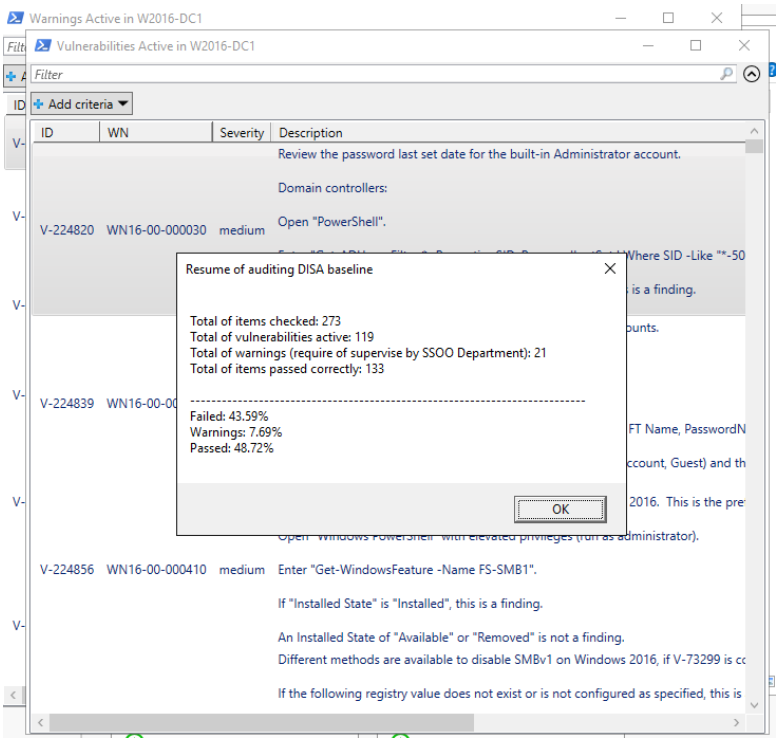


Figura 37. Medición de cumplimiento de DISA con herramienta propia previo al bastionado

A continuación, se procede a mostrar la ejecución de la herramienta una vez implementadas las políticas de grupo ofrecidas por DISA.

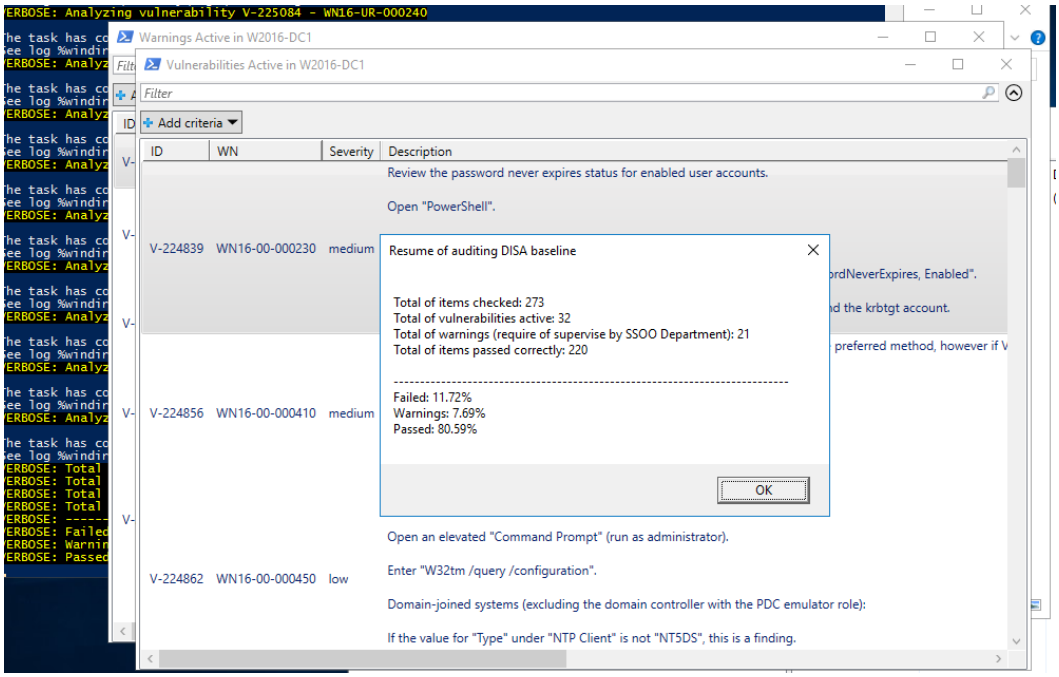


Figura 38. Medición de cumplimiento de DISA con herramienta propia tras importación de políticas

Finalmente, se muestra una ejecución de la herramienta una vez implementado el bastionado completamente. Como se indicó en el apartado

Implementación de bastionado, en este caso, por tratarse de virtualización en VirtualBox, quedarán ítems que incumplen.

Además, estarán vigentes las 21 advertencias que dependerán de la estructura y documentación de la organización para que cumplan o no estos. Estas advertencias constituyen elementos que no pueden ser analizados por la herramienta, ya que, son cuentas de usuario, aplicaciones etc específicas de cada organización, y, por tanto, deben ser revisadas por el departamento correspondiente de la organización.

Si alguna aplicación específica de negocio necesita ciertos permisos específicos de ejecución, por ejemplo, no constituirá un problema de seguridad, siempre y cuando esté recogido en la documentación y en los informes de evaluación de riesgos.

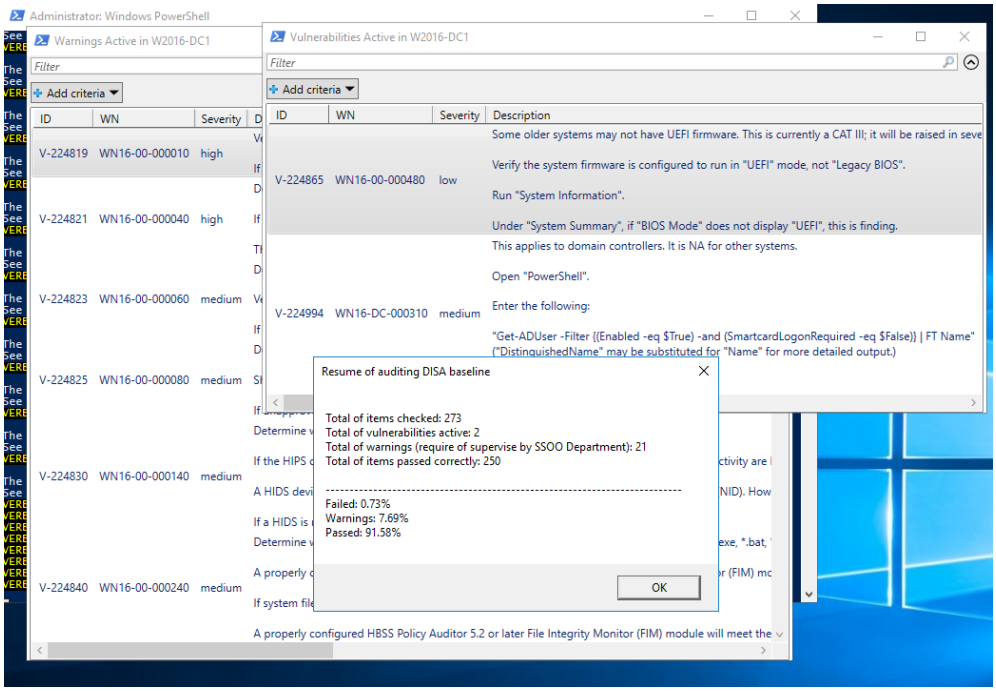


Figura 39. Medición de cumplimiento de DISA con herramienta propia tras bastionado

8. Revisión, corrección y ajuste de nivel de medición de la herramienta

Con el objetivo de mejorar la eficacia de la herramienta desarrollada, se ha realizado una revisión en los diferentes estados del servidor (sin bastionado, con algunos elementos aplicados, y completamente aplicado) utilizando como comparativa la herramienta de Nessus.

Como se observa en el siguiente caso, había un desfase en las verificaciones correctas respecto a la herramienta de Nessus, por lo que se han hecho revisiones en el código fuente.

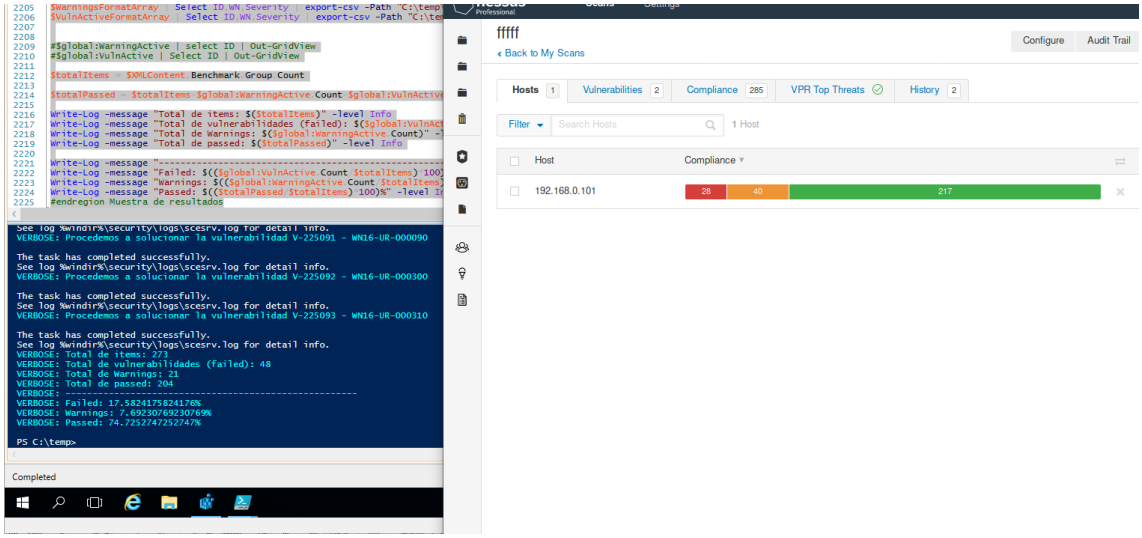


Figura 40. Corrección y ajuste de medición - previo

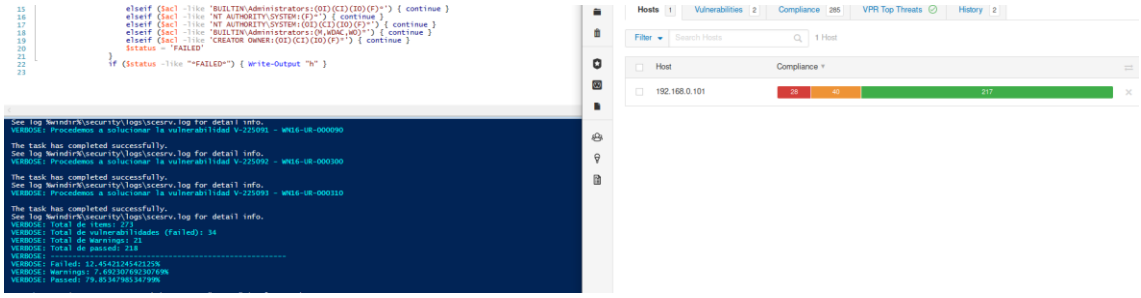


Figura 41. Corrección y ajuste de medición - posterior

9. Conclusiones

En la actualidad, las entidades cada vez toman más conciencia de la necesidad de una correcta gestión de riesgos de seguridad. Esto provoca que, sea necesaria la constante monitorización del estado de seguridad de los diferentes elementos de la infraestructura, ya sea servidores, equipos cliente, servicios, aplicaciones etc.

Aunque las distintas organizaciones descritas en el presente proyecto proveen de guías técnicas de seguridad para implementar una correcta configuración de seguridad, no proveen de una opción viable de auditoria de estas. Este hueco de mercado ha sido aprovechado por algunas empresas que en los últimos años han crecido exponencialmente, realizando herramientas de auditoria de cumplimiento de estas guías técnicas.

Sin embargo, como hemos visto en el presente proyecto, las herramientas de terceros, escritas normalmente en lenguaje propio y de difícil acceso, genera distintos problemas a la hora de la modificación de algunos chequeos y verificaciones que deben ser adaptadas por las diferentes empresas. A esto se suma el alto presupuesto que estas empresas demandan por la contratación y mantenimiento de estos servicios.

Por tanto, actualmente, salvo proyectos abandonados, no hay proyectos de software comunitario que provean de la necesidad de una herramienta de auditoria de cumplimiento de las guías técnicas de seguridad.

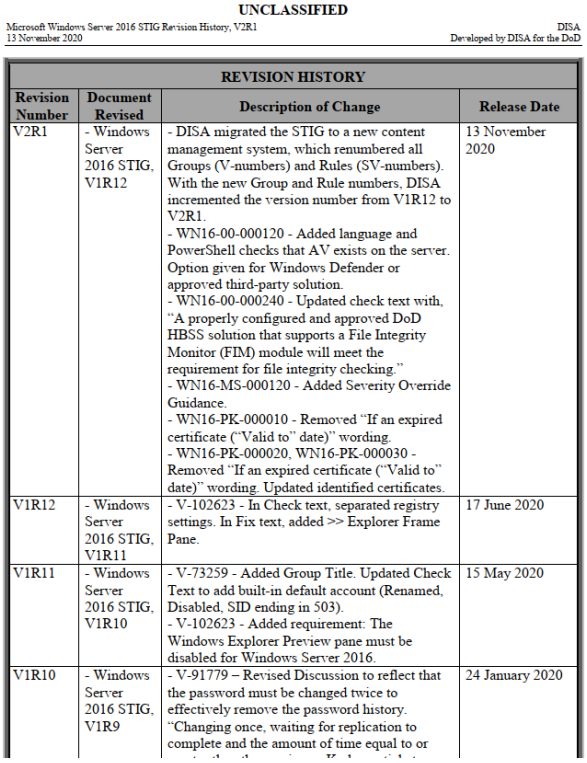
En este proyecto, en primera instancia se ha afrontado un marco teórico de la necesidad de implementación de guías técnicas de seguridad, y su auditoria. Seguidamente, se ha conseguido afrontar el desarrollo de una herramienta de auditoria para estas guías técnicas, que como hemos visto, es capaz de auditar el sistema con un margen de error bastante más pequeño que el provisto por software de terceros como Nessus.

10. Trabajo futuro

La elaboración de la herramienta de auditoria de bastionado descrita en este proyecto es actualmente válida, pero debe ser actualizada conforme las guías de bastionado de la organización correspondiente se actualicen.

La actualización de estas guías suele ser cada 90 días. Aunque en primera instancia únicamente sería necesaria la descarga del archivo XML correspondiente a la nueva versión, y su copiado al directorio del código fuente del proyecto, es cierto que, hay ciertos ítems que con corregidos o modificados por las organizaciones.

Este aspecto podemos verlo en cada nueva versión (NIST, Microsoft Windows Server 2016 Checklist Revisions, 2021), en un documento llamado "U_MS_Windows_Server_2016_V2R1_Revision_History", donde se indicarán los cambios realizados en cada versión.



UNCLASSIFIED

Microsoft Windows Server 2016 STIG Revision History, V2R1 — DISA
13 November 2020 — Developed by DISA for the DoD

REVISION HISTORY

Revision Number	Document Revised	Description of Change	Release Date
V2R1	- Windows Server 2016 STIG, V1R12	- DISA migrated the STIG to a new content management system, which renumbered all Groups (V-numbers) and Rules (SV-numbers). With the new Group and Rule numbers, DISA incremented the version number from V1R12 to V2R1. - WN16-00-000120 - Added language and PowerShell checks that AV exists on the server. Option given for Windows Defender or approved third-party solution. - WN16-00-000240 - Updated check text with, "A properly configured and approved DoD HBSS solution that supports a File Integrity Monitor (FIM) module will meet the requirement for file integrity checking." - WN16-MS-000120 - Added Severity Override Guidance. - WN16-PK-000010 - Removed "If an expired certificate ("Valid to" date)" wording. - WN16-PK-000020, WN16-PK-000030 - Removed "If an expired certificate ("Valid to" date)" wording. Updated identified certificates.	13 November 2020
V1R12	- Windows Server 2016 STIG, V1R11	- V-102623 - In Check text, separated registry settings. In Fix text, added >> Explorer Frame Pane.	17 June 2020
V1R11	- Windows Server 2016 STIG, V1R10	- V-73259 - Added Group Title. Updated Check Text to add built-in default account (Renamed, Disabled, SID ending in 503). - V-102623 - Added requirement: The Windows Explorer Preview pane must be disabled for Windows Server 2016.	15 May 2020
V1R10	- Windows Server 2016 STIG, V1R9	- V-91779 – Revised Discussion to reflect that the password must be changed twice to effectively remove the password history. "Changing once, waiting for replication to complete and the amount of time equal to or	24 January 2020

Figura 42. Cambios por revisión de DISA

En este último caso, por ejemplo, hubo cambio en 5 ítems que habrá que revisar, pero en versiones anteriores únicamente hubo cambio en 1-2 ítems.

Esto podría ser fácilmente mantenido subiendo el código a GitHub y estableciendo como un proyecto de software libre en el que se pueda participar abiertamente corrigiendo errores y actualizando con dichas versiones.

De esta forma, provee a las empresas de una forma de auditar los servidores sin suponer un coste adicional contratando software como Nessus con un presupuesto alto, y que como hemos visto, contiene diferentes problemas asociados.

Por otra parte, en este proyecto, únicamente se ha abordado la implementación y auditaje de la guía técnica de la organización más relevante (DISA), sin embargo, un punto de mejora sería ampliar dicha herramienta para la implementación de otras guías de cumplimiento de otras organizaciones como CIS.

Bibliografía

Defense information Systems Agency. (2010). *STIGs, SCAP and Data Metrics*. Obtenido de https://www.disa.mil/~/media/files/disa/news/conference/cif/briefing/ia_stig_scap_and_data_metrics.pdf

Department of Defense. (2003). *Number 8500.2*. Obtenido de https://fas.org/irp/doddir/dod/d8500_2.pdf

Department of Defense. (2014). *Number 8500.1*. Obtenido de https://www.esd.whs.mil/portals/54/documents/dd/issuances/dodi/850001_2014.pdf

INCIBE-CERT. (26 de 02 de 2019). *La importancia del bastionado de sistemas*. Obtenido de https://www.incibe-cert.es/blog/importancia-bastionado-sistemas

Insights, H. (06 de 03 de 2021). *Companies using Nessus*. Obtenido de https://discovery.hgdata.com/product/tenable-nessus

Kali. (02 de 03 de 2021). *About Kali Linux*. Obtenido de https://www.kali.org/docs/introduction/what-is-kali-linux/

Kaspersky. (2016). Obtenido de Report: Measuring the Financial Impact of IT Security on Businesses: https://www.kaspersky.com/blog/security_risks_report_financial_impact/#report

Krause, J. (2016). Mastering Windows Server 2016. En J. Krause. Van Haren Publishing.

Lemesle, R., & Petitjean, A. (2018). *PowerShell Core y Windows PowerShell.* Ediciones ENI.

McCabe, J. (2016). *Introducing Windows Server 2016 Technical Preview.* Pearson Education.

Microsoft. (17 de 05 de 2014). *Buenas prácticas en scripting Powershell en entornos compartidos*. Obtenido de https://devblogs.microsoft.com/scripting/weekend-scripter-best-practices-for-powershell-scripting-in-shared-environment/

Microsoft. (31 de 05 de 2017). *Introducción a Active Directory Domain Services*. Obtenido de https://docs.microsoft.com/es-es/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview

Microsoft. (08 de 2017). *Windows Server Security Guide.* Obtenido de https://download.microsoft.com/download/5/8/5/585DF9E9-D3D6-410A-8B51-81C7FC9A727C/Windows_Server_2016_Security_Guide_EN_US.pdf

Microsoft. (22 de 03 de 2021). *¿Qué es PowerShell?* Obtenido de https://docs.microsoft.com/es-es/powershell/scripting/overview?view=powershell-7.1

Microsoft. (02 de 05 de 2021). *Center for Internet Security (CIS) Benchmarks*. Obtenido de https://docs.microsoft.com/en-us/compliance/regulatory/offering-CIS-Benchmark?view=o365-worldwide

Microsoft. (2021). *Productos Windows Server y recursos*. Obtenido de https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server-2016

NIST. (2021). *Microsoft Windows Server 2016 Checklist Revisions*. Obtenido de https://ncp.nist.gov/checklist/753/archive

NIST. (2021). *Microsoft Windows Server 2016 STIG Checklist Details*. Obtenido de https://ncp.nist.gov/checklist/revision/4012

Oracle. (2020). *Creación de máquina virtual en VirtualBox*. Obtenido de https://docs.oracle.com/cd/E26217_01/E26796/html/qs-create-vm.html

Scarfone, K., Jansen, W., Department of Commerce, U., Tracy, M., & Commerce, U. (2008). *Guide to General Server Security.* Createspace Independent Pub.

Security, Center of Internet. (2020). *Mapping and Compliance*. Obtenido de https://www.cisecurity.org/cybersecurity-tools/mapping-compliance/

Security, I. (04 de 12 de 2018). *Server Hardening Standard (Windows)*. Obtenido de https://security.uconn.edu/server-hardening-standard-windows/

Security, O. (01 de 02 de 2021). *Máquina virtual Kali Linux*. Obtenido de Máquina virtual Kali Linux: https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/#1572305786534-030ce714-cc3b

Statista. (13 de 5 de 2020). *Distribución de mercado en base a sistema operativo servidor en 2018 y 2019*. Obtenido de https://www.statista.com/statistics/915085/global-server-share-by-os/

Stöckle, P., Grobauer, B., & Pretschner, A. (2020). *Automated Implementation of Windows-related Security-Configuration Guides*. Obtenido de https://i4.pages.gitlab.lrz.de/conferences-public/preprints/2020/ASE/automated-implementation-of-windows-related-security-configuration-guides.pdf

Tenable. (02 de 04 de 2021). *Nessus Professional Datasheet*. Obtenido de https://static.tenable.com/marketing/datasheets/Datasheet-Nessus_Professional_es-la.pdf

Tenable. (05 de 03 de 2021). *Tenable - Descargas*. Obtenido de https://www.tenable.com/downloads/nessus?loginAttempted=true

Vacca, J. R. (2013). *Cyber Security and IT Infrastructure Protection.* Elsevier Gezondheidszorg.

Warner, T. &. (2016). *Exam Ref 70-744 Securing Windows Server 2016.* Pearson Education.

Anexo I. Creación de máquina virtual Windows Server en VirtualBox.

En el actual Anexo se muestra el proceso de creación de una máquina virtual Windows Server 2016 en VirtualBox.

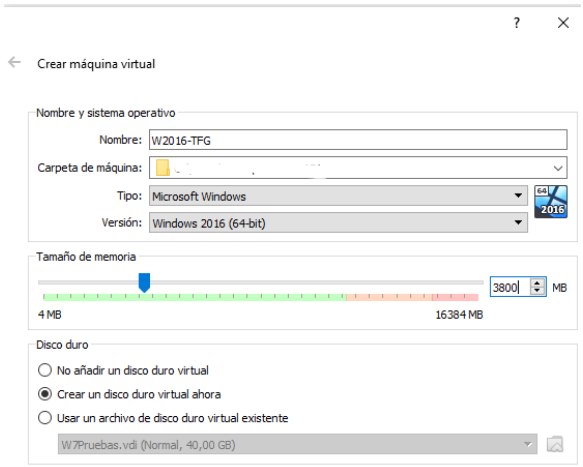


Figura 43. Creación de máquina virtual - Paso 1

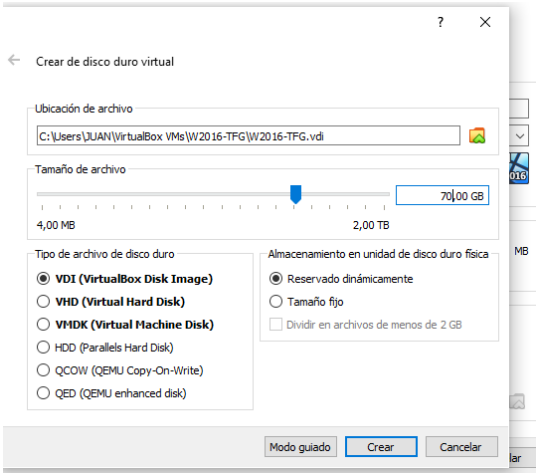


Figura 44. Creación de máquina virtual - Paso 2

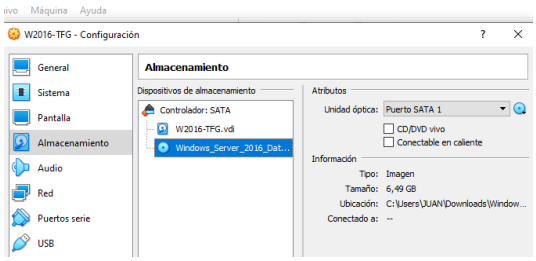


Figura 45. Creación de máquina virtual - Paso 3

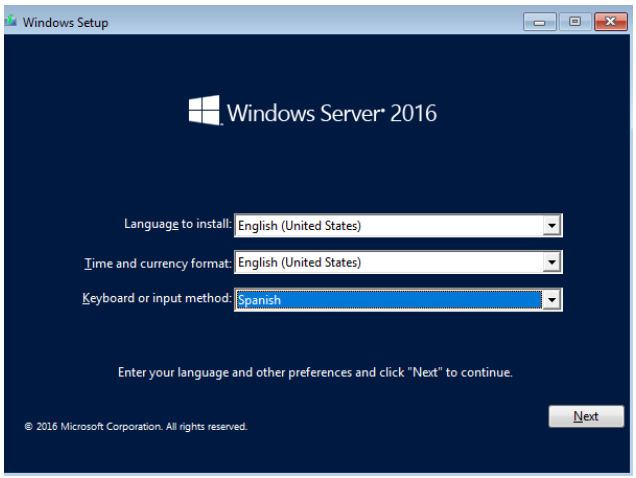


Figura 46. Creación de máquina virtual - Paso 4

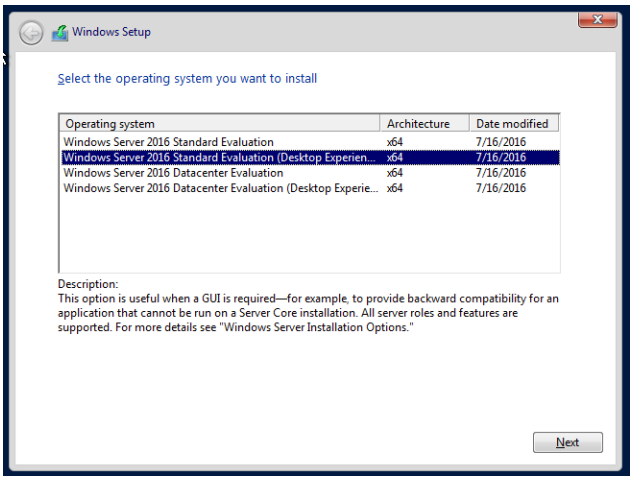


Figura 47. Creación de máquina virtual - Paso 5

Finalmente, procedemos a configurar la red en modo adaptador puente, con el que replicaremos la red a la que está conectado nuestro equipo e instalaremos las Guest Additions de VirtualBox para una mejor experiencia visual.

Anexo II. Instalación de servicios ADDS y DNS en Windows Server.

Una vez instalada la máquina virtual que actuará de servidor en el entorno de pruebas, debemos configurar la red para establecer una dirección ipv4 estática por normativa en servidores.

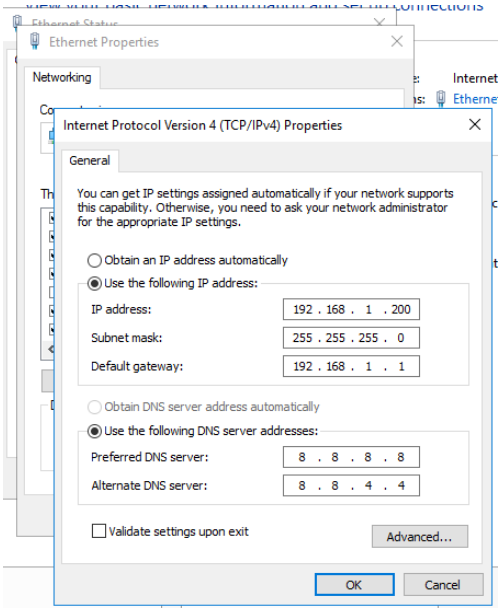


Figura 48. Configuración de IP estática Windows Server

Tras esto, procederemos a implementar algunos servicios básicos que recopilará la mayoría de los entornos empresariales. Para ello, instalaremos los servicios de DNS y ADDS.

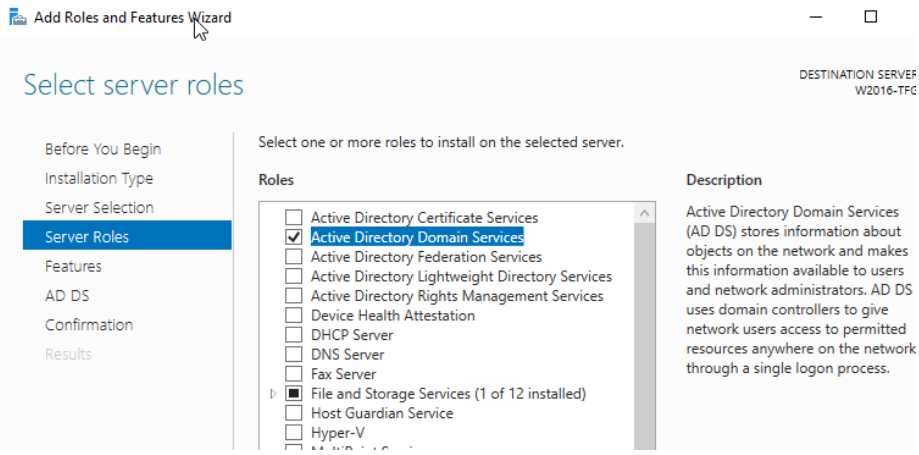


Figura 49. Instalación de servicios de ADDS y DNS - paso 1

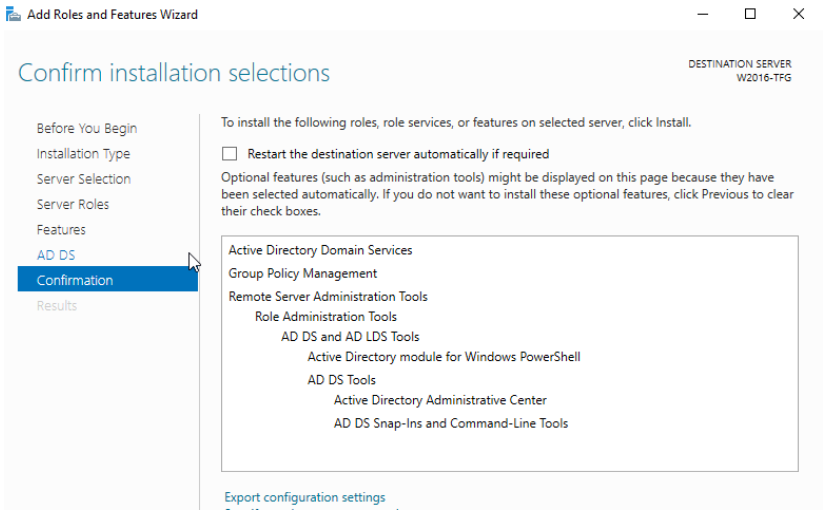


Figura 50. Instalación de servicios de ADDS y DNS - paso 2

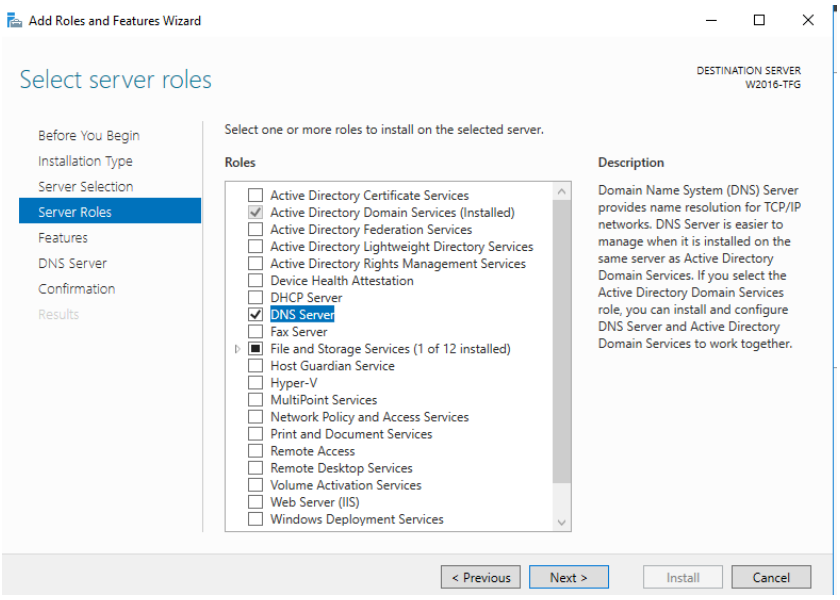


Figura 51. Instalación de servicios de ADDS y DNS - paso 3

Una vez instalados tendremos que configurar un nuevo dominio principal.

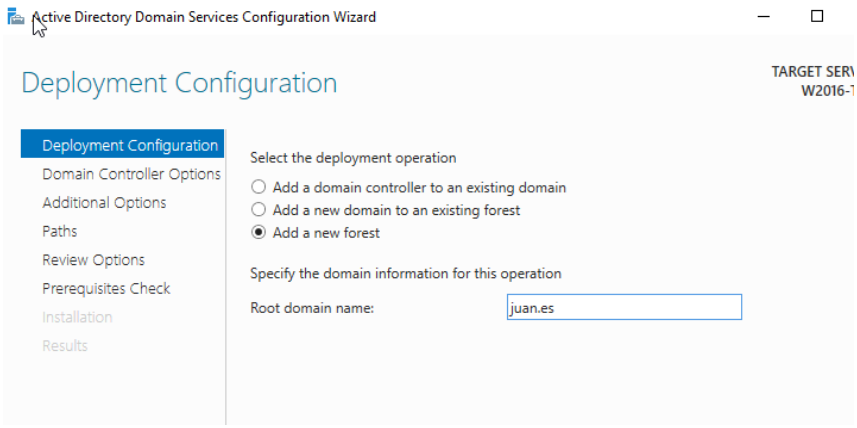


Figura 52. Instalación de servicios de ADDS y DNS - paso 5

Anexo III. Importación de Kali Linux en VirtualBox.

Para importar la máquina virtual de Kali Linux ofrecida por Offensive Security simplemente tendremos que utilizar la opción de importación de VirtualBox y seleccionar el archivo ".ova" descargado.

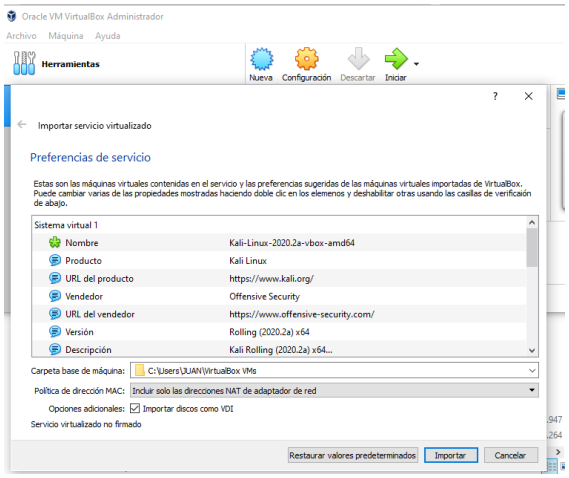


Figura 53. Importación de Kali Linux en VirtualBox - paso 1

Tras esto, al igual que con Windows Server, procedemos a poner la tarjeta de red en adaptador puente para replicar la red del equipo que alberga las máquinas virtuales, y configuramos una dirección ipv4 estática.

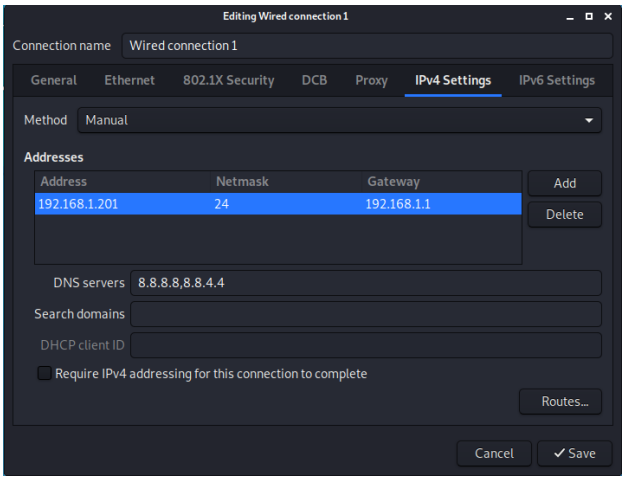


Figura 54. Configuración de IP estática en Kali Linux

Anexo IV. Instalación de Tenable Nessus en Kali Linux e inicialización.

Para ello, iremos a la página de descargas oficial de Tenable (Tenable, Tenable - Descargas, 2021) y descargaremos el paquete de Nessus para Debian.

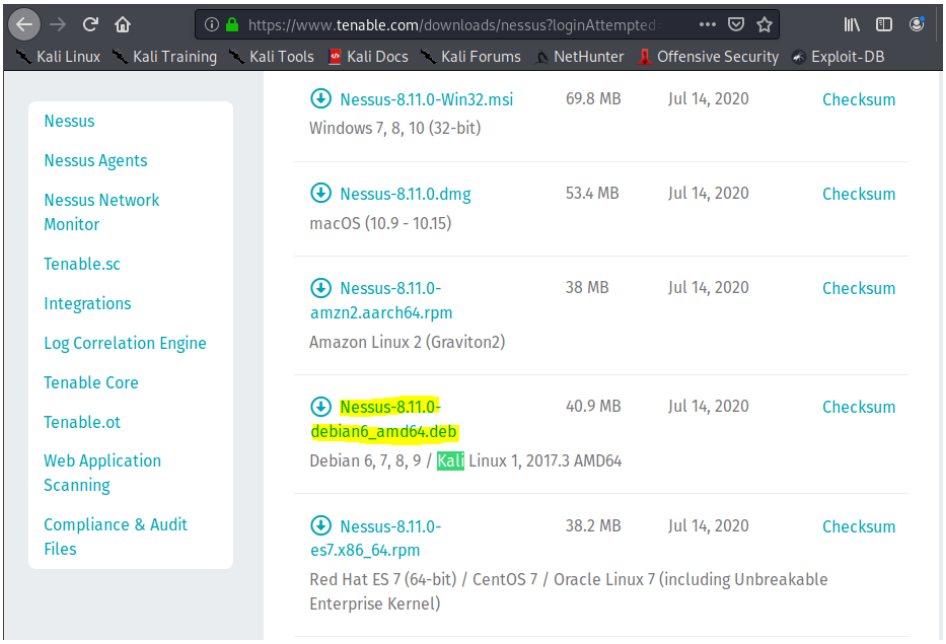


Figura 55. Instalación de Nessus en Kali

A continuación, procedemos a su instalación en Kali Linux, y la ejecución del servicio.

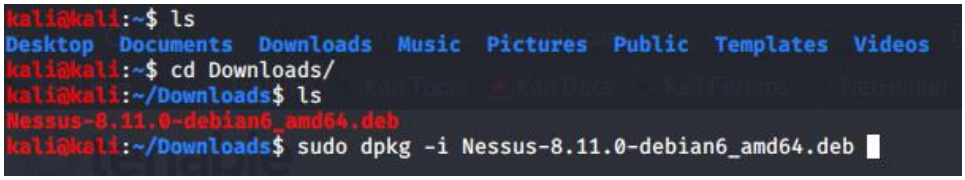


Figura 56. Instalación de Nessus en Kali - paso 2

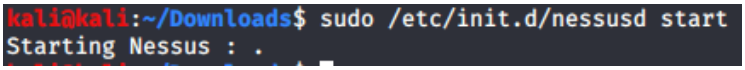


Figura 57. Ejecución de servicio Nessus

Una vez hecho esto, simplemente tendremos que registrar una prueba de 7 días en Nessus. De esta forma, tendremos 7 días para utilizar este software y poder comparar resultados con la herramienta desarollada de forma propia.

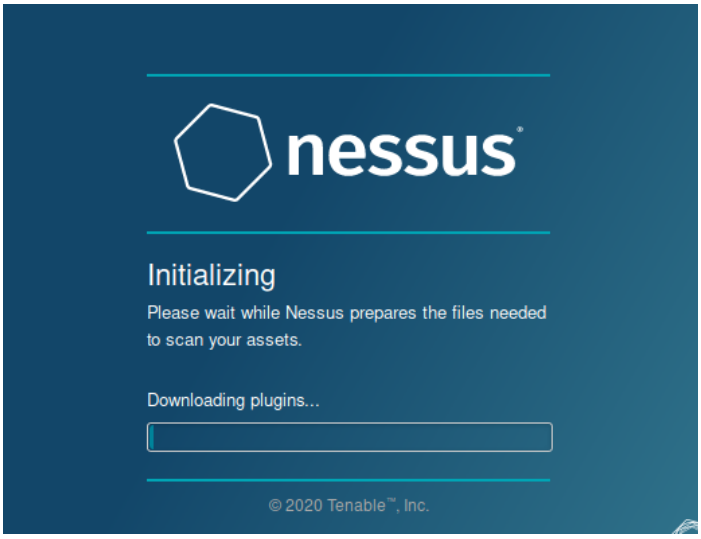


Figura 58. Inicialización y ejecución de Nessus

Anexo V. Código fuente de herramienta de auditoria.

Bloque de cabecera y parámetros

```
<#
.SYNOPSIS
  Permite implementar lineas base de DISA, CIS, etc y auditarlas
.DESCRIPTION
  Implementa o audita baseline de seguridad. Bastionado de servidores
.PARAMETER
  -Implement (optional)
.INPUTS
.OUTPUTS
  Log file stored in .\Logs\BaselineAuditingTool_<Date>.log>
.NOTES
  Version:        1.0
  Author:         Juan Merlos García
  Creation Date:  08/10/2020
  Purpose/Change: Creación inicial del script

  Change control:
                    Juan Merlos García. 26/04/2021. Se implementan las funciones necesarias vistas en función de los items a comprobar.
                    Juan Merlos García. 18/05/2021. Se implementan el bloque de ejecución.
                    Juan Merlos García. 13/06/2021. Se corrigen errores en la verificación de algunos items.
                    Juan Merlos García. 26/06/2021. Se implementa automatización para modificación de importtable.
                    Juan Merlos García. 27/06/2021. Se cambia la forma de obtención de la ruta del XML de la baseline.

.EXAMPLE
  BaselineAuditingTool.ps1 -Baseline "C:\pruebas\U_MS_Windows_Server_2016_STIG_V2R1_Manual-xccdf.xml"
#>

<#
Parametro opcional para implementar bastionado en lugar de auditar.
Si no se recibe parámetro, auditará.
Ejemplo: BaselineAuditingTool.ps1 -Implement
#>
param(
    # Parámetro que se recibe si se quiere implementar al guía de cumplimiento
    # Por defecto si no se recibe, audita (medir el nivel de cumplimiento)
    [Parameter(Mandatory=$false)]
    [switch] $Implement
)
```

Código 1. Herramientas y auditoria - Cabecera y parámetros

Bloque de funciones

```
<#
.SYNOPSIS Función con parámetros para creación de Log
.DESCRIPTION Lo único necesario es el mensaje, el resto son parametros opcionales,
    si no se envía el argumento $path escribiá en el valor por defecto del $path
.EXAMPLE Write-Log -message 'La carpeta no existe' -path c:\Logs\Script.log -level Error
    Escribe el mensaje al archivo de log especificado como un mensaje de error.
#>
function Write-Log {
    [CmdletBinding()]
    Param (
        # Mensaje
        [Parameter(Mandatory=$true,
                   ValueFromPipelineByPropertyName=$true)]
        [ValidateNotNullOrEmpty()]
        [string]$message,

        # Ruta por defecto sino se pasa como parametro
        [Parameter(Mandatory=$false)]
        [string]$path = "$($LogFile)",

        # Tipo de mensaje ("Error, Warning, Info")
        [Parameter(Mandatory=$false)]
        [ValidateSet("Error","Warn","Info")]
        [string]$level = "Info"
    )
    # Si se asigna se mostraran los log en la consola
    $VerbosePreference = 'Continue'

    # Si se va a escribir el archivo de log en una ruta que no existe, se crea la ruta.
    if (!(Test-path $path)) {
        Write-Verbose "Creando $path"
        New-Item $path -Force -ItemType File
    }

    # Formato de fecha para el log
    $formattedDate = Get-Date -Format "yyyy-MM-dd HH:mm:ss"

    # Escribir el tipo de mensaje que es en el log o en consola con $levelText
    switch ($level) {
        'Error' {
            Write-Error $message
            $levelText = 'ERROR:'
            }
        'Warn' {
            Write-Warning $message
            $levelText = 'WARNING:'
            }
        'Info' {
            Write-Verbose $message
            $levelText = 'INFO:'
            }
    }
    # Escritura del log en $path
    "$formattedDate $levelText $message" | Out-File -FilePath $path -Append
}
```

Código 2. Bloque de funciones - función de escritura de log

```powershell
<#
.SYNOPSIS Función que permite obtener la ruta del XML que definirá la guía técnica a auditar.
.DESCRIPTION No recibe parámetros
             Devuelve el valor de la ruta del XML en string
.EXAMPLE $variable = GetComputerType
#>
function Get-OptionBaseline (){
    [System.Reflection.Assembly]::LoadWithPartialName("System.windows.forms") | Out-Null
    $openfiledialog = New-Object System.Windows.Forms.OpenFileDialog
    $openfiledialog.InitialDirectory = $myfolder
    $openfiledialog.ShowDialog() | Out-Null
    #$openfiledialog.FileName

    if ($openfiledialog.FileName -eq "") {
        Write-Log -message "Select the correct XML. Exiting Script. Press any key to continue..."
        $null = $Host.UI.RawUI.ReadKey('NoEcho,IncludeKeyDown')
        Exit
    } else {
        return $openfiledialog.FileName
    }
}

<#
.SYNOPSIS Función que permite obtener información acerca de si el equipo es controlador de dominio,
          servidor miembro o equipo de escritorio
.DESCRIPTION No recibe parámetros
             Devuelve el valor del tipo de equipo en string
.EXAMPLE $variable = GetComputerType
#>
function GetComputerType {
    $ProductType = (Get-WmiObject win32_OperatingSystem).ProductType
    switch ($ProductType) {
        1 {
            $ComputerType = "Workstation"
        }
        2 {
            $ComputerType = "Domain Controller"
        }
        3 {
            $ComputerType = "Member Server"
        }
    }

    return $ComputerType
}
```

Código 3. Bloque de funciones - función de obtener guía técnica y tipo de equipo

```powershell
<#
.SYNOPSIS Función que comprueba si el equipo tiene UEFI activo o no
.DESCRIPTION No recibe parámetros
             True = tiene UEFI activo
             False = no tiene UEFI activo
.EXAMPLE $variable = CheckUEFI
#>
function CheckUEFI {
    $CheckUefi = bcdedit | Select-String 'path\s*.*winload'
    if ($CheckUefi -like "*.efi") {
        return $true
    }elseif ($CheckUefi -like "*.exe") {
        return $false
    }
}

<#
.SYNOPSIS Función que permite comprobar si una característica de Windows está en el estado correcto
          según lo que dicta la guía técnica de seguridad
.DESCRIPTION
    Recibe dos parámetros, uno para el nombre de la característica a comprobar, y otro indicando el estado incorrecto para incumplir la guía técnica
.EXAMPLE
    Check-WindowsFeature -Feature Web-Ftp-Service -InCorrectState Installed
#>
function Check-WindowsFeature ($Feature, $InCorrectState){
    $WNCheck = Get-WindowsFeature | where Name -like $Feature
    if ($WNCheck.InstallState -eq $InCorrectState) { $global:VulnActive+=$Vuln }
}
```

Código 4. Bloque de funciones - comprobación de UEFI y características de Windows

```
<#
.SYNOPSIS Función que permite comprobar las políticas de permisos de usuario
          según lo que dicta la guía técnica de seguridad
.DESCRIPTION
    Recibe 4 parámetros:
        1. Item a comprobar
        2. Política de usuario a comprobar
        3. Valor correcto que debe contener para cumplir la guía técnica de seguridad
        4. Parámetro opcional. Indica si siendo nulo (no estando definida esa política de permiso) se incumple o no.
           Si puede ser nulo, se debe enviar como parámetro en este caso "true"
.EXAMPLE
    Check-UsersRightsPolicy -Vuln $Vuln -Policy SeNetworkLogonRight -Value "S-1-5-11,S-1-5-32-544,S-1-5-9"
    Check-UsersRightsPolicy -Vuln $Vuln -Policy SeNetworkLogonRight -Value "S-1-5-11,S-1-5-32-544,S-1-5-9" -CanBeNull true

#>
function Check-UsersRightsPolicy ($Vuln, $Policy, $Value, $CanBeNull){
    # Si recibe en el parametro $CanBeNull "true" significa que puede ser nulo y estaría OK.
    # Si recibe "false" significa que no puede ser nulo
    Secedit /Export /Areas USER_RIGHTS /CFG C:\Windows\temp\seceditConfig_UsersRights.txt
    $SeceditExport = Get-content -Path "C:\Windows\temp\seceditConfig_UsersRights.txt" | select-string $Policy

    if($SeceditExport){
        #Está configurado, obtenemos valor
        $valueSecedit = $seceditExport.Line.Split("=").Trim()
        $ValueConfigured = $valueSecedit[1].Replace("*","")
        #Si no esta configurado como deberia, incumple
        if ($valueConfigured -notlike $Value) {
            $global:VulnActive+=$Vuln
        }
    }else {
        if ($CanBeNull -notlike "true") {
            #Si no está configurado, incumple
            $global:VulnActive+=$Vuln
        }

    }
}
```

Código 5. Bloque de funciones - comprobación de políticas de permisos de usuario

```
<#
.SYNOPSIS Función que permite comprobar las políticas de seguridad del equipo
          según lo que dicta la guía técnica de seguridad
.DESCRIPTION
    Recibe 3 parámetros:
        1. Item a comprobar
        2. Política de seguridad a comprobar
        3. Valor. Este valor puede ser recibido de las siguientes formas:
            - "!X". CUmpliría con todos los valores excepto el valor X.
            - ">X". CUmpliría siempre y cuando no supere el valor de X.
            - X..Y. Array con los valores inválidos.
            - X. Valor inválido
.EXAMPLE
    Check-SecurityPolicy -Vuln $Vuln -Policy TicketValidateClient -Value 0
    Check-SecurityPolicy -Vuln $Vuln -Policy MaxTicketAge -Value "!0"
    Check-SecurityPolicy -Vuln $Vuln -Policy MaxTicketAge -Value ">10"
    Check-SecurityPolicy -Vuln $Vuln -Policy LockoutDuration -Value 1..14
#>
function Check-SecurityPolicy ($Vuln, $Policy, $Value){
    #REVISAR# EN EL SECEDIT APARECE ADMINSITRATORS O EL SID??S-1-5-32-544 (Administrators)
    Secedit /Export /Areas SecurityPolicy /CFG C:\Windows\temp\seceditConfig_SecurityPolicy.txt
    $SeceditExport = Get-content -Path "C:\Windows\temp\seceditConfig_SecurityPolicy.txt" | select-string $Policy

    if($SeceditExport){
        #Está configurado, obtenemos valor
        $valueSecedit = $seceditExport.Line.Split("=").Trim()

        #EL value que se puede pasar puede ser de dos tipos:
        #Los valores que están permitidos (1,2,3,4,5,6)
        #Todo es válido menos el valor X: se pasa con una exclamación antes (!1)

        #Se pasa el valor que no es válido
        if ($value -like "!*") {
            if ($valueSecedit[1] -like $value.Substring(1)) {
                $global:VulnActive+=$Vuln
            }

        } elseif ($value -like ">*") {
        #Valor inválido mayor que X. Si es mayor que X, incumple
            if ($valueSecedit[1] -gt $value.Substring(1)) {
                $global:VulnActive+=$Vuln
            }
        } else {
            #En este caso se le pasan los valores inválidos
            if ($value -contains $valueSecedit[1]) {
                $global:VulnActive+=$Vuln
            }
        }

    }else {
        #Si no está configurado, incumple
        $global:VulnActive+=$Vuln
    }
}
```

Código 6. Bloque de funciones - comprobación de políticas de seguridad del equipo

```
<#
.SYNOPSIS Función que permite recibiendo la descripción del item de tipo clave de registro de la guía técnica en bruto,
          dividir la información y obtener los valores de la clave de registro a comprobar
.DESCRIPTION
    Recibe 1 parámetro:
        1. Item a comprobar
.EXAMPLE
    ObtainRegeditData -Item $vuln
#>
function ObtainRegeditData ($Item) {
    $RegistryObject = [PSCustomObject]@{
        HKEY = ''
        Path = ''
        Name = ''
        Type = ''
        Value = ''
    }

    $Data = $item.Rule.Check."Check-content".Split([Environment]::NewLine)
    foreach ($element in $data) {
        switch -wildcard ($element) {
            '*Registry Hive*' {
                $HKEYWithoutFormat = $element.Split(":").Trim()[1]
                if ($HKEYWithoutFormat -like "HKEY_LOCAL_MACHINE") { $RegistryObject.HKEY = "HKLM" }
                elseif ($HKEYWithoutFormat -like "HKEY_CURRENT_USER") { $RegistryObject.HKEY = "HKCU"}
            }
            '*Registry Path*' {
                $RegistryObject.Path = $element.Split(":")[1].Trim()
            }
            '*Value Name*' {
                $RegistryObject.Name = $element.Split(":").Trim()[1]
            }
            '*Type*:*' {
                $RegistryObject.Type = $element.Split(":").Trim()[1]
            }
            'Value:*' {
                $ValueWithoutFormat = $element.Split(":").Trim()[1]
                if ($ValueWithoutFormat -match "^\d+x\d+") {
                    $RegistryObject.Value = [int]$ValueWithoutFormat.Split(" ")[0]
                } else {
                    $RegistryObject.Value = $ValueWithoutFormat
                }
            }
        }
    }
    return $RegistryObject

}
```

Código 7. Bloque de funciones - división de información para obtención de claves de registro

```
<#
.SYNOPSIS Función que permite, recibiendo un item de tipo clave de registro de la guía técnica en bruto,
          comprobar si la clave de registro se encuentra correctamente configurada
.DESCRIPTION
    Recibe 2 parámetros:
        1. Item a comprobar
        2. Parámetro opcional. Indica si siendo nulo (no estando definida esa clave de registro) se incumple o no.
           Si puede ser nulo, se debe enviar como parámetro en este caso "true"
.EXAMPLE
    Check-RegistryVuln -Item $vuln
#>
function Check-RegistryVuln ($Vuln, $CanBeNull) {
    #Obtain de data of registry item
    $RegeditKeyData = ObtainRegeditData -Item $Vuln

    #Check registry item
    $WNCheck = Get-ItemProperty -Path "$($RegeditKeyData.HKEY):$($RegeditKeyData.Path)" -Name $RegeditKeyData.Name -ErrorAction Ignore
    $Name = $RegeditKeyData.Name

    if ($CanBeNull -eq "true") {
        if (($WNCheck) -and ($WNCheck.$Name -notlike $RegeditKeyData.Value)) { $global:VulnActive+=$Vuln }
    } else {
        if ((!$WNCheck) -or ($WNCheck.$Name -notlike $RegeditKeyData.Value)) { $global:VulnActive+=$Vuln }
    }

    # Debug lines
    #Write-Log -message "RegeditKeyData: $($RegeditKeyData.HKEY):$($RegeditKeyData.Path), Clave:$($RegeditKeyData.Name), $($RegeditKeyD
    #Write-Log -message "loquehay configurado: $($WNCheck.$Name)" -level Warn
    #Read-Host
}
```

Código 8. Bloque de funciones - verificación de claves de registro

```
<#
.SYNOPSIS Función que permite, auditar el item de la guía técnica
.DESCRIPTION
    Recibe 1 parámetros:
        1. Item a comprobar
.EXAMPLE
    OthersVulnerabilities -Vuln $vuln
#>
function OthersVulnerabilities ($Vuln) {
    switch -wildcard ($Vuln.Rule.Version) {
        #region WN16-DC...

        #region WN16-00...

        #region WN16-AC...

        #region WN16-AU...

        #region WN16-CC...

        #region WN16-MS...

        #region WN16-SO...

        #region WN16-UC...

        #region WN16-UR...

        #region WN16-PK...
    }

}
```

Código 9. Bloque de funciones – resumen de función principal de comprobación de ítems de guía técnica

```
function OthersVulnerabilities ($Vuln) {
    switch -wildcard ($Vuln.Rule.Version) {
        #region WN16-DC
        'WN16-DC-000010' {
            if ((GetComputerType) -eq "Domain Controller"){
                $WNCheck = Get-ADGroupMember "Administrators" | Where-Object {($_.objectClass -like "User") -and ($_.Name -notlike "Administrator")}
                if ($WNCheck){
                    Write-Log -message "Hay usuarios en el grupo de administradores locales. No deben haber para solventar la vulnerabilidad $($Vuln)" -level Warn
                    $global:VulnActive+=$Vuln
                }
            }
        }
        'WN16-DC-000020' {
            if ((GetComputerType) -eq "Domain Controller"){
                Check-SecurityPolicy -Vuln $Vuln -Policy TicketValidateClient -Value 0
            }
        }
        'WN16-DC-000030' {
            if ((GetComputerType) -eq "Domain Controller"){
                Check-SecurityPolicy -Vuln $Vuln -Policy MaxServiceAge -Value "!0"
                Check-SecurityPolicy -Vuln $Vuln -Policy MaxServiceAge -Value ">600"
            }
        }
        'WN16-DC-000040' {
            if ((GetComputerType) -eq "Domain Controller"){
                Check-SecurityPolicy -Vuln $Vuln -Policy MaxTicketAge -Value "!0"
                Check-SecurityPolicy -Vuln $Vuln -Policy MaxTicketAge -Value ">10"
            }
        }
        'WN16-DC-000050' {
            if ((GetComputerType) -eq "Domain Controller"){
                Check-SecurityPolicy -Vuln $Vuln -Policy MaxRenewAge -Value ">7"
            }
        }
        'WN16-DC-000060' {
            if ((GetComputerType) -eq "Domain Controller"){
                Check-SecurityPolicy -Vuln $Vuln -Policy MaxClockSkew -Value ">5"
            }
        }
```

Código 10. Bloque de funciones - parte de verificaciones de ítems WN16-DC

```
'WN16-DC-000170' {
    if ((GetComputerType) -eq "Domain Controller"){
        $AllGPO = Get-Gpo -All
        foreach ($GPO in $AllGPO) {
             Get-Acl -Path ("AD:\" + $GPO.Path) -Audit | Select Audit | foreach {
                $flag = $false
                $_.Audit | foreach {
                    if (($_.IdentityReference -like "Everyone") -and ($_.AuditFlags -like "*Fail*")){
                        $flag = $true
                    }
                }

             }
             if (!$flag) {$global:VulnActive+=$Vuln;break;}
        }
    }
}
'WN16-DC-000180' {
    if ((GetComputerType) -eq "Domain Controller"){
        import-module ActiveDirectory
        Set-Location ad:
        $DSOU = (Get-ADDomain).DistinguishedName
        $AuditPolicies = (get-acl -Path $DSOU -Audit).Audit | select IdentityReference,AuditFlags
        $flag = $false
        foreach ($auditPolicy in $AuditPolicies) {
            if (($auditPolicy.IdentityReference -like "Everyone") -and ($auditPolicy.AuditFlags -like "*Fail*")){
                $flag = $true
            }
        }
        if (!$flag) {$global:VulnActive+=$Vuln}
        Set-Location C:
    }
}
```

Código 11. Bloque de funciones - parte de verificaciones de ítems WN16-DC. Parte 2

```
#region WN16-00
'WN16-00-000010' {
    $global:WarningActive+=$Vuln
}
'WN16-00-000030' {
    if ((GetComputerType) -eq "Domain Controller"){
        $WNCheck = Get-ADUser -Filter * -Properties PasswordLastSet | Where SID -Like "*-500" | where {$_.PasswordLastSet -le (Get-Date).AddDays(-60)} | Format-Table -property
        if ($WNCheck) { $global:VulnActive+=$Vuln }
    }
}
'WN16-00-000040' {
    $global:WarningActive+=$Vuln
}
'WN16-00-000050' {
    if (Get-ADGroupMember "Backup Operators") {
        #Manual
        $global:WarningActive+=$Vuln
    }
}
'WN16-00-000060' {
    $global:WarningActive+=$Vuln
}
'WN16-00-000070' {
    $WNCheck = Get-AdUser -Filter * -Properties PasswordLastSet | where-object {($_.PasswordLastSet -lt ((Get-Date).AddYears(-1))) -and ($_.Enabled -like "True")}
    if ($WNCheck) { $global:VulnActive+=$Vuln }
}
'WN16-00-000080' {
    $global:WarningActive+=$Vuln
}
'WN16-00-000090' {
    if (((GetComputerType) -eq "Member Server") -or ((GetComputerType) -eq "Workstation")){
        Import-Module AppLocker
        $AppLockerList = Get-AppLockerPolicy -Effective -XML | select -Skip 1
        if (-not($AppLockerList)) {
            $global:VulnActive+=$Vuln
        }
    }
}
'WN16-00-000100' {
    $NotCompatibleTPM = (Get-WmiObject Win32_ComputerSystem).Model

    if ($NotCompatibleTPM -notlike "Virtualbox") {
        $WNCheck = (Get-tpm -errorAction Ignore).Enabled
        if (!$WNCheck) { $global:VulnActive+=$Vuln }
    }

}
```

Código 12. Bloque de funciones - parte de verificaciones de ítems WN16-00

```
    }
'WN16-00-000140' {
    $global:WarningActive+=$Vuln
}
'WN16-00-000150' {
     $WNCheck = Get-Volume | WHERE {$_.DriveType -eq 'Fixed'} | WHERE {$_.FileSystem -ne 'NTFS'} | WHERE {$_.FileSystem -ne 'ReFS'} | SELECT DriveLetter, FileSystemLabel,FileSystem
     if ($WNCheck) { $global:VulnActive+=$Vuln }
}
'WN16-00-000160' {
    $WNCheck = (get-acl "C:\").access | where-object {($_.IdentityReference -like "*Users*" -or $_.IdentityReference -like "Authenticated Users") -and ($_.RegistryRights -like "FullControl")}
    if ($WNCheck) { $global:VulnActive+=$Vuln }
}
'WN16-00-000170' {
    $WNCheck = (get-acl "C:\Program Files").access | where-object {($_.IdentityReference -like "*Users*" -or $_.IdentityReference -like "Authenticated Users") -and ($_.RegistryRights -like "Ful
    $WNCheck2 = (get-acl "C:\Program Files (x86)").access | where-object {($_.IdentityReference -like "*Users*" -or $_.IdentityReference -like "Authenticated Users") -and ($_.RegistryRights -li
    if ($WNCheck -or $WNCheck2) { $global:VulnActive+=$Vuln }
}
'WN16-00-000180' {
    $WNCheck = (get-acl "C:\Windows").access | where-object {($_.IdentityReference -like "*Users*" -or $_.IdentityReference -like "Authenticated Users") -and ($_.RegistryRights -like "FullContr
    if ($WNCheck) { $global:VulnActive+=$Vuln }
}
'WN16-00-000190' {
    $Folders = @("HKLM:\SECURITY", "HKLM:\SOFTWARE", "HKLM:\SYSTEM")

    foreach ($folder in $Folders) {
        #$WNCheck = (get-acl $folder).access | where-object {($_.IdentityReference -like "*Users*" -or $_.IdentityReference -like "Everyone") -and ($_.RegistryRights -notlike "ReadKey")}
        $WNCheck = (get-acl $folder).access | where-object {($_.IdentityReference -like "*Users*" -or $_.IdentityReference -like "Everyone") -and (($_.RegistryRights -notlike "ReadKey") -and ($
        if ($WNCheck) { $global:VulnActive+=$Vuln }
    }
}
'WN16-00-000200' {
    if (Get-Printer | where-object {$_.Name -notlike "*Microsoft*" -and $_.Shared -eq "True"}) {
        #REVISAR falta comprobar los permisos de la compartida
        if ($WNCheck) { $global:VulnActive+=$Vuln }
    }
}
'WN16-00-000210' {
    $WNCheck = Search-ADAccount -AccountInactive -UsersOnly -TimeSpan 35.00:00:00 | where-object {($_.SID -notlike "*500" -and $_.SID -notlike "*501" -and $_.SID -notlike "*503") -and ($_.Enabl
    if ($WNCheck) { $global:VulnActive+=$Vuln }
}
'WN16-00-000220' {
    $WNCheck = Get-Aduser -Filter * -Properties Passwordnotrequired | where-object {$_.PasswordNotrequired -like "True" -and $_.Enabled -like "True"}
    if ($WNCheck) { $global:VulnActive+=$Vuln }
}
'WN16-00-000230' {
    $WNCheck = Search-ADAccount -PasswordNeverExpires -UsersOnly | where-object {$_.PasswordNeverExpires -like "True" -and $_.Enabled -like "True"}
    if ($WNCheck) { $global:VulnActive+=$Vuln }
}
```

Código 13. Bloque de funciones - parte de verificaciones de ítems WN16-00. Parte 2

```
'WN16-00-000330' {
    $WNCheck = Search-ADAccount -AccountExpiring | FT Name, AccountExpirationDate
    if ($WNCheck) { $global:VulnActive+=$Vuln }
}
'WN16-00-000340' {
    $WNCheck = Search-ADAccount -AccountExpiring | FT Name, AccountExpirationDate
    if ($WNCheck) { $global:VulnActive+=$Vuln }
}
'WN16-00-000350' {
    Check-WindowsFeature -Feature Fax -InCorrectState Installed
}
'WN16-00-000360' {
    Check-WindowsFeature -Feature Web-Ftp-Service -InCorrectState Installed
}
'WN16-00-000370' {
    Check-WindowsFeature -Feature PNRP -InCorrectState Installed
}
'WN16-00-000380' {
    Check-WindowsFeature -Feature Simple-TCPIP -InCorrectState Installed
}
'WN16-00-000390' {
    Check-WindowsFeature -Feature Telnet-Client -InCorrectState Installed
}
'WN16-00-000400' {
    Check-WindowsFeature -Feature TFTP-Client -InCorrectState Installed
}
'WN16-00-000410' {
    Check-WindowsFeature -Feature FS-SMB1 -InCorrectState Installed
}
'WN16-00-000411' {
    if ((Get-WindowsFeature | where Name -like "FS-SMB1").INstallState -like "Installed") {
        Check-RegistryVuln $Vuln
    }
}
'WN16-00-000412' {
    Check-RegistryVuln $Vuln
}
'WN16-00-000420' {
    Check-WindowsFeature -Feature PowerShell-v2 -InCorrectState Installed
}
'WN16-00-000430' {
    if ((Get-WindowsFeature | where name -like "*ftp*").InstallState -eq "Installed"){
        $WNCheck = Get-WebConfigurationProperty -Filter "/system.webServer/security/authentication/anonymousAuthentication" -Name Enabled | select Value
        if ($WNCheck) { $global:VulnActive+=$Vuln }
    }
}
```

Código 14. Bloque de funciones - parte de verificaciones de ítems WN16-00. Parte 3

```
'WN16-AU-000286' {
    $WNCheck = auditpol /get /category:* | Select-String "Other Object Access Events"
    if ($WNCheck -notlike "*Failure*") { $global:VulnActive+=$Vuln }
}
'WN16-AU-000290' {
    $WNCheck = auditpol /get /category:* | Select-String "Removable Storage"
    if ($WNCheck -notlike "*Success*") { $global:VulnActive+=$Vuln }
}
'WN16-AU-000300' {
    $WNCheck = auditpol /get /category:* | Select-String "Removable Storage"
    if ($WNCheck -notlike "*Failure*") { $global:VulnActive+=$Vuln }
}
'WN16-AU-000310' {
    $WNCheck = auditpol /get /category:* | Select-String "Audit Policy Change"
    if ($WNCheck -notlike "*Success*") { $global:VulnActive+=$Vuln }
}
'WN16-AU-000320' {
    $WNCheck = auditpol /get /category:* | Select-String "Audit Policy Change"
    if ($WNCheck -notlike "*Failure*") { $global:VulnActive+=$Vuln }
}
'WN16-AU-000330' {
    $WNCheck = auditpol /get /category:* | Select-String "Authentication Policy Change"
    if ($WNCheck -notlike "*Success*") { $global:VulnActive+=$Vuln }
}
'WN16-AU-000340' {
    $WNCheck = auditpol /get /category:* | Select-String "Authorization Policy Change"
    if ($WNCheck -notlike "*Success*") { $global:VulnActive+=$Vuln }
}
'WN16-AU-000350' {
    $WNCheck = auditpol /get /category:* | Select-String "Sensitive Privilege Use"
    if ($WNCheck[1] -notlike "*Success*") { $global:VulnActive+=$Vuln }
}
'WN16-AU-000360' {
    $WNCheck = auditpol /get /category:* | Select-String "Sensitive Privilege Use"
    if ($WNCheck[1] -notlike "*Failure*") { $global:VulnActive+=$Vuln }
}
```

Código 15. Bloque de funciones - parte de verificaciones de ítems WN16-AU

```
'WN16-SO-000300' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000320' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000330' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000340' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000350' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000360' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000380' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000390' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000400' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000410' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000420' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000430' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000450' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000460' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000470' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000480' {
    Check-RegistryVuln $Vuln
}
'WN16-SO-000490' {
    Check-RegistryVuln $Vuln
```

Código 16. Bloque de funciones - parte de verificaciones de ítems WN16-SO

```
'WN16-UR-000270' {
    Check-UsersRightsPolicy -Vuln $Vuln -Policy SeSystemEnvironmentPrivilege -Value "S-1-5-32-544"
}
'WN16-UR-000280' {
    Check-UsersRightsPolicy -Vuln $Vuln -Policy SeManageVolumePrivilege -Value "S-1-5-32-544"
}
'WN16-UR-000290' {
    Check-UsersRightsPolicy -Vuln $Vuln -Policy SeProfileSingleProcessPrivilege -Value "S-1-5-32-544"
}
'WN16-UR-000300' {
    Check-UsersRightsPolicy -Vuln $Vuln -Policy SeRestorePrivilege -Value "S-1-5-32-544"
}
'WN16-UR-000310' {
    Check-UsersRightsPolicy -Vuln $Vuln -Policy SeTakeOwnershipPrivilege -Value "S-1-5-32-544"
}
#endregion WN16-UR

#region WN16-PK
'WN16-PK-000010' {
    $DODCerts = Get-ChildItem -Path Cert:Localmachine\root | Where Subject -Like "*DoD Root CA*"
    if (!$DODCerts) {
        $global:VulnActive+=$Vuln
    }
}
'WN16-PK-000020' {

    #$DODCerts = Get-ChildItem -Path Cert:Localmachine\disallowed | Where Subject -Like "*DoD*" | FL Subject, Issuer, Thumbprint, NotAfter
    $DODCerts = Get-ChildItem -Path Cert:Localmachine\disallowed | Where {$_.Issuer -Like "*DoD Interoperability*" -and $_.Subject -Like "*DoD*"}
    if (!$DODCerts) {
        $global:VulnActive+=$Vuln
    }
}
'WN16-PK-000030' {
    $DODCerts3 = Get-ChildItem -Path Cert:Localmachine\disallowed | Where Issuer -Like "*CCEB Interoperability*"
    if (!$DODCerts3) { $global:VulnActive+=$Vuln }
}
#endregion WN16-PK
```

Código 17. Bloque de funciones - parte de verificaciones de ítems WN16-UR y WN16-PK

Bloque de inicializaciones

```
#---------------------------------------------------------[Initialisations]--------------------------------------------------------

$Baseline = Get-OptionBaseline

#Set Error Action to Silently Continue
$ErrorActionPreference = "Inquire"

# Log
$DirLog = ".\Logs"
$ScriptName = $MyInvocation.MyCommand.Name.Split('.')[0]
$LogFile = "$($DirLog)\$($ScriptName)_$(Get-Date -Format "yyyy-MM-dd").log"
```

Código 18. Bloque de inicializaciones

Bloque de ejecución

```
Write-Log -Message "Running $($ScriptName) with $($env:USERNAME). Start: $(Get-Date -Format "yyyy-MM-dd HH:mm:ss")" -Level Info

if ($Implement) {
    #Importación de certificados utilizando lautilidad de DOD DISA
    Start-Process -FilePath "msiexec" -ArgumentList "/i .\CertificatesUtility\InstallRoot_5.5x64.msi /qn" -Wait
    Start-Process -FilePath "C:\Program Files\DoD-PKE\InstallRoot\InstallRoot.exe" -ArgumentList "--insert" -Wait
    Start-Process -FilePath "msiexec" -ArgumentList "/x .\CertificatesUtility\InstallRoot_5.5x64.msi /qn" -Wait

    #Modify lDAP atributes
    ntdsutil "LDAP policies" connections "connect to server W2016-DC1" quit "Set MaxConnIdleTime to 200" "Commit Changes" quit quit


    #resetear pass de administrator
    if ((Get-WindowsFeature | where name -like "*SMB1*").InstallState -eq "Installed"){ Get-WindowsFeature -Name "*SMB1*" | Remove-WindowsFeature }
    #auditings
    #Domain policy importar

    # Importar políticas DOD
    Add-Type -AssemblyName Microsoft.VisualBasic

    $DomainAdminsGroup = [Microsoft.VisualBasic.Interaction]::InputBox('Enter the name of DOMAIN ADMINS group of
    your organization

    Eg: PRUEBA\DOMAIN ADMINS)',
    'Importing DISA STIGs')

    $EnterpriseAdminsGroup = [Microsoft.VisualBasic.Interaction]::InputBox('Enter the name of ENTERPRISE ADMINS group of
    your organization

    Eg: PRUEBA\ENTERPRISE ADMINS)',
    'Importing DISA STIGs')


    [xml] $ImportTable = Get-Content ".\U_STIG_GPO_Package_November_2020\Support Files\importtable.migtable"
    $ImportTable.MigrationTable.Mapping | foreach {
        if ($_.Source -like "ADD YOUR DOMAIN ADMINS") {
            $_.Source = $DomainAdminsGroup
        }
        if ($_.Source -like "ADD YOUR ENTERPRISE ADMINS") {
            $_.Source = $EnterpriseAdminsGroup
        }
    }
    $ImportTable.Save("C:\Users\JUAN\Downloads\Mimporttable.migtable")

    Start-process ".\U_STIG_GPO_Package_November_2020\Support Files\DISA_GPO_Baseline_Import.ps1" -ArgumentList "-gpoimportFile '.\U_STIG_GPO_Packag
```

Código 19. Bloque de ejecución - implementación de bastionado

```
} else {
    # Comprobamos la introducción del XML del paráemtro
    if (!(Test-path $Baseline)){
        Write-Log "Specified file not found. $($Baseline) " -level Warn
        Exit 1
    }

    # Importamos el fichero XML y creamos los arrays principales que almacenan los items failed y warnings
    [xml]$XMLContent = Get-Content -Path $Baseline
    $global:VulnActive = @()
    $global:WarningActive = @()

    # Recorremos los items del XML
    foreach ($VulnItem in $XMLContent.Benchmark.Group){
        #Se muestra por pantalla el item que está analizando en cada momento
        $VulnID = $VulnItem.id
        $VulnWN = $VulnItem.Rule.Version
        Write-Log -message "Analyzing vulnerability $($VulnID) - $($VulnWN)" -level Info

        # Analizamos los items
        Set-Location C:
        OthersVulnerabilities -Vuln $VulnItem
    }

    #region Muestra de resultados
    ### MUESTRA DE RESULTADOS

    $VulnActiveFormatArray = @()
    foreach ($VulnWithoutFormat in $global:VulnActive) {
        $VulnActiveFormat = [PSCustomObject]@{
            ID = $VulnWithoutFormat.ID
            WN = $VulnWithoutFormat.Rule.Version
            Severity = $VulnWithoutFormat.Rule.Severity
            Description = $VulnWithoutFormat.Rule.Check."Check-content"
        }
        $VulnActiveFormatArray+=$VulnActiveFormat
    }

    $WarningsFormatArray = @()
    foreach ($WarningWithoutFormat in $global:WarningActive) {
        $WarningFormat = [PSCustomObject]@{
            ID = $WarningWithoutFormat.ID
            WN = $WarningWithoutFormat.Rule.Version
            Severity = $WarningWithoutFormat.Rule.Severity
            Description = $WarningWithoutFormat.Rule.Check."Check-content"
        }
        $WarningsFormatArray+=$WarningFormat
    }
```

Código 20. Bloque de ejecución - auditoria de ítems

```
    $WarningsFormatArray | Out-GridView -Title "Warnings Active in $($env:COMPUTERNAME)"
    $VulnActiveFormatArray | Out-GridView -Title "Vulnerabilities Active in $($env:COMPUTERNAME)"
    $WarningsFormatArray | Select ID,WN,Severity | export-csv -Path ".\ResultsCSV\warnings_$($(Get-Date -Format "yyyy-MM-dd_HH-mm-ss")).csv"
    $VulnActiveFormatArray | Select ID,WN,Severity | export-csv -Path ".\ResultsCSV\vulnes_$($(Get-Date -Format "yyyy-MM-dd_HH-mm-ss")).csv"

    $totalItems = $XMLContent.Benchmark.Group.Count
    $totalPassed = $totalItems-$global:WarningActive.Count-$global:VulnActive.Count

    Write-Log -message "Total de items: $($totalItems)" -level Info
    Write-Log -message "Total de vulnerabilidades (failed): $($global:VulnActive.Count)" -level Info
    Write-Log -message "Total de Warnings: $($global:WarningActive.Count)" -level Info
    Write-Log -message "Total de passed: $($totalPassed)" -level Info
    Write-Log -message "-------------------------------------------------------" -level Info
    Write-Log -message "Failed: $(($global:VulnActive.Count/$totalItems)*100)%" -level Info
    Write-Log -message "Warnings: $(($global:WarningActive.Count/$totalItems)*100)%" -level Info
    Write-Log -message "Passed: $(($totalPassed/$totalItems)*100)%" -level Info

    [System.Windows.MessageBox]::Show(
    "Total of items checked: $($totalItems)
    Total of vulnerabilities active: $($global:VulnActive.Count)
    Total of warnings (require of supervise by SSOO Department): $($global:WarningActive.Count)
    Total of items passed correctly: $($totalPassed)

    -------------------------------------------------------------
    Failed: $([math]::Round((($global:VulnActive.Count/$totalItems)*100),2))%
    Warnings: $([math]::Round((($global:WarningActive.Count/$totalItems)*100),2))%
    Passed: $([math]::Round((($totalPassed/$totalItems)*100),2))%",
    'Resume of auditing DISA baseline','OK','None')
    #endregion Muestra de resultados
```

Código 21. Bloque de ejecución - muestra de resultados tras auditoria de ítems

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

En el presente trabajo de fin de grado se analizará un marco teórico de la seguridad de la infraestructura a nivel empresarial, se profundizará en el bastionado de servidores y, por último, se estudiará y evaluará la necesidad y el impacto de su implementación.

Una vez hecho esto, se replicará la implementación de alguna de las líneas base de configuración ofrecidas por organizaciones prestigiosas en el ámbito de la seguridad informática evitando producir inestabilidades en el sistema. Todo esto se realizará en un laboratorio personal que replicará la infraestructura empresarial típica.

Finalmente, en este trabajo de fin de grado se procederá, de forma adicional, a desarrollar una herramienta mediante programación con Powershell, con la que se evaluará y medirá el nivel de cumplimiento implementado en el servidor objetivo. Esta herramienta permitirá realizar auditorías de seguridad a servidores Windows.

In this Final Degree Project, I am going to analyze a theoretical framework of infrastructure security at the business level, delving into the hardening of servers and, finally, I will study and evaluate the need and the impact of its implementation.

Once this is done, I will replicate the implementation of some of the configuration baselines offered by renowned organizations in the field of computer security, avoiding producing instabilities in the system. All of this will be done in a personal lab that will replicate typical business infrastructure.

To conclude this Final Degree Project, I will proceed -in addition- to develop a tool using Powershell´s programming, which will evaluate and measure the level of compliance implemented in the target server. This tool will allow security audits performed in Windows servers.

GRADO EN INGENIERÍA INFORMÁTICA, 2020/2021