	HARDENING MongoDB	
		V.1

CONTROL DE VERSIONES

Elaborado por:	David Chamorro Portilla	**No. de Versión:**	1
Revisado por:		**Fecha de revisión:**	
Aprobado por:		**Fecha de aprobación:**	

Historia de Modificaciones

No. de Versión	Fecha de Versión	Autor	Revisado por	Aprobado por	Descripción
1	Febrero 24, 2023	David Chamorro Portilla			

1. DESCRIPCIÓN GENERAL

Objetivo

Presentar los puntos primordiales para ejecutar un proceso de gestión de seguridad y auditoria de los servicios del motor de Base de Datos MongoDB, establecidos por SETI, basados en documentos generados por el Fabricante y en la experiencia de SETI S.A.S.

Definiciones

Observaciones

Este documento está elaborado de tal forma que puede ser aplicado desde la versión 4.2.6 a la versión 6.0.4 del motor de base de datos MongoDB en el sistema operativo Linux redhat o basados en redhat.

Tener en cuenta:

- Los Hardening por demanda realizados por parte del cliente pueden generar oportunidades de mejora puesto que los servicios están en constante cambio o evolución.
- Cualquier punto tratado en este documento podría representar un Riesgo el cual demandará un análisis más profundo del tema específico.
- La ejecución de esta actividad de Hardening puede llevar a la planeación y ejecución de actividades de PAYM.

Las tareas presentadas en este documento también aplican cuando se desea ajustar y mejorar los lineamientos vigentes conforme lo demanden las buenas prácticas del fabricante o la experiencia de SETI.

La Plantillas Default mencionada en este documento, se relacionan a continuación:
CIS MongoDB 5 Benchmark.

El documento asociado llamado "Lista de Chequeo adicional.xlsx" se utiliza para plasmar información a la cual no se tenga acceso directo por parte de SETI o que estén fuera de nuestro alcance, pero que igual, permiten confrontar y llevar un registro del estado de aseguramiento de los servicios de Bases de datos en sus diferentes niveles.

CALIFICACION GENERAL DEL HARDENING

Teniendo en cuenta los puntos tratados en este documento , su nivel de criticidad y las evidencias obtenidas , resultado del proceso ejecutado, se presenta a continuación una tabla con los hallazgos que permitirá calificar el nivel de cumplimiento de este Hardening.

		NO CUMPLIMIENTO DE VULNERABILIDADES		
CAPA		**ALTA**	**MEDIA**	**BAJA**
BASES DE DATOS	**VULNERABILIDADES**	**##%**	**##%**	**##%**
	Sistema Operativo			
	Instancia			
Total ítems Evaluados:	**##**	**#**	**#**	**#**

1. VULNERABILIDADES A NIVEL DE SISTEMA OPERATIVO

TAREA	CRITICIDAD
1. Instalación: En MongoDB no se manejan parches sino cambios de versión, por ejemplo: estar en la versión 5.2 y pasa a la versión 6.0.1, lo que conlleva la instalación completa de MongoDB server. Además de verificar que se cuente con las últimas versiones liberadas por el Fabricante, es necesario determinar que los paquetes se hayan obtenido de repositorios autorizados y confiables, según manual del fabricante. **NOTA:** Si se utilizan repositorios no autorizados o confiables se corre el riesgo de utilizar productos manipulados por terceros, con altas fallas de seguridad o rastreo, incluso, sin ningún tipo de documentación de soporte valida. **AUDITORIA:** A. Determine que los repositorios listados a continuación son válidos y autorizados: `cat /etc/yum.repos.d/mongodb-org-*` B. Asegure que las versiones de los paquetes principales coincidan todas con la versión actual del producto: `# Obtener la lista de paquetes de MongoDB` `mongo_packages=$(rpm -qa --queryformat '%{NAME}\n' | grep mongo)` `# Iterar por cada paquete y obtener su URL` `for package in $mongo_packages; do` `  package_name=$(echo $package \| cut -d " " -f 1)` `  # Obtener la version del paquete` `  package_version=$(rpm -qi $package_name | grep Version | awk '{print $3}')` `  # Obtener la URL del paquete` `  package_url=$(rpm -qi $package_name | grep URL | awk '{print $3}')` `  # Obtener el proveedor del paquete` `  package_vendor=$(rpm -qi $package_name | grep Vendor | awk '{print $3}')`	**Alta**

TAREA	CRITICIDAD
`# Imprimir la información del paquete` `echo "Package Name: $package_name"` `echo "Version: $package_version"` `echo "URL: $package_url"` `echo "Vendor: $package_vendor"` `echo "------------------------------"` `done` `#Version Mongodb` `mongod --version` **EVIDENCIAS:**	
2. Registro de servicio de MongoDB en SYSTEMD: En caso de que el servidor sufra algún cambio de estado, es necesario que los servicios de MongoDB estén configurados para actuar en consecuencia, para evitar que los tiempos de indisponibilidad se extiendan más de lo necesario, así mismo, garantizando que su cambio de estado se corrija de forma automática. **NOTA:** El no hacer uso de automatizaciones mediante SYSTEMD no es posible asegurar que los servicios de MongoDB vuelvan a estar activos rápidamente en caso de cambios de estado en el servidor. **AUDITORIA:** A. Establecer el Target a consultar: `systemctl get-default` B. Listar las dependencias sobre el Target encontrado: `systemctl list-dependencies multi-user.target | grep -i mongo` **EVIDENCIAS:**	**Media**

TAREA	CRITICIDAD
3. <u>Permisos en Carpetas y Binarios:</u> Todo servicio de MongoDB que este correctamente implementado a nivel de propietario y permisos debe cumplir los siguientes criterios: ➢ Los archivos y carpetas ubicados en el directorio de dbPath, deben ser de propiedad del usuario propietario de la instalación del motor de MongoDB. (Generalmente es el usuario: mongod) ➢ Sobre el directorio de dbPath ningún usuario o grupo distinto al propietario debe poseer permisos de ninguna índole. ➢ El directorio dbPath no puede ser de propiedad del usuario **root** del sistema operativo. ➢ El servicio o procesos de mongod no debe ser invocado o activado por el usuario **root** o por cualquier otro distinto al propietario de los Binarios de instalación. **NOTA:** El no contar con estas pautas, permitirá la inestabilidad tanto de seguridad como del servicio mismo de la instancia y de los Datos contenidos en la misma, incluso, llegar al punto de fallar en el momento de iniciarla. **AUDITORIA:** A. Verificar que los mínimos procesos de MongoDB estén en pleno funcionamiento y se tenga un usuario dedicado para el servicio mongod ó mongos diferente a root: `ps -fea|grep mongo` `(mínimo: mongod)` B. Determinar el propietario, grupo y permisos de los archivos de la instancia (variable dbPath del archivo de cofiguracion): `cat /etc/mongod.conf|grep dbPath` `Tomar la ruta de la variable dbPath en este caso:` `ls -la /var/lib/mongo` **EVIDENCIAS:**	**Alta**

4. <u>Estado de autenticación:</u> Esta configuración asegura que para acceder a la base de datos de MongoDB, todos los clientes, usuarios y servidores deben autenticarse primero. La autenticación implica comprobar la identidad de un cliente, y cuando la autorización está activada, MongoDB requerirá que todos los clientes se autentiquen para determinar su nivel de acceso a la base de datos. En resumen, esta configuración asegura que solo los usuarios autorizados y autenticados puedan acceder a la base de datos de MongoDB. MongoDB en su version community soporte diferentes métodos de autenticación: • SCRAM (Default) • x.509 Certificate Authentication. **AUDITORIA:** A. Ejecutar el siguiente comando para verificar si al autenticación SCRAM esta habilitada: `cat /etc/mongod.conf | grep "authorization"` B. Ejecutar el siguiente comando para verificar si al autenticación x509 esta habilitada: `cat /etc/mongod.conf | grep "clusterAuthMode"` **EVIDENCIAS:**	**Alto**
5. <u>Autenticacion sin password por localhost:</u> Es importante que no se configure MongoDB para que salte la autenticación a través de la excepción de localhost. Esta excepción permite que los usuarios habiliten la autorización antes de crear el primer usuario en el sistema. Si se activa, la excepción de localhost permitirá que todas las conexiones desde la interfaz de localhost tengan acceso completo a esa instancia de MongoDB.	**Alto**

NOTA: Es importante tener en cuenta que esta recomendación solo es relevante en caso de que no existan usuarios creados en la instancia de MongoDB. . **AUDITORIA:** A. Verificar la existencia la autenticación este desactivada: `cat /etc/mongod.conf |grep "enableLocalhostAuthBypass"` **EVIDENCIAS:**	
	Alto

2. VULNERABILIDADES A NIVEL DE INSTANCIA

TAREA	CRITICIDAD
1. <u>Mínimos privilegios de roles:</u> MongoDB utiliza un enfoque basado roles para controlar el acceso a los datos y comandos. Viene con funciones integradas que proporcionan diferentes niveles de acceso a la base de datos, y también puede crear roles personalizados según sus necesidades. Algunos roles integrados, como el rol dbOwner y el rol UserAdmin, permiten a los usuarios asignar privilegios a sí mismos en cualquier base de datos, siempre y cuando estén en el ámbito de la base de datos de administración. Sin embargo, es importante tener en cuenta que estos roles deben asignarse cuidadosamente y limitarse solo a usuarios confiables y autorizados. **Revisión:** A. Verificar la existencia la autenticación este desactivada: `db.system.users.find(` `{"roles.role":{$in:["dbOwner","userAdmin","userAdminAnyDatabase"]},"roles.db": "admin" } )`	**Media**

TAREA	CRITICIDAD
2. Super usuarios: Los roles son una forma conveniente de administrar los privilegios de acceso en un sistema de base de datos, ya que permiten a los administradores de seguridad controlar el acceso a las bases de datos de manera que refleje la estructura organizativa de sus organizaciones. Esto significa que pueden crear roles que se correspondan con las funciones laborales de sus organizaciones y asignar esos roles a los usuarios en lugar de otorgar el mismo conjunto de privilegios a cada usuario individualmente. Esto simplifica la asignación de privilegios y hace que la administración sea más eficiente. La revisión de los roles de Superusuario/Administrador dentro de una base de datos ayuda a minimizar la posibilidad de acceso privilegiado no deseado. **AUDITORIA:** A. Verificar que los usuarios listados : `db.runCommand( { rolesInfo: "dbOwner" } )` `db.runCommand( { rolesInfo: "userAdmin" } )` `db.runCommand( { rolesInfo: "userAdminAnyDatabase" } )` `db.runCommand( { rolesInfo: "readWriteAnyDatabase" } )` `db.runCommand( { rolesInfo: "dbAdminAnyDatabase" } )` `db.runCommand( { rolesInfo: "userAdminAnyDatabase" } )` `db.runCommand( { rolesInfo: "clusterAdmin" } )` `db.runCommand( { rolesInfo: "hostManager" } )` **EVIDENCIAS:**	**Alta**
3. Encripción de data en transito TLS o SSL: Para garantizar la seguridad en las conexiones entrantes y salientes, se debe utilizar TLS o SSL. Esto significa que se debe cifrar la comunicación entre los componentes de MongoDB y las aplicaciones, lo que incluye la comunicación entre mongod y mongos. MongoDB admite TLS/SSL, lo que permite cifrar todo el tráfico de red de	**Media**

TAREA	CRITICIDAD
MongoDB. Cuando se utiliza TLS/SSL, se asegura que el tráfico de red solo pueda ser leído por el cliente previsto, lo que ayuda a proteger los datos sensibles que se transmiten. **AUDITORIA:** A. Verificar verificar que TLS este activado y TLS 1.0 y 1.1 este desactivado: `cat /etc/mongod.conf` Revisar en la sección net: `net:` `  tls:` `  mode: requireTLS` `  certificateKeyFile: /etc/ssl/mongodb.pem` `  CAFile: /etc/ssl/caToValidateClientCertificates.pem` `  disabledProtocols: TLS1_0,TLS1_1` **EVIDENCIAS:**	
4. Encripción de datos en reposo: El cifrado de datos en reposo significa que la información almacenada en una base de datos MongoDB está protegida mediante técnicas de cifrado. Esto es importante para cumplir con los estándares de seguridad y privacidad, como HIPAA, PCI-DSS y FERPA. El cifrado en reposo debe ser utilizado junto con el cifrado de transporte, que protege el tráfico de red entre los componentes MongoDB, y con buenas prácticas de seguridad que protejan las cuentas, contraseñas y claves de cifrado relevantes para mantener la integridad y confidencialidad de la información almacenada. **AUDITORIA:** A. Verificar estado de la rotación del Log: `cat mongodnodo1.conf  | grep enableEncryption` `cat mongodnodo1.conf  | grep encryptionKeyFile`	**Media**

TAREA	CRITICIDAD
EVIDENCIAS:	