Hardening de bases de datos documentales: Caso practico MongoDB

Lucía Rotela
Instituto de Computación
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
lucia.rotela@fing.edu.uy

Agustín Sanchez
Instituto de Computación
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
javier.sanchez@fing.edu.uy

***Resumen*—El desconocimiento sobre seguridad en bases de datos documentales facilita ataques exitosos, como consecuencia se ve un incremento en investigaciones sobre seguridad en bases de datos como MongoDB. Es de interés recolectar la información y obtener una respuesta a qué se puede hacer para asegurar estos sistemas contra usuarios maliciosos. En este artículo se investiga el estado del arte en la seguridad sobre bases de datos documentales mostrando evidencia respecto al impacto de los ataques, así como recabando investigaciones académicas, herramientas existentes y recomendaciones para subsanar vulnerabilidades sobre estos sistemas. El resultado es un conjunto de recomendaciones que al ser aplicadas brindan garantías de seguridad sobre confidencialidad, integridad y disponibilidad de los datos, el activo más importante en una base de datos. La importancia de este resultado está en presentar conocimiento sobre seguridad que permita, a quienes usan bases de datos documentales y en particular MongoDB, prevenir las vulnerabilidades a las que están expuestos.**

I. Introducción

Como consecuencia del uso de celulares, redes sociales y aplicaciones populares en general, cada momento se intercambian mayores volúmenes de datos. Con intención de mejorar la escalabilidad y performance en este intercambio, surge una nueva forma de almacenar información, las bases de datos no relacionales. Por otro lado, el aumento en la popularidad de estas bases de datos trae consigo nuevos requerimientos, nuevas necesidades. Ya no es suficiente con garantizar performance, es fundamental que esta información sea almacenada de forma segura. Poner en riesgo información sensible de los usuarios puede terminar en la pérdida de grandes sumas de dinero o incluso en el final de un negocio. Productos como MongoDB en su primera versión no estaban preparados, ni pretendían, dar solución a estos problemas; en este trabajo analizaremos, entre otras cosas, el estado actual del producto, posibilidades, fallas y recomendaciones para poder usarlo de forma óptima. [Abramov(2011)] [Chaturvedi(2018)]

El resto del documento se organiza de la siguiente forma. En la Sección II se introducen las motivaciones para investigar este tema. En la Sección III se presentan definiciones importantes a ser utilizadas en el resto del documento. A continuación, en la sección IV, se analiza el estado del arte. En la sección V se listan las problemáticas con las que se trabajará y las recomendaciones a seguir para reducir riesgo e impacto. Por último, en la sección VI se presentan conclusiones y trabajo futuro.

II. Motivación

En mayo del presente año, un análisis de cybernews[1] detectó más de 29000 bases de datos abiertas y más de 19 petabytes de datos expuestos al mundo. La búsqueda estuvo acotada a bases que no presentaran requerimientos de autenticación y fueran de MongoDB, Hadoop o Elasticsearch. Quedaron excluidas todas aquellas con una autenticación débil o con las credenciales por defecto. Por otro lado, en 2020 se popularizó el ataque "Meow", vaciaban bases y dejaban un único archivo como identificador del atacante, alrededor de 4000 bases sufrieron este ataque, 97 % de ellas de MongoDB y Elasticsearch. Sorprendentemente, 59 de las bases *Meowed* el año pasado, fueron detectadas nuevamente en el análisis de cybernews; aún no se corrigió la vulnerabilidad en la autenticación. [Mikalauskas(2021)] [Owaida(2020)]

También se vio perjudicada Medicare, una empresa estadounidense que pretende encontrarle a cada usuario el seguro de salud que mejor se adapte a ellos. Recopilan información mediante un formulario web que los usuarios completan con datos personales. En junio de 2019 comparitech[2] detectó una base de datos pública relacionada con la web de Medicare. La información encontrada era suficiente para identificar a las personas: nombre, dirección, número de celular, fecha de nacimiento, género, dirección IP; también para identificar algunos de sus intereses comerciales o intereses en el ámbito del seguro de salud precisamente. Los investigadores no tenían forma de asegurar si alguien más accedió a esta información antes que ellos por lo que, aún cuando desde la empresa se corrigieron las vulnerabilidades de seguridad, los usuarios cuyos datos fueron expuestos podían ser víctimas de spam, phishing dirigido o fraude. [Bischoff(2019)]

Los casos presentados permiten visualizar el impacto de tener una falla en seguridad. Así como la importancia de tener un buen plan para disminuir riesgos, o daños una vez que se haya logrado vulnerar un sistema.

[1] https://cybernews.com
[2] https://www.comparitech.com/

III. Marco teórico

Hablar de base de datos documental refiere a la forma en que se almacenan los datos en una base no relacional; no aporta información alguna sobre transacciones específicas, únicamente da la pauta de que la información será almacenada en documentos estructurados, usualmente XML o JSON. [Harrison(2015)]

MongoDB es una base de datos documentales que se apoya fuertemente en la estructura JSON. No solo en los datos almacenados, también en su propio lenguaje para plantear consultas. Entre las ventajas de MongoDB, encontramos la escalabilidad y la performance de la base de datos. [mongoDB(2021)]

Entendemos por vulnerabilidad aquella propiedad de un sistema que, potenciada por alguna amenaza interna o externa, puede generar un fallo en la seguridad del mismo. [Anderson(2008)]

El término hardening hace referencia a reducir la ocurrencia de vulnerabilidades en un sistema, por consiguiente, incrementar su seguridad. Modificaciones de código, metodologías o productos, así como combinaciones de estos tres puntos para aumentar la resistencia a posibles ataques. [Debbabi(2006)]

IV. Estado del arte

Chaturvedi en *Security Challenges and Solutions in MongoDB* [Chaturvedi(2018)] estudia las principales características que provee MongoDB, en un contexto equivalente a versiones posteriores a 2018. Evidencia falencias de seguridad al punto que destaca lo débil o hasta inexistente de la autenticación y encriptación. Se deja a cargo del sistema operativo o de los desarrolladores, responsabilidades de seguridad propias del producto como la protección de los datos persistidos o la prevención de inyección NoSQL.

No cuenta con encriptación automática del tráfico, por defecto no soporta SSL y por ser un desarrollo sumamente ligado al uso de javascript es vulnerable a ataques de inyección. Aunque se identifica un escenario nada favorable para el uso de MongoDB con garantías de seguridad, Chaturvedi puede concluir con una lista de recomendaciones haciendo uso de las pocas funcionalidades con las que se contaba. Incluye recomendaciones tales como el uso de certificados x.509 para encriptar el tráfico o autorización según RBAC.

IV-A. Seguridad de MongoDB en la actualidad

Para extender y actualizar el tipo de análisis de Chaturvedi a la última versión de MongoDB se acude a la documentación oficial [MongoDB(2021c)] donde se dispone de una guía con las funcionalidades de seguridad que provee actualmente, junto a un checklist de recomendaciones donde hace uso de las mismas.

Se subdivide en cinco grupos de funcionalidades: autenticación, autorización, TLS/SSL, uso empresarial y encriptación:

- Autenticación [MongoDB(2021a)], para verificar la identidad del usuario, se brinda soporte para los mecanismos x.509 [Stalling(2005)] y SCRAM [IETF(2010)], siendo esta última la opción por defecto. También se puede usar membresías (archivos clave con un formato preestablecido) para autenticación interna entre réplicas y shares [MongoDB(2021b)].
- Autorización, para asignar permisos a los usuarios, se brinda soporte para RBAC [MongoDB(2021d)]. Para hacer uso de esta funcionalidad se debe activar el control de acceso, y se puede usar los roles con privilegios ya existentes o definir roles a demanda. Los privilegios serán un conjunto de recursos y acciones que se pueden hacer sobre los mismos, como la lectura de una tabla.
- TLS/SSL [MongoDB(2021e)], o encriptación en capa de transporte, se cuenta con soporte para TLS 1.0 o superior y se destaca que la versión 1.0 debe ser deshabilitada por las vulnerabilidades reportadas sobre la misma[3]. También se recomienda el uso de claves efímeras para mantener la seguridad del cifrado de sesiones finalizadas con el servidor de base de datos, incluso ante compromiso de la clave privada.
- Uso empresarial, opciones no gratuitas, se destacan otros mecanismos de autenticación (Kerberos y LDAP mediante proxy), encriptación *at rest* o de datos persistidos y auditoría para hacer un seguimiento de uso de la base de datos.
- Encriptación de campos del lado cliente, permite una mayor granularidad, sin necesidad de encriptar un documento en su totalidad [MongoDB(2021)].

IV-B. Herramientas y soluciones

Reconocidas las funcionalidades que incluye MongoDB, el siguiente paso es recolectar investigaciones y herramientas que sustenten la seguridad sobre este tipo de bases de datos.

Bajo un enfoque de mejoras respecto al control de acceso, Pietro Colombo y Elena Ferrari de la Universidad de Insuria-Italia presentan dos papers. Primero, en *Complementing MongoDB with Advanced Access Control Features: Concepts and Research Challenges* [Ferrari(2015a)], estudiaron de qué capacidades se dispone para extender el control de acceso RBAC de MongoDB. Fue de interés estudiar el control de acceso según contexto en NoSQL o CAAC por sus siglas en inglés, ya que el mismo es muy utilizado en RDBMSs como Oracle. CAAC, abrió paso a K-VAC, su alternativa para *key-value store* como Cassandra. Respecto a bases documentales, el paper plantea un roadmap que incluye aplicar políticas según el contexto para fortalecer mecanismos de monitoreo y acceso, formalizar la identificación de contextos a partir de la semántica propia de cada contexto, para luego poder discernir sobre qué política aplicar. Por último, empezar a disminuir la granularidad para llegar a nivel de colecciones, documentos y campos.

Este análisis terminó siendo el primer paso hacia un desarrollo específico de CAAC o como hicieron llamar *Prupose-Based Access Control*. El segundo paper, *Enhancing Mon-*

[3] https://media.defense.gov/2021/Jan/05/2002560140/1/-1/0/ELIMINATING_OBSOLETE_TLS_UOO19744320.PDF

goDB with Purpose-based Access Control [Ferrari(2015b)], es donde resaltan que MongoDB con RBAC no admite un nivel de granularidad suficiente para aplicar políticas de privacidad. Por lo tanto, proponen el desarrollo de control de acceso según su fin o propósito. En resumen, brindan las bases para la implementación de un proxy, denominado MEM, con capacidad de monitorear y hasta manipular los mensajes entre cliente y servidor MongoDB. MEM gestiona las sesiones para crear perfiles de usuarios según el uso que le darán a la base de datos, ese perfil será asociado a un rol limitando las colecciones a las cuales podrá acceder.

Respecto a la confidencialidad de los datos, se hace uso de la encriptación porque hoy en día es obligatoria cuando se trata de resguardar datos en tránsito o ya persistidos. En *Combining Fragmentation and Encryption to Ensure Big Data at Rest Security* [Gargouri(2018)] de la Universidad de Sfaz-Túnez se presenta una extensión al uso de cifrado donde se incorpora el uso de fragmentación. La idea clave está en fragmentar los datos según una categorización de datos sensibles, para cifrar esos fragmentos. Para el reconocimiento de datos sensibles se hace uso de PLN (procesamiento de lenguaje natural), luego se fragmenta según niveles de confidencialidad otorgados a cada dato sensible.

En cuanto a las herramientas que se pueden incorporar al servidor de base de datos se destacaron los siguientes ejemplos:

- Los desarrolladores de Studio 3T [Studio3T(2021)] decidieron crear una herramienta que ayude al *data masking*, o encubrir los datos sensibles, en particular es muy útil para trabajar con datos reales en ambientes de testing. Manejar los datos reales sin una cobertura adecuada puede derivar en fuga de información por lo tanto es una posible vulnerabilidad. Además de presentar su herramienta, también comentan desarrollos similares como son IRI, DataSunrise e Informatica con sus dos opciones Dynamic y Persistent.
- MongoDB Monitor [ManageEngine(2021)] propuesto por Application Manager destaca el interés en dar seguimiento al rendimiento de la base de datos, para identificar por ejemplo cuellos de botella. Además se pueden generar reportes y mantener un historial, por lo tanto esta información será de utilidad para agrupar un conjunto de huellas que identifiquen comportamiento sospechoso a pesar que no sea un objetivo de esta herramienta.
- En cuanto a auditoría se suele citar a mongoaudit [4], una herramienta desarrollada en Python para auditar servidores MongoDB con el fin de detectar configuraciones de seguridad inadecuadas o vulnerables y realizar pruebas de penetración automatizadas.
- También sobre auditoría, pero más específica, se puede considerar mongodb-brute [Karlsson(2012)], que es un script de la herramienta Nmap[5] desarrollado para auditar el uso de fuerza bruta sobre autenticación en MongoDB. Para el mismo objetivo también se tiene todo un proyecto opensource denominado Go-mbf o MongoDB Brute Force[6].

[4]https://github.com/stampery/mongoaudit

[5]https://nmap.org/book/man.html#man-description

V. Problemáticas y recomendaciones

V-A. Problemáticas

Hasta el momento se investigó qué avances se tienen respecto a hardening en bases de datos documentales, en particular en MongoDB. A continuación se especificará una lista de problemáticas frecuentes y en la próxima sección se listarán recomendaciones para reducir el impacto de estas. Una problemática engloba un conjunto de vulnerabilidades de características similares.

La lista presentada a continuación está basada en el proyecto *OWASP TOP TEN* para aplicaciones web [OWASP(2017)] y un artículo sobre ataques a bases de datos de *BCS The Chartered Institute for IT* [BCS(2019)]. Dichas fuentes inspiraron nombres y definiciones.

1. **Unprotected to NoSQL Injection and XSS**
   Las fallas de inyección NoSQL ocurren cuando se envían datos no fiables a un intérprete como parte de una consulta. Los datos hostiles del atacante pueden engañar al intérprete para que acceda a datos sin la autorización correspondiente.
   Los defectos de XSS ocurren cuando los datos hostiles no pasaron por una validación acdecuada, permite a los atacantes ejecutar scripts en el servidor de base de datos de la víctima. De esta forma pueden robar o modificar su contenido, así como evadir el proceso de autenticación.
2. **Broken or Weak Authentication**
   Las funciones relacionadas con la autenticación y la administración de sesiones a menudo presentan falencias de seguridad que permite a los atacantes comprometer contraseñas o aprovechar otras fallas del propio manejador de base de datos para asumir las identidades de otros usuarios de forma temporal o permanente.
3. **Inadequate or Weak Encryption Strength**
   Los atacantes pueden robar o modificar datos confidenciales débilmente protegidos para cometer fraude con tarjetas de crédito, robo de identidad u otros delitos. Los datos confidenciales pueden verse comprometidos por carecer de protección adicional como el cifrado, tanto al ser almacenados (*at rest*) como en tránsito.
4. **Broken Access Control**
   Las restricciones sobre lo que pueden hacer los usuarios autenticados a menudo se aplican de forma incorrecta. Los atacantes pueden aprovechar estas fallas para acceder a funciones y / o datos no autorizados, acceder a las cuentas de otros usuarios, ver archivos confidenciales, modificar los datos de otros usuarios o cambiar los derechos de acceso.

[6]https://github.com/c0nrad/go-mbf

5. **Inadequate Privilege Management**
   Abuso malintencionado de privilegios legítimos o privilegios otorgados por sobre los mínimos indispensables para los requerimientos de sus funciones.
6. **Insecure System Architecture and Deployment**
   Controles contra amenazas específicas que forman parte del diseño y el despliegue del servidor de base de datos, incluyendo su acceso a nivel de red.
7. **Inadequate Backup Management**
   La falta de garantías de seguridad sobre los backups por localizarse en un servidor sin protección, no manejar cifrado y/o exponer información sensible. Así mismo mala configuración de los procedimientos de creación y recuperación de backups, generando una copia de respaldo inapropiada.
8. **Using Components with Known Vulnerabilities**
   Los componentes, como bibliotecas, plugins y otros módulos de software, se pueden ejecutar con los mismos privilegios que el manejador de base de datos. Si se explota un componente vulnerable, se puede facilitar cualquier ataque que provoque pérdida de datos o la toma de control del servidor.
9. **Insufficient Logging & Monitoring**
   El registro y la supervisión insuficiente, junto con una respuesta a incidentes ineficiente o incluso inexistente, permite aún más ataques sobre los sistemas, mantener la persistencia del ataque, propagarse a más sistemas y manipular, extraer o destruir datos.
10. **Unprotected to Denial of Services**
    Los ataques de denegación de servicio (DoS) a nivel de red pueden perjudicar el desempeño de su base de datos a raíz de un consumo excesivo de recursos.

V-B. Recomendaciones

Presentadas las problemáticas a considerar, resta plantear las recomendaciones de hardening para disminuir la ocurrencia de vulnerabilidades. Aplicar estas recomendaciones permiten incorporar garantías de seguridad sobre propiedades como la confidencialidad, integridad y disponibilidad. Confidencialidad porque *se prevé la difusión no autorizada de la información*, integridad porque *se prevé la modificación no autorizada de la información* y disponibilidad porque *se prevé la apropiación no autorizada de información o recursos* [Betarte(2020)].

A continuación se presentan las recomendaciones para cada una de las diez problemáticas consideradas:

1. **Unprotected to NoSQL Injection and XSS**
   - Usar una librería para sanitizar la entrada. Esto es, validar, filtrar y limpiar la entrada previo a ser utilizada [Andrzej(2020)].
   - Validar toda entrada de usuario para corroborar que sea aceptada [Liu(2016)].
   - Limpiar la entrada corresponde a pasar toda entrada de usuario al tipo esperado, por ejemplo nombres de usuario a string [Andrzej(2020)].
   - Usar consultas parametrizadas. Esto permite comprobar cuál es el formato de la entrada y filtrar la misma, previo a ser considerada como parte de una consulta [Liu(2016)].
   - En la medida de lo posible no utilizar los operadores *where*, *mapReduce* o *group* sobre las entradas de usuario. [Andrzej(2020)]
   - En la medida de lo posible deshabilitar la ejecución de código javascript del lado del servidor. [Andrzej(2020)]
2. **Broken or Weak Authentication**
   - Limitar a seis la cantidad de intentos fallidos al momento de iniciar sesión con el manejador NoSQL [Baykara(2020)].
   - Bloquear por un periodo de 30 minutos la cuenta del usuario que superó el límite de intentos fallidos en el inicio de sesión [Baykara(2020)].
   - Crear y forzar el uso de una política de contraseñas fuerte que incluya los cuatro tipos de caracteres (minúscula, mayúscula, números y símbolos) con una longitud no menor a los ocho caracteres, evitando el uso de palabras del diccionario [NIST(2021)].
   - Considerar la autenticación multifactor, se podría incorporar el uso de certificados de clave privada o tokens de sesión [BCS(2019)].
   - Almacenamiento de contraseñas haciendo uso de un algoritmo de hash fuerte, como SHA-256[7], y salt[8] en la contraseña con el fin de obtener siempre un valor distinto de hash aunque sea sobre la misma contraseña [BCS(2019)].
3. **Inadequate or Weak Encryption Strength**
   - Encriptar, como mínimo, los datos sensibles de todas las comunicaciones. A modo de ejemplo, la comunicación entre réplicas o *shares*. [MongoDB(2020)]
   - Encriptar, como mínimo, los datos sensibles almacenados [LIFARS(2020)].
   - Encriptar, como mínimo, los datos sensibles en todo backup que se genere [LIFARS(2020)].
   - En capa de transporte usar encriptación TLS 1.2 o superior[9].
   - Encriptar datos con algoritmos fuertes como AES o 3DES [Turner(2017)].
4. **Broken Access Control**
   - Habilitar el control de acceso y limitar el acceso a datos según una política de acceso por roles RBAC [MongoDB(2020)].
   - Evitar exponer interfaces que permitan consultas arbitrarias y exportación masiva de datos [BCS(2019)].
   - De necesitar una interfaz se debe registrar su uso, auditar periódicamente y limitar la cantidad de usuarios con acceso a dicha interfaz [BCS(2019)].

[7]https://csrc.nist.gov/Projects/Hash-Functions/NIST-Policy-on-Hash-Functions

[8]https://csrc.nist.gov/glossary/term/salt

[9]https://datatracker.ietf.org/doc/html/rfc8996

- Implementar una lista de control de acceso a nivel de usuarios para dar permisos individuales por usuario [Vaillancourt(2019)].

5. **Inadequate Privilege Management**
   - Definir roles según las reglas de negocio para dar acceso a la mínima cantidad de recursos y funcionalidades que cada rol requiere [Vaillancourt(2019)].
   - Crear procedimientos a seguir para asegurar el cambio de privilegios al cambiar de roles [BCS(2019)].
   - Revisiones regulares, pero no necesariamente frecuentes, de quién tiene qué roles para confirmar que los procedimientos están funcionando [BCS(2019)].
   - Comenzar con usuarios de rol administrador, crear nuevos roles y usuarios a partir de un pedido explícito [Vaillancourt(2019)].
   - Revocar los permisos de un usuario inmediatamente después de terminar su vínculo contractual [Vaillancourt(2019)].
6. **Insecure System Architecture and Deployment**
   - Tener un solo punto de acceso a la información [Hartson(2003)].
   - Autorización descentralizada. Los accesos se solicitan a diferentes entidades, con responsabilidades diferentes. En caso de información sensible, se puede requerir la autorización de más de una entidad [Hartson(2003)].
   - Los mecanismos de seguridad deben ser contemplados desde el principio. Las decisiones de diseño deben considerar factores de seguridad [Hartson(2003)].
   - Información expuesta a Internet debe ser almacenada en una base de datos aislada.
7. **Inadequate Backup Management**
   - La información debe ser encriptada [Crowther(2013)].
   - No debe poder sobrescribirse el dispositivo en el que se hace el backup [Anderson(2008)].
   - Backups a intervalos regulares [Anderson(2008)].
   - Mantener registros de las operaciones realizadas entre checkpoints [Anderson(2008)].
   - Limitar el acceso al backup a aquellas personas directamente relacionadas con el mismo. Incluye acceso digital y físico, tanto al software utilizado como a los archivos per se [Beaver(2020)].
   - Mantener backups en diferentes puntos geográficos [Beaver(2020)].
   - Mantener el backup en una red diferente, física o lógicamente separada del resto [Beaver(2020)].
   - Testear regularmente el backup, asegurarse de que todo se está respaldando correctamente [Beaver(2020)].
8. **Using Components with Known Vulnerabilities**
   - Eliminar dependencias, componentes y funcionalidades que no se utilicen [OWASP(2017)].
   - Inventario frecuente de versiones de los componentes utilizados [OWASP(2017)].
   - Automatizar proceso de monitoreo continuo a los componentes utilizados para verificar que no se hayan detectado vulnerabilidades [OWASP(2017)].
   - Obtener los componentes de sitios oficiales [OWASP(2017)].
   - Monitorear librerías que están fuera de mantenimiento o a las que no se le están implementando parches de seguridad [OWASP(2017)].
9. **Insufficient Logging & Monitoring**
   - Seleccionar criteriosamente la información a loguear. La cantidad de información puede escalar rápidamente, afectar la performance y dificultar la utilización de los datos auditados [Oracle(2021)].
   - Considerar la privacidad de la información logueada, seguramente incluya datos personales de los usuarios [Oracle(2021)].
   - Proteger los logs para evitar modificaciones en la información recabada [Oracle(2021)].
10. **Unprotected to Denial of Services**
    - Aumentar el ancho de banda [Anderson(2008)].
    - Contratar ISP que identifica y filtra paquetes *spoofed* o falsificados [Anderson(2008)].
    - Utilizar versión superior a 3.1.3. En versiones anteriores MongoDB presenta una falla al intentar buscar en una colección que no existe.
    - Restringir el tráfico, balanceadores de carga, listas de accesos, firewalls [AWS(2021)].
    - Conocer el tráfico normal para poder examinar los paquetes recibidos, verificarlos y filtrarlos en caso de que corresponda [AWS(2021)].

VI. Conclusiones y trabajo futuro

La seguridad en bases de datos NoSQL, en particular las bases de datos documentales, está siendo foco de diversas investigaciones a raíz de algunos ataques exitosos, como se comprobó al analizar el estado del arte. Dado que el objetivo inicial de este tipo de bases de datos era la escalabilidad y eficiencia, la consideración de seguridad en este contexto es reciente y surge como consecuencia de la gran popularidad ganada en los últimos años. Se detecta la necesidad de centralizar una primera aproximación al hardening de bases de datos documentales, objetivo principal de este trabajo. En el presente trabajo se agruparon las vulnerabilidades en problemáticas, considerando en total diez problemáticas, a las que se les asignó un conjunto de recomendaciones a seguir. El equipo entiende que este aporte es sumamente valioso porque permite prevenir falencias graves de seguridad que, por desconocimiento, terminan siendo la puerta de entrada de usuarios malintencionados.

Como extensión de este trabajo se resalta la necesidad de obtener la aprobación de la sociedad científica. Esta tarea se puede realizar por dos caminos paralelos, contactando a la fundación OWASP por su experiencia en la seguridad

informática, o al equipo de desarrollo de MongoDB por su experiencia en bases de datos documentales. Adicionalmente, sería interesante el desarrollo de un toolkit que incorpore herramientas con capacidad para auditar las recomendaciones sobre un despliegue en producción de una base de datos MongoDB. Esto permitiría automatizar procesos para reconocer la completitud del hardening.

REFERENCIAS

[Abramov(2011)] Okman & GalOz & Gonen & Gudes & Abramov. Security issues in nosql databases, 2011.

[Anderson(2008)] Ross Anderson. *Security Engineering*. Wiley, 2008.

[Andrzej(2020)] Tomasz Andrzej. Nosql injections and how to avoid them. https://www.acunetix.com/blog/web-security-zone/nosql-injections/, 2020.

[AWS(2021)] AWS. https://aws.amazon.com/es/shield/ddos-attack-protection/, 2021.

[Baykara(2020)] Surkay Baykara. What are the pci dss password requirements. https://www.pcidssguide.com/what-are-the-pci-dss-password-requirements/, 2020.

[BCS(2019)] BCS. Top 10 database attacks. https://www.bcs.org/content-hub/top-ten-database-attacks/, 2019.

[Beaver(2020)] Kevin Beaver. 10 guidelines to secure your data backup, 2020.

[Betarte(2020)] Gustavo Betarte. Clase 2 - introduccion. In *Fundamentos de la Seguridad Informática*, Montevideo, Uruguay, 2020.

[Bischoff(2019)] Paul Bischoff. 5 million personal records belonging to medicaresupplement.com exposed to public. https://www.comparitech.com/blog/information-security/medicare-supplement-data-breach/, 2019.

[Chaturvedi(2018)] Alankrit Chaturvedi. *Security Challenges and Solutions in MongoDB*. International Journal of Science and Research, Carnegie Mellon University, Pittsburgh, 2018.

[Crowther(2013)] Lake & Crowther. *Concise Guide to Databases*. Springer, 2013.

[Debbabi(2006)] Mourad & Laverdiere & Debbabi. Security hardening of open source software, 2006.

[Ferrari(2015a)] Colombo & Ferrari. 23rd italian symposium on advanced database systems. In *Complmenting MongoDB with Advanced Access Control Features: Concepts and Research Challenges*, Gaeta, Italy, 2015a.

[Ferrari(2015b)] Colombo & Ferrari. Enhancing mongodb with purpose-based access control, 2015b.

[Gargouri(2018)] Heni & Abdallah & Gargouri. Combining fragmentation and encryptation to ensure big data at rest security, 2018.

[Harrison(2015)] Guy Harrison. *Next Generation Databases: NoSQL, NewSQL, and BigData*. Apress, 2015.

[Hartson(2003)] H.Rex Hartson. Database security—system architectures, 2003.

[IETF(2010)] Internet Engineering Task Force IETF. Rfc5802. https://datatracker.ietf.org/doc/html/rfc5802, 2010.

[Karlsson(2012)] Patrik Karlsson. mongodb-brute nse script. https://nmap.org/nsedoc/scripts/mongodb-brute.html, 2012.

[LIFARS(2020)] LIFARS. Boost mongodb security and keep hackers away with these 10 powerful tips. https://lifars.com/2020/05/boost-mongodb-security-and-keep-hackers-away-with-these-10-powerful-tips/, 2020.

[Liu(2016)] Hou & Qiun & Li & Shi & Tao & Liu. Mongodb nosql injection analysis and detection. In *IEEE 3rd International Conference on Cyber Security and Cloud Computing*, Beijing, China, 2016.

[ManageEngine(2021)] ManageEngine. Mongodb monitor tool, 2021. URL https://www.manageengine.com/products/applications_manager/mongodb-monitoring.html.

[Mikalauskas(2021)] Edvardas Mikalauskas. Research: 19 petabytes of data exposed across 29,000+ unprotected databases. https://cybernews.com/security/19-petabytes-of-data-exposed-worldwide/, 2021.

[MongoDB(2020)] MongoDB. Security checklist. https://docs.mongodb.com/manual/administration/security-checklist/, 2020.

[MongoDB(2021)] MongoDB. Client-side field level. https://docs.mongodb.com/manual/core/security-client-side-encryption/, 2021.

[mongoDB(2021)] mongoDB, 2021. URL https://www.mongodb.com/.

[MongoDB(2021a)] MongoDB. *Authentication - MongoDB Manual*, 2021a. URL https://docs.mongodb.com/manual/core/authentication/.

[MongoDB(2021b)] MongoDB. *Interlan/Membership Authentication - MongoDB Manual*, 2021b. URL https://docs.mongodb.com/manual/core/security-internal-authentication/#std-label-inter-process-auth.

[MongoDB(2021c)] MongoDB. *Security MongoDB - Manual*, 2021c. URL https://docs.mongodb.com/manual/security/.

[MongoDB(2021d)] MongoDB. *Roled-Based Access Control - MongoDB Manual*, 2021d. URL https://docs.mongodb.com/manual/core/authorization/.

[MongoDB(2021e)] MongoDB. *TLS/SSL (Transport Encryptation) - MongoDB Manual*, 2021e. URL https://docs.mongodb.com/manual/core/security-transport-encryption/.

[NIST(2021)] National Institute Of Standards & Tecnology NIST. 2021 nist password recommendations. https://securityboulevard.com/2021/03/nist-password-guidelines-2021-challenging-traditional-password-management/, 2021.

[Oracle(2021)] Oracle. Database security guide. https://docs.oracle.com/cd/B19306_01/index.htm, 2021.

[Owaida(2020)] Amer Owaida. Almost 4000 databases wiped in 'meow' attacks. https://www.welivesecurity.com/2020/07/27/almost-4000-databases-wiped-meow-attacks/, 2020.

[OWASP(2017)] OWASP. Owasp top ten. https://owasp.org/www-project-top-ten/, 2017.

[Stalling(2005)] William Stalling. *Cryptography and Network Security*. Prentice Hall, Upper Saddle River, NJ 07458, 2005.

[Studio3T(2021)] Studio3T. The best data masking tools for mongodb. https://studio3t.com/knowledge-base/articles/best-data-masking-tools/, 2021.

[Turner(2017)] Dawn M. Turner. Strong cryptography and key management requirements for emv and pci dss compliance. https://www.cryptomathic.com/news-events/blog/strong-cryptography-requirements-for-emv-and-pci-dss-compliance, 2017.

[Vaillancourt(2019)] Tim Vaillancourt. The definitive guide to mongodb security. https://opensource.com/article/19/1/mongodb-security, 2019.