بسمه تعالی

سازمان فناوری اطلاعات ایران

معاونت امنیت فضای تولید و تبادل اطلاعات

مرکز ماهر

مقاوم‌سازی امنیتی MongoDB

فهرست مطالب

پیشگفتار

در این مستند، مقاوم‌سازی امنیتی MongoDB نسخه ۳,۴ مورد بررسی قرار می‌گیرد. به این منظور، نحوه وارسی و ایمن‌سازی مقادیر و تنظیمات مربوط به تعداد از پارامترهای تاثیرگذار در عملکرد این پایگاه‌داده معرفی می‌گردند. در مورد هر پارامتر، کاربرد، ارزش امنیتی، نحوه آگاهی از مقدار کنونی و در نهایت چگونگی مقداردهی امن پارامتر تعیین می‌گردد.

بررسی پارامترهای مربوط به مقاوم‌سازی MongoDB در شش بخش متمایز صورت می‌گیرد. در بخش اول، نیازمندی‌های مرتبط با امن‌سازی محیط اجرا ارائه می‌شود. بخش دوم به تشریح پارامترهای نصب و پیکربندی MongoDB می‌پردازد. بخش سوم به معرفی محدودیت‌هایی اختصاص دارد که باید بر روی فرآیند اتصال و ورود کاربران به MongoDB اعمال گردد. در بخش چهارم، پارامترهای کنترل دسترسی و مجازشماری مورد بررسی قرار می‌گیرند. پارامترهای مربوط به رویدادنگاری امن در بخش پنجم بررسی می‌شوند. در بخش ششم، تنظیمات مربوط به رمزنگاری مورد توجه قرار می‌گیرند. در پایان نیز، نحوه اجرای اسکریپت‌ها و اعمال تنظیمات مورد نیاز برای مقاوم‌سازی پایگاه‌داده بیان می‌گردد.

۱ امن‌سازی محیط اجرا

در این فصل، پیشنهاداتی برای مقاوم‌سازی پایگاه‌داده MongoDB که مربوط به سیستم‌عامل است؛ مانند تنظیمات پیشنهادی برای مجوزهای فایل کلید و فایل پایگاه‌داده؛ تشریح می‌گردد.

۱ – ۱ راه‌اندازی پایگاه‌داده MongoDB با حداقل مجوزها

باید این اطمینان حاصل شود که سرویس mongod توسط کاربری با حداقل مجوز در حال اجرا است.

تهدید/توجیه امنیتی:

هرکسی که به نوعی قربانی ویروس‌ها، کرم‌ها، و سایر نرم‌افزارهای مخرب و بدافزارها بوده است، قانون امنیتی حداقل مجوزها را تأیید می‌کند. در صورتی که تمامی پروسه‌ها با حداقل مجوزهای موردنیاز به فعالیت بپردازند، آلوده‌سازی یک ماشین توسط بدافزارها و انتشار آن به سایر ماشین‌ها دشوارتر خواهد بود.

اطلاع از وضعیت فعلی:

با اتصال به سرویس MongoDB و مشاهده‌ی خروجی آن، می‌توان متوجه شد که آیا سرویس با مجوز root در حال اجرا است یا خیر. در صورتی که سرویس MongoDB با مجوز کاربر root اجرا شده باشد، هشدار زیر هنگام اتصال به سرویس نمایش داده می‌شود.

```
WARNING: You are running this process as the root user, which is not
recommended.
```

مقاوم‌سازی:

می‌بایست تنها یک حساب کاربری برای اجرای MongoDB و پروسه‌های مرتبط با آن ایجاد و مورد استفاده قرار گیرد.لازم به ذکر است که کاربر فوق الذکر نباید دارای مجوزهای مدیریتی در سیستم باشد. برای ایجاد چنین کاربری از دستور زیر استفاده می‌شود.

```
useradd -m -d /home/mongodb -s /bin/bash -g mongodb -u 1234 mongodb
```

حال مالکیت <dbpath> را به کاربر mongodb می‌توان اعطا کرد.

```
sudo chown -R mongodb:mongodb <dbpath>
```

۱-۲ مجوزهای فایل کلید

فایل کلید به منظور احراز اصالت در خوشه‌بندی انحصاری[1] استفاده می‌شود. پیاده‌سازی مجوزهای مناسب بر روی فایل کلید از دسترسی‌های غیرمجاز به آن جلوگیری می‌کند.

تهدید/توجیه امنیتی:

حفاظت از فایل کلید موجب تقویت احراز اصالت در خوشه‌بندی انحصاری و جلوگیری از دسترسی‌های غیرمجاز به پایگاه‌داده می‌شود.

اطلاع از وضعیت فعلی:

ابتدا با استفاده از دستور زیر می‌توان محل فایل کلید را به دست آورد:

```
cat /etc/mongod.conf |grep "keyFile:"
```

آدرسی که در دستور فوق به دست آمده است، در دستور زیر قرار گرفته و مجوزهای فایل کلید استخراج می‌شوند:

```
ls -l <keyFile>
```

مقاوم‌سازی:

مالکیت فایل کلید باید به کاربر mongodb اعطا شود و سایر مجوزها بر روی این فایل حذف گردند.

```
chmod 600 <keyFile>
sudo chown mongodb:mongodb <keyFile>
```

۱-۳ مجوزهای فایل پایگاه‌داده

فایل‌های پایگاه‌داده باید توسط تخصیص مجوزهای امن به آنها حفاظت شوند.

تهدید/توجیه امنیتی:

با حفاظت از فایل‌های پایگاه‌داده از طریق مجوزهای مناسب می‌توان از دسترسی‌های غیرمجاز به پایگاه‌داده جلوگیری کرد.

اطلاع از وضعیت فعلی:

---

[1] Sharded Cluster

با استفاده از دستور زیر می‌توان محل فایل پایگاه‌داده را به دست آورد.

```
cat /etc/mongod.conf |grep "dbPath:"
```

آدرسی که به کمک دستور فوق به دست آمده است، در دستور زیر قرار گرفته و مجوزهای فایل پایگاه‌داده استخراج می‌شوند.

```
ls -l <dbpath>
```

مقاوم‌سازی:

مالکیت فایل پایگاه‌داده باید به کاربر mongodb اعطا شود و سایر مجوزها بر روی این فایل حذف گردند.

```
chmod 760 <dbpath>
sudo chown mongodb:mongodb <dbpath>
```

لازم به ذکر است که بر طبق [۱]، تغییر مجوزهای فایل پایگاه‌داده می‌بایست به ۶۶۰ باشد ولی در عمل در صورتی که مجوزهای فایل پایگاه‌داده به ۶۶۰ تغییر کنند، خطای زیر تولید می‌شود.

```
exception in initAndListen: xx unable to determine status of lock
file in the data directory /var/lib/mongodb:
boost::filesystem::status: Permission denied:
"/var/lib/mongodb/mongod.lock", terminating
```

۱-۴ جمع‌بندی

در این فصل به تشریح پارامترهای امنیتی محیط اجرای سمپاد که به طور مستقیم بر عملکرد آن تاثیرگذار است، پرداختیم. در این راستا، مجوزهای فایل کلید و فایل پایگاه‌داده مورد بحث و بررسی قرار گرفت. مدیر سامانه به منظور بررسی پارامترهای مقاوم‌سازی و تهیه گزارش در این زمینه می‌تواند از چک لیست زیر استفاده نماید.

	عنوان	تنظیم صحیح	
		بله	خیر
۱	**ایمن سازی محیط اجرا**		
۱-۱	راه‌اندازی پایگاه‌داده MongoDB با حداقل مجوزها	☐	☐
۱-۲	مجوزهای فایل کلید	☐	☐
۱-۳	مجوزهای فایل پایگاه‌داده	☐	☐

۲ نصب و پیکربندی امن پایگاه‌داده

در این فصل، به مقاوم‌سازی امن پارامترهای مربوط به نصب و پیکربندی پایگاه‌داده MongoDB؛ مانند عدم استفاده از پورت پیش‌فرض و به‌روزرسانی بسته‌های تکمیلی؛ می‌پردازیم.

۲ – ۱ عدم استفاده از پورت پیش‌فرض

تغییر پورت پیش‌فرض استفاده شده توسط MongoDB، یافتن پایگاه‌داده و مورد حمله قرار گرفتن آن توسط مهاجمین را دشوارتر می‌کند.

تهدید/توجیه امنیتی:

پورت‌های استاندارد در اجرای حملات خودکار، استفاده می‌شوند. همچنین مهاجمین از طریق پورت‌های استاندارد می‌توانند متوجه شوند که کدامیک از برنامه‌های کاربری بر روی سرور در حال اجرا هستند.

اطلاع از وضعیت فعلی:

دستور زیر برای یافتن شماره پورت استفاده شده توسط MongoDB به کار می‌رود.

```
cat /etc/mongod.conf |grep "port";
```

مقاوم‌سازی:

شماره پورت سرور MongoDB باید به عددی غیر از ۲۷۰۱۷ تغییر یابد. بدین ترتیب در فایل پیکربندی /etc/mongod.conf تنظیم زیر اعمال می‌شود:

```
port: <new_port>
```

همچنین با استفاده از دستور زیر می‌توان پورت را از ۲۷۰۱۷ به عدد مورد نظر تغییر داد:

```
sudo sed -i "s/port\s*:\s*27017/port: <new_port>/" "/etc/mongod.conf"
```

۲ – ۲ به‌روزرسانی نرم‌افزار MongoDB و نصب وصله‌ها

نسخه‌ی نصب شده MongoDB و بسته‌های تکمیلی باید آخرین موارد منطبق با نیازمندی‌های عملیاتی سازمان باشند.

تهدید/توجیه امنیتی:

نصب آخرین نسخه نرم‌افزار MongoDB به همراه بسته‌های تکمیلی، احتمال سوء استفاده از آسیب‌پذیری‌های محتمل در نرم‌افزار را کاهش می‌دهد.

اطلاع از وضعیت فعلی:

با اجرای دستور زیر در پوسته[2] MongoDB نسخه نرم‌افزار MongoDB به دست می‌آید.

```
> db.version()
```

مقاوم‌سازی:

به‌روزرسانی نرم‌افزار MongoDB با گام‌های زیر انجام می‌شود:

۱. تهیه پشتیان از داده‌ها
۲. دانلود فایل‌های دودویی[3] از آخرین نسخه MongoDB
۳. متوقف کردن نمونه[4] MongoDB
۴. جایگزینی فایل‌های دودویی جدید با فایل‌های دودویی موجود
۵. راه‌اندازی مجدد نمونه MongoDB

۲-۳ جمع‌بندی

در این فصل به تشریح برخی از مهم‌ترین تنظیمات مربوط به پیکربندی سمپاد قبل از بکارگیری عملیاتی آن پرداختیم. در این راستا عدم استفاده از پورت پیش‌فرض و به‌روزرسانی بسته‌های تکمیلی MongoDB مورد بحث و بررسی قرار گرفت. مدیر سامانه به منظور بررسی پارامترهای مقاوم‌سازی و تهیه گزارش در این زمینه می‌تواند از چک لیست زیر استفاده نماید.

	عنوان	تنظیم صحیح	
		بله	خیر
۲	**پیکربندی امن پایگاه‌داده**		
۲-۱	عدم استفاده از پورت پیش‌فرض	☐	☐
۲-۲	به‌روزرسانی نرم‌افزار MongoDB و نصب وصله‌ها	☐	☐

[2] Shell
[3] Binary
[4] Instance

۳ امن‌سازی اتصال به پایگاه‌داده

در این فصل پیشنهاداتی برای احراز اصالت کاربران پیش از اعطای مجوز برای دسترسی به پایگاه‌داده MongoDB ارائه شده است.

۳-۱ فعال‌سازی احراز اصالت

با فعال‌سازی احراز اصالت برای پایگاه‌داده MongoDB می‌توان مطمئن بود که تمامی کاربران پیش از دسترسی به پایگاه‌داده MongoDB باید اصالتشان تصدیق شود.

تهدید/توجیه امنیتی:

عدم احراز اصالت کاربران امکان دسترسی‌های غیرمجاز به پایگاه‌داده MongoDB را فراهم کرده و می‌تواند از ردگیری فعالیت‌های انجام شده جلوگیری کند.

اطلاع از وضعیت فعلی:

خروجی دستور زیر مشخص می‌کند که آیا احراز اصالت بر روی سرور MongoDB فعال است یا خیر.

```
cat /etc/mongod.conf | grep “authorization:”
```

در صورتی که مقدار authorization برابر enabled باشد، احراز اصالت بر روی سرور MongoDB فعال است.

مقاوم‌سازی:

پیش از آنکه دسترسی به سرور MongoDB انجام شود، باید مکانیزم احراز اصالت پیاده‌سازی شده باشد.

به منظور فعال‌سازی مکانیزم احراز اصالت گام‌های زیر باید طی شوند:

۱. نمونه MongoDB بدون احراز اصالت راه‌اندازی شود.

```
mongod --port 27017 --dbpath /data/db1
```

۲. کاربری به عنوان مدیر سیستم ایجاد شود. رمز عبور تعریف شده برای کاربر باید مطابق با نیازمندی‌ها و خط‌مشی‌های تعریف شده‌ی سازمان باشد.

```
use admin
db.createUser(
{
user: "siteUserAdmin",
pwd: "password",
roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]
```

```
})
```

۳. نمونه MongoDB با فعال‌سازی احراز اصالت مجدداً راه‌اندازی شود.

```
mongod --auth --config /etc/mongod.conf
```

۳-۲ اطمینان حاصل شود که ورود به صورت localhost، احراز اصالت را کنار نمی‌گذارد

پایگاه‌داده MongoDB نباید احراز اصالت را با استفاده از استثنای localhost کنار گذارد. استثنای localhost این امکان را فراهم می‌کند که پیش از آنکه اولین کاربر در سیستم ایجاد شود، بتوان به localhost وصل شد و به پایگاه‌داده دسترسی پیدا کرد. تنها از استثنای localhost زمانی که هنوز هیچ کاربری بر روی نمونه MongoDB ایجاد نشده است، باید استفاده شود.

تهدید/توجیه امنیتی:

با غیرفعال‌سازی استثنای localhost از دسترسی محلی غیرمجاز به پایگاه‌داده MongoDB جلوگیری می‌شود. همچنین می‌توان مطمئن بود که هر فعالیت در پایگاه‌داده قابل ردگیری است.

اطلاع از وضعیت فعلی:

به منظور بررسی وضعیت استثنای localhost از دستور زیر استفاده می‌شود. خروجی دستور زیر باید false باشد.

```
cat /etc/mongod.conf |grep "enableLocalhostAuthBypass"
```

مقاوم‌سازی:

به منظور غیرفعال‌سازی استثنای localhost، تنظیمات زیر در فایل پیکربندی می‌بایست اعمال گردد.

```
setParameter:
enableLocalhostAuthBypass: false
```

۳-۳ اطمینان از فعال بودن احراز اصالت در خوشه منشعب شده

در صورتی که فایل‌های کلید[۵] برای تمامی مؤلفه‌ها ایجاد و پیکربندی شده باشند، احراز اصالت در خوشه منشعب شده فعال است. بدین ترتیب می‌توان مطمئن بود که هر کارخواهی که به خوشه دسترسی پیدا

[۵] Keyfiles

می‌کند، باید احراز اصالت شود. این مورد شامل دسترسی نمونه‌های MongoDB در خوشه به یکدیگر هم می‌شود.

تهدید/توجیه امنیتی:

استفاده از کلید در خوشه منشعب شده از دسترسی‌های غیرمجاز به پایگاه‌داده MongoDB جلوگیری کرده و امکان ردگیری فعالیت‌های پایگاه‌داده و نسبت دادن آن‌ها به کاربر یا مؤلفه مشخص را فراهم می‌کند.

اطلاع از وضعیت فعلی:

با استفاده از دستور زیر می‌توان بررسی کرد که آیا پارامتر keyFile تنظیم شده است یا خیر.

```
cat /etc/mongod.conf | grep "keyFile:"
```

مقاوم‌سازی:

به منظور فعال‌سازی احراز اصالت در خوشه منشعب شده گام‌های زیر باید طی شوند:

۱. ایجاد یک فایل کلید
۲. بر روی هر مؤلفه در خوشه منشعب شده باید در فایل پیکربندی /etc/mongod.conf مقدار پارامتر keyFile را برابر مسیر فایل کلید قرار داد و سپس با استفاده از دستور زیر مؤلفه را راه‌اندازی کرد:

```
keyFile = /srv/mongodb/keyfile
```

همچنین می‌توان هنگام راه‌اندازی مؤلفه، گزینه keyFile-- را تنظیم کرد. این گزینه برای نمونه‌های mongos و mongod به کار می‌روند. مقدار keyFile-- برابر مسیر فایل کلید است.

۳-۴ اطمینان از استفاده از مکانیزم احراز اصالت استاندارد صنعتی

استفاده از یک یا چندین مکانیزم احراز اصالت استاندارد صنعتی[۷] به سازمان‌ها کمک می‌کند تا خط‌مشی‌های رمز عبور و حساب‌های کاربری خود را برای کاربران MongoDB به کار ببرند.

تهدید/توجیه امنیتی:

بدون استفاده از مکانیزم احراز اصالت استاندارد صنعتی، مدیریت حساب‌های کاربری و رمز عبور آن‌ها کاری خسته‌کننده و طاقت‌فرسا خواهد بود و ممکن است احراز اصالت با خط‌مشی‌های سازمان منطبق نباشد.

---

[۶] Sharded Cluster – مجموعه ای از گره‌ها (Node) یک استقرار توزیع شده MongoDB را شکل می دهند.

[۷] Industry Standard Authentication Mechanism

اطلاع از وضعیت فعلی:

به منظور بررسی مکانیزم احراز اصالت به کار برده شده در پایگاه‌داده MongoDB از دستورات زیر استفاده می‌شود.

```
cat /etc/mongod.conf | grep “clusterAuthMode:”
cat /etc/mongod.conf | grep “mode:”
cat /etc/mongod.conf | grep “authorization:"
cat /etc/mongod.conf | grep “authenticationMechanisms:”
```

مقاوم‌سازی:

به منظور پیاده‌سازی مکانیزم احراز اصالت استاندارد صنعتی، می‌توان از نمونه‌های زیر به عنوان الگویی برای تعیین مکانیزم احراز اصالت در فایل پیکربندی MongoDB استفاده کرد. به عنوان نمونه می‌توان از گواهینامه‌های x.509 برای احراز اصالت کاربران و اعمال تنظیمات زیر در فایل پیکربندی MongoDB استفاده کرد.

```
security:
clusterAuthMode: x509
net:
ssl:
mode: requireSSL
PEMKeyFile: <path to TLS/SSL certificate and key PEM file>
CAFile: <path to root CA PEM file>
```

همچنین تنظیمات زیر برای اعمال احراز اصالت Kerberos در لینوکس به کار می‌روند.

```
security:
authorization: enabled
setParameter:
authenticationMechanisms: GSSAPI
storage:
dbPath: /opt/mongodb/data
```

۳-۵ جمع‌بندی

در این فصل به تشریح تنظیمات مربوط به امن‌سازی اتصال به سمپاد MongoDB پرداختیم. در این راستا، برخی از تنظیمات مرتبط مورد بحث و بررسی قرار گرفتند. مدیر سامانه به منظور بررسی پارامترهای مقاوم‌سازی و تهیه گزارش در این زمینه می‌تواند از چک لیست زیر استفاده نماید.

	عنوان	تنظیم صحیح	
		بله	خیر
۳	**امن‌سازی اتصال به پایگاه‌داده**		
۳-۱	فعال‌سازی احراز اصالت برای پایگاه‌داده MongoDB	☐	☐
۳-۲	اطمینان حاصل شود که MongoD از طریق استثنای localhost احراز اصالت را کنار نمی گذارد	☐	☐
۳-۳	اطمینان از فعال بودن احراز اصالت در خوشه منشعب شده	☐	☐
۳-۴	اطمینان از استفاده از مکانیزم احراز اصالت استاندارد صنعتی	☐	☐

۴ کنترل دسترسی و مجازشماری

طبق اصل اعطای حداقل مجوزها به عنوان یک اصل کلی امنیتی، هر کاربر باید تنها مجوزهایی را دارا باشد که واقعا برای اجرای مسوولیت‌های روزمره‌اش به آن مجوزها نیاز دارد. در این فصل، پیشنهاداتی به منظور محدودسازی دسترسی به پایگاه‌داده MongoDB مورد بحث و بررسی قرار می‌گیرد.

۴ – ۱ استفاده از کنترل دسترسی مبتنی بر نقش

کنترل دسترسی مبتنی بر نقش[8]یک روش تنظیم دسترسی به منابع بر مبنای نقش‌های کاربران در یک سازمان است. به یک کاربر می‌توان یک یا تعداد بیشتری نقش اعطا کرد که دسترسی کاربر به منابع پایگاه‌داده را مشخص می‌کنند. MongoDB می‌تواند از کنترل دسترسی مبتنی بر نقش برای مدیریت دسترسی‌ها استفاده کند.

تهدید/توجیه امنیتی:

در صورتی که کنترل دسترسی مبتنی بر نقش به درستی پیاده‌سازی شده باشد، این امکان برای کاربران فراهم می‌شود که مجموعه اعمال مجازی را انجام دهند و دسترسی کارمندان سازمان به جداول پایگاه‌داده به درستی کنترل شود.

اطلاع از وضعیت فعلی:

با دستور مناسبی همچون دستور زیر به MongoDB متصل شوید.

```
mongo --port 27017 -u <siteUserAdmin> -p <password> --
authenticationDatabase
<database name>
```

با استفاده از دستور زیر نقش‌ها و مجوزهای کاربران به دست می‌آید.

```
> db.getUser()
> db.getRole()
```

باید بررسی شود که نقش یا نقش‌های مناسبی برای هر یک از کاربران تنظیم شده باشد.

با استفاده از دستورات زیر می توان لیست تمامی کاربران و نقش‌های موجود در پایگاه‌داده‌ی جاری را مشاهده کرد.

---

[8] Role-based Access Control (RBAC)

```
db.getUsers()
db.getRoles()
```

برای مشاهده‌ی لیست تمامی کاربران و نقش‌ها در تمامی پایگاه‌های داده از دستورات زیر استفاده می‌شود:

```
use admin
db.system.roles.find()
db.system.users.find()
```

مقاوم‌سازی:

مراحل زیر برای پیاده‌سازی مناسب کنترل دسترسی مبتنی بر نقش باید طی شوند.

۱. ایجاد نقش‌ها
۲. تخصیص مجوزهای مناسب به هر نقش
۳. تخصیص کاربران مناسب به هر نقش
۴. حذف هر یک از مجوزهای منفرد تخصیص داده شده به کاربران که به نقش‌ها تخصیص داده شده‌اند

برای اطلاعات بیشتر می‌توانید به پیوست ۱ مراجعه نمایید.

۲-۴ اطمینان از اینکه MongoDB تنها بر روی اتصالات مجاز شبکه شنود می‌کند

باید این اطمینان حاصل شود که MongoDB در محیط شبکه قابل اعتماد اجرا می‌شود و اتصال‌های شبکه که نمونه‌های MongoDB بر روی آن‌ها به درخواست‌های اتصال شنود می‌کنند، محدود باشند. هر نوع اتصال شبکه غیرقابل اعتماد باید توسط MongoDB رد شوند.

تهدید/توجیه امنیتی:

با تنظیم صحیح این پیکربندی، درخواست اتصال از سوی شبکه‌های غیرقابل اعتماد بلاک شده و تنها سیستم‌های موجود در شبکه‌های مجاز و مورد اعتماد می‌توانند به MongoDB متصل شوند. در صورتی که پیکربندی به درستی انجام نشده باشد، امکان برقراری ارتباط‌های غیرمجاز از شبکه‌های غیرقابل اعتماد به MongoDB وجود دارد.

اطلاع از وضعیت فعلی:

ابتدا باید تنظیمات موجود در فایل پیکربندی MongoDB مورد بررسی قرار بگیرند.

```
cat /etc/mongod.conf |grep -A12 "net" | grep "bindIp"
```

همچنین تنظیمات شبکه‌ای مرتبط بر روی سیستم عامل لینوکس نیز باید بررسی شوند.

```
iptables -L
```

مقاوم‌سازی:

باید تنظیمات مربوط به پارامتر bindIp موجود در فایل پیکربندی MongoDB به گونه‌ای انجام شود که نمونه‌های MongoDB تنها بر روی شبکه‌های مورد اعتماد، قابل دسترس باشند. به عنوان نمونه، اگر خط‌مشی سازمان این است که پایگاه‌داده MongoDB تنها از طریق شبکه‌های محلی قابل دسترس باشد، باید آدرس IP هایی از سرور MongoDB در پارامتر bindIp تنظیم شوند که در شبکه‌های محلی سازمان قرار دارند. بدین ترتیب می‌توان تنظیمات را به صورت زیر اعمال کرد. هر تعداد آدرس IP را می‌توان با کاما از هم جدا کرد.

```
net:
        bindIp: <authorized_interface_ip>,<authorized_interface_ip>
```

۳-۴ استفاده از حساب سرویس اختصاصی و غیرممتاز

سرویس MongoDB نباید با استفاده از حساب ممتاز[9] همچون root اجرا شود زیرا سیستم عامل را در معرض ریسک قرار می‌دهد.

تهدید/توجیه امنیتی:

استفاده از حساب سرویس اختصاصی و غیر ممتاز، دسترسی پایگاه‌داده به نواحی حساس سیستم عامل را محدود می‌کند.

اطلاع از وضعیت فعلی:

دستور زیر لیستی از نمونه‌های mongo، شماره PID و مالک PID را نمایش می‌دهد.

```
ps -ef | grep -E "mongos|mongod"
```

مقاوم‌سازی:

مراحل زیر برای استفاده از حساب سرویس اختصاصی و غیرممتاز به منظور اجرای فعالیت‌های پایگاه‌داده MongoDB باید طی شوند:

۱. ایجاد کاربر اختصاصی برای انجام فعالیت‌های پایگاه‌داده مطابق با دستور زیر:

[9] Privileged account

```
useradd -m -d /home/mongodb -s /bin/bash -g mongodb -u 1234 mongodb
```

۲. فایل‌های داده پایگاه‌داده، فایل کلید و فایل‌های کلید خصوصی SSL تنها توسط کاربر mongod/mongos قابل خواندن باشند.

```
chmod 760 <dbpath>
chown mongodb:mongodb <dbpath>

chmod 600 <keyFile>
chown mongodb:mongodb <keyFile>
```

۳. فایل‌های رویدادنگاری تنها توسط کاربر mongod/mongos قابل نوشتن باشند.

```
chmod 644 <logpath>
chown mongodb:mongodb <logpath>
```

۴ – ۴ بررسی نقش‌ها و مجوزهای آن‌ها

بررسی دوره‌ای نقش‌ها، جلوگیری از وجود نقش‌های غیرضروری و همچنین مجوزهای غیرضروری بر روی نقش‌ها به کاهش مجوزهای هر کاربر کمک خواهد کرد.

تهدید/توجیه امنیتی:

اگرچه کنترل دسترسی مبتنی بر نقش، دسترسی به منابع را کنترل و تنظیم می‌کند، در طول زمان برخی از نقش‌ها و برخی از مجوزهای اعطا شده به نقش‌ها دیگر نیاز نیستند و باید حذف شوند.

اطلاع از وضعیت فعلی:

با استفاده از دستور زیر می‌توان تمامی نقش‌های پایگاه‌داده، شامل نقش‌های تعریف شده توسط کاربر و نقش‌های از پیش تعریف شده و مجوزهای اعطا شده به آن‌ها را مشاهده کرد. باید این اطمینان حاصل شود که نقش‌ها و مجوزهای نمایش داده شده، ضروری هستند.

```
db.runCommand(
{
rolesInfo: 1,
showPrivileges: true,
```

```
showBuiltinRoles: true
}
)
```

مقاوم‌سازی:

به منظور لغو مجوزهای اعطا شده به نقش‌های تعریف شده توسط کاربر می‌توان از دستور زیر استفاده کرد.

```
{
revokePrivilegesFromRole: "<role>",
privileges:
[
{ resource: { <resource> }, actions: [ "<action>", ... ]},
  ...
],
}
```

۵ – ۴ بررسی نقش‌های تعریف شده توسط کاربر

بررسی دوره‌ای تمامی نقش‌ها و حذف تمامی کاربران از نقش‌هایی که نباید بیش از این عضوی از نقش باشند، مجوزهای اعطا شده به هر کاربر را کاهش می‌دهد.

تهدید/توجیه امنیتی:

اگرچه کنترل دسترسی مبتنی بر نقش، دسترسی به منابع را کنترل و تنظیم می‌کند، در طول زمان برخی از کاربران به دلایل مختلف همچون تغییر شغل در سازمان، دیگر نباید یک نقش را دارا باشند. کاربرانی که مجوزهای بیش از نیاز داشته باشند، می‌توانند سازمان را در معرض ریسک قرار دهند.

اطلاع از وضعیت فعلی:

نقش‌ها می‌بایست در هر یک از پایگاه‌های داده مورد بررسی قرار گیرند. برای بررسی یک نقش در پایگاه‌داده جاری می‌توان از دستور زیر استفاده کرد.

```
db.runCommand( { rolesInfo: "<rolename>" } )
```

برای بررسی نقش در پایگاه‌داده دیگری می‌توان از دستور زیر استفاده کرد.

```
db.runCommand( { rolesInfo: { role: "<rolename>", db: "<database>" }
} )
```

مقاوم‌سازی:

به منظور حذف یک کاربر از یک یا چندین نقش در پایگاه‌داده جاری دستور زیر اجرا شود.

```
use <dbName>
db.revokeRolesFromUser( "<username>", [ <roles> ])
```

۶ – ۴ بررسی کاربر فوق العاده و نقش‌های مدیریتی

نقش‌ها، مدیریت مجوزها در سیستم پایگاه‌داده را آسان می‌سازند. مدیران امنیتی می‌توانند دسترسی به پایگاه‌های داده را با ایجاد نقش‌هایی در پایگاه‌داده منطبق با مشاغل موجود در سازمان کنترل کنند. به جای اعطای مجوزها به هر یک از کاربران یک شغل کاری، می‌توان مجموعه‌ای از مجوزها را به نقش منطبق با هر شغل کاری در سازمان تخصیص داد. پس از آن می‌توان نقش‌ها را به کاربران مطابق با شغل کاریشان تخصیص داد.

تهدید/توجیه امنیتی:

بررسی کاربران فوق العاده[0] و نقش‌های مدیریتی در پایگاه‌داده، احتمال دسترسی‌ها و مجوزهای ناخواسته را کاهش می‌دهد.

اطلاع از وضعیت فعلی:

نقش‌های کاربر فوق العاده[1] این امکان را فراهم می‌کنند که به هر کاربری، هر مجوزی بر روی هر پایگاه‌داده‌ای اعطا شود. این بدان معنا است که یک کاربر با نقش کاربر فوق العاده می‌تواند هر نوع مجوزی بر روی هر پایگاه‌داده‌ای را به خود تخصیص دهد.

```
db.runCommand( { rolesInfo: "dbOwner" } )
db.runCommand( { rolesInfo: "userAdmin" } )
db.runCommand( { rolesInfo: "userAdminAnyDatabase" } )
```

نقش ریشه[2] دسترسی به عملیات و منابع نقش‌های readWriteAnyDatabase، dbAdminAnyDatabase، userAdminAnyDatabase و clusterAdmin را فراهم می‌کند.

```
db.runCommand( { rolesInfo: "readWriteAnyDatabase" } )
db.runCommand( { rolesInfo: "dbAdminAnyDatabase" } )
db.runCommand( { rolesInfo: "userAdminAnyDatabase" } )
```

---

[1] Superuser
[1] Superuser Roles
[1] Root role

[0]
[1]
[2]

```
db.runCommand( { rolesInfo: "clusterAdmin" } )
```

نقش‌های مدیریت خوشه[3]برای مدیریت کل سیستم به جای مدیریت یک پایگاه‌داده منفرد استفاده می‌شوند.

```
db.runCommand( { rolesInfo: "hostManager" } )
```

مقاوم‌سازی:

به منظور حذف یک کاربر از یک یا چندین نقش در پایگاه‌داده جاری از دستور زیر استفاده می‌شود.

```
use <dbName>
db.revokeRolesFromUser( "<username>", [ <roles> ])
```

۴-۷ جمع‌بندی

در این فصل مجوزهایی که ممکن است به کاربر یا نقش اعطا شود در حالی که نیازی به اعطای آن‌ها نیست مورد بحث و بررسی قرار گرفت. مدیر سامانه به منظور بررسی پارامترهای مقاوم‌سازی و تهیه گزارش در این زمینه می‌تواند از چک لیست زیر استفاده نماید.

	عنوان	تنظیم صحیح	
		بله	خیر
۴	**کنترل دسترسی و مجازشماری**		
۴-۱	استفاده از کنترل دسترسی مبتنی بر نقش	☐	☐
۴-۲	اطمینان از اینکه MongoDB تنها بر روی اتصالات مجاز شبکه شنود می‌کند	☐	☐
۴-۳	استفاده از حساب سرویس اختصاصی و غیرممتاز	☐	☐
۴-۴	بررسی نقش‌ها و مجوزهای آن ها	☐	☐
۴-۵	بررسی نقش‌های تعریف شده توسط کاربر	☐	☐

[1] Cluster Administration Roles [3]

۴-۶	بررسی کاربر فوق العاده و نقش‌های مدیریتی	☐	☐

۵ تنظیمات رویدادنگاری/ممیزی

ثبت وقایع سیستم و بازبینی آن‌ها در صورت رخداد مشکلات فنی یا امنیتی یکی از نیازمندی‌های اصلی در پایگاه‌داده است. با توجه به این‌که انواع وقایعی که در سیستم رخ می‌دهند از درجه اهمیت متفاوتی برخوردارند و ثبت کلیه وقایع (بدون توجه به ارزش هر یک) می‌تواند منجر به کاهش کارایی سرور یا هدر رفتن فضای دیسک گردد، بنابراین لازم است یک زیرمجموعه از وقایع مهم شناسایی و رویدادنگاری در مورد آن وقایع همواره انجام شود. در مورد رویدادنگاری سایر وقایع، بسته به توانمندی‌های پردازشی و ذخیره‌سازی سرور، می‌توان تصمیم‌گیری کرد. در این فصل به تشریح برخی از مهم‌ترین پارامترهای امنیتی مربوط به پیکربندی تنظیمات رویدادنگاری می‌پردازیم.

۵ – ۱ ممیزی فعالیت‌های سیستم

دسترسی و تغییر پیکربندی پایگاه‌داده و داده‌ها باید همواره ردیابی شود. MongoDB Enterprise شامل امکان ممیزی برای ثبت رویدادهای سیستمی بر روی نمونه MongoDB است.

تهدید/توجیه امنیتی:

رویدادهای ثبت شده در سطح سیستم برای عیب‌یابی مشکلات عملیاتی و رسیدگی به حوادث امنیتی به کار برده می‌شوند.

اطلاع از وضعیت فعلی:

به منظور بررسی ثبت فعالیت‌های سیستمی در MongoDB دستور زیر برای بررسی مقدار auditLog.destination اجرا می‌شود.

```
cat /etc/mongod.conf |grep -A4 "auditLog" | grep "destination"
```

مقاوم‌سازی:

برای پارامتر auditLog.destination مطابق با گزینه‌های زیر باید مقدار مناسبی تنظیم شود.

- Syslog: به منظور فعال‌سازی ممیزی و ثبت رویدادهای ممیزی در syslog دستور زیر اجرا می‌شود:

```
mongod --dbpath data/db --auditDestination syslog
```

- کنسول: با اجرای دستور زیر ممیزی فعال شده و رویدادها در خروجی استاندارد چاپ می‌شوند:

```
mongod --dbpath data/db --auditDestination console
```

- فایل JSON: به منظور فعال‌سازی ممیزی و ثبت رویدادها در فایلی با فرمت JSON دستور زیر اجرا می‌شود. ثبت رویدادها در فایلی با فرمت JSON کارایی سرور را نسبت به حالتی که رویدادها در فایلی با فرمت BSON ثبت شوند، کاهش می‌دهد.

```
mongod --dbpath data/db --auditDestination file --auditFormat JSON --
auditPath data/db/auditLog.json
```

- فایل BSON: با اجرای دستور زیر ممیزی فعال شده و رویدادها در فایلی با فرمت دودویی BSON ذخیره می‌شوند.

```
mongod --dbpath data/db --auditDestination file --auditFormat BSON --
auditPath data/db/auditLog.bson
```

۲ – ۵ اطمینان از پیکربندی مناسب برای ممیزی

پایگاه‌داده MongoDB Enterprise، ممیزی از انواع عملیات را پشتیبانی می‌کند. هنگامی که ممیزی فعال می‌شود، به صورت پیش‌فرض تمامی اعمال قابل ممیزی، ثبت می‌شوند. به منظور تعیین اینکه کدامیک از رویدادها ثبت شوند، می‌توان از گزینه auditFilter– استفاده کرد. این قابلیت تنها در نسخه‌های Enterprise موجود است.

تهدید/توجیه امنیتی:

در صورتی که تمامی انواع عملیات ثبت شوند، ردیابی هر نوع حادثه‌ای امکان‌پذیر خواهد بود. فیلتر ممیزی باید به طور مناسب و مطابق با سیاست‌های سازمان انجام شود.

اطلاع از وضعیت فعلی:

به منظور بررسی فیلترهای ممیزی تنظیم شده بر روی پایگاه‌داده MongoDB و تطابق آن‌ها با نیازمندی‌های سازمان، دستور زیر اجرا می‌شود.

```
cat /etc/mongod.conf |grep -A10 "auditLog" | grep "filter"
```

مقاوم‌سازی:

فیلترهای ممیزی باید مطابق با نیازمندی‌های سازمان اعمال شوند. به صورت پیش‌فرض، MongoDB تمامی رویدادها را ثبت و ردگیری می‌کند. با تعیین فیلترهای ممیزی در فایل پیکربندی می‌توان تنها رویدادهای مورد نظر را ثبت کرد. فیلترها در واقع پرس‌وجوهایی به صورت JSON بوده و در تعریف آن‌ها می‌توان از

انتخابگرهای جستجوهای استاندارد[4]همچون $eq، $in و $ne استفاده کرد. به عنوان نمونه، اگر بخواهیم تنها رویدادهای مربوط به یک کاربر مشخص ثبت شوند، تنظیم زیر را در فایل پیکربندی وارد می‌کنیم:

```
auditLog:
        filter: '{ "users.user": "<username>" }'
```

به عنوان مثالی دیگر، اگر قصد ثبت انواع رویدادهای خاصی همچون dropCollection و dropDatabase را داشته باشیم، فیلتر ممیزی به صورت زیر تعریف می‌شود:

```
auditLog:
  filter: '{ atype: { $in: [ "dropCollection", "dropDatabase" ] } }'
```

نیاز به تعریف فیلترهای ممیزی، کاملاً وابسته به نیازمندی‌ها و خط‌مشی سازمان‌ها است.

۵-۳ جمع‌بندی

در این فصل به تشریح برخی از مهم‌ترین تنظیمات مربوط به پیکربندی رویدادنگاری پرداختیم. مدیر سامانه به منظور بررسی پارامترهای مقاوم‌سازی و تهیه گزارش در این زمینه می‌تواند از چک لیست زیر استفاده نماید.

	عنوان	تنظیم صحیح	
		بله	خیر
۵	**تنظیمات رویدادنگاری**		
۵-۱	ممیزی فعالیت‌های سیستم	☐	☐
۵-۲	اطمینان از پیکربندی مناسب فیلترهای ممیزی	☐	☐

[1] Standard Query Selector [4]

۶ تنظیمات رمزنگاری

این فصل شامل پیشنهاداتی برای امن‌سازی داده‌های ذخیره شده و داده‌های در حال انتقال در پایگاه‌داده MongoDB است.

۶ – ۱ حفاظت از ارتباطات شبکه‌ای با SSL یا TLS

برای حفاظت از تمامی ارتباطات ورودی و خروجی از SSL یا TLS استفاده شود. این مورد شامل استفاده از TLS یا SSL به منظور رمزگذاری ارتباطات بین مؤلفه‌های mongod و mongos و همچنین تمامی ارتباطات میان برنامه‌های کاربردی و MongoDB است.

تهدید/توجیه امنیتی:

رمزگذاری ارتباطات با کمک TLS یا SSL از شنود ترافیک آشکار بین مؤلفه‌های MongoDB یا انجام حمله شخص میانی[۵]جلوگیری می‌کند.

اطلاع از وضعیت فعلی:

برای تأیید اینکه سرور از SSL یا TLS استفاده می‌کند، دستور زیر اجرا شود:

```
cat /etc/mongos.conf | grep -A20 'net' | grep -A10 'ssl' | grep
'mode'
```

مقدار پارامتر net.ssl.mode باید برابر requireSSL باشد.

مقاوم‌سازی:

به منظور رمزگذاری تمامی ارتباطات شبکه‌ای MongoDB با استفاده از SSL یا TLS گام‌های زیر باید طی شوند. گام‌های زیر برای تنظیم mongod به کار می‌روند:

۱. در فایل پیکربندی /etc/mongod.conf پارامتر PEMKeyFile با مقداری برابر مسیر فایل گواهینامه تنظیم شود.

```
ssl:
mode: requireSSL
```

[1] Man-in-the-middle Attack [5]

```
PEMKeyFile: /etc/ssl/mongodb.pem
CAFile: /etc/ssl/ca.pem
```

۲. نمونه mongod با یکی از دستورات زیر مجدداً اجرا شود.

```
mongod --config /etc/mongod.conf
mongod --sslMode requireSSL --sslPEMKeyFile /etc/ssl/mongodb.pem --sslCAFile
/etc/ssl/ca.pem
```

۲ – ۶ اطمینان از فعال‌سازی FIPS

استاندارد پردازش اطلاعات فدرال[۶](FIPS) ، یک استاندارد امنیتی رایانه‌ای است که برای تأیید ماژول‌های نرم‌افزاری و کتابخانه‌هایی است که رمزگذاری و رمزگشایی داده‌ها را به صورت امن انجام می‌دهند. می‌توان MongoDB را به صورتی پیکربندی کرد که با کتابخانه‌ی OpenSSL گواهی شده با FIPS 140-2 اجرا شود.

تهدید/توجیه امنیتی:

FIPS یک استاندارد صنعتی است که مشخص می‌کند داده‌ها هنگام ذخیره‌سازی و در حین انتقال چگونه باید رمزگذاری شوند.

اطلاع از وضعیت فعلی:

برای تأیید اینکه سرور از حالت FIPS استفاده می‌کند (مقدار net.ssl.FIPSMode باید true باشد)، دستور زیر می‌بایست اجرا شود.

```
mongos --config /etc/mongos.conf
net:
ssl:
FIPSMode: true
```

همچنین می‌توان فایل رویدادنگاری سرور را به منظور یافتن پیامی مبنی بر فعال بودن FIPS بررسی کرد.

```
FIPS 140-2 mode activated
```

مقاوم‌سازی:

---

[1] Federal Information Processing Standard

با به کارگیری حالت FIPS می‌توان مطمئن شد که گواهینامه استفاده شده منطبق با استاندارد FIPS است. نمونه‌های mongod و mongod باید در حالت FIPS اجرا شوند. بدین منظور در فایل پیکربندی تغییرات زیر اعمال می‌شود تا نمونه‌های mongod و mongos در حالت FIPS اجرا شوند. بنابراین نمونه را متوقف کرده و در فایل پیکربندی تغییرات زیر را اعمال کنید.

```
net:
ssl:
FIPSMode: true
```

حال نمونه mongod و mongos را با فایل پیکربندی راه‌اندازی نمایید.

```
mongod --config /etc/mongod.conf
```

۶-۳ جمع‌بندی

در این فصل به تشریح برخی از مهم‌ترین پارامترهای امنیتی مربوط به پیکربندی تنظیمات رمزنگاری پرداختیم. مدیر سامانه به منظور بررسی پارامترهای مقاوم‌سازی و تهیه گزارش در این زمینه می‌تواند از چک لیست زیر استفاده نماید.

	عنوان	تنظیم صحیح	
		بله	خیر
۶	**تنظیمات رویدادنگاری**		
۶-۱	حفاظت از ارتباطات شبکه‌ای با SSL یا TLS	☐	☐
۶-۲	اطمینان از فعال‌سازی FIPS	☐	☐

۷ راهنمای اعمال مقاوم‌سازی

این پروژه دارای سه فایل اجرایی است که در ادامه به بررسی هر یک می‌پردازیم.

۷-۱ فایل start.sh

این فایل، تنها فایلی است که کاربر باید آن را اجرا کند. برای آنکه فرآیند آزمودن پایگاه‌داده و امن‌سازی آن صورت گیرد، ابتدا لازم است از وجود MongoDB روی سیستم اطمینان حاصل گردد. پس از آن، کاربر می‌تواند نام کاربری و رمز عبور مورد نظر خود را برای اتصال به پایگاه‌داده وارد کند. همچنین، در صورتی که تنظیمات موجود در فایل پیکربندی MongoDB به صورتی باشد که سرور تنها اتصالات SSL را قبول کند، از کاربر مسیر فایل CA و مسیر فایل گواهینامه‌ی کارخواه درخواست می‌شود. با توجه به آنکه نام میزبانی که سرور Mongo بر روی آن قرار دارد، در اتصالات SSL و انطباق این نام با نام وارد شده در گواهینامه سرور اهمیت دارد، نام میزبان نیز از کاربر دریافت می‌شود که مقدار پیش‌فرض آن localhost است. توجه به این نکته حائز اهمیت است که در اسکریپت‌ها فرض بر آن است که اجرای سرور MongoDB از طریق سرویس MongoDB انجام می‌شود و تمامی تنظیمات مربوطه در فایل پیکربندی /etc/mongod.conf قرار دارند. پس از اجرای start.sh و دریافت اطلاعات مورد نظر از کاربر، اسکریپت script.sh اجرا می‌شود. در اثر اجرای این فایل، دو پوشه حاوی نتایج آزمایش و نتایج مورد انتظار ایجاد می‌شوند. در این اسکریپت قصد داریم این دو پوشه را با هم مقایسه کنیم تا مغایرت‌های سیستم با موارد امنیتی مورد انتظار مشخص شوند. نتایج این آزمایش در فایل first_test_result ثبت می‌شود. نمونه‌ای از خروجی این برنامه در شکل (۱) نشان داده شده است.

"MongoDB RESULT"	"EXPECTED RESULT"
<------1-1-Mongodb Database Running with Least Privileges --- mongodb	<------1-1-Mongodb Database Running with Least Privileges --- not root user
<------1-2-key file permissions ------> NONE	<------1-2-key file permissions ------> NONE
<------1-3-database file permissions ------> 760 mongodb mongodb	<------1-3-database file permissions ------> 760 mongodb mongodb
<------2-1-Uses a non-default port ------> 27017	<------2-1-Uses a non-default port ------> port != 27017
<------2-2-MongoDB software version/patches ------> db version v3.4.20	<------2-2-MongoDB software version/patches ------> db version v3.4.20
<------3-1-Authentication for MongoDB databases ------> disabled	<------3-1-Authentication for MongoDB databases ------> enabled
<------3-2-Authentication via the localhost exception ------> true	<------3-2-Authentication via the localhost exception ------> false

شکل ۱: محتوای فایل first_test_result

ستون سمت چپ نشان دهنده تنظیمات فعلی و ستون سمت راست نشان دهنده تنظیمات مورد انتظار است. پس از اجرای این اسکریپت، در صورت تمایل کاربر، اسکریپت repair اجرا می‌شود. سپس هر مورد امنیتی که در آن نتیجه مورد انتظار و نتیجه حاصل از آزمودن پارامترهای امنیتی مغایر باشند، با موافقت کاربر امن‌سازی شده و در آخر نیز بررسی دوباره‌ای روی سیستم انجام می‌شود. نتیجه تست دوم در فایل second_test_result ذخیره خواهد شد.

"MongoDB RESULT"	"EXPECTED RESULT"
<------1-1-Mongodb Database Running with Least Privileges --- mongodb	<------1-1-Mongodb Database Running with Least Privileges --- not root user
<------1-2-key file permissions ------> NONE	<------1-2-key file permissions ------> NONE
<------1-3-database file permissions ------> 760 mongodb mongodb	<------1-3-database file permissions ------> 760 mongodb mongodb
<------2-1-Uses a non-default port ------> 2727	<------2-1-Uses a non-default port ------> port != 27017
<------2-2-MongoDB software version/patches ------> db version v3.4.20	<------2-2-MongoDB software version/patches ------> db version v3.4.20
<------3-1-Authentication for MongoDB databases ------> enabled	<------3-1-Authentication for MongoDB databases ------> enabled
<------3-2-Authentication via the localhost exception ------> false	<------3-2-Authentication via the localhost exception ------> false

شکل ۲: محتوای فایل second_test_result

۷-۲ فایل script.sh

در این فایل دو متغیر با نام‌های result_path و expected_path تعریف شده است. این دو متغیر به پوشه‌هایی اشاره می‌کنند که در آن‌ها به ترتیب نتایج هر آزمایش و نتیجه مورد انتظار آن آزمایش قرار می‌گیرند. در اینجا لازم است بنا به نیازمندی سیستم و نکات گفته شده حین توضیح هر مورد، نتایج مورد انتظار برای هر مورد امنیتی تنظیم شود. این کد توسط برنامه start.sh اجرا می‌شود.

۷-۳ فایل repair.sh

فایل آخر مربوط به تغییر تنظیمات سیستم می‌باشد. در صورتی که تنظیمات به درستی انجام شده باشد، انتظار می‌رود که مورد امنیتی در ستون دوم وجود نداشته باشد و تنها چند پیشنهاد برای امنیت بیشتر در آن باقی بماند. در شکل (۲) می‌توان نمونه‌ای از فایل second_test-result را پس از اعمال تغییرات مشاهده کرد.

۸ جمع‌بندی

در این مستند به بررسی موارد امنیتی مربوط به مقاوم‌سازی پایگاه‌داده‌ی MongoDB پرداخته شد. تنظیمات مربوط به مقاوم‌سازی MongoDB در شش بخش مختلف دسته‌بندی شدند. در بخش اول، امن‌سازی محیط اجرا، بخش دوم نصب و پیکربندی امن پایگاه‌داده، بخش سوم امن‌سازی اتصال به پایگاه‌داده، بخش چهارم تنظیمات کنترل دسترسی و مجازشماری، بخش پنجم تنظیمات رویدادنگاری و بخش ششم تنظیمات رمزنگاری بررسی شدند. در مورد هر پارامتر، کاربرد، ارزش امنیتی و نحوه آگاهی از مقدار کنونی آن پارامتر و چگونگی مقداردهی امن آن توضیحاتی داده شد. در پایان نیز نحوه اجرای اسکریپت‌ها و خروجی‌های سیستم بیان شدند. خلاصه‌ای از گزارش ارائه شده، به صورت یک چک لیست در ادامه آورده شده است.

	عنوان	تنظیم صحیح	
		بله	خیر
۱	**ایمن‌سازی محیط اجرا**		
۱-۱	راه‌اندازی پایگاه‌داده MongoDB با حداقل مجوزها	☐	☐
۱-۲	مجوزهای فایل کلید	☐	☐
۱-۳	مجوزهای فایل پایگاه‌داده	☐	☐
۲	**پیکربندی امن پایگاه‌داده**		
۲-۱	عدم استفاده از پورت پیش‌فرض	☐	☐
۲-۲	به‌روزرسانی نرم‌افزار MongoDB و نصب وصله ها	☐	☐
۳	**امن‌سازی اتصال به پایگاه‌داده**		
۳-۱	فعال‌سازی احراز اصالت برای پایگاه‌داده MongoDB	☐	☐
۳-۲	اطمینان حاصل شود که ورود به صورت localhost، احراز اصالت را کنار نمی‌گذارد	☐	☐
۳-۳	اطمینان از فعال بودن احراز اصالت در خوشه منشعب شده	☐	☐
۳-۴	اطمینان از استفاده از مکانیزم احراز اصالت استاندارد صنعتی	☐	☐
۴	**کنترل دسترسی و مجازشماری**		
۴-۱	استفاده از کنترل دسترسی مبتنی بر نقش	☐	☐

۴-۲	اطمینان از اینکه MongoDB تنها بر روی اتصالات مجاز شبکه شنود می‌کند	☐	☐
۴-۳	استفاده از حساب سرویس اختصاصی و غیرممتاز	☐	☐
۴-۴	بررسی نقش‌ها و مجوزهای آن ها	☐	☐
۴-۵	بررسی نقش‌های تعریف شده توسط کاربر	☐	☐
۴-۶	بررسی کاربر فوق العاده و نقش‌های مدیریتی	☐	☐
۵	**تنظیمات رویدادنگاری**		
۵-۱	ممیزی فعالیت‌های سیستم	☐	☐
۵-۲	اطمینان از پیکربندی مناسب فیلترهای ممیزی	☐	☐
۶	**تنظیمات رمزنگاری**		
۶-۱	حفاظت از ارتباطات شبکه‌ای با SSL یا TLS	☐	☐
۶-۲	اطمینان از فعال‌سازی FIPS	☐	☐

۹ مراجع

[1] https://www.cisecurity.org/

پیوست ۱

پایگاه‌داده MongoDB نقش‌های از پیش تعریف شده‌ای دارد که امکان کنترل دسترسی بـر روی سیسـتم را برای مدیران فراهم می‌کند. در صورتی که نقش‌های از پیش تعریف شده، مجموعه‌ای از مجوزهای مورد نظـر مدیر پایگاه داده را نداشته باشند می‌توان نقش‌های جدیدی تعریف کرد. تابع db.createRole() برای تعریف نقش جدید در MongoDB به کار بـرده می‌شـود. در تعریـف نقـش جدیـد، مجموعـه‌ای از مجوزهـا و سـایر نقش‌هایی که این نقش مجوزهای آن‌ها را به ارث می برد، تعیین می‌شوند. جزئیات تابع db.createRole() در ادامه و در جدول ۱ توضیح داده شده‌اند.

```
db.createRole({
        role: "<name>",
       privileges: [
       { resource: { db: "<db-name>", collection: "<collection-name>" }
      , actions: [ "<action>", ... ] },
          ...
       ],
      roles: [
       { role: "<role>", db: "<database>" } | "<role>",
         ...
      ]
})
```

جدول ۱ - فیلدهای تعریف نقش جدید

فیلد	توضیحات
role	نام نقش جدید
privileges	آرایه‌ای از مجوزهای تخصیص داده شده به نقـش جدیـد. یـک مجـوز شـامل resource و actions است. به عنوان مثال می‌توان مجوز update بر روی کالکشن یک پایگاه داده را به نقش تعریف شده، اعطا کرد.
roles	آرایه‌ای از نقش‌هایی که نقش جدید از آن‌ها ارث‌بری می‌کند.

پس از تعریف نقش جدید، می‌توان نقش تعریف شده را به یک کاربر با استفاده از تابع db.grantRolesToUser() اعطا کرد. در تابع زیر می‌توان نقش یا نقش‌های مورد نظر را بر روی یک پایگاه‌داده به کاربر اعطا کرد.

```
Db.grantRolesToUser(
   “<user>”,
   [
     { role: “<role>”, db: “<db-name>” }
   ]
)
```

همچنین برای حذف یک نقش بر روی یک پایگاه داده از یک کاربر می توان از تابع db.revokeRolesFromUser() استفاده کرد.

```
db.revokeRolesFromUser(
   "<user>",
   [
     { role: "<role>", db: "<db-name>" }
   ]
)
```