بسمه تعالی

سازمان فناوری اطلاعات ایران

معاونت امنیت فضای تولید و تبادل اطلاعات

مرکز ماهر

مقاوم‌سازی امنیتی PostgreSQL

فهرست مطالب

پیشگفتار

در این گزارش، مقاوم‌سازی امنیتی PostgreSQL نسخه ۹,۵ بـر روی سیسـتم‌عامل Ubuntu 17.04 64-bit مورد بحث و بررسی قرار می‌گیرد. در این راستا، برای هر یک از پارمترهای امنیتی تاثیرگذار، شرح مختصری ارایه می‌گردد و سپس تهدید/توجیه امنیتی آن مورد بررسـی قـرار می‌گیـرد. در نهایـت نیـز نحـوه اطـلاع از وضعیت فعلی پارامتر و چگونگی مقاوم‌سازی آن تشریح می‌گردد.

بررسی پارامترهای مربوط به مقاوم‌سازی PostgreSQL در پنج بخش مختلـف صـورت می‌گیـرد. بخـش اول، تنظیمات مربوط به امن‌سازی محیط اجرا را در بر می‌گیـرد. در بخـش دوم، پارامترهـای مربـوط بـه نصـب و پیکربندی امن پایگاه داده تشریح می‌گردد. در بخش سوم، تنظیمات مربوط به احـراز اصـالت کـاربران مـورد بحث و بررسی قرار می‌گیرد. در بخش چهارم، تنظیمات مربوط به رویـدادنگاری تشـریح می‌گـردد. در بخـش پنجم نیز برخی از پیکربندی‌های با سطح حساسیت پایین‌تر دسته‌بندی شده است. در پایان گزارش نیز، نحوه اجرای اسکریپت‌ها و اعمال تنظیمات مورد نیاز برای مقاوم سازی پایگاه داده بیان می‌گردد.

۱ امن سازی محیط اجرا

در هر سیستم عامل، ابزارها و روش‌های مختلفی برای کنترل دسترسی وجود دارد. پیکربندی نرم‌افزارهایی چون پایگاه داده نیز باید بسیار دقیق و با رویکرد امنیتی انجام شود. با این حال، برای ایجاد یک مدل جامع برای محافظت از اطلاعات و سرویس‌های اطلاعاتی، باید از سطح سیستم‌عامل تمهیدات اندیشیده شود. چرا که اگر تنظیمات کارپذیر پایگاه داده به خوبی صورت گرفته باشد اما کنترل دسترسی بر روی فایل‌های آن اعمال نشده باشد، یک مهاجم می‌تواند به سادگی از داده‌ها نسخه‌برداری کرده و از این طریق اطلاعات زیادی فاش شود.

در این بخش بر روی کنترل دسترسی فایل‌ها و پردازه‌ها در سیستم‌های مبتنی بر یونیکس که شاخص‌ترین آن‌ها لینوکس است، متمرکز می‌شویم. در این سیستم‌ها مجوزها بر روی سه دسته از کاربران مشخص می‌شود. اولین دسته مربوط به مالک است (u)، دسته دوم گروهی از کاربران که در گروه مالک قرار می‌گیرند (g) و دسته سوم دیگر کاربران (o) را مشخص می‌کند. مجوزهای اصلی نیز شامل خواندن (r)، نوشتن (w) و اجرا (x) می‌شود. لذا مجوزهای مربوط به هر فایل یا دایرکتوری با یک نه‌تایی نشان داده می‌شود. به عنوان مثال اگر مجوزهای فایلی به صورت rw-r--x باشد، بدین معنی است که مالک این فایل حق خواندن و نوشتن، گروه کاربران متعلق به گروه مالک حق خواندن، و دیگر کاربران حق اجرای این فایل را دارند. تنظیم دقیق این پارامترها نقش بسزایی در تأمین امنیت پایگاه دارد.

۱-۱ پیکربندی فایل تنظیمات

تنظیمات اصلی PostgreSQL در فایل postgresql.conf قرار دارد، از این رو حفاظت از این فایل مهم است.

تهدید/توجیه امنیتی:

از آنجایی که در این فایل تنظیمات اصلی PostgreSQL وجود دارد، تنها مالک این فایل یعنی root که مدیر سیستم است، حق تغییر این فایل را دارد. پس باید دقت کرد که اولاً مالک این فایل root باشد، ثانیا مجوز نوشتن تنها به مالک داده‌شده باشد.

اطلاع از وضعیت فعلی:

با استفاده از دستورات زیر می‌توان مسیر تنظیمات اصلی PostgreSQL را به دست می آورد:

```
sudo  su    postgres
psql   -c    'show config_file'
```

خروجی این دستور برای نسخه PostgreSQL انتخاب شده در این سند، به صورت زیر است:

```
/etc/postgresql/10/main/postgresql.conf
```

با استفاده از دستور زیر می‌توان حقوق دسترسی مربوط به فایل تنظیمات اصلی PostgreSQL را مشاهده نمود:

```
ls   -lh     /etc/postgresql/10/main/postgresql.conf
```

مقاوم‌سازی:

دستورات زیر به ترتیب مالکیت، گروه کاربری مالکیت و حقوق دسترسی به فایل را به صورت امن مشخص می‌نمایند:

```
sudo  su
chown            -R    root    /etc/postgresql/10/main/postgresql.conf
chgrp            -R    root    /etc/postgresql/10/main/postgresql.conf
chmod       644          /etc/postgresql/10/main/postgresql.conf
```

۲-۱ پیکربندی دایرکتوری ذخیره داده

در پایگاه داده PostgreSQL، می‌توان مسیر پیش فرض داده‌های اصلی (مانند جداول و غیره) را با استفاده از دستور زیر به دست آورد:

```
sudo  su     postgres
psql   -c    'show data_directory'
```

خروجی دستور فوق برای نسخه انتخابی گزارش به صورت زیر است:

```
data_directory
/var/lib/postgresql/10/main
(1 row)
```

تهدید/توجیه امنیتی:

برخلاف فایل‌های دیگر، مدیر پایگاه داده نباید مالک این فایل باشد. مالکیت فایل باید متعلق به کاربری بدون هرگونه حقوق ممتاز (مثلا کاربری با نام Postgres) باشد؛ زیرا این کاربر اجازه انجام هیچ عملیاتی داخل سیستم عامل لینوکس را ندارد. همچنین علاوه بر مدیر، هیچ‌کس دیگری نیز حق خواندن، تغییر یا اجرای این فایل را نباید داشته باشد. درنتیجه تمامی حقوق روی این فایل را از همه گرفته و مجوزهای مربوطه را تنها به کاربر postgres می‌دهیم.

اطلاع از وضعیت فعلی:

برای اطلاع از وضعیت فعلی این پارامتر می‌توان از دستور زیر بهره گرفت.

```
ls   -lh     /var/lib/postgresql/10/main
```

مجوزهای مربوط به فایل‌های موجود در این دایرکتوری باید به صورت rwxrwx--- postgres postgres باشد.

مقاوم‌سازی:

دستورات زیر به ترتیب مالکیت، گروه کاربری مالکیت و حقوق دسترسی به داده‌های اصلی (مانند جدوال و غیره) را به صورت امن تعیین می‌نماید.

```
sudo   su
chown       -R     postgres    /var/lib/postgresql/10/main
chgrp       -R     postgres    /var/lib/postgresql/10/main
chmod       770                /var/lib/postgresql/10/main
```

۳-۱ پیکربندی اجرای دستور از طریق خط فرمان توسط کاربر postgres

در این بخش به نحوه پیکربندی امکان/عدم امکان اجرای دستورات از طریق خط فرمان برای کاربر postgres پرداخته می‌شود.

تهدید/توجیه امنیتی:

پردازه postgres نباید حق اجرای دستورات shell را داشته باشد. در این صورت حتی اگر مهاجم بتواند به این به پردازه نفوذ کند، باز هم قادر نخواهد بود به سطر فرمان کارپذیر دسترسی پیدا کند. برای این کار باید در فایل /etc/passwd/ آخرین ستون مربوط به کاربر postgres را از /bin/bash به /bin/false تغییر دهیم.

اطلاع از وضعیت فعلی:

با اجرای دستور زیر می‌توان از وضعیت فعلی کاربر postgres برای اجرای دستور از طریق shell مطلع گردید.

```
grep   postgres    /etc/passwd
```

خروجی دستور فوق به صورت زیر می‌باشد:

```
postgres:x:125:134:PostgreSQL
    administrator,,,:/var/lib/postgresql:/bin/bash
```

مقاوم‌سازی:

با اجرای دستور زیر، مقدار مورد نظر را جایگزین مقدار فعلی می‌کنیم:

```
sed   -i    's/postgres\(.*\):\/bin.*/postgres\1:\/bin\/false/'
      /etc/passwd
```

خروجی حاصل از اجرای دستور فوق برای فایل passwd به صورت زیر می‌باشد:

```
postgres:x:125:134:PostgreSQL
    administrator,,,:/var/lib/postgresql:/bin/false
```

۴-۱ اجرای سرویس به عنوان یک کاربر عادی

در سیستم عامل لینوکس (Ubuntu)، امکان اجرای سرویس PostgreSQL توسط کاربر root همانند دیگر سرویس‌ها، با استفاده از دستورهای زیر وجود دارد:

```
systemctl start postgresql
service postgresql start
```

در این بخش به ضرورت اجرای سرویس postgres از طریق یک کاربر غیر root می‌پردازیم.

تهدید/توجیه امنیتی:

اگر سرویس postgres توسط یک کاربر عادی (غیر root) اجرا نشده باشد، این تهدید وجود دارد که یک کاربر سرویس postgres با مجوز FILE بتواند فایلهایی را با عنوان root ایجاد نماید (به عنوان مثال می‌تواند محتوای فایل root/.bashrc را تغییر داده و روال عادی کار سیستم عامل را مختل نماید).

اطلاع از وضعیت فعلی:

برای اطلاع از اینکه سرویس postgres توسط کاربر root در حال اجرا است یا خیر، دستورهای زیر را می‌توان به کار برد.

```
service postgresql status
systemctl status postgresql
```

خروجی حاصل از اجرای دستورهای فوق (در صورت فعال یا غیر فعال بودن سرویس) به صورت زیر است.

```
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled;
    vendor prese
   Active: active (exited) since Tue 2018-05-29 12:12:50 +0430; 31min
    ago
  Process: 30931 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 30931 (code=exited, status=0/SUCCESS)
```

مقاوم‌سازی:

برای این منظور می‌توان یک حساب جداگانه تحت عنوان postgres ایجاد نموده و از این حساب به منظور انجام اعمال مدیریتی در سرویس PostgreSQL استفاده نمود. به این منظور گام‌های زیر باید طی شود:

۱. سرویس postgres را متوقف نمایید. برای این منظور می‌توانید از دستورهای زیر استفاده نمایید:

```
systemctl stop postgresql
service postgresql status
```

۲. مجوزهای مربوط به دایرکتوریها و فایلهای پایگاه داده را به گونه‌ای تغییر دهید که کاربر مورد نظر امکان خواندن و نوشتن در آنها را داشته باشد. در صورتی که این کار را انجام ندهید، کاربر قادر به دسترسی به فایلهای پایگاه داده و جداول نخواهد بود.

```
chown  -R  postgres     /var/lib/postgresql/10/main
```

۳. سرویس postgres را با استفاده از کاربر مورد نظر اجرا نمایید. برای این منظور از مجموعه دستورات زیر استفاده می‌کنیم:

```
sudo su postgres
/usr/lib/postgresql/10/bin/pg_ctl -D /var/lib/postgresql/10/main -l
    logfile restart
```

۴. پس از طی کردن مراحل فوق، با استفاده از دستور زیر می‌توان وضعیت اجرای سرویس postgres اطمینان حاصل کرد:

```
/usr/lib/postgresql/10/bin/pg_ctl -D /var/lib/postgresql/10/main -l
    logfile status
```

خروجی حاصل از اجرای دستور فوق به صورت زیر می‌باشد:

```
pg_ctl: server is running (PID: 346)
/usr/lib/postgresql/10/bin/postgres "-D"
    "/var/lib/postgresql/10/main" "-c"
    "config_file=/etc/postgresql/10/main/postgresql.conf"
```

لازم به ذکر است که در صورت بررسی وضعیت سرویس postgres توسط دستورات systemctl و service که در بالا به آن‌ها اشاره شده، سرویس postgres همچنان به صورت غیرفعال نشان داده می‌شود.

۵-۱ پیکربندی فایل‌های رویدادنگاری

در PostgreSQL تمام رویدادهای مرتبط با پایگاه داده در فایل‌های از قبل تعیین شده‌ای ثبت می‌شوند.

تهدید/توجیه امنیتی:

هیچ کاربری به جز postgres نباید حق خواندن یا نوشتن روی فایل‌های رویدادنگاری را داشته باشد. رعایت این مورد امنیتی از نشت اطلاعات این فایل‌ها جلوگیری می‌کند. به عنوان مثال، ممکن است یک دستورGRANT شامل اطلاعات مهمی باشد که به صورت رمز نشده در فایل رویدادنگاری ذخیره‌شده است.

اطلاع از وضعیت فعلی:

برای اطلاع از وضعیت فعلی مجوزهای مربوط به فایل رویدادنگاری، از دستور زیر استفاده می‌شود:

```
ls -lh /var/log/postgre*
```

خروجی دستور فوق به صورت زیر است.

```
-rw-r----- 1 postgres adm  16K  مه30 16:00 postgresql-10-main.log
```

مقاوم‌سازی:

برای این منظور می‌بایست به ترتیب دستورات زیر را برای تعیین مالکیت و حقوق دسترسی فایل مذکور انجام داد.

```
chown       -R    postgres    /var/log/postgres*
chgrp       -R    adm         /var/log/postgres*
chmod       640               /var/log/postgres*
```

۱-۶ جمع‌بندی

در این فصل به تشریح برخی از مهم‌ترین پارامترهای امنیتی محیط اجرای سمپاد که به طور مستقیم بر عملکرد آن تاثیرگذار است، پرداختیم. در این راستا، تنظیمات مربوط به حقوق دسترسی فایل تنظیمات، دایرکتوری ذخیره‌داده، امن‌سازی کاربر PostgreSQL، راه‌اندازی سرویس PostgreSQL توسط یک کاربر عادی و پیکره‌بندی فایل رویدادنگاری 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	**امن سازی محیط اجرا**		
۱-۱	پیکربندی فایل تنظیمات	☐	☐
۱-۲	پیکربندی دایرکتوری ذخیره داده	☐	☐
۱-۳	پیکربندی اجرای دستور از طریق خط فرمان توسط کاربر postgres	☐	☐
۱-۴	راه اندازی سرویس postgres توسط یک کاربر عادی	☐	☐
۱-۵	پیکربندی فایل رویدادنگاری	☐	☐

۲ نصب و پیکربندی امن پایگاه داده

تنظیمات پیش‌فرض، یکی از مسایلی است که باعث بروز مشکلات امنیتی زیادی می‌شود. از این رو تغییر مقادیر پیش‌فرض پارامترهای مختلف به مقادیر امن، یکی از نکات ضروری در ایجاد امنیت است. دسته‌ای از این تنظیمات مربوط به پارامترهای اتصال به سِرور است. اینکه کاربران روی چه آدرسی، چه پورت و سوکتی بتوانند به سِرور متصل شوند. در ادامه در مورد هر یک از این پارامترها جزییات بیشتری بیان می‌شود.

۲-۱ پارامتر listen_addresses

این پارامتر آدرس‌های TCP/IP ای را مشخص می‌کند که سِرور روی آن‌ها گوش‌داده و منتظر برقراری ارتباط با برنامه‌های کاربردی کاربران است. در واقع این آدرس واسط بین سِرور و کاربران است که کاربران برای برقراری ارتباط با سِرور باید این آدرس را وارد کنند. این پارامتر می‌تواند تمامی آدرس‌ها، آدرس‌های محلی، یا مجموعه‌ای از آدرس‌های مورد اعتماد باشد. اگر آدرس سِرور ثابت باشد، می‌توان مقدار آن را برابر با آدرس سِرور گذاشت، به عنوان مثال:

```
listen_addresses = '10.10.100.5'
```

در صورتی که این مقدار برابر با localhost قرار داده شود، تنها به آدرس‌های TCP/IP محلی که با loopback متصل شده‌اند، اجازه ارتباط داده می‌شود و تمامی ارتباطات به آدرس‌های دیگر برگردانده می‌شوند. به عبارت دیگر این پارامتر مشخص می‌کند که کدام تلاش برای برقراری ارتباط را بپذیرد.

تهدید/توجیه امنیتی:

در صورتی که مقدار این پارامتر محدودشده باشد، زمانی که یک حمله‌کننده درخواست‌های متعددی به شبکه ارسال می‌کند، تنها درخواست‌هایی که سِرور روی آن آدرس‌ها در حال گوش دادن است به سِرور ارسال می‌شود. با این حال، اگر سِرور روی تمامی آدرس‌ها گوش دهد، ممکن است مقدار زیادی درخواست از طرف حمله‌کننده داشته باشد که این امر می‌تواند باعث کاهش کارایی سِرور شده و حتی باعث حمله منع سرویس شود.

اطلاع از وضعیت فعلی:

برای اطلاع از آدرس‌هایی که سرویس postgres در حال شنود درخواست های کاربران است، از دستورات زیر استفاده می‌شود:

```
tmp1=$(psql -c 'show config_file;'); \
tmp2=($tmp1); \
config_path=${tmp2[2]}; \
grep listen $config_path
```

خروجی دستور بالا، برای تنظیمات پیش‌فرض به صورت زیر است:

```
#listen_addresses = 'localhost'     # what IP address(es) to listen
    on;
```

مقاوم‌سازی:

از آنجا که مقدار پیش‌فرض برای این پارامتر localhost است و این مقدار، امن به حساب می‌آید، پیشنهاد می‌شود مقدار پارامتر به پیش فرض تغییر کند. مجموعه دستورات زیر تنظیم پارامتر شنود به localhost را انجام می‌دهد:

```
sudo su
sed   -i    's/\(.*\)listen_addresses = \(.*\)/#listen_addresses =
    \2/'  $config_path
```

در صورتی که سیاست‌های امنیتی به نحوی باشد که نیاز باشد به این پارامتر مقادیر دیگری اختصاص داده شود، می‌تواند از این دستور استفاده کرد و بجای <trusted-ip> مقادیر مورد نظر را وارد کرد.

```
sudo su
sed -i 's/\(.*\)listen_addresses = '"'.*.'"'\(.*\)/listen_addresses =
    '"'<trusted-ip>'"'\2/'  $config_path
```

نهایتاً نیز برای اعمال تغییرات انجام شده، می بایست سرویس را مجدداً راه اندازی نمود.

۲-۲ پارامتر port

این پارامتر نشان دهنده شماره پورتی است که سِرور روی آن گوش می‌دهد. مقدار پیش‌فرض آن 5432 است.

تهدید/توجیه امنیتی:

لازم است از شماره پورت دیگری استفاده گردد تا توانایی مهاجمان در شناسایی مشخصات سِرور و حمله به آن کاهش یابد.

اطلاع از وضعیت فعلی:

برای اطلاع از اینکه سرویس postgres در حال حاضر از طریق چه پورتی درخواست‌ها را پیگیری می‌کند، از دستور زیر استفاده می‌شود.

```
grep port $config_path
```

خروجی دستور بالا، برای تنظیمات پیش‌فرض به صورت زیر است.

```
port = 5432             # (change requires restart)
                        # supported by the operating system:
```

```
                                # supported by the operating system:
                                #   %r = remote host and port
```

مقاوم‌سازی:

با اجرای دستورات زیر، شماره پورت به شماره مورد نظر تغییر می‌یابد. لازم است بجای <port number> مقدار مناسب و دلخواهی قرار گیرد.

```
sudo su
sed -i 's/\(.*\)port = 5432\(.*\)/port = <port number>\2/'
    $config_path
```

نهایتاً نیز برای اعمال تغییرات انجام شده در سرویس postgres، می‌بایست سرویس را مجدداً راه اندازی نمود.

۲-۳ پارامتر max_connections

پارامتر بیشترین تعداد ارتباطات همزمان به پایگاه داده را نشان می‌دهد. مقدار پیش‌فرض این پارامتر 100 است، اما مقدار آن باید متناسب با تنظیمات کِرنِل باشد.

تهدید/توجیه امنیتی:

اگر مقدار این پارامتر کوچک باشد، به راحتی می‌توان حمله منع سرویس روی آن اعمال کرد. به عنوان مثال اگر مقدار این پارامتر 10 باشد، یک مهاجم می‌تواند با برقراری 10 ارتباط با سرور، سِرور را مشغول کند و نگذارد با کاربران دیگر ارتباط برقرار کند و در نتیجه از سرویس‌دهی به کاربران مجاز جلوگیری می‌شود.

اطلاع از وضعیت فعلی:

برای اطلاع از اینکه سرویس postgres در حال حاضر توانایی برقراری چند ارتباط را دارد، از مجوعه دستورات زیر استفاده می‌شود:

```
tmp1=$(psql -c 'show config_file;'); \
tmp2=($tmp1); \
config_path=${tmp2[2]}; \
grep max_connections $config_path
```

خروجی دستور بالا، برای تنظیمات پیش‌فرض به صورت زیر است:

```
max_connections = 100                   # (change requires restart)
```

مقاوم‌سازی:

با اجرای دستورات زیر، پارامتر مذکور با مقدار مورد نظر تغییر می‌یابد. لازم است بجای < appropriate value> مقدار مناسبی بسته به میزان بار معمول روی سرور، قرار گیرد.

```
sudo su
sed -i 's/\(.*\)max_connections = 100\(.*\)/max_connections =
    <appropriate value>\2/'    $config_path
```

نهایتاً نیز برای اعمال تغییرات انجام شده در سرویس postgres، می‌بایست سرویس را مجدداً راه اندازی نمود.

۲-۴ پارامتر unix_socket_permissions

با استفاده از این پارامتر می‌توان دسترسی به سوکت‌های یونیکس را کنترل کرد. این سوکت‌ها از مجوزهای موجود در فایل سیستم استفاده می‌کنند و بر اساس آن‌ها مقداردهی می‌شوند. مقدار پیش‌فرض این پارامتر ۰۷۷۷ است، بدین معنی که هرکسی می‌تواند به آن متصل شود.

تهدید/توجیه امنیتی:

برای اتصال به سرور، پس از آنکه مقدار IP و درگاه به درستی بیان شد، باید به سوکتی وصل شد که سِرور روی آن برای متصل شدن برنامه‌های کاربران گوش می‌دهد. اگر بنا به نیاز سیستم، لازم باشد دسترسی به سوکت محدود شود، می‌توان به چند طریق این کار را کرد. اول اینکه مقدار این پارامتر را تغییر داد، به عنوان مثال با مقدار ۰۷۷۷ تنها کاربران و گروه‌ها و با مقدار ۰۷۰۰ تنها کاربران می‌توانند به سوکت متصل شوند. از طرفی دیگر اگر بخواهیم تنها سوکت‌های خاصی محدود شوند و مقدار این سوکت‌ها هم مشخص شود، باید ابتدا در پارامتر unix_socket_directories مقادیر مورد نظر را وارد کرده و سپس مجوز آن‌ها را در این پارامتر تغییر داد. در سامانه‌هایی مثل solaris که socket_permissions ها را نادیده می‌گیرند، تنها راه‌حل دوم قابل اعمال است.

اطلاع از وضعیت فعلی:

برای اطلاع از مقدار پارامتر مذکور از دستور زیر استفاده می‌گردد.

```
grep unix_socket_permissions $config_path
```

خروجی دستور بالا، برای تنظیمات پیش‌فرض به صورت زیر است.

```
#unix_socket_permissions = 0777# begin with 0 to use octal notation
```

مقاوم‌سازی:

با اجرای دستورات زیر، پارامتر مذکور به مقدار مورد نظر تغییر می‌یابد. لازم است بجای <appropriate value> مقدار مناسبی بسته به نیازمندی‌های امنیتی سیستم، قرار گیرد.

```
sudo su
```

```
sed -i 's/\(.*\)unix_socket_permissions =
    0777\(.*\)/unix_socket_permissions = <appropriate value>\2/'
    $config_path
```

نهایتاً نیز برای اعمال تغییرات انجام شده، می‌بایست سرویس را مجدداً راه اندازی نمود.

۲-۵ پارامتر authentication_timeout

این پارامتر بیشترین زمانی را که یک کاربر فرصت دارد تا فرآیند احراز هویت خود را کامل کند، نشان می‌دهد. اگر یک کاربر نتواند در زمان مقرر شده قدم‌های پروتکل احراز هویت را کامل کند، سِرور ارتباطش را با او قطع می‌کند. مقدار پیش‌فرض این پارامتر یک دقیقه است.

تهدید/توجیه امنیتی:

اگر این زمان طولانی باشد، مهاجم می‌تواند یک ارتباط را ایجاد کرده و فرآیند احراز هویت را تا زمان نامحدودی ادامه دهد. در نتیجه با این کار منابع سیستم به هدر رفته و از طرفی به دلیل محدود بودن تعداد ارتباط‌های همزمان با یک سرور، در صورت ایجاد تعداد زیادی از این ارتباط‌های نیمه احرازشده، حمله منع سرویس رخ می‌دهد.

اطلاع از وضعیت فعلی:

برای اطلاع از مقدار پارامتر مذکور، از دستور زیر استفاده می‌گردد.

```
grep authentication_timeout $config_path
```

خروجی دستورات بالا برای تنظیمات پیش‌فرض به صورت زیر است.

```
#authentication_timeout = 1min          # 1s-600s
```

مقاوم‌سازی:

برای مقدار دهی این پارامتر به مقدار پیش‌فرض (یک دقیقه)، از دستورات زیر استفاده می‌شود.

```
sudo su
sed -i 's/\(.*\)authentication_timeout =
    .*.min\(.*\)/#authentication_timeout = 1min\2/'    $config_path
```

همچنین برای ارزش دهی به یک مقدار مناسب امنیتی چون <appropriate value> از دستورات زیر استفاده می‌گردد.

```
sudo su
```

```
sed -i 's/\(.*\)authentication_timeout =
    .*.min\(.*\)/authentication_timeout = <appropriate value>min\2/'
    $config_path
```

نهایتاً نیز برای اعمال تغییرات انجام شده می بایست سرویس را مجدداً راه اندازی نمود.

۲-۶ پارامتر password_encryption

هنگامی‌که از متد password برای ایجاد کاربر استفاده می‌شود (در دستورات CREATE USER و CREATE ROLE)، باید مشخص گردد که گذرواژه به صورت رمز شده نگهداری شود یا خیر. با مقداردهی این پارامتر می‌توان مطمئن بود که حتی اگر در زمان ایجاد کاربر یا نقش، ذکر رمز شدن گذرواژه فراموش شد، بازهم گذرواژه به صورت رمز شده ذخیره شود. این پارامتر به صورت پیش‌فرض روشن است.

تهدید/توجیه امنیتی:

در صورتی‌که گذرواژه‌ها به صورت رمز نشده ذخیره شوند، مهاجم می‌تواند به سادگی آن‌ها را بخواند و از آن‌ها سوءاستفاده کند.

اطلاع از وضعیت فعلی:

برای اطلاع از وضعیت فعلی پارامتر مذکور، از دستورات زیر استفاده می‌گردد.

```
grep 'password_encryption ='  $config_path
```

خروجی دستور فوق برای تنظیمات پیش‌فرض به صورت زیر است.

```
#password_encryption = md5          # md5 or scram-sha-256
```

مقاوم‌سازی:

برای فعال کردن رمزنگاری مربوط به کلمه عبور کاربران، از مجموعه دستورات زیر استفاده می‌شود.

```
sudo su
sed -i 's/\(.*\)password_encryption = off\(.*\)/#password_encryption
    = on\2/'    $config_path
```

نهایتاً نیز برای اعمال تغییرات انجام شده، می بایست سرویس را مجدداً راه اندازی نمود.

۲-۷ پارامتر ssl

با این پارامتر می‌توان مشخص کرد که ارتباط‌های بین کاربران و سِرور به صورت رمز شده باشد یاخیر. به صورت پیش‌فرض، این پارامتر غیر فعال است و بنا بر نیاز سیستم می‌توان آن را فعال کرد. توجه داشته باشید

که در صورت فعال کردن این گزینه باید گواهی های معتبری برای سِرور ارائه کنید، در غیر این صورت سِرور PostgreSQL هنگام راه اندازی خطا می‌دهد.

تهدید/توجیه امنیتی:

از ssl برای امن کردن یک ارتباط و جلوگیری از افشاء اطلاعات استفاده می‌شود. از آنجا که ارتباط رمز شده است، حتی اگر مهاجمی بتواند داده‌های ارتباط را شنود کند، قادر به دستیابی به اطلاعات آن نخواهد بود.

اطلاع از وضعیت فعلی:

برای اطلاع از وضعیت پارامتر مذکور، از دستور زیر استفاده می‌شود.

```
grep 'ssl '  $config_path
```

خروجی دستور فوق برای تنظیمات پیش‌فرض به صورت زیر است.

```
ssl = on
```

مقاوم‌سازی:

اگر در کاربردی لازم باشد که ارتباطات به صورت رمز شده بین کاربر و سِرور انتقال یابد، اجرای دستور زیر این امکان را فراهم می‌کند.

```
sudo su
sed -i 's/\(.*\)ssl = off\(.*\) /ssl = on\1/'    $config_path
```

نهایتاً نیز برای اعمال تغییرات انجام شده، می بایست سرویس را مجدداً راه اندازی نمود.

۸-۲ جمع‌بندی

در این فصل به تشریح برخی از مهم‌ترین پارامترهای امنیتی مربوط به پیکربندی سمپاد قبل از بکارگیری عملیاتی آن پرداختیم. در این راستا، تنظیمات مربوط به کانال رسیدگی به درخواست‌ها، شماره پورت سرویس‌دهی، حداکثر تعداد کاربران قابل پشتیبانی در یک زمان، مجوزهای دسترسی، مهلت زمانی برای احراز اصالت کاربران، رمزنگاری کلمه عبور کاربران، تنظیمات مربوط به SSL 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	پارامتر listen_addresses	☐	☐
۲-۲	پارامتر port	☐	☐
۲-۳	پارامتر max_connections	☐	☐
۲-۴	پارامتر unix_socket_permissions	☐	☐
۲-۵	پارامتر authentication_timeout	☐	☐
۲-۶	پارامتر password_encryption	☐	☐
۲-۷	پارامتر ssl	☐	☐

۳ امن سازی اتصال به پایگاه داده

احراز اصالت کاربران به هنگام ورود به پایگاه داده در postgresql با استفاده از فایل pg_hba.conf کنترل می‌شود. دستورات زیر برای مشخص کردن آدرس فایل مذکور استفاده می‌شود.

```
tmp1=$(psql -c 'show hba_file;'); \
tmp2=($tmp1); \
hba_path=${tmp2[2]}; \
echo  $hba_path
```

خروجی دستورات بالا به صورت زیر است.

```
/etc/postgresql/10/main/pg_hba.conf
```

هر سطر از فایل pg_hba.conf، نشان دهنده نوع ارتباط، نام پایگاه داده، نام کاربر، بازه آدرس IP کاربر و روش احراز هویت استفاده شده برای کاربر است. اولین سطری که با پارامترهای موجود در یک درخواست دسترسی مطابقت داشته باشد، برای احراز هویت کاربر، به کاربرده می‌شود. اگر هیچ یک از سطرها با درخواست تطبیق پیدا نکند، درخواست دسترسی رد می‌شود. قالب هر سطر می‌تواند به صورت یکی از حالت‌های زیر باشد.

TYPE	DATABASE	USER	ADDRESS	METHOD	OPTIONS
local	Database	User	-	Auth-method	[OPTIONS]
host	Database	User	Address	Auth-method	[OPTIONS]
hostssl	Database	User	Address	Auth-method	[OPTIONS]
hostnossl	Database	User	Address	Auth-method	[OPTIONS]

در ادامه به بررسی هر یک از مولفه های این جدول می‌پردازیم.

مولفه type : این مولفه بیانگر نوع ارتباط می‌باشد که می‌تواند در قالب چهار دسته: local، Host، Hostssl، Hostnossl رده‌بندی شود. در ادامه تشریح هر یک از آن‌ها آورده شده است.

local	این رکورد، با ارتباط‌هایی منطبق می‌شود که از سوکت‌های دامنه یونیکس صورت گرفته باشند.
Host	این رکورد با ارتباط‌هایی منطبق می‌شود که از آدرس‌های TCP/IP برای اتصال اقدام کرده‌اند. به علاوه این رکورد در مورد تمامی ارتباطات ssl و non-ssl صدق می‌کند.
Hostssl	این رکورد با ارتباط‌هایی منطبق می‌شود که از آدرس‌های TCP/IP برای اتصال اقدام کرده‌اند و از نوع ssl هستند.
Hostnossl	این رکورد رفتاری مخالف رفتار رکورد قبلی دارد و تنها با ارتباط‌هایی منطبق می‌شود که از یک آدرس TCP/IP باشد و در آن از ssl استفاده نشده باشد.

مولفه Database : نام پایگاه داده‌ای را مشخص می‌کند که به هنگام احراز اصالت کاربر می‌بایست با این رکورد منطبق باشد.

All	تمامی پایگاه داده‌ها با این رکورد منطبق می‌شوند.
Sameuser	در صورتی که Sameuser در این قسمت قرار داده شود، تنها در حالتی که پایگاه داده درخواست شده هم نام با کاربر درخواست دهنده باشد، درخواست با این رکورد منطبق می‌شود.
Samerole	در این حالت باید کاربر درخواست دهنده، عضو نقشی باشد که هم نام با پایگاه داده است.
	برای نام بردن چندین پایگاه داده نیز می‌توان آن‌ها را با کارکتر "," از هم جدا کرد.

مولفه User : در این قسمت مشخص می‌شود که کدامیک از کاربران پایگاه داده در این رکورد منطبق می‌شوند. در صورتی که مقدار آن all قرار داده شود، تمامی کاربران پایگاه داده در این رکورد صدق می‌کنند. در غیر این صورت، می‌تواند نام یک کاربر خاص یا گروهی از کاربران قرار داده شود.

مولفه Address: آدرس ماشین کاربری که با این رکورد منطبق می‌شود، در این قسمت مشخص می‌شود. مقدار این قسمت می‌تواند نام یک میزبان، یک بازه آدرس IP، یا all باشد.

مولفه Method : در این قسمت روشی که برای احراز هویت است، مشخص می‌شود.

trust	در صورتی که در قسمت روش، trust قرار داده‌شده باشد، به افراد بدون هیچ احراز هویتی اجازه برقراری ارتباط داده می‌شود. این روش به هرکسی اجازه می‌دهد با هر سطح کاربری که می‌خواهد به سِرور پایگاه داده وصل شود، بدون اینکه از آن‌ها گذرواژه یا راهی برای احراز هویتش نیاز باشد.

reject	برخلاف روش قبل، در این روش هر ارتباطی برگردانده می‌شود و اجازه برقراری ارتباط داده نمی‌شود. این گزینه برای فیلتر کردن گروه مشخصی از میزبان‌ها مفید است. به عنوان مثال می‌توان در این رکورد اجازه ارتباط یک سری از آدرس‌های IP را رد کرد و در رکورد بعد، به بقیه آدرس‌ها مجوز ارتباط داد.
md5	در این حالت کاربر باید یک گذرواژه رمز شده با md5 را برای احراز هویت ارائه کند.
password	در این حالت نیز کاربر باید یک گذرواژه رمز نشده را برای احراز هویت خود ارائه کند. ازآنجاکه گذرواژه به صورت رمز نشده از شبکه می‌گذرد، لازم است نوع ارتباط قابل‌اعتماد باشد.
gss	برای احراز هویت از GSSAPI استفاده می‌کنند. این گزینه تنها برای ارتباطات TCP/IP قابل اعمال است.
sspi	برای احراز هویت از SSPI استفاده می‌کنند. این گزینه تنها برای Windows قابل اعمال است.
krb5	برای احراز هویت از کربراس V5 استفاده می‌کنند. این گزینه تنها برای ارتباطات TCP/IP قابل اعمال است.
ident	در این روش از نام کاربری سیستم‌عامل کاربر استفاده می‌شود. برای این کار از سِرور ident که روی سیستم کاربر است استفاده می‌شود. در این حالت باید نام کاربری با کاربران مربوط به پایگاه داده یکی باشد. این گزینه تنها برای ارتباطات TCP/IP قابل‌اعمال است. برای ارتباطات داخلی لازم است از گزینه peer استفاده شود.
peer	در این روش از نام کاربری سیستم‌عامل کاربر استفاده می‌شود. در این حالت باید نام کاربری با کاربران مربوط به پایگاه داده یکی باشد.
ldap	برای احراز هویت از سِرور LDAP استفاده می‌شود.
radius	برای احراز هویت از سِرور RADIUS استفاده می‌شود.
cert	برای احراز هویت از گواهی SSL کاربر استفاده می‌شود.
pam	احراز هویت از طریق یک ماژول جایگزین احراز هویت (PAM) که توسط سیستم‌عامل ایجاد می‌شود، انجام می‌شود.

همان‌طور که گفته شد، زمانی که یک درخواست ارتباط ارسال می‌شود، درخواست به ترتیب با هر سطر این جدول مقایسه می‌شود. در اولین سطری که انطباق صورت گرفت، یا سِرور درخواست را بدون هیچ شرطی می‌پذیرد، یا نحوه احراز هویت مشخص‌شده را روی آن اعمال می‌کند و یا درخواست رد می‌شود. به همین دلیل توصیه می‌شود سطرهای اولیه دارای پارامترهای محدودتر باشند و از طرفی روش احراز هویت آسان‌گیرانه‌تری داشته باشند و هر چه به سطرهای آخر می‌رویم، این پارامترها آزادتر و روش احراز هویت و سیاست‌های از این دست سخت‌گیرانه‌تر باشند. نحوه قرارگیری این سطرها وابستگی زیادی به نوع کاربرد پایگاه داده در سیستم دارد و باید بنا بر نیازمندی سیستم و سیاست‌های دسترسی آن، به دقت مشخص شود زیرا هر اشتباه کوچک در این فایل می‌تواند یک کاربر غیرمجاز را وارد سیستم کند و یا از سرویس‌دهی به یک

کاربر مجاز جلوگیری کند. در این قسمت به ذکر چند مورد ضروری در مورد نحوه قرارگیری این سطرها بسنده می‌کنیم و تنظیمات دقیق‌تر را بسته به نوع نیازمندی‌ها به مدیر سیستم واگذار می‌کنیم.

۳-۱ ارتباطات trust

یکی از چالش‌ها در خصوص احراز اصالت، روش trust است. استفاده از این روش، بستگی زیادی به سیاست‌های احراز اصالت دارد. در ادامه به چند سناریو در این رابطه می‌پردازیم.

سناریوی اول:

در برخی از کاربردها تمامی کسانی که به سِرور متصل می‌شوند، قابل‌اعتماد فرض می‌شوند. بدین معنی که تمامی ارتباطات مشخص‌شده از نوع trust هستند و هیچ روش احراز هویتی برای آن‌ها در نظر گرفته نشده است. این سیاست بسیار نا امن و آسیب‌پذیر است و توصیه نمی‌شود.

سناریوی دوم:

اگر سامانه‌ای که PostgreSQL در آن استفاده می‌شود ارتباطات داخلی امن به حساب بیایند و اعتماد کاملی به کاربران داخلی (local) داشته باشیم، می‌توانیم ارتباط آن‌ها را از نوع trust انتخاب کنیم. در این صورت این کاربران بدون نیاز به هیچ روش احراز هویتی می‌توانند با هر نقشی که می‌خواهند به سِرور متصل شوند. این رویکرد در بسیاری از سامانه‌ها وجود دارد اما باید دقت کرد که خطر مهاجم داخلی کمتر از مهاجم خارجی نیست و این کار ریسک بالایی می‌تواند داشته باشد. در صورتی‌که این رویکرد استفاده شود سطر اول فایل به صورت زیر خواهد بود:

```
local       all         all                 trust
```

سناریوی سوم:

دسته دیگری از سامانه‌ها وجود دارند که نه‌تنها به کاربران خارجی و host ها اعتماد ندارند، بلکه به کاربران داخلی و local نیز اعتمادشان به حدی نیست که آن‌ها را trust قرار دهند. از این رو در این سامانه‌ها تمامی ارتباط‌ها از انواع دیگری غیر از trust هستند. ازلحاظ امنیتی توصیه می‌شود که از md5 بجای trust استفاده شود زیرا گذرواژه در طول راه نیز به صورت درهم شده جریان می‌یابد در حالی‌که همان‌طور که در قسمت قبل نیز گفته شد، روش password به صورت درهم نشده در شبکه جریان دارد. برای اطلاع از وضعیت فعلی می‌توان از دستور زیر استفاده کرد.

```
grep trust  $hba_path | grep -v '#'
```

خروجی دستور بالا به صورت زیر است.

```
local   all     postgres     trust
```

اگر در سامانه‌ای نیاز به رعایت این سطح از امنیت وجود داشته باشد، دستور زیر می‌تواند به اعمال این سیاست کمک کند و تمامی ارتباطات trust را به md5 تغییر دهد.

```
sudo su
sed -i 's/^\([^#].*\)trust\(.*\)$/\1md5\2/'      $hba_path
```

توجه داشته باشید که پس از این تغییر می‌بایست به کاربران گذرواژه معتبر مطابق با md5 تخصیص دهید در غیر این صورت کاربران هر گذرواژه‌ای که وارد کنند، قادر به ورود به سیستم نخواهند بود. نهایتاً نیز برای اعمال تغییرات انجام شده می بایست سرویس را مجدداً راه اندازی نمود.

۳-۲ درخواست ارتباطات superuser از راه دور

فرض کنید در پایگاه داده، کاربری به نام admin وجود دارد که دسترسی‌های مدیریتی برای او ایجاد شده است. ازآنجاکه نمی‌خواهیم هیچ کاربری با استفاده از نقش admin وارد سیستم شود، لازم است اولین قانون در این فایل به صورت زیر نوشته شود:

```
local       all         admin             md5
```

سیاست‌های متفاوتی در خصوص این نقش وجود دارد که بسته به نیازهای امنیتی می‌توان یکی از آن‌ها را اعمال کرد.

سناریوی اول:

در این سناریو فرض شده است که به نقش admin این حق را می‌دهیم که بتواند از بیرون از سازمان نیز، نقش‌های مدیریتی‌اش را ایفا کند. در این صورت می‌توان خط زیر را نیز به قواعد اضافه کرد.

```
host  all   admin       0.0.0.0/0         md5
```

این خط بدین معنی است که admin با هر IP ای که بود، می‌تواند با واردکردن گذرواژه md5 خود وارد سیستم شود.

سناریوی دوم:

اگر بخواهیم نسبت به رویکرد قبلی کمی سخت‌گیرانه‌تر باشیم، بهتر است دسترسی مدیر از خارج local را به یک آدرس مشخص محدود کنیم. به عنوان مثال اگر می‌دانیم که مدیر ممکن است از منزلش به سِرور وصل شود، تنها ارتباط از آدرس IP مدیر و نه تمامی ارتباطات از هر آدرسی را مجاز بشماریم. دستور زیر می‌تواند این سیاست را نشان دهد:

```
host  all         admin       213.233.183.181        md5
host  all         admin       0.0.0.0/0            reject
```

این خط تنها ارتباط admin از این آدرس را با md5 می‌پذیرد و اگر درخواست دیگری با این نام کاربری از آدرس دیگری بیاید، درخواست رد می‌گردد.

سناریوی سوم (پیشنهادی):

پیشنهاد ما این است که کاربر admin تنها بتواند از داخل سازمان وارد سیستم شود چرا که در غیر این صورت احتمال حملات بیشتر می‌شود و به عنوان مثال با جعل آدرس IP، مهاجم می‌تواند به صورت admin به سِرور متصل شود. پس به صورت کلی تنظیمات زیر امن‌تر هستند و پیشنهاد می‌شود برای مقاوم کردن سیستم از این تنظیمات استفاده شود.

```
local       all         admin                               md5
host        all         admin       0.0.0.0/0               reject
```

نهایتاً نیز برای اعمال تغییرات انجام شده در سرویس postgres، برای هر یک از سه سناریوی شرح داده شده در بالا، می بایست سرویس را مجدداً راه اندازی نمود.

۳-۳ درخواست‌های تعریف‌نشده

در PostgreSQL گفته‌شده است در صورتی‌که درخواستی ارایه شود که در هیچ‌یک از قواعد گفته‌شده صدق نکند، درخواست رد می‌شود؛ اما اکتفا کردن به این گفته کافی نیست و بهتر این است که از قاعده آخر برای تعریف رفتار پیش‌فرض سیستم (برای حالاتی که با هیچ‌یک از قواعد موجود تطبیقی حاصل نشود) استفاده نمود. توجه نمایید که برخی سامانه‌ها دارای خط‌مشی دسترسی باز هستند، به این معنی که قاعده آخر آن‌ها به همه اجازه ورود می‌دهد.

```
host        all         all         0.0.0.0/0   trust
```

برخی دیگر دارای خط‌مشی دسترسی بسته هستند، یعنی قاعده آخر آن‌ها reject است، بدین معنی که تمامی آدرس‌های ناشناخته باید رد شوند؛ به عبارت دیگر اگر برای آدرسی قاعده‌ای تعریف نشده باشد، پس لازم نبوده که این آدرس با سِرور postgres ارتباطی داشته باشد.

```
host        all         all         0.0.0.0/0   reject
```

اما در برخی سامانه‌ها نیز قاعده‌ای مابین این دو استفاده‌شده است. درخواست ناشناخته نه رد می‌شود و نه قبول، بلکه اگر از فیلتر احراز هویت گذشت، می‌تواند وارد سیستم شود.

```
host        all         all         0.0.0.0/0           md5
```

نهایتاً نیز برای اعمال تغییرات انجام شده برای هر یک از سه سناریوی شرح داده شده در بالا، می‌بایست سرویس را مجدداً راه اندازی نمود.

۳-۴ ترتیب خطوط در فایل

ترتیب خط‌ها در فایل بسیار مهم است و جابجایی هر یک می‌تواند به کلی معنی قواعد را تغییر دهد. به عنوان مثال ترتیب خطوط زیر را در نظر بگیرید.

```
local       all         @admins                         md5
local       all         all                             trust
host        all         @admins     0.0.0.0/0           reject
host        all         all         127.0.0.1/32        trust
host        all         all         ::1/128             trust
host        all         all         0.0.0.0/0           md5
```

بنابراین هرگاه درخواستی به سِرور ارسال گردد، قدم‌های زیر دنبال می‌شود:

۱. ابتدا بررسی می‌شود که کاربر از نوع admin هست یا نه. اگر admin بود باید گذرواژه خود را به صورت md5 ارسال کند.
۲. اگر کاربر از نوع admin نبود، اما از شبکه داخلی local ارتباط برقرار کرده بود، از نوع trust محسوب می‌شود و نیاز به احراز هویت ندارد.
۳. در مرحله بعد، تمامی دسترسی‌ها به admin از خارج از local مسدود شده است.
۴. در قاعده چهارم، آدرس 127.0.0.1/32 همان آدرس شبکه داخلی است، در اینجا منظور loopback بوده است. ارتباطات از این دست نیز trust محسوب شده‌اند.
۵. در این سطر نیز همان قاعده قبلی نوشته‌شده است، با این تفاوت که در اینجا آدرس IPv6 مدنظر بوده است.
۶. در آخر نیز گفته‌شده است که هرکسی خارج از این قوانین گفته‌شده بخواهد وارد سیستم شود، چه رفتاری رویش انجام شود. در این مثال متد md5 برای آدرس‌های ناشناخته قرار داده‌شده است.

توجه کنید که ترتیب سطر اول و دوم بسیار مهم است. با استفاده از این ترتیب از دسترسی کاربران داخلی به admin جلوگیری شده است. اگر این دو خط جابجا شوند، هرکسی از داخل می‌تواند بانام کاربری admin وارد شود و هیچ احراز هویتی هم برایش لازم نباشد! بعلاوه اگر قاعده 6 بجای 4 قرار می‌گرفت، دیگر ارتباط‌های loopback نیز نمی‌توانستند به سِرور متصل شوند و باید با md5 احراز هویت می‌شدند. پس باید توجه زیادی به ترتیب این خطوط داشت.

۳-۵ جمع‌بندی

در این فصل به تشریح تنظیمات مربوط به امن‌سازی اتصال به سمپاد پرداختیم. در این راستا، ارتباط امن کاربران به سمپاد به صورت محلی و از راه دور، امکان اتصال امن و از راه دور کاربران Superuser، نحوه

سرویس‌دهی به درخواست‌های کاربران تعریف نشده و الویت دهی به سیاست‌های اتصال کاربران به سمپاد مورد بحث و بررسی قرار گرفت.

۴ تنظیمات رویدادنگاری

ثبت وقایع سیستم و بازبینی آن‌ها در صورت رخداد مشکلات فنی و یا امنیتی یکی از نیازمندی‌های اصلی در پایگاه داده‌ها 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 سِرور و یا هدر رفتن فضای دیسک گردد، لازم است یک زیرمجموعه از وقایع مهم شناسایی گردد تا 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 بخش تعدادی از وقایع مهم که ثبت آن‌ها از ارزش امنیتی بالایی برخوردار است، معرفی می‌شوند.

۴-۱ پارامتر logging_collector

این پارامتر تمامی پیام‌هایی را که قرار است به stderr برود را به فایل‌های رویدادنگاری ارسال می‌کند. این رویکرد بهتر از این است که پیام‌ها در syslog ثبت شوند زیرا برخی از اطلاعات ممکن است در syslog ثبت نشود، در حالی که این اتفاق هرگز در این رویکرد اتفاق نمی‌افتد. در واقع این پارامتر برای این ایجادشده است که هیچ پیامی از دست نرود و به طور کامل ثبت شود. به عنوان مثال زمانی که بار روی سِرور خیلی زیاد است، syslog ترجیح می‌دهد پیام‌هایی را که نمی‌تواند ثبت کند را دور بریزد، اما در این رویکرد پروسه بلاک می‌شود تا بتواند تمامی پیام‌ها را ثبت کرد.

تهدید/توجیه امنیتی:

درصورتی‌که بار زیاد سِرور باعث عدم رویدادنگاری شود، یک مهاجم می‌تواند ابتدا کاری کند که بار روی سِرور زیاد شود و بعد با سوءاستفاده از این موقعیت به حملات دیگر دست بزند. ازآنجاکه ممکن است این حملات ثبت‌نشده باشند، خسارات زیادی ممکن است به سیستم وارد شود.

اطلاع از وضعیت فعلی:

برای اطلاع از وضعیت فعلی پارامتر مذکور، از دستور زیر استفاده می‌شود.

```
grep   'logging_collector = '   $config_path
```

خروجی حاصل از اجرای دستورات فوق به صورت زیر است.

```
#logging_collector = off            # Enable capturing of stderr
                                    # and csvlog
```

مقاوم‌سازی:

برای فعال کردن پارامتر مذکور، از دستور زیر استفاده می‌کنیم. لازم به ذکر است که اگر در سیستم موجود، نیاز باشد که هرگز بلاک فرآیندی رخ ندهد، می‌توان این پارامتر را غیر فعال کرد.

```
sed   -i    's/\(.*\)#logging_collector = off\(.*\)/logging_collector
   = on\2/'  $config_path
```

خروجی حاصل از اجرای دستور فوق به صورت زیر است.

```
logging_collector = on # Enable capturing of stderr and csvlog
```

نهایتاً نیز برای اعمال تغییرات می‌بایست سرویس را مجددا راه اندازی کرد.

۴-۲ پارامتر log_directory

زمانی که پارامتر logging_collector فعال شده باشد، به وسیله log_directory می‌توان مشخص کرد که رویدادهای ثبت‌شده در کجا ذخیره شوند.

تهدید/توجیه امنیتی:

اگر رویدادها در مکانی ناامن ذخیره شوند، مهاجم به راحتی می‌تواند به آن‌ها دسترسی پیدا کرده و آن‌ها را حذف کند.

اطلاع از وضعیت فعلی:

برای اطلاع از وضعیت فعلی پارامتر مذکور از دستور زیر استفاده می‌شود.

```
grep 'log_directory ='   $config_path
```

خروجی دستور فوق به صورت پیش‌فرض به صورت زیر است.

```
#log_directory = 'log' # directory where log files are written
```

مقاوم‌سازی:

برای تغییر مسیر پیش‌فرض به مسیر مناسبی چون <appropriate value>، از دستور زیر استفاده می‌شود.

```
sed  -i     's/\(.*\)#log_directory ='"'.*.'"'\
(.*\)/log_directory = '"'<appropriate value>'"'\2/' $config_path
```

نهایتاً برای اعمال تغییرات می بایست سرویس را مجدداً راه اندازی کنیم.

۴-۳ پارامتر log_connections

فعال کردن این پارامتر باعث می‌شود تمامی تلاش‌ها برای اتصال به سِرور ثبت شوند.

تهدید/توجیه امنیتی:

ثبت ورود و خروج کاربران به پایگاه داده می‌تواند در شناسایی وقایع امنیتی موثر باشد.

اطلاع از وضعیت فعلی:

برای اطلاع از وضعیت فعلی پارامتر مذکور، از دستور زیر استفاده می‌شود.

```
grep ' log_connections ='   $config_path
```

خروجی حاصل از دستور فوق به صورت زیر است.

```
log_connections = on
```

مقاوم‌سازی:

برای فعال کردن این پارامتر در صورت غیرفعال بودن، از دستور زیر استفاده می‌شود.

```
sed   -i    's/\(.*\)log_connections = off\(.*\)/log_connections =
   on\2/'  $config_path
```

نهایتاً برای اعمال تغییرات می‌بایست سرویس را مجددا راه اندازی کنیم.

۴-۴ پارامتر log_disconnections

با فعال کردن این پارامتر، هر زمان که یک جلسه خاتمه یابد، جلسه و طول مدت جلسه ثبت می‌شود.

تهدید/توجیه امنیتی:

ثبت ورود و خروج کاربران به پایگاه داده می‌تواند در شناسایی وقایع امنیتی موثر باشد.

اطلاع از وضعیت فعلی:

برای اطلاع از وضعیت فعلی پارامتر مذکور، از دستور زیر استفاده می‌شود.

```
grep ' log_disconnections ='   $config_path
```

خروجی حاصل از دستور فوق به صورت زیر است که نشان می‌دهد این پارامتر به صورت پیش‌فرض فعال است.

```
log_disconnections = on
```

مقاوم‌سازی:

برای فعال کردن پارامتر مذکور، از دستور زیر استفاده می‌شود.

```
sed   -i    's/\(.*\)log_connections = off\(.*\)/log_disconnections =
   on\2/'  $config_file
```

در نهایت نیز برای اعمال تغییرات، می‌بایست سرویس را مجدداً راه اندازی کنیم.

۴-۵ پارامتر log_hostname

به صورت پیش‌فرض، پیام‌هایی که برای رویدادنگاری اتصالات ذخیره می‌شوند، تنها بر اساس آدرس IP هستند. با روشن کردن این پارامتر می‌توان نام میزبان را نیز ذخیره کرد.

تهدید/توجیه امنیتی:

داشتن اطلاعات بیشتر در مورد اتصال‌ها می‌تواند در شناسایی وقایع امنیتی موثر باشد.

اطلاع از وضعیت فعلی:

برای اطلاع از وضعیت فعلی پارامتر مذکور از دستور زیر استفاده می‌شود.

```
grep 'log_hostname ='    $config_path
```

خروجی حاصل از اجرای دستور فوق به صورت زیر است.

```
#log_hostname = off
```

مقاوم‌سازی:

برای فعال کردن پارامتر مذکور، از دستور زیر استفاده می‌شود.

```
sed   -i    's/\(.*\)#log_hostname = off\(.*\)/log_hostname = on\2/'
    $config_path
```

نهایتاً نیز برای اعمال تغییرات می‌بایست سرویس را مجدداً راه اندازی کرد.

۴-۶ جمع‌بندی

در این فصل به تشریح برخی از مهم‌ترین پارامترهای امنیتی مربوط به پیکربندی تنظیمات رویدادنگاری پرداختیم. در این راستا، تنظیمات مربوط به رویدادنگاری از پوشه ذخیره‌سازی داده و ورود/خروج به سمپاد 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	تنظیمات رویدادنگاری		

۴-۱	پارامتر logging_collector	☐	☐
۴-۲	پارامتر log_directory	☐	☐
۴-۳	پارامتر log_connections	☐	☐
۴-۴	پارامتر log_disconnections	☐	☐
۴-۵	پارامتر log_hostname	☐	☐

۵ تنظیمات ویژه

در این فصل به تشریح برخی از پارامترهای امنیتی ویژه که در آخرین بروزرسانی اسناد استاندارد مقاوم‌سازی سمپاد postgres مطرح شده است، می‌پردازیم. در این راستا، تنظیمات مربوط به سطح اجرای سرویس postgres، پیکربندی آرشیو رویدادها و نصب افزونه‌های مهم جهت بازیابی و پشتیبان‌گیری مورد بحث و بررسی قرار گرفت.

۵-۱ عضویت در گروه pg_wheel

به هنگام نصب سرویس postgres، یک گروه کاربری با عنوان pg_wheel ایجاد می‌شود. عضویت در این گروه سبب می‌شود که کاربران عادی عضو آن بتوانند با استفاده از دستور su به یک ماشین دسترسی در سطح کاربران ارشد را داشته باشند. بنابراین تنها کاربرانی که مجاز به چنین سطحی از دسترسی هستند می‌بایست عضو این گروه بوده و ما بقی کاربران می‌بایست از آن حذف شوند.

تهدید/توجیه امنیتی:

در صورت وجود کاربران غیر مجاز در گروه کاربری pg_wheel، این امکان برای این دسته از کاربران وجود دارد که بتوانند همانند مالک پایگاه داده و مدیر پایگاه داده به اسکریپت‌ها، فایل‌ها و دیگر فایل‌های اجرایی دسترسی داشته باشند. از این رو ضروری است که از چنین دسترسی‌های غیرمجازی جلوگیری شود.

اطلاع از وضعیت فعلی:

برای اطلاع از کاربران فعلی که عضو گروه کاربری pg_whell هستند، از دستور زیر استفاده می‌کنیم.

```
getent group pg_wheel | awk -F":" '{print $4}' | \
awk '{split($1,a,","); for (i in a) print a[i]; }'
```

مقاوم‌سازی:

برای این منظور تمامی کاربران، به جز کاربر postgres را از گروه کاربری pg_wheel حذف می‌کنیم. لازم به ذکر است در صورتی که نام کاربری postgres در این گروه کاربری نباشد، می‌بایست آن را به این گروه کاربری اضافه کرد. برای حذف و اضافه نام کاربری به ترتیب از دستورات زیر استفاده می‌کنیم.

```
gpasswd -d [user name] pg_wheel
gpasswd -a [user name] pg_wheel
```

۵-۲ سطح اجرای سرویس postgres

سطح اجرا در واقع یک عدد صحیح تک رقمی از پیش تعیین شده‌ای است که حالت عملیاتی یک سیستم عامل لینوکس/یونیکس را تعریف می‌کند. هسته لینوکس استاندارد از هفت سطح مختلف پشتیبانی می‌کند که عباتند از:

- صفر برای توقف سیستم
- یک برای حالت تک کاربری
- دو برای چند کاربری بدون فایل سیستم NFS
- سه برای چند کاربری تحت خط فرمان
- چهار برای تعریف پذیر توسط کاربر
- پنج برای چند کاربری تحت محیط گرافیکی
- شش برای راه اندازی مجدد

تهدید/توجیه امنیتی:

با تنظیم سطح اجرایی مناسب برای سرویس postgres می‌توان همواره از اجرای آن اطمینان داشت و هر تغییر در سطح اجرای سیستم‌عامل موجب می‌شود سرویس postgres همچنان فعال باقی بماند.

اطلاع از وضعیت فعلی:

برای اطلاع از سطح اجرایی فعلی سیستم عامل از دستور زیر استفاده می‌کنیم.

```
runlevel | awk -F" " '{print $2}'
```

مقاوم‌سازی:

برای تنظیم سطح اجرایی سرویس postgres به سطح اجرایی سیستم عامل، از دستور زیر استفاده می‌کنیم.

```
update-rc.d -f postgresql enable [runlevel OS]
```

۵-۳ پیکربندی و نصب ابزار بازیابی و پشتیبان‌گیری

ابزار pgBackRest به عنوان یک ابزار ساده و قابل اعتماد برای بازیابی و پشتیبان‌گیری برای سرویس postgres ارایه شده است. ابزار pgBackRest به عنوان جایگزینی برای ابزارهای پشتیبان‌گیری قدیمی چون tar و rsync، تمامی ویژگی‌های مورد نیاز برای پشتیبان‌گیری امن را فراهم آورده است.

تهدید/توجیه امنیتی:

داشتن یک ابزار جامع برای بازیابی و پشتیبان‌گیری در سِرور postgres از اهمیت بالایی برقرار است. لازم به ذکر است که برخی از ابزارهای پشتیبان‌گیری همچون pg_dump یا pg_basebackup در سِرور postgres فراهم شده است اما هر کدام با هدف متفاوتی طراحی شده و نیاز به یک ابزار جامع با تنظیمات بیشتر وجود دارد.

اطلاع از وضعیت فعلی:

برای اطمینان از نصب بودن ابزار pgBackRest از مجموعه دستورات زیر استفاده می‌شود.

```
pgbackrest > /dev/null
if [ $? -eq 0 ]; then
echo "pgBackRest is installed"
else
echo "pgBackRest is not installed"
fi
```

مقاوم‌سازی:

برای نصب و پیکربندی ابزار pgBackRest از مجموعه دستورات زیر استفاده می‌شود.

```
find /usr/share/perl5/pgBackRest -type d -exec chmod 755 {} +
cp   pgbackrest-release-1.25/bin/pgbackrest /usr/bin/pgbackrest
chmod 755 /usr/bin/pgbackrest
mkdir -m 770 /var/log/pgbackrest
chown postgres:postgres /var/log/pgbackrest
touch /etc/pgbackrest.conf
chmod 640 /etc/pgbackrest.conf
chown postgres:postgres /etc/pgbackrest.conf
```

۵-۴ پیکربندی آرشیو رویدادها

به فرآیند ارسال فایل‌های رویداد مربوط به تراکنش‌ها از میزبان اصلی[1] به یک یا چند میزبان آماده به کار[2] یا یک منبع ذخیرسازی راه دور برای استفاده‌های آتی WAL[3] اطلاق می‌شود. راهکارهای زیادی همانند استفاده از cp، scp، sftp و rynsc برای نسخه‌برداری محتوای WALها استفاده می‌شود. در واقع سِرور از یک مجموعه پارامترهای زمان اجرا پیروی می‌کند که این پارامترها تعیین می‌کنند که چه هنگام WALها می‌بایست با استفاده از ابزارهای معرفی شده در بالا نسخه‌برداری شوند.

تهدید/توجیه امنیتی:

اگر سِرور به درستی پیکربندی نشده باشد، همواره ریسک ارسال WALها به صورت نا امن و رمز نشده وجود دارد.

اطلاع از وضعیت فعلی:

برای اطلاع از فعال بودن این پیکربندی و جزییات مربوط به تظیمات آن، از دو دستور زیر استفاده می‌کنیم. لازم به ذکر است که پارامترهای مذکور در فایل postgresql.conf قرار دارند.

```
grep "archive_mode = " $conf_path
grep "archive_command = " $conf_path
```

مقاوم‌سازی:

برای فعال کردن آرشیو رویدادها و پیکربندی امن آن از دستورات زیر استفاده می‌شود.

```
sed -i "s/\(.*\)archive_mode = *.*# \(.*\)/#archive_mode = on  \     #
    \2/" $conf_path


sed -i "s~\(.*\)archive_command = .*.# \(.*\)~archive_command = \
    'rsync \-e ssh \-a \%p postgres@remotehost:/var/lib/pgsql/WAL/%f'
            # \2~"  $conf_path
```

[1] Primary host
[2] Standby host
[3] Write Ahead Log

۵-۵ جمع‌بندی

در این فصل به تشریح برخی از پارامترهای امنیتی ویژه که در آخرین بروزرسانی اسناد استاندارد مقاوم‌سازی سمپاد postgres مطرح شده است، پرداختیم. در این راستا، تنظیمات مربوط به سطح اجرای سرویس postgres، پیکربندی آرشیو رویدادها و نصب افزونه‌های مهم جهت بازیابی و پشتیبان‌گیری 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
۵	**تنظیمات ویژه**		
۵-۱	عضویت در گروه کاربری pg_wheel	☐	☐
۵-۲	سطح اجرای سرویس postgres	☐	☐
۵-۳	پیکره‌بندی و نصب ابزار بازیابی و پشتیبان‌گیری	☐	☐
۵-۴	پیکره‌بندی آرشیو رویدادها	☐	☐

۶ راهنمای ابزار مقاوم‌سازی

این ابزار دارای چندین فایل اجرایی است که از میان آنها، سه فایل start.sh، script.sh و repair.sh اجزای اصلی ابزار را تشکیل داده است. دیگر فایل‌های موجود با هدف ساده کردن فایل‌های اصلی در نظر گرفته شده و توسط فایل‌های اصلی در روال مقاوم‌سازی فرخوانی می‌گردد. در ادامه هر یک از فایل‌های اصلی مورد بحث و بررسی قرار گرفته است.

۶-۱ فایل start.sh

این فایل اسکریپت، تنها فایلی است که کاربر باید آن را اجرا کند و بقیه اسکریپتها از درون این فایل فراخوانی و اجرا می‌شوند. برای آنکه فرآیند تست پایگاه داده و امن‌سازی آن صورت گیرد، ابتدا لازم است از وجود PostgreSQL روی سیستم اطمینان حاصل گردد در صورتی که PostgreSQL روی سیستم نصب بوده و فایل‌های مربوط به تنظیمات موجود باشند، اسکریپت script.sh اجرا می‌شود. در اثر اجرای این فایل، دو پوشه حاوی نتایج آزمایش و نتایج مورد انتظار ایجاد می‌شوند. حال 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. نمونه ای از خروجی این برنامه در شکل (۱) نشان داده شده است.

```
 GNU nano 2.7.4                                    File: first_test_result

     "POSTGRESQL RESULT"                            "EXPECTED RESULT"

<------1-1-Configuration File Access Rights ------>  <------1-1-Configuration File Access Rights ------>
-rw-r--r--                                           -rw-r--r--



<------1-2-Data storage folder Access Rights ------>  <------1-2-Data storage folder Access Rights ------>
total                                              | drwx------
drwxr-xr-x 9.5                                     <



<------1-3-Event log file Access Rights ------>      <------1-3-Event log file Access Rights ------>
total                                              | -rw-r-----
-rw-r----- postgresql-9.5-main.log                 <
-rw-r----- postgresql-9.5-main.log.1               <



<------2-1-Listen Addresses Parameter ------>       <------2-1-Listen Addresses Parameter ------>
#listen_addresses = 'localhost'                    | listen_addresses = 'localhost'
                                                   > OR
                                                   > listen_addresses = 'trusted ip'



<------2-2-Port Parameter ------>                   <------2-2-Port Parameter ------>
port = 5432                                        | port = '<Appropriate value>'



<------2-3-Max Connections Parameter ------>        <------2-3-Max Connections Parameter ------>
```

شکل ۱: محتوای فایل first_test_result

ستون سمت چپ نشان دهنده تنظیمات فعلی است. مواردی که با رنگ قرمز نشان داده شده است، مواردی هستند که مقابل آنها نتیجه مورد انتظار با نتیجه حاصل از آزمایش مغایرت داشته 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 تست مغایر باشند، با موافقت کاربر امن سازی شده و در آخر نیز تست دوباره‌ای روی سیستم انجام می‌شود. نتیجه تست دوم در فایل second_test_result ذخیره خواهد شد.

```
 GNU nano 2.7.4                                          File: second_test_result

     "POSTGRESQL RESULT"                                  "EXPECTED RESULT"

<------1-1-Configuration File Access Rights ------>      <------1-1-Configuration File Access Rights ------>
-rw-r--r--                                               -rw-r--r--



<------1-2-Data storage folder Access Rights ------>      <------1-2-Data storage folder Access Rights ------>
total                                                  | drwx------
drwxr-xr-x 9.5                                         <



<------1-3-Event log file Access Rights ------>          <------1-3-Event log file Access Rights ------>
total                                                  | -rw-r-----
-rw-r----- postgresql-9.5-main.log                     <
-rw-r----- postgresql-9.5-main.log.1                   <



<------2-1-Listen Addresses Parameter ------>           <------2-1-Listen Addresses Parameter ------>
#listen_addresses = 'localhost'                        | listen_addresses = 'localhost'
                                                       > OR
                                                       > listen_addresses = 'trusted ip'



<------2-2-Port Parameter ------>                       <------2-2-Port Parameter ------>
port = 5432                                            | port = '<Appropriate value>'



<------2-3-Max Connections Parameter ------>            <------2-3-Max Connections Parameter ------>
```

شکل ۲: محتوای فایل second_file_result

۶-۲ 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، ساخته می‌شود. 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.

۶-۳ فایل repair.sh

فایل آخر مربوط به تغییر تنظیمات سیستم می‌شود. 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 نمونه ای از فایل second_test-result را پس از اعمال تغییرات مشاهده کرد.

۷ جمع‌بندی

در این مستند به بررسی موارد امنیتی مربوط به مقاوم‌سازی PostgreSQL پرداخته شد. تنظیمات مربوط به مقاوم‌سازی PostgreSQL در پنج بخش مختلف دسته بندی شدند. در بخش اول، امن‌سازی محیط اجرا، بخش دوم نصب و پیکربندی امن پایگاه‌داده، بخش سوم امن‌سازی اتصال به پایگاه‌داده، بخش چهارم تنظیمات رویداد نگاری و بخش پنجم برخی از تنظیمات ویژه 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 ارایه شده، به صورت یک چک لیست در ادامه آورده شده است.

	عنوان	تنظیم صحیح	
		بله	خیر
۱	**امن سازی محیط اجرا**		
۱-۱	پیکربندی فایل تنظیمات	☐	☐
۱-۲	پیکربندی دایرکتوری ذخیره داده	☐	☐
۱-۳	پیکربندی اجرای دستور از طریق خط فرمان توسط کاربر postgres	☐	☐
۱-۴	راه اندازی سرویس postgres توسط یک کاربر عادی	☐	☐
۱-۵	پیکربندی فایل رویدادنگاری	☐	☐
۲	**پیکربندی امن پایگاه داده**		
۲-۱	پارامتر listen_addresses	☐	☐
۲-۲	پارامتر port	☐	☐
۲-۳	پارامتر max_connections	☐	☐
۲-۴	پارامتر unix_socket_permissions	☐	☐
۲-۵	پارامتر authentication_timeout	☐	☐
۲-۶	پارامتر password_encryption	☐	☐
۲-۷	پارامتر ssl	☐	☐
۳	**امن‌سازی اتصال به پایگاه داده**		
۳-۱	ارتباط امن	☐	☐
۳-۲	ارتباط از راه دور	☐	☐

۴	**تنظیمات رویدادنگاری**		
۴-۱	پارامتر logging_collector	☐	☐
۴-۲	پارامتر log_directory	☐	☐
۴-۳	پارامتر log_connections	☐	☐
۴-۴	پارامتر log_disconnections	☐	☐
۴-۵	پارامتر log_hostname	☐	☐
۵	**تنظیمات ویژه**		
۵-۱	عضویت در گروه کاربری pg_wheel	☐	☐
۵-۲	سطح اجرای سرویس postgres	☐	☐
۵-۳	پیکره‌بندی و نصب ابزار بازیابی و پشتیبان‌گیری	☐	☐
۵-۴	پیکره‌بندی آرشیو رویدادها	☐	☐

۸ مراجع

[۱] https://www.cisecurity.org/