Postgres Enterprise Manager

Release 8.0

PEM Security Guide

Dec 08, 2020

Contents

This document provides information about security practices you should consider when configuring PEM. PEM functionality does not require you to enforce these practices; however, EDB recommends these practices to enhance the overall system's security.

PEM is dependent on third-party components from the vendor repository, including the Apache web server, OpenSSL, snmp++, libcurl, etc. To ensure these components are up to date, you should update your operating system and regularly apply security updates to avoid any security vulnerability. Without the most recent security patches, your system is potentially vulnerable to cyberattacks. Security patches protect your devices and their data by applying the latest updates to guard against the latest threats.

Some of the benefits of regularly applying security patches include:

- Reduced exposure to cyberattacks
- Avoiding lost productivity
- Data protection from malware (like ransomware)
- Avoid worm infections that use security loopholes to spread over the network

CHAPTER 1

Apache HTTPD Security Configurations

On a Windows system, Apache HTTPD is named PEM HTTPD, and the Apache httpd configuration file (`pme.conf`) is located in the `<Apache_Installation_Path>/conf/addons` directory. The ssl configuration file (`httpd-ssl-pem.conf`) is located in the `<Apache_Installation_Path>/conf/addons` directory.

On a Linux system, the apache httpd configuration file (`edb-pem.conf`) is located in the `<Apache_Installation_Path>/conf.d` directory and the SSL configuration file (`edb-ssl-pem.conf`) is located in the `<Apache_Installation_Path>/conf.d` directory.

1.1 Disable SSLv2 and SSLv3

You should disable SSL versions SSLv2, SSLv3, TLS 1, and TLS 1.1 as these versions are the most vulnerable and are affected by cryptographic concerns.

To disable the versions, please add the following command to apache httpd configuration file:

```
SSLProtocol -ALL +TLSv1.2
```

You need to restart the Web Server to apply the changes to the configuration file.

By default, PEM adds the following lines to the SSL configuration file to allow use of TLS 1.2 for security:

```
SSLProtocol -All TLSv1.2

SSLProxyProtocol -All TLSv1.2
```

1.2 Secure httpd with SSL Certificates

EDB recommends having an additional layer of SSL security for the web application.

By default, during PEM installation, PEM will generate and use self-signed certificates for Apache/HTTPD server. If you want to use your own SSL certificate for PEM, you must update the Apache configuration file.

On a Linux system, you need to open the Apache httpd configuration file (`edb-ssl-pem.conf`) in a text editor as a user with write permission on the file. You must also change the server name and file names in the configuration file to match your certificate files.

There are two SSL Directives within the PEM VirtualHost section which you need to update:

- `SSLCertificateFile` is your DigiCert certificate file (e.g.,your_domain_name.crt).
- `SSLCertificateKeyFile` is the .key file generated when you created the CSR (e.g., your_private.key).

For example, update:

```
SSLEngine on

SSLCertificateFile /path/to/your_domain_name.crt

SSLCertificateKeyFile /path/to/your_private.key
```

You can also replace the httpd self-signed SSL certificates with trusted CA-signed certificates in Postgres Enterprise Manager. Follow the link shown below for the steps:

https://www.enterprisedb.com/postgres-tutorials/how-replacing-httpd-self-signed-ssl-certificates-trusted-ca-sign

1.3 Disable Web Server Information Exposure

EDB recommends disabling all web server signatures as part of web server security. The web server will expose a software signature; to disable the signature, add the following parameters to the Apache httpd configuration file. By default, PEM disables exposure of the information by adding the below parameters to the Apache httpd configuration file:

```
ServerTokens Prod

ServerSignature Off
```

The `ServerTokens` directive controls the server response header field, which is returned to the client. We recommend hiding the Apache server version by adding this parameter in the Apache httpd configuration file.

The `ServerSignature` directive includes a footer for server-produced documents. The footer contains information regarding the Apache configuration, like the Apache and operating system

version. To limit the display of such information, we recommend disabling this directive in the Apache httpd configuration file.

You need to restart the web server to apply any changes to the Apache httpd configuration file.

1.4 Disable Directory Listing

The directory listing can allow an attacker to view complete directory contents. By default, the web server enables this option, and an attacker can discover and view any file. This listing could lead to the attacker reverse engineering an application to obtain the source code, analyze it for possible security flaws, and discover more information about an application.

To avoid this, you should disable the directory listing by setting the `Options` directive in the Apache httpd configuration file. By default, PEM disables the directory listing by setting the option below in the web server configuration file:

```
<Directory /application/directory> Options -Indexes </
Directory>
```

You need to restart the web server to apply the changes made to the configuration file.

1.5 Restrict the Access to a Network or IP Address

Apache provides access control based on the client hostname or IP address. To view the application by specific IP address or network, a user can modify the Apache configuration file as shown below to provide your network address within the `Allow` directive:

```
<Directory /application/hostname>

Options None

AllowOverride None

Order deny,allow

Deny from all

Allow from 192.168.0.0/24

    </Directory>
```

PEM uses the `ALLOWED_HOSTS` configuration parameter in the application configuration file to provide the allowed hosts' list of IP addresses. The application configuration `config_local.py` file is located in `<PEM_INSTALLATION_PATH>/web`.

By default, PEM allows all the hosts to connect with the application.

For example:

You can set the range of IP addresses in the configuration file as below:

```
ALLOWED_HOSTS = ['225.0.0.0/8', '226.0.0.0/7', '228.0.0.
0/6']
```

You can set the IP adresses to allow a host on a subnet level in the configuration file as below:

```
ALLOWED_HOSTS = ['192.0.2.0/28', '::192.0.2.0/124']
```

You can set a specific individual host address (based on the IP address) in the configuration file as below:

```
ALLOWED_HOSTS = ['127.0.0.1', '192.168.0.1']
```

You need to restart the web server to apply the changes to the application configuration file.

1.6 Cross-site Tracing

There are two HTTP methods to debug the web server connections - TRACE and TRACK. When an HTTP TRACE request is sent to a web server that supports it, that server will respond, echoing the data passed to it, including any HTTP headers. We recommend disabling these methods within the Apache Configuration.

To disable the TRACE method for all virtual hosts, add the following line to the Apache httpd configuration file:

```
TraceEnable off
```

To disable these methods for a specific virtual host, add the following lines for each virtual host in the Apache configuration file. PEM does add the following lines to the Apache httpd configuration file:

```
RewriteEngine on

RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK|OPTIONS)

RewriteRule .\* - [F]
```

1.7 Run Web Server from a Non-privileged User Account

Running the Apache web server as a root user creates a security issue. We always recommend running the web server as a unique non-privileged user. This helps to secure other services running in the event of any security breaches.

PEM runs as a WSGI application. To delegate the WSGI applications that are running, you can create distinct daemon processes using the `WSGIDaemonProcess` directive.

On Linux, the Apache web server starts as the root user, but the daemon processes which PEM application runs as `pem` user. On Windows, the `WSGIDaemonProcess` directive and its features are not available. PEM HTTPD is installed as a service during the installation, and the user needs to be a member of the `Administrators` group for the service installation to succeed.

By default the Apache services are registered to run as the system user (the `LocalSystem` account).

1.8 Customize Security HTTP Headers in PEM WebServer

PEM contains its own configuration file to fix the following security issues. We recommend overriding the configuration only of `config_local.py` and not of `config.py`. The `config_local.py` is not present on the systems in most of the cases; hence users need to create it to override the application-level configurations. Please note that during a PEM upgrade, `config_local.py` will not be overwritten, but changes in `config.py` and `config_distro.py` will be overridden. Users need to remove `config_local.py` after uninstalling the PEM.

By default, `config_local.py` is located in `/usr/edb/pem/web` on Linux, and at `C:\ProgramFiles\edb\pem\server\share\web` on Windows.

1.9 Host Header Injection Attacks

HTTP host header attacks exploit vulnerable websites that handle the host header value in an unsafe way. If the server implicitly trusts the host header and fails to validate or escape it properly, an attacker may be able to use this input to inject harmful payloads that manipulate server-side behavior. The web applications typically don't know what domain they are deployed on unless specified in a configuration file during setup. When they need to know the current domain, for example, they may resort to retrieving the domain from the host header to generate an absolute URL. The host header is a potential vector for exploiting a range of other vulnerabilities, most notably web cache poisoning & SQL injections.

1.9.1 X-Frame-Options

X-Frame-Options can be used to indicate if a browser should be allowed to render a page in an <iframe> tag. It was designed specifically to protect against clickjacking. PEM has a host validation `X_FRAME_OPTIONS` option to prevent such kind of attacks, which you can configure in the `config_local.py` file. The default is:

```
X_FRAME_OPTIONS = "SAMEORIGIN"
```

1.9.2 Content-Security-Policy

Content-Security-Policy is part of the HTML5 standard and provides a broader range of protection than the X-Frame-Options header (which it replaces). It is designed in such a way that website authors can whitelist individual domains from which resources (like scripts, stylesheets, and fonts) can be loaded, and also domains that are permitted to embed a page.

PEM has a host validation `CONTENT_SECURITY_POLICY` option to prevent such kinds of attacks, which you can configure in the `config_local.py` file. The default is:

```
CONTENT_SECURITY_POLICY = "default-src https:  data:
blob:  'unsafe-inline' ''unsafe-eval';"
```

1.9.3 Strict-Transport-Security

The Strict-Transport-Security response header (often abbreviated as HSTS) allows a web site or web application to tell browsers that it should only be accessed using HTTPS instead of HTTP. This option allows you to prevent a man-in-the-middle attack. The default is:

```
STRICT_TRANSPORT_SECURITY = "max-age=31536000;
includeSubDomains"
```

Note: Adding this parameter may cause problems if config is changed. Hence it is recommended to add this only after PEM installation is complete and tested.

1.9.4 X-Content-Type-Options

The X-Content-Type-Options response HTTP header is a marker used by the server to indicate that the MIME types advertised in Content-Type headers should not be changed and followed. This is a way to opt out of MIME type sniffing, or, in other words, to say that the MIME types are deliberately configured. The default is:

```
X_CONTENT_TYPE_OPTIONS = "nosniff"
```

1.9.5 X-XSS-Protection

Cross-site scripting (XSS) is one of the most common application-layer vulnerabilities in the web servers. XSS enables attackers to inject client-side script into web pages viewed by other users. The HTTP X-XSS-Protection response to the header is a feature of Internet Explorer, Chrome, and Safari that stops pages from loading when they detect reflected cross-site scripting (XSS) attacks. Although these protections are largely unnecessary in modern browsers when sites implement a

strong Content-Security-Policy that disables the use of inline JavaScript ('unsafe-inline') can still provide protections for users of older web browsers that don't yet support CSP. The default is:

```
X_XSS_PROTECTION = "1; mode=block"
```

To avoid this, you need to add the following options to the Apache configuration file:

```
<IfModule mod_headers.c>

Header set X-XSS-Protection "1; mode=block"

</IfModule>
```

A web server restart is required for configuration file changes to be applied.

By default, PEM sets `X-XSS-Protection to "1; mode=block"` in the application configuration file which is located at `/usr/edb/pem/web/config.py`.

This changes requires an Apache service restart in order to take effect

For more detailed information on `config.py` file you can see PEM Online Help.

1.9.6 Cookie Security

Cookies are small packets of data that a server can send to your browser to store configuration data. The browser automatically sends them along with all requests to the same server, so it's important to know how to secure cookies. There are multiple configuration options provided by PEM to make cookies secure which you can refer to in `config.py` but the three that follow are most important.

(a) **SESSION_COOKIE_SECURE** - Setting the secure flag prevents the cookie from ever being sent over an unencrypted connection. It basically tells the browser to never add the cookie to any request to the server that does not use an encrypted channel. The browser will only add cookies to connections such as HTTPS. The default is:

```
SESSION_COOKIE_SECURE = True
```

(b) **SESSION_COOKIE_HTTPONLY** - By default the content of cookies can be read via JavaScript. The `HTTPOnly` flag prevents scripts from reading the cookie. As the name `HTTPOnly` implies, the browser will only use the cookie in HTTP(S) requests. This prevents hackers from using XSS vulnerabilities to learn the contents of the cookie. For example, for the `sessionId` cookie it is never necessary to read the cookie with a client-side script, so for `sessionId` cookies, you can always set the `HTTPOnly` flag. The default is:

```
SESSION_COOKIE_HTTPONLY = True
```

(c) **ENHANCED_COOKIE_PROTECTION -** When you set this option to `True` then a token will be generated according to the IP address and user agent. In all subsequent requests, the token will be recalculated and checked against the one computed for the first request. If the session cookie is stolen and the attacker tries to use it from another location, the generated

token will be different, and in that case, the extension will clear the session and block the request. The default is:

```
ENHANCED_COOKIE_PROTECTION = True
```

Note: This option can cause problems when the server is deployed in dynamic IP address hosting environments, such as Kubernetes or behind load balancers. In such cases, this option should be set to `False`.

This changes requires an Apache service restart in order to take effect.

For more detailed information on `config.py` file you can see PEM Online Help.

CHAPTER 2

PEM application Security Configurations

2.1 Session Timeout

Insufficient session expiration by the web application increases the exposure of other session-based attacks, as it allows time for the attacker to be able to reuse a valid session ID and hijack the associated session. The shorter the session interval is, the lesser the time an attacker has to use the valid session ID. We recommend setting the inactivity timeout for the web application to a low value to avoid this security issue.

Postgres Enterprise Manager provides a way to set the timeout value for a user session. When there is no user activity for a specified duration on the web console, PEM will log out the user from the web console. A PEM Administrator can set the length of time for inactivity. This value is application-wise, rather than for an individual user. To configure the timeout duration, modify the `USER_INACTIVITY_TIMEOUT` parameter in `config_local.py` file, located in the `<PEM_INSTALLATION_PATH>/web` directory. By default this functionality is disabled.

For example, to specify that an application should log out a user after 15 minutes of inactivity, set:

```
USER_INACTIVITY_TIMEOUT = 900
```

Note: The timeout value is specified in seconds.

This changes requires an Apache service restart in order to take effect.

For more detailed information on `config.py` file you can see PEM Online Help.

2.2 RestAPI Header Customization

You can customize the RestAPI token headers as per your requirements. The default values are not exposed by the `config.py` file; customize the following headers in the `config_local.py` file:

2.2.1 PEM_HEADER_SUBJECT_TOKEN_KEY

This configuration option will allow you to change the HTTP header name to get the generated token. By default, when the user sends a request to create a token, the server response will have an 'X-Subject-Token' header, which will contain the value of a newly generated token. If you want to customize the header name, then you can update the `config_local.py` file as shown:

```
PEM_HEADER_SUBJECT_TOKEN_KEY = 'Pem-RestAPI-Generate-Token'
```

Which will give you the following output:

```
curl -ik -X POST -d '{"username":"enterprisedb",
"password":"edb"}' -H "Content-Type: application/json"
https://localhost:8443/pem/api/token/

HTTP/1.1 201 CREATED

Date: Thu, 29 Oct 2020 11:03:48 GMT

Server: Apache

Content-Length: 326

Pem-RestAPI-Generate-Token: 997aef95-d46d-4d84-932a-a80146eaf84f
```

2.2.2 PEM_HEADER_TOKEN_KEY

This configuration option will allow you to change the HTTP request header name through which you will send the token to the PEM server. By default, when the user sends a request to generate a token, the token header will be `X-Auth-Token`. If you want to customize the RestAPI request header name then you can update the `config_local.py` file as follows:

```
PEM_HEADER_TOKEN_KEY = 'Pem-Token'
```

Which will allow you to send the token:

```
$ curl -Lk -X GET -H "Pem-Token: gw5rzaloxydp91ttd1c97w24b5sv60clic24s
https://localhost:8443/pem/api/v4/agent
```

2.2.3 PEM_TOKEN_EXPIRY

This configuration option will allow you to change the PEM RestAPI token expiry time after it is generated. By default, the token expiry time is set to 20 minutes (1200 seconds). If you want to change the token expiry time to 10 minutes then you can update the `config_local.py` file as shown:

```
PEM_TOKEN_EXPIRY = 600
```

This changes requires an Apache service restart in order to take effect.

2.3 Role-based Access Control in PEM

Role-based access control (RBAC) restricts application access based on a user's role within an organization and is one of the primary methods for access control. The roles in RBAC refer to the levels of access that users have to the application. Users are only allowed to access the information necessary to effectively perform their job duties. Roles in PEM are inheritable and additive rather than subscriptive. In simple terms, as a PEM admin you need to grant the lowest level role to the user and then grant the roles which are required for the user to perform their respective tasks. For example, to give access only to SQL profiler:

```
CREATE ROLE user_sql_profiler WITH LOGIN NOSUPERUSER
NOCREATEDB NOCREATEROLE INHERIT NOREPLICATION CONNECTION
LIMIT -1 PASSWORD 'xxxxxx';

GRANT pem_user, pem_comp_sqlprofiler TO
user_sql_profiler;
```

For more detailed information on roles, you can see PEM Roles.

2.4 SQL/Protect Plugin

Preventing a SQL injection attack is usually the responsibility of the application developer. The database administrator typically has little or no control over the potential threat. The difficulty for database administrators is that the application must have access to the data to function properly.

`SQL/Protect` is a module that allows a database administrator to protect a database from SQL injection attacks. SQL/Protect provides a layer of security in addition to the standard database security policies by examining incoming queries for typical SQL injection profiles.

There are several different techniques used to perpetrate SQL injection attacks. A specific signature characterizes each technique. SQL/Protect examines queries for Unauthorized Relations, Utility Commands, SQL Tautology, Unbounded DML Statements. SQL/Protect gives the control back to the database administrator by alerting the administrator to potentially dangerous queries and blocking them.

Note: This plugin works only on the EPAS server, so this is useful only when your PEM database is hosted on the EPAS server.

For more detailed information about the SQL Profiler plugin, see the PEM Online Help - SQL Profiler.

2.5 Password Management

One security tip for PEM administrative users is to change your PEM login passwords to something new regularly. Changing your password avoids a number of dangers including:

- breaches of multiple accounts
- prevents constant access
- prevents the use of saved passwords on a physically unsecured system
- limits access gained by keystroke loggers

2.6 Run pemAgent Jobs with a Non-root User

In most cases, `pemAgent` is installed as a root user, and runs as a daemon process with root privileges. By default, PEM disables running the scheduled jobs/task. PEM provides support for running scheduled jobs as a non-root user by changing the pemAgent configuration file.

To run scheduled jobs as a non-root user, modify the entry for the `batch_script_user` parameter in the `agent.cfg` file and specify the user that should be used to run the script. You can either specify a non-root user or root user identity. If you do not specify a user, or the specified user does not exist, then the script will not execute. Restart the agent after modifying the file. If a non-root user is running `pemagent`, then the value of `batch_script_user` will be ignored, and the same non-root user used for running the `pemagent` will execute the script.

To invoke a script on a Windows system, set the registry entry for `AllowBatchJobSteps` to `true` and restart the PEM agent. PEM registry entries are located in:

```
HKEY_LOCAL_MACHINE\Software\Wow6432Node\EnterpriseDB\PEM\
agent
```

2.7 Changing the pemAgent and PEM Backend Database Server Certificates

By default, when you install PEM, the installer will generate and use self signed certificates for the pemAgent and PEM database server. PemAgent uses these certificates when connecting to the PEM database server. To use your own SSL certificate for the pemAgent and PEM database server, follow the steps mentioned in the `Managing Certificates` section of the `PEM Administrators's Guide`:

https://www.enterprisedb.com/edb-docs/d/edb-postgres-enterprise-manager/user-guides/administrators-guide/7.16/managing_certificates.html<Here we need to provide the link to managing_certificates.html file>.

Note: PEM does not support placing the SSL CA certificates at a custom location; you should not change the location of `ca_certificate.crt` and `ca_key.key`.

CHAPTER 3

Conclusion

Postgres Enterprise Manager Security Guide

EnterpriseDB® Corporation 34 Crosby Drive, Suite 201, Bedford, MA 01730, USA

T +1 781 357 3390 F +1 978 467 1307 E info@enterprisedb.com www.enterprisedb.com

- EDB designs, establishes coding best practices, reviews, and verifies input validation for the logon UI for Postgres Enterprise Manager where present. EDB follows the same approach for additional input components, however the nature of the product may require that it accepts freeform SQL, WMI or other strings to be entered and submitted by trusted users for which limited validation is possible. In such cases it is not possible to prevent users from entering incorrect or otherwise dangerous inputs.
- EDB reserves the right to add features to products that accept freeform SQL, WMI or other potentially dangerous inputs from authenticated, trusted users in the future, but will ensure all such features are designed and tested to ensure they provide the minimum possible risk, and where possible, require superuser or equivalent privileges.
- EDB does not that warrant that we can or will anticipate all potential threats and therefore our process cannot fully guarantee that all potential vulnerabilities have been addressed or considered.

Index