Fortify Security Report

2024-6-21

ASUS

Executive Summary

Issues Overview

On 2024-6-21, a source code review was performed over the PolarDB-for-PostgreSQL code base. 1,500 files, 15,991 LOC (Executable) were scanned and reviewed for defects that could lead to potential security vulnerabilities. A total of 4 reviewed findings were uncovered during the analysis.

Issues by Fortify Priority Order	
Critical	3
High	1

Recommendations and Conclusions

The Issues Category section provides Fortify recommendations for addressing issues at a generic level. The recommendations for specific fixes can be extrapolated from those generic recommendations by the development group.

Project Summary

Code Base Summary

Code location: C:/Users/ASUS/Desktop/Gitrepo/PolarDB-for-PostgreSQL
Number of Files: 1500
Lines of Code: 15991
Build Label: <No Build Label>

Scan Information

Scan time: 01:20
SCA Engine version: 20.1.1.0007
Machine Name: DESKTOP-MK5UPFE
Username running scan: ASUS

Results Certification

Results Certification Valid

Details:

Results Signature:

SCA Analysis Results has Valid signature

Rules Signature:

There were no custom rules used in this scan

Attack Surface

Attack Surface:
Command Line Arguments:
null.null.null

Environment Variables:
null.null.null

File System:
null.null.open
null.file.read
null.file.readline
null.file.readlines

Private Information:
null.null.null

System Information:
null.null.null
os.null.getcwd
os.null.sysconf

Filter Set Summary

Current Enabled Filter Set:
Quick View

Filter Set Details:

Folder Filters:
If [fortify priority order] contains critical Then set folder to Critical
If [fortify priority order] contains high Then set folder to High
If [fortify priority order] contains medium Then set folder to Medium
If [fortify priority order] contains low Then set folder to Low
Visibility Filters:
If impact is not in range [2.5, 5.0] Then hide issue
If likelihood is not in range (1.0, 5.0] Then hide issue

Audit Guide Summary

J2EE Bad Practices

Hide warnings about J2EE bad practices.
Depending on whether your application is a J2EE application, J2EE bad practice warnings may or may not apply. AuditGuide can hide J2EE bad practice warnings.
Enable if J2EE bad practice warnings do not apply to your application because it is not a J2EE application.

Filters:
If category contains j2ee Then hide issue
If category is race condition: static database connection Then hide issue

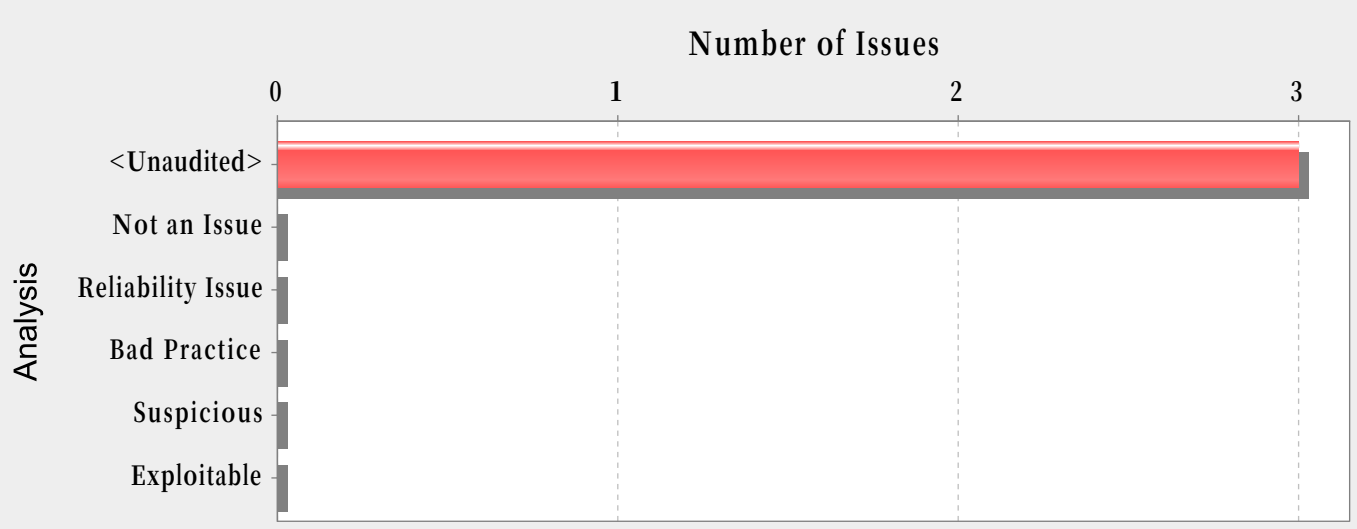
Results Outline

Overall number of results

The scan found 4 issues.

Vulnerability Examples by Category

Category: Password Management: Hardcoded Password (3 Issues)



Abstract:

Hardcoded password 可能会削弱系统安全性，并且无法轻易修正出现的安全问题。

Explanation:

使用硬编码方式处理密码绝非好方法。这不仅是因为所有项目开发人员都可以使用通过硬编码方式处理的密码，而且还会使解决这一问题变得极其困难。在代码投入使用之后，除非对软件进行修补，否则将无法更改密码。如果受密码保护的帐户遭受入侵，系统所有者将必须在安全性和可用性之间做出选择。

示例：以下代码对密码进行了硬编码：

```
DECLARE
pwd VARCHAR(20);
BEGIN
pwd := "tiger";
END;
```

该代码可以正常运行，但是有权访问该代码的任何人都能得到这个密码。一旦程序发布，除非修补该程序，否则可能无法更改密码“ tiger ”。雇员可以利用手中掌握的信息访问权限入侵系统。更糟的是，如果攻击者能够访问应用程序的二进制码，他们就可以利用多种常用的反编译器来访问经过反汇编的代码，而在这些代码中恰恰包含着用户使用过的密码值。

Recommendations:

绝不能对密码进行硬编码。通常情况下，应对密码加以模糊化，并在外部资源文件中进行管理。在系统中采用明文的形式存储密码，会造成任何有充分权限的人读取和无意中误用密码。

Tips:

1. 在识别 null 密码、空密码和硬编码密码时，默认规则只会考虑包含 password 一词的字段和变量。但是，使用 Fortify Custom Rules Editor（Fortify 自定义规则编辑器）提供的“ Password Management（密码管理） ”向导可轻松创建用于在自定义命名字段和变量中检测 password management 问题的规则。

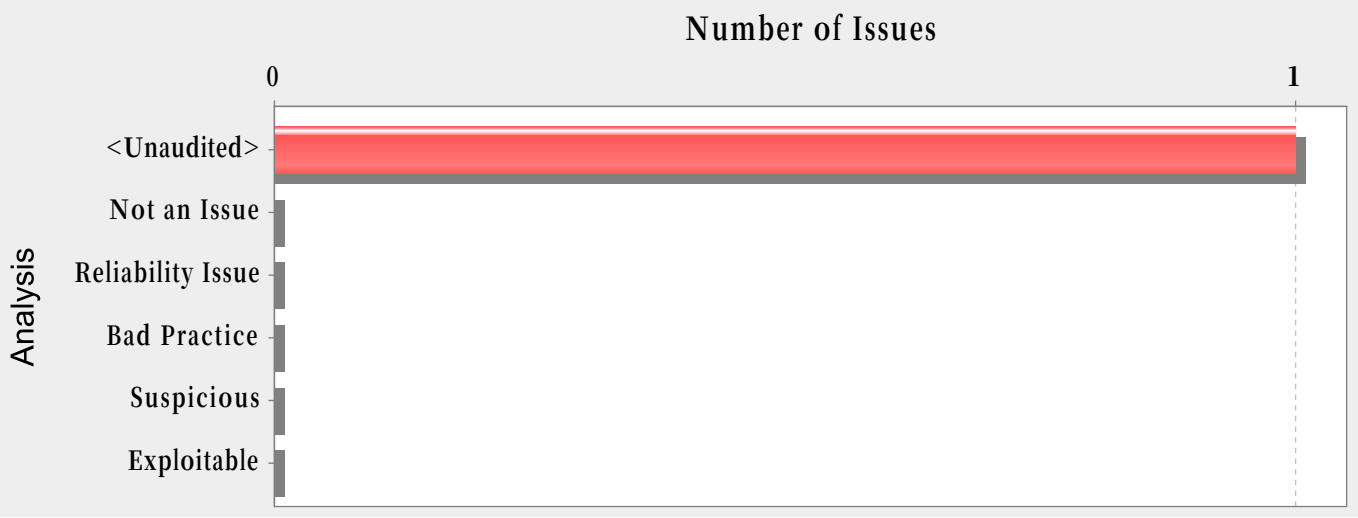
system_views.sql, line 27 (Password Management: Hardcoded Password)

Fortify Priority:	High	**Folder**	High
Kingdom:	Security Features		
Abstract:	Hardcoded password 可能会削弱系统安全性，并且无法轻易修正出现的安全问题。		
Sink:	system_views.sql:27 VariableAccess: rolpassword()		

```
25      rolreplication,
26      rolconnlimit,
27      '********'::text as rolpassword,
```

28	rolvaliduntil,
29	rolbypassrls,

Category: Privacy Violation: iOS Property List (1 Issues)



Abstract:

org.postgresql.postgres.plist 中的数据可能表示 12 行上不受保护的 iOS 属性列表中的私人信息。

Explanation:

当用户私人信息存储在一个不受保护的位置时，则会发生 Privacy Violation。

示例 1：下列 XML 将某个用户的私人信息存储在一个 plist 文件中。除存储的其他值外，MyCreditCard 键还会存储由用户提供的、与该帐户相关的纯文本信用卡卡号。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>password</key>
<string>BASICSECRET</string>
<key>credentials</key>
<dict>
<key>pin</key>
<string>2345</string>
<key>MyCreditCard</key>
<string>1111 11 2321 1112</string>
<key>MysSn</key>
<string>1111-22-3333</string>
<key>ssn</key>
<string>2345-22-3345</string>
<key>userid</key>
<string>12345</string>
</dict>
</dict>
</plist>
```

Example 1 中的代码会将移动设备中的私人信息存储在该设备上的一个不受支持的 plist 文件中。虽然许多开发人员认为 plist 文件是存储所有数据的安全位置，但这不是绝对的，特别是涉及到隐私问题时，因为 plist 文件可被持有该设备的任何人读取。

可以通过多种方式将私人数据输入到程序中：

— 以密码或个人信息的形式直接从用户处获取。

— 由应用程序访问数据库或者其他数据存储形式。

— 间接地从合作者或者第三方处获取。

- 从移动数据存储中检索如下信息：地址簿、拍摄的照片、地理位置、配置文件（包括 plist）、存档的 SMS 消息等。

有时，某些数据并没有贴上私人数据标签，但在特定的上下文中也有可能成为私人信息。比如，通常认为学生的学号不是私人信息，因为学号中并没有明确而公开的信息用以定位特定学生的个人信息。但是，如果学校用学生的社会保障号码生成学号，那么这时学号就应被视为私人信息。

安全和隐私似乎一直是一对矛盾。从安全的角度看，您应该记录所有重要的操作，以便日后可以鉴定那些非法的操作。然而，当其中牵涉到私人数据时，这种做法就会带来额外的风险。

虽然私人数据处理不当的方式多种多样，但常见风险来自于盲目信任。程序员通常会信任运行程序的操作环境，因此认为可以将私人信息存放在文件系统、注册表、plist 内部或其他本地控制的资源中。然而，尽管限制了对某些资源的访问权限，也仍无法保证所有可访问这些资源的个人都是可信赖的。例如，2004 年，一个不道德的 AOL 员工将大约 9200 万个私有客户电子邮件地址卖给了一个通过垃圾邮件进行营销的境外赌博网站 [1]。

鉴于此类备受瞩目的信息盗取事件，私人信息的收集与管理正日益规范化。要求各个组织应根据其经营地点、所从事的业务类型及其处理的私人数据性质，遵守下列一个或若干个联邦和州的规定：

- Safe Harbor Privacy Framework [3]

- Gramm-Leach Bliley Act (GLBA) [4]

- Health Insurance Portability and Accountability Act (HIPAA) [5]

- California SB-1386 [6]

尽管制定了这些规范，Privacy Violation 漏洞仍时有发生。

Recommendations:

当安全和隐私的需要发生矛盾时，通常应优先考虑隐私的需要。为满足这一要求，同时又保证信息安全的需要，应在退出程序前清除所有私人信息。

为加强隐私信息的管理，应不断改进保护内部隐私的原则，并严格地加以执行。这一原则应具体说明应用程序应该如何处理各种私人数据。在贵组织受到联邦或者州法律的制约时，应确保您的隐私保护原则尽量与这些法律法规保持一致。即使没有针对贵组织的相应法规，您也应当保护好客户的私人信息，以免失去客户的信任。

保护私人数据的最好做法就是最大程度地减少私人数据的暴露。不应允许应用程序、流程处理以及员工访问任何私人数据，除非是出于职责以内的工作需要。正如最小授权原则一样，不应该授予访问者超出其需求的权限，对私人数据的访问权限应严格限制在尽可能小的范围内。

Tips:

1. 要彻底审计所有的 Privacy Violation 漏洞，措施之一就是确保自定义的规则可以识别所有进入程序的私人或敏感信息。无法自动识别多数私人数据信息。若不使用自定义规则，您执行的 Privacy Violation 漏洞检查可能是不完整的。

org.postgresql.postgres.plist, line 12 (Privacy Violation: iOS Property List)

Fortify Priority:	Critical	**Folder**	Critical
Kingdom:	Security Features		
Abstract:	org.postgresql.postgres.plist 中的数据可能表示 12 行上不受保护的 iOS 属性列表中的私人信息。		
Sink:	org.postgresql.postgres.plist:12 null()		

```
10      <string>/usr/local/pgsql/bin/postgres-wrapper.sh</string>
11    </array>
12    <key>UserName</key>
13    <string>postgres</string>
14    <key>KeepAlive</key>
```

Issue Count by Category	
Issues by Category	
Password Management: Hardcoded Password	3
Privacy Violation: iOS Property List	1

Issue Breakdown by Analysis

Issues by Analysis