Fortify Security Report

2024-6-21

ASUS

Executive Summary

Issues Overview

On 2024-6-21, a source code review was performed over the dosbox-x code base. 243 files, 3,991 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

High	4

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/dosbox-x
Number of Files: 243
Lines of Code: 3991
Build Label: <No Build Label>

Scan Information

Scan time: 01:17
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

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

Standard Input Stream:
null.file.readlines

System Information:
null.null.null
java.lang.System.getProperty
java.lang.Throwable.getMessage

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

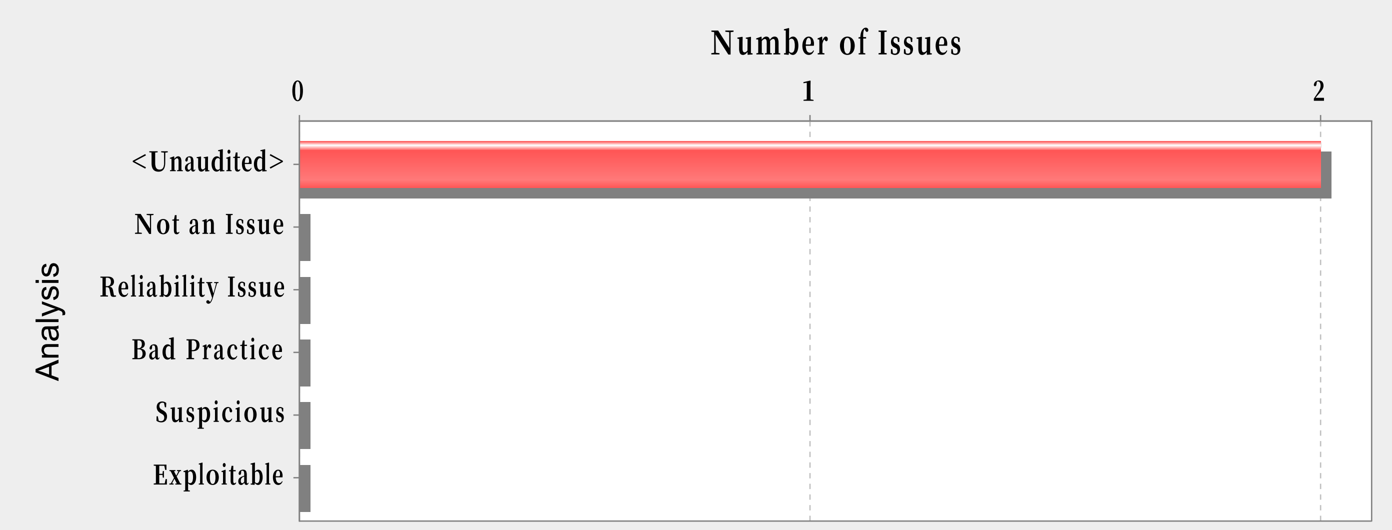
Results Outline

Overall number of results

The scan found 4 issues.

Vulnerability Examples by Category

Category: Privilege Management: Unnecessary Permission (2 Issues)



Abstract:

应用程序若不能遵守最低权限原则，便会大大增加引发其他漏洞的风险。

Explanation:

应用程序应仅拥有正常执行所需的最小权限。权限过多会导致用户不愿意安装该应用程序。此权限对于该程序可能是不必要的。

Recommendations:

考虑应用程序是否需要请求的权限来保证正常运行。如果不需要，则应将相应的权限从 AndroidManifest.xml 文件中删除。除了请求应用程序真正需要的权限之外，切忌因请求更多权限而导致对应用程序过度授权。这会导致在设备上安装的其他恶意应用程序利用这种过度授权的应用程序对用户体验及存储的数据造成负面影响。另外，设置过多的权限可能会适得其反，导致客户不愿意安装您的应用程序。

AndroidManifest.xml, line 30 (Privilege Management: Unnecessary Permission)

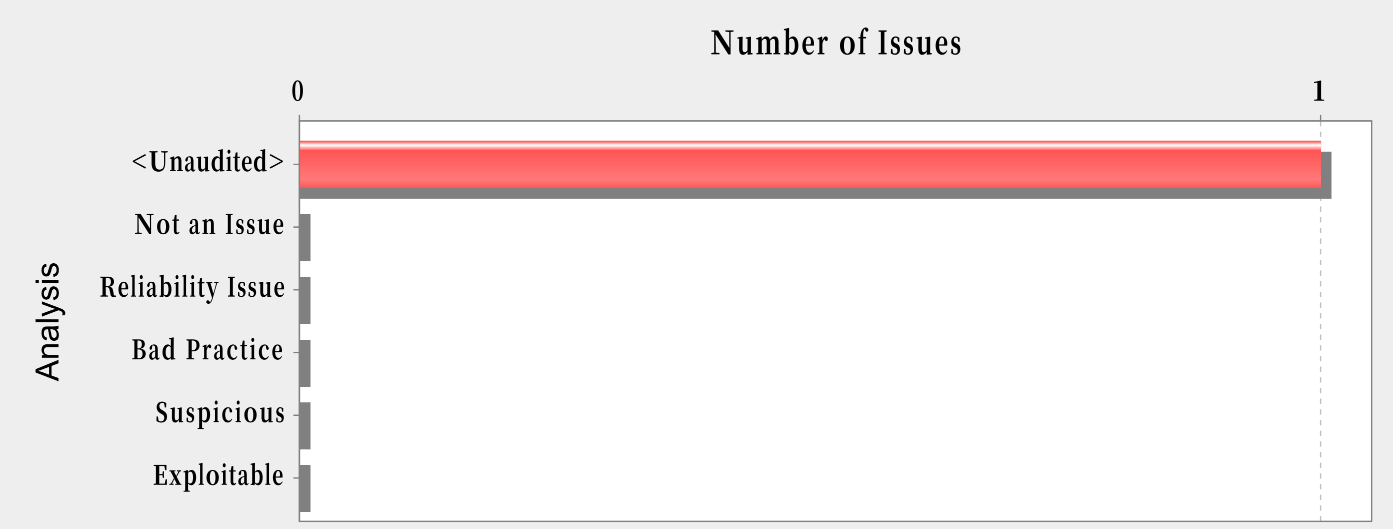
Fortify Priority:	High	**Folder**	High
Kingdom:	Security Features		

Abstract: 应用程序若不能遵守最低权限原则，便会大大增加引发其他漏洞的风险。

Sink: AndroidManifest.xml:30 null()

```
28
29    <!-- Allow writing to external storage -->
30    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
31    <!-- Allow access to the vibrator -->
32    <uses-permission android:name="android.permission.VIBRATE" />
```

Category: Android Bad Practices: Missing Google Play Services Updated Security Provider (1 Issues)



Abstract:

应用程序不使用 Google Play 服务更新的安全提供程序，这可能使其未来易遭受 OpenSSL 库中漏洞的攻击。

Explanation:

Android 依赖于可提供安全网络通信的安全提供程序。 但是，有时漏洞存在于默认安全提供程序中。 为了防范这些漏洞，Google Play 服务可提供用于自动更新设备安全提供程序的方法，以防御已知盗取手段。 通过调用 Google Play 服务方法，您的应用程序可以确保其在具有最新更新的设备上运行，以防御已知盗取手段。

Recommendations:

修补安全提供程序最简单的方法是调用同步法 installIfNeeded()。 如果在等待操作完成的过程中用户体验不会受到线程阻止的影响，则此方法适用，否则它应该以异步方式完成。

示例： 以下代码可实现用于更新安全提供程序的同步适配器。 由于同步适配器在后台运行，因此在等待安全提供程序更新的过程中若出现线程阻止也没有影响。 同步适配器调用 installIfNeeded() 以更新安全提供程序。 如果方法正常返回，则同步适配器了解安全提供程序为最新程序。 如果方法抛出异常，则同步适配器可采取相应的操作（如提示用户更新 Google Play 服务）。

```
public class SyncAdapter extends AbstractThreadedSyncAdapter {

...

// This is called each time a sync is attempted; this is okay, since the
// overhead is negligible if the security provider is up-to-date.
@Override
public void onPerformSync(Account account, Bundle extras, String authority, ContentProviderClient provider, SyncResult
syncResult) {
try {
ProviderInstaller.installIfNeeded(getContext());
} catch (GooglePlayServicesRepairableException e) {

// Indicates that Google Play services is out of date, disabled, etc.

// Prompt the user to install/update/enable Google Play services.
GooglePlayServicesUtil.showErrorNotification(e.getConnectionStatusCode(), getContext());

// Notify the SyncManager that a soft error occurred.
syncResult.stats.numIOExceptions++;
return;

} catch (GooglePlayServicesNotAvailableException e) {
// Indicates a non-recoverable error; the ProviderInstaller is not able
// to install an up-to-date Provider.

// Notify the SyncManager that a hard error occurred.
syncResult.stats.numAuthExceptions++;
return;
}

// If this is reached, you know that the provider was already up-to-date,
// or was successfully updated.
}
}
```

AndroidManifest.xml, line 49 (Android Bad Practices: Missing Google Play Services Updated Security Provider)

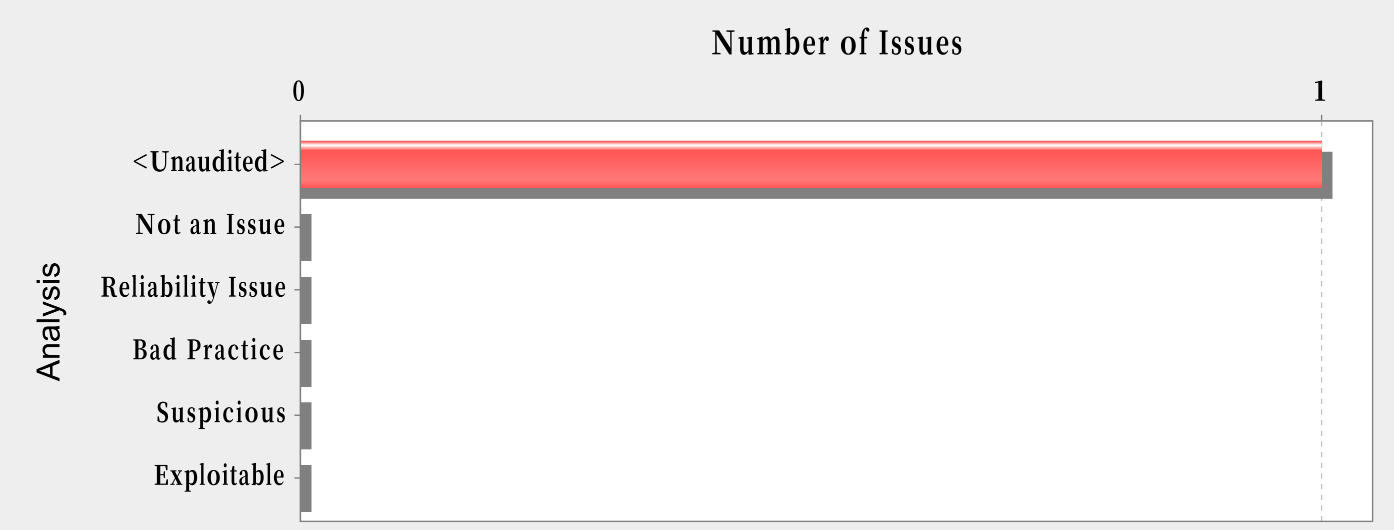
Fortify Priority:	High	**Folder**	High
Kingdom:	Security Features		

Abstract: 应用程序不使用 Google Play 服务更新的安全提供程序，这可能使其未来易遭受 OpenSSL 库中漏洞的攻击。

Sink: AndroidManifest.xml:49 null()

```
47      android:allowBackup="true"
48      android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
49      android:hardwareAccelerated="true" >
50
51      <!-- Example of setting SDL hints from AndroidManifest.xml:
```

Category: Privilege Management: Android Data Storage (1 Issues)



Abstract:

程序在 AndroidManifest.xml 的第 30 行请求将数据写入 Android 外部存储的权限。

Explanation:

写入外部存储的文件可被任意程序与用户读写。程序不可将个人可识别信息等敏感信息写入外部存储中。通过 USB 将 Android 设备连接到电脑或其他设备时，就会启用 USB 海量存储模式。在此模式下，可以读取和修改写入外部存储的任意文件。此外，即使卸载了写入文件的应用程序，这些文件仍会保留在外部存储中，因而提高了敏感信息被盗用的风险。

例 1：AndroidManifest.xml 的 <uses-permission .../%gt; 元素包含危险属性。

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Recommendations:

请勿将以后要使用的受信敏感信息或数据写入外部存储中。而应将其写入程序特定的位置，例如 SQLite 数据库（由 Android 平台提供）。程序内的任意类都可以按名称访问您所创建的任意数据库，而程序外的类则不能。

例 2.通过创建 SQLiteOpenHelper 的子类和替代 OnCreate() 方法来创建一个新的 SQLite 数据库。

```
public class MyDbOpenHelper extends SQLiteOpenHelper {

private static final int DATABASE_VERSION = 2;
private static final String DICTIONARY_TABLE_NAME = "dictionary";
private static final String DICTIONARY_TABLE_CREATE =
"CREATE TABLE " + DICTIONARY_TABLE_NAME + " (" +
KEY_WORD + " TEXT, " +
KEY_DEFINITION + " TEXT);";

DictionaryOpenHelper(Context context) {
super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL(DICTIONARY_TABLE_CREATE);
}
}
```

另一种选择则是写入该设备的内部存储中。默认情况下，保存到内部存储中的文件为该程序专用的，其他程序和用户无法直接访问。用户卸载程序时，保存在内部存储中的文件也会随之删除，保证不会留下任何重要的信息。

例 3：以下代码创建了一个专用文件并将其写入设备的内部存储中。此 Context.MODE_PRIVATE 声明会创建一个文件（或是替换同名文件），并将其设定为当前程序的专用文件。

```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

AndroidManifest.xml, line 30 (Privilege Management: Android Data Storage)

Fortify Priority:	High	**Folder**	High
Kingdom:	Security Features		

Abstract: 程序在 AndroidManifest.xml 的第 30 行请求将数据写入 Android 外部存储的权限。

Sink: AndroidManifest.xml:30 null()

```
28
29      <!-- Allow writing to external storage -->
30      <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
31      <!-- Allow access to the vibrator -->
32      <uses-permission android:name="android.permission.VIBRATE" />
```

Issue Count by Category

Issues by Category	
Privilege Management: Unnecessary Permission	2
Android Bad Practices: Missing Google Play Services Updated Security Provider	1
Privilege Management: Android Data Storage	1

Issue Breakdown by Analysis

Issues by Analysis