8-17-2021

Forensicast: A Non-intrusive Approach & Tool for Logical Forensic Acquisition & Analysis of the Google Chromecast TV

Alex Sitterer
University of New Haven

Nicholas Dubois
University of New Haven

Ibrahim Baggili
University of New Haven, ibaggili@newhaven.edu

Publisher Citation

Alex Sitterer, Nicholas Dubois, and Ibrahim Baggili. 2021. Forensicast: A Non-intrusive Approach & Tool For Logical Forensic Acquisition & Analysis of The Google Chromecast TV. In The 16th International Conference on Availability, Reliability and Security (ARES 2021). Association for Computing Machinery, New York, NY, USA, Article 50, 1–12. DOI:https://doi.org/10.1145/3465481.3470060

Comments

This is the Author's Accepted Manuscript.
Article part of the International Conference Proceeding Series (ICPS), *ARES 2021: The 16th International Conference on Availability, Reliability and Security*, published by ACM.

Forensicast: A Non-intrusive Approach & Tool For Logical Forensic Acquisition & Analysis of The Google Chromecast TV

Alex Sitterer
Connecticut Institute of Technology
University of New Haven
United States of America
asitt1@unh.newhaven.edu

Nicholas Dubois
Connecticut Institute of Technology
University of New Haven
United States of America
ndubo2@unh.newhaven.edu

Ibrahim Baggili
Connecticut Institute of Technology
University of New Haven
United States of America
ibaggili@newhaven.edu

ABSTRACT

The era of traditional cable Television (TV) is swiftly coming to an end. People today subscribe to a multitude of streaming services. Smart TVs have enabled a new generation of entertainment, not only limited to constant on-demand streaming as they now offer other features such as web browsing, communication, gaming etc. These functions have recently been embedded into a small IoT device that can connect to any TV with High Definition Multimedia Interface (HDMI) input known as Google Chromecast TV. Its wide adoption makes it a treasure trove for potential digital evidence. Our work is the primary source on forensically interrogating Chromecast TV devices. We found that the device is always unlocked, allowing extraction of application data through the backup feature of Android Debug Bridge (ADB) without device root access. We take advantage of this minimal access and demonstrate how a series of artifacts can stitch together a detailed timeline, and we automate the process by constructing Forensicast – a Chromecast TV forensic acquisition and timelining tool. Our work targeted (n=112) of the most popular Android TV applications including 69% (77/112) third party applications and 31% (35/112) system applications. 65% (50/77) third party applications allowed backup, and of those 90% (45/50) contained time-based identifiers, 40% (20/50) invoked some form of logs/activity monitoring, 50% (25/50) yielded some sort of token/cookie, 8% (4/50) resulted in a device ID, 26% (13/50) produced a user ID, and 24% (12/50) created other information. 26% (9/35) system applications provided meaningful artifacts, 78% (7/9) provided time based identifiers, 22% (2/9) involved some form of logs/activity monitoring, 22% (2/9) yielded some form of token/-cookie data, 22% (2/9) resulted in a device ID, 44% (4/9) provided a user ID, and 33% (3/9) created other information. Our findings also illustrated common artifacts found in applications that are related to developer and advertising utilities, mainly WebView, Firebase, and Facebook Analytics. Future work and open research problems are shared.

KEYWORDS

Digital Forensics, Artifacts, Google TV, Android TV, ADB

ARES 2021, August 17–20, 2021, Vienna, Austria

ACM ISBN 978-1-4503-9051-4/21/08…$15.00
https://doi.org/10.1145/3465481.3470060

ACM Reference Format:
Alex Sitterer, Nicholas Dubois, and Ibrahim Baggili. 2021. Forensicast: A Non-intrusive Approach & Tool For Logical Forensic Acquisition & Analysis of The Google Chromecast TV. In *The 16th International Conference on Availability, Reliability and Security (ARES 2021), August 17–20, 2021, Vienna, Austria.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3465481.3470060

1 INTRODUCTION

The Google Chromecast was first released in 2013 and quickly caught on as a means to easily share content from a smartphone, tablet, or other device to a Television (TV) screen. As of October 2017, over 55 million Chromecasts and Chromecast-enabled devices have been sold [13].

While older Chromecast devices ran Google's own Google TV, in September of 2020 the Android-based Google Chromecast with Google TV[1] was released, offering a stand alone streaming device with a remote, similar to offerings like the Amazon Fire TV and Roku TV line of products. The Chromecast TV is a device that was designed to accommodate all streaming services, and applications, under one interface. It also provides the ability for users to download and install different Chromecast TV applications (entertainment, browsing, communication etc.).

Internet of Things (IoT) devices such as the Chromecast TV offer great value due to their frequency of use and ability to provide evidence on what a user may have been doing at a given time. There is already precedence for the use of digital evidence extracted from IoT devices in cases, such as evidence obtained from an Amazon Alexa in a homicide investigation in 2017 [6].

While past work focused on the extraction of digital evidence from smart TVs and streaming devices, no work has explored a non-intrusive extraction and analysis of digital evidence from the Chromecast TV. We present the primary work on the forensic analysis of a Chromecast TV. Our contributions are as follows:

- We test and share a non-intrusive logical acquisition approach for a Chromecast TV without the need for root access, nor any authentication between Chromecast TV and the forensic workstation used to acquire it.
- We share our findings from the analysis of a logical extraction of 112 Chromecast TV applications.
- We provide investigators with a suggested investigative approach for logically analyzing a Chromecast TV.
- We construct Forensicast, a tool that parses data from artifacts of forensic relevance and presents a timeline of user activity in a readable manner.

[1]Henceforth referred to as the Chromecast TV

- We share common artifacts along with retrieval methods with the Digital Forensics community through the Artifact Genome Project (AGP) [10].

The rest of the paper is organized as follows. In Section 2, we share our methodology. We follow that up with our results in Section 3. We then present Forensicast in Section 4 and a suggested investigator workflow in Section 5. We then discuss our work in Section 6, and present our future work in Section 7. We end our paper with the related work in Section 8. Note that Appendix A lists all applications and their respective version numbers used in this research and Appendix B lists the artifacts recovered.

2 METHODOLOGY

Our work embodied the following step-wise procedure for each application tested:

- **Experimental Setup:** The Chromecast TV was set up and prepared for developer debugging and application backup extraction (See Section 2.1)
- **Scenario Creation:** Artifacts were manually created on the device by simulating normal usage (See Section 2.2)
- **Artifact Extraction & Analysis:** Artifacts were extracted using the Android Debug Bridge (ADB) and then manually analysed (See Section 2.3)

2.1 Experimental Setup

We employed an experimental setup with the Chromecast TV connected to a monitor with the built in High Definition Multimedia Interface (HDMI) connector on the Chromecast TV to view the display output. The Chromecast TV was also physically connected to a forensic workstation for power using a Universal Serial Bus (USB) Type C (USB-C) cable. The same power cable was employed in the acquisition of data from the device using ADB. Due to the storage limitations of the Chromecast TV, applications were loaded onto the device in batches of 30. User actions were simulated via the Chromecast TV remote, data was acquired, and then applications were deleted to free up space for other applications. This process was repeated until we tested 112 applications. In order to download applications, the Chromecast TV was connected to a WiFi network.

2.2 Scenario Creation

Before recovering any artifacts normal usage had to be simulated. This was accomplished by using each application for at least five minutes. A full list and count of the applications installed can be found in A. Author accounts were used to log into applications that required a paid subscription. Applications that used accounts but prevented usage without payment were still used as user credentials could be captured. A live feed from an Oculus Quest 2 Virtual Reality (VR) headset as well as a YouTube video from an iPhone were streamed (casted) to the device to determine if these casting actions produced any unique artifacts.

2.3 Artifact Extraction & Analysis

To logically acquire the Chromecast TV device, a few settings must first be changed on the Chromecast itself. This would be indicative of an investigator having physical access to the device. First, developer mode must be activated (Figure 1). USB debugging is then allowed in the developer mode menu (Figure 2 and Figure 3).

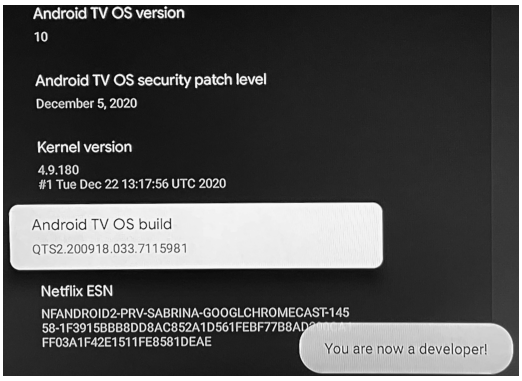

Figure 1: Activating Developer Mode on the Chromecast TV

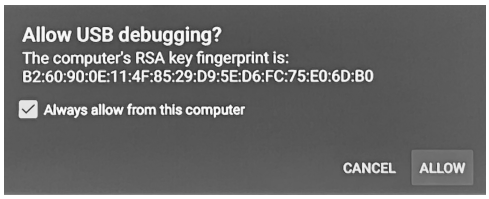

Figure 2: Allowing Debugging Permissions

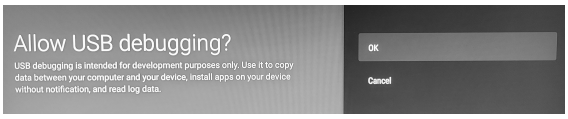

Figure 3: Allowing USB Debugging on the Chromecast TV

The Chromecast was connected to a computer and the ADB command `adb devices` was ran to find the list of devices connected to the computer, which returned the Chromecast.[2] The command `adb shell pm list packages` was run to list the packages installed on the device. After selecting a target application (in this case YouTube) the command `adb backup -f Backup.ab com.package.name` was executed to backup the application data. This was run twice, once before accessing the application and once after. After pulling the backup from the Chromecast TV, a tool called Android Backup Toolkit [3] was utilized to extract the contents of the backup file. Diff-based forensics as well as manual exploration were used against the two recovered files to determine what artifacts may have been created. [4] The accounts were then deleted from the device in order

[2] Some programs even see the Chromecast as a Google Nexus 7, suggesting similar physical/digital attributes.

[3] https://sourceforge.net/projects/adbextractor/files/latest/download

[4] Android application PacKage (APK) files can be installed directly on an Android phone and run correctly. It is not user-friendly or responsive to touch gestures aside from tapping the screen and is locked to a 16 by 9 aspect ratio, however, it can run just as any other application further proving the similarities in Android Mobile and Android TV

to explore if artifacts could still be recovered. The methodology is summarized in the following steps, and is demonstrated in Figure 4:

(1) Download desired application
(2) Backup entire device with ADB before using service
(3) Login, use application
(4) Backup again
(5) Convert ADB backups to readable files
(6) Use diff to find changed data and extract artifacts
  (a) Compare the two pulled directories for changes
  (b) Collate different files
(7) Examine collated files
  (a) Explore Structured Query Language (SQL)ite files
  (b) Explore eXtensible Markup Language (XML) files

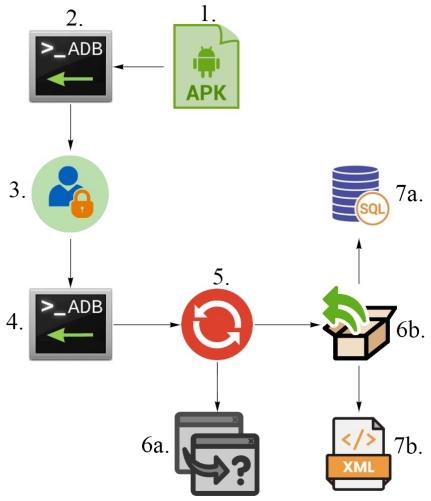

Figure 4: Artifact Acquisition Flowchart

This resulted in several interesting files, and allowed for the collection of artifacts from the Chromecast TV. A full list of forensic artifacts collected from third-party applications can be found in B and a full list of artifacts collected from system applications is shared in C. Applications that did not produce useful artifacts (or did not produce any artifacts at all, as detailed in Section 6) are not listed. An abbreviated list of system application artifacts can be found in Table 1.

3 RESULTS

Of the 112 applications used in our work, 69% (77/112) were third party applications and 31% (35/112) were system applications. 35% (27/77) third party applications did not allow backup, and the reasoning for this is explained in Section 6. While we were able to backup 35 total system applications only the 9 applications listed (26%) provided meaningful results. There are more system applications on the device, however, we will only focus on system applications which can provide backups. Of the 9 system applications, a brief overview of the artifacts they provide is located in Table 1. Of the 50 third party applications that allow backups, 90% (45/50) contained time-based identifiers, 40% (20/50) invoked some form of logs/activity monitoring, 50% (25/50) yielded some sort of token/cookie, 8% (4/50) resulted in a device ID, 26% (13/50) produced a user ID, and 24% (12/50) created other information. The overall results are visually displayed in Figure 5.

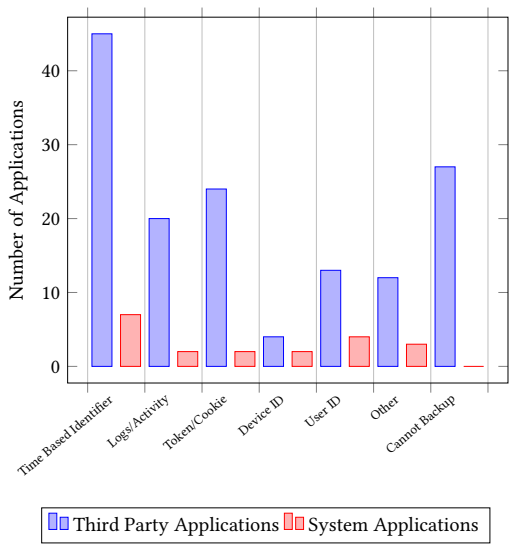

Figure 5: Application Artifact Results (n=112)

3.1 Common Artifacts

Some artifacts are very similar or identical across multiple applications. If the developers are employing Facebook Analytics, Google Firebase, or Android Webview they will generally follow the same file structure. In total 8% (9/112) of applications use Facebook analytics, 15% (17/112) of applications use Firebase, and 28% (32/112) of applications utilize Android WebView.

- **Facebook Analytics:** Many of the most popular applications on the Google TV store utilize some form of advertisement sharing or analytical data gathering. Facebook Analytics saves numerous files to the application folder. These can be found in the applications `sp` folder normally following the naming scheme `facebook.IMPLEMENTATION.xml`. These files do not store any Personal Identifiable Information, however, they provide timestamps which can be used to piece together user activity. The times are stored in UNIX time, as are all other timestamps other than custom logging applications.
- **Firebase:** Google Firebase, often used in correlation with Google Analytics and AdMob, is an application development tool which can handle most of the application data in the cloud. This means that any of the applications that employ Firebase have the same logging implementation. Firebase creates a number of logs that data can be extracted from. Firebase files can be found in the applications `sp` folder as well as the `/f/.com.google.firebase.crashlytics` folder if

Table 1: Android System Artifacts

Application	Time Based Identifier	Logs/Activity	Token/Cookie	Device ID	UserID	Other
Android System WebView	✓					
Remote Control Logging				✓		
Android Play Games	✓				✓	
Google app for Android TV	✓	✓	✓		✓	✓
Android TV Ambient Mode					✓	
Chromecast built-in	✓	✓				✓
Android Intelligence Settings	✓					
Captive Portal Login	✓		✓	✓	✓	✓
Android Providers	✓					

Firebase is utilizing Crashlytics. Every application which utilizes Firebase creates a `/sp/FirebaseAppHeartBeat.xml` file which houses a global and installation id, which are generated from the UNIX time in which Firebase was first run for that application. This is a signal that the application was installed and run for the first time at that timestamp. Firebase may create optional Google Analytics XML files in the `/sp/` directory. Firebase also creates a `/f/PersistedInstallation.UNIQUEID.json` file which refreshes a token while updating a timestamp to keep track of a user. Applications which utilize Firebase generate a plethora of useful artifacts for user tracking

- **WebView:** Android System WebView is an Android Component to implement content-invoking within applications. WebView may be utilized when presenting browser data or to call web-pages in the back-end for application use. WebView by default stores all of the history in log files as well as cookies. Cookie data as well as some occasional log files (dependant on the application) are stored in `/r/app_webview`. Local and Session storage database logs are often stored in this folder giving a full account of everything the user has done recently. WebView may create preference files in the applications `sp` folder. Applications which utilize WebView provide the richest artifacts.

3.2 Casting

After streaming data to the device using both an iPhone and an Oculus Quest 2 it became apparent that data can be found in the Android Media Shell which defines exactly which user was on the device, the associated display name of that user, which website they were casting (if applicable), and which type of device they were using. This is stored in `com.google.android.apps.mediashell/r/app_cast_shell/Local Storage/leveldb/000003.log`. Although stored in a log file, when restarting the device the system will automatically recover and keep the log intact. This means that even after removing power from the device the log may be recoverable.

3.3 Browsers

While the data collection for most of the applications followed the same methodology, browser data is likely to hold the most sensitive information. All of the popular browsers that were tested (TVWeb, Jiopages, Downloader) utilize WebView. Each contains a full log of every site visited as well as every stored cookie. The Local, and Session, storage folders house even more logs which go in greater depth. The history may be cleared from these applications in settings, however, this does not actually remove the history from logs but instead removes the Uniform Resource Locator (URL)s from the front-end only and ignores WebView. This means that even if the user clears their history, as long as they do not uninstall the application, all of their activity may still be retrieved.

4 FORENSICAST

We constructed Forensicast – a tool used to acquire, discover, and present artifacts from the Chromecast TV. As well as including some basic ADB management tools, Forensicast can dynamically create a log-style timeline of actions by piecing together artifacts from various applications. Forensicast can be downloaded from the following GitHub repository: https://github.com/unhcfreg/ForensiCast.

Algorithm 1 Common Artifact Discovery

```
1:  procedure FINDART(bPaths,bNames,WebView, Facebook,Firebase)
2:      WebView ← WebView                          ▷ Has WebView
3:      Facebook ← Facebook                        ▷ Has Facebook
4:      Firebase ← Firebase                        ▷ Has Firebase
5:      Paths ← bPaths                             ▷ List of Applicaton Paths
6:      Names ← bNames                             ▷ List of Applicaton Names
7:      results ← []                               ▷ Blank List for Results for
    Paths, Names = 1, 2, ..., N do
8:
          end
          Path ← Paths                             ▷ Current File Path
9:      Name ← Names                               ▷ Current File Name
10:     Web ← WebView                              ▷ Has WebView
11:     Book ← Facebook                            ▷ Has WebView
12:     Base ← Firebase                ▷ Has WebView if Web ≠ 0 then
13:
          end
          results+ = path/r/app_webview
14:     break if Book ≠ 0 then
15:
          end
          results+ = path/sp/facebook.*
16:     break if Base ≠ 0 then
17:
          end
          results+ = path/sp/Firebase*
18:     results+ = path/f/Persisted*
19:     results+ = path/f/com.google.firebase.crashlytics
20:     break
21: end procedure
```

4.1 Algorithm Overview

Algorithm 1 shows our constructed approach for Forensicast. Applications which utilize the common packages WebView, Firebase,

and Facebook Software Development Kit (SDK) share common artifacts which can be recovered using the given algorithm. For **N** applications currently being researched, we will extract relevant artifacts.

The procedure utilizes `bPaths`, the path to the applications extracted backup folder, and `bNames`, the name of each application. WebView, Firebase, and Facebook variables are used to indicate whether the application is utilizing each package or not. This will have to be known ahead of time by looking briefly at the applications extracted backup directory.

In Algorithm 1, lines 2-7 set up our variables for later use. The bulk of the algorithm begins on line 8 with the start of a `for` loop which is run through for all `N` packages.

If an application is utilizing the selected package, then the algorithm will direct the user to artifacts produced by the package. If WebView is set to 1/true then the algorithm will collect the required files on lines 14-17. For WebView this is an entire folder as the contents of this folder may vary greatly for each application.

Lines 18-21 execute the extraction of Facebook-related artifacts. If Facebook packages are used and Facebook is set to true then files in the `/sp/` directory which start with the phrase "facebook" will be returned.

If Firebase is set to true then the algorithm returns the `com.google.firebase.crashlytics` folder on lines 22-27. Firebase artifacts also include files in the `/sp/` directory which contain the phrase "Firebase" at the beginning.

4.2 Usage

The options in the menu shown in Figure 6 perform the following actions:

(1) Create Full Timeline - Create a big timeline using any and all data gathered (Listing 1)
(2) Create Partial Timeline - Create a timeline based on a given start/end time
(3) Dump All Recoverable Data - Pull every file in the file system that ADB has permissions for
(4) Backup App - Backup all packages
(5) Backup Package - Given Package name, backup specific package
(6) List All Installed Packages
(7) Install APK - Push APK to device for installation

4.3 Timelines

Using artifacts extracted in Section 2, a timeline of user activity can be created. Timelining is imperative in digital forensics as it helps tell the story of what happened [15]. This functionality was built into Forensicast, and an example timeline is shown in Listing 1.

```
1
2 com.google.android.youtube.tv ==> Youtube 2020-02-03
      10:04:34+00:00 -> 2020-02-03 10:37:56+00:00
3 {
4 2020-02-03 10:04:34+00:00 == Application Opened
5 2020-02-03 10:05:17+00:00 == User 'nnoots93@gmail.com' (
      Google ID BcSy54k8BMOgV9FAntBCRA) Logged In
6 2020-02-03 10:37:56+00:00 == Application Closed
7 }
8
```

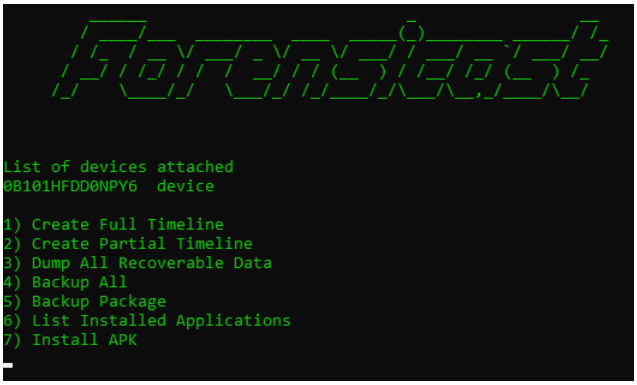

Figure 6: Forensicast Tool Menu

```
9
10 com.radioline.android.radioline ==> Radioline 2020-02-03
      10:42:29+00:00 -> 2020-02-03 14:21:31+00:00
11 {
12 2020-02-03 10:42:29+00:00 == Application Opened, First
      Installation
13 Favorites DB == Last Listened To 'Cherie Romantic'
      Channel
14 2020-02-03 14:21:31+00:00 == Application Closed
15 }
16
17
18 jp.co.rarity.tvweb ==> TVWeb 2020-02-04 11:21:44+00:00 ->
       2020-02-04 11:34:03+00:00
19 {
20 2020-02-04 11:21:44+00:00 == Application Opened
21 2020-02-04 11:22:01+00:00 == https://www.google.com/
22 2020-02-04 11:22:10+00:00 == Sign in - Google Accounts
      https://accounts.google.com/ServiceLogin?hl
23 2020-02-04 11:23:07+00:00 == Sign in - Google Accounts
24 2020-02-04 11:23:50+00:00 == Noot Noot - Google Search
25 2020-02-04 11:34:47+00:00 == Application Closed
26 }
```

Listing 1: Example Timeline Output

The example timeline in Listing 1 shows a user opening the YouTube package (line 2) at 10:04 AM (line 4) and logging in with the account "nnoots93@gmail.com" at 10:05 AM on March 2nd 2021 (line 5). The user then closes the application at 10:37 AM (line 6). Forensicast will attempt to resolve the application name (YouTube) from the given package name (com.android.youtube.tv), however, if it cannot be found `Unknown Application` will be listed instead. The same formula is applied to other applications, as first the package name is stated then artifact information is stated.

5 RECOMMENDED INVESTIGATOR WORKFLOW

The workflow for an investigator is similar to the experimental workflow used in this paper (detailed in Sections 2.1 and 2.3) and includes the Forensicast tool (Section 4).

(1) Connect Chromecast TV to a monitor for visual data and a forensic workstation with ADB installed using a USB-C cable to pull backups.[5]
(2) Activate Developer mode on the Chromecast TV by interacting with the `Android TV OS Build` info box in the `About` section of the `Settings` menu ten times (Shown in Figure 1).
(3) In the `Developer` section of the `Settings` menu allow USB Debugging (Shown in Figure 3).
(4) Run the command `adb devices` on the connected laptop and select `Allow` on the Chromecast TV to allow the backing up of data (Shown in Figure 2).
(5) Run the command `adb shell pm list packages` to get a list of all installed applications.
(6) Run the command `adb shell pm path <package name>` to get the path of the desired package.
(7) Run the command `adb pull <path to package>` to pull the APK to the investigating computer.
(8) Extract SQLite database, XML, and log files.
(9) Determine if these files contain valid artifacts to reconstruct any event history
(10) Use the information gained to create a timeline.

Forensicast aims at automating steps 4-10 to speed up the investigative process and automate the artifact extraction and timeline creation.

5.1 Example Scenario

Forensicast and the forensic analysis of a Chromecast TV could be used to determine the time, location, viewer, and content in a forensic investigation, which in turn provide:

- Evidence to prove or disprove an alibi.
- Viewing history which provides evidence a suspect viewed media that encouraged them to commit a crime.
- Location history to show where a suspect actually was.
- Account history to show who was watching or that account information had been shared between the owner of the Chromecast TV and an account holder.

We set up a fictitious scenario with a student that was accused of murdering a person in New York. After creating the user based scenario, our tool was run and produced the following artifacts:

- Emails - nnoots93@gmail.com
- Google Search - how to hide a body
- YouTube - How To Hide A Body - Apr 25 2021 20:55:04 GMT
- Google Search - food near Bethpage New York
- Notes - "1. Go to NY, 2. Kill Jimmy"

6 DISCUSSION

With Android TV, especially the Google TV version, being a newer and less common operating system there is still much to be asked for in terms of security. This is partially due to third party app developers not fully understanding what plugins do in the back-end or simply not caring as much and partially due to the simplification and user intuitiveness features of the Google TV operating system coming at the cost of security. While this can be seen as bad for the general population this is a great thing for forensic researchers.

Android TV is still somewhat of an untapped field and can offer much more potential than just what has been outlined in this paper. The work we performed was only a simple dive into backup data. While producing valid results, this is only the beginning and there is still much to be discovered.

35% (27/77) of third party applications tested did not provide backups. Adding the line `android_allowBackup="false"` to the `Android Manifest.XML` file in the application will disallow users from making a backup of the app without root permissions. This is beneficial for app developers who wish to hide sensitive data from the methods used in this paper, however, it presents a challenge to investigators as the device in question must be rooted in order to access the information stored in those applications.

It is also quite apparent that most of the artifacts are not created directly by the application themselves but by the implementation of plugins and system calls. A developer could, if so inclined, hide these calls or get rid of them all together.

As there were no functioning methods to gain root access to the Chromecast TV at the time of writing (August 25, 2021) there is no way to access the `/data/` folder directly. This means that an investigator cannot gather many important files from the system and various applications.

The data stored on a Chromecast TV contains more than just usage time (when a user was watching), as it holds account information (who was watching/listening), history (what they were watching/listening to), tokens/cookies (which can be used to identify them elsewhere), and in some circumstances location information (where they were). As the Chromecast TV is portable and consumes little electricity, it may be used by travelers to consume media on a hotel TV or in a vehicle with media capability, such as an Recreational Vehicle (RV) or some models of cars.

There can only be one user account associated with a Chromecast TV, and there are no users, but individual applications can have separate user profiles (E.g., Netflix). There is also no password needed to authenticate to the device, however it is necessary to provide user input.

7 FUTURE WORK

In our work we focused on data that could be easily accessible that was not obscured or encrypted. Since encoded or encrypted artifacts show up often, future work may want to explore methods of decrypting or deobfuscating such data.

Other versions of Android TV such as the official copy and those offered by Samsung have parental locks and other restriction options to lock the device behind a password. Future work should explore how parental restrictions may prevent a backup and how to negate or bypass these restrictions.

As the device was new during our work, there were no functioning rooting methods that were found. This rendered some applications and artifacts to be inaccessible as detailed in Section 6. With rooting, an investigator could explore more than just the backup directory, and may uncover deleted data since a physical image maybe acquired.

[5] In the absence of the remote for the Chromecast TV, a USB gamepad or keyboard can be plugged in using a USB-C hub in order to give input to the device.

Recovering artifacts through hardware hacking and debugging could also be a worthy future work endeavour. Lastly, our work did not explore the network traffic of the Chromecast TV. While it appears to be similar to an Android device in terms of netcode, there may be artifacts unique to this device that future research may uncover.

8 RELATED WORK

Little work has explored the area of smart TV forensics. At the time of writing, prior research did not explore a non-intrusive acquisition and analysis method of the Chromecast TV. In this section, we cover past efforts related to our work in the areas of smart TV exploits and forensics, and non-intrusive forensic acquisition and analysis.

8.1 Chromecast Exploits

ghostlulz [sic.] [9] detailed how to replicate the infamous Chromecast hack that occurred in early January of 2019. The attack was performed by sending a POST request to a Chromecast with a YouTube URL which will cause the Chromecast to automatically play that video.

Marck [14] discussed some of the vulnerabilities present in an off the shelf Android TV box from 2017. This box was different from the Chromecast TV in that it was rooted in its default state. The researchers installed Busybox (a set of custom binaries for UNIX devices) and then proceeded to install various malware with the intent to simulate an attack on a home network. They were successful in attacking several devices on the network, as well as capturing displayed information on the device. They further discussed potential mitigation techniques to prevent the exploitation of the device.

8.2 Smart TV Forensics

Morrison et al. [17] described attempts to acquire forensic images of various devices, including an original Chromecast, a Measy A2W Miracast, and an Amazon Fire TV Stick. The researchers concluded that there was little possibility to pull data from the Chromecast, a slim possibility to acquire data from the Miracast. The researchers developed a framework to acquire forensic images from an Amazon Fire TV Stick. Their method involved rooting the Fire TV Stick and editing the file permissions to access the user data on the device. This is similar to the goal of this paper, however, at the time of writing (August 25, 2021) there were no functioning root methods that could be found for the Chromecast TV.

Tekeoglu and Tosun [18] detailed their attempts to extract network and other vulnerability information from a 2013 model Chromecast without physical exploitation. The research concluded that the older version of a Chromecast does not encrypt control packets and leaks network data.

Boztas et al. [5] researched the acquisition of forensic artifacts from a Samsung SmartTV. Their research employed several methods to directly read the data from the flash memory chip located on the device. Their first attempt was to wire the chip to an Secure Digital (SD) card reader connected to a write blocker, however, this did not work. The researchers then de-soldered the memory chip and used a Netherlands Forensic Institute (NFI) Memory Toolkit II (MTK II) (A combination of hardware and software that is able to directly read memory chips[6]) to access the data. Understanding that this is not feasible for many investigators, they also presented a software method that involved rooting the smart TV to access the entire file system. This allowed the researchers to collect artifacts which detailed most, if not all, of the activity on the device.

Falayeh [7] explored common usage of smart TVs, how they can be compromised, and an overview of some data recovery methods. The researchers found that there are many issues facing smart TV forensics, mainly the lack of standardization of tools, devices, and approaches.

Hadgkiss et al. [11] listed methods of collecting artifacts from an Amazon Fire TV Stick. Initially, software methods were used in an attempt to image the device, however, without root access this did not work so hardware methods similar to what was outlined in [5] were used.

8.3 Non-intrusive Acquisition & Analysis

Feng et al. [8] described a method to subvert the issues of having a device that cannot be rooted. In their work they employ a manufacturer's system level data transfer solution to transfer data to a surrogate rooted device. The surrogate device can then be attached to a computer for forensic acquisition.

Bader and Baggili [4] detailed the primary account for the logical acquisition of artifacts from an iPhone 3GS using the Apple iTunes Utility. This approach was also generalized in [12], and prior work has also explored the difference between manually examining the backup, versus using an automated tool [3]. Building on the iOS past work, researchers then created an open source source tool, Logical iOS Forensic Examiner (LiFE) for automatically analyzing the backup files of an iOS device [1].

Marzougy et al. [16] explored the logical acquisition of forensic artifacts from a Blackberry PlayBook backup file. Their approach detailed that the backup file was essentially a zip file, and a number of forensically relevant artifacts were extractable.

Lastly, Mutawa et al. [2] explored the analysis of social networking applications on iPhones, Blackberry phones, and an Android phone. The researchers acquired a forensic image of each of these devices and were able to extract social networking artifacts from the Android phone and the iPhones, but were unsuccessful with the Blackberry devices. Mutawa et al. noted that there were no unified backup solutions at that time for Android phones, forcing the researchers to use a third party application to backup Android devices.

ACKNOWLEDGMENTS

This material is based upon work supported by the Office of Naval Research under Grant Number N00014-20-1-2296. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research.

REFERENCES

[1] Shadi Al Awawdeh and Jason Moore. 2014. LiFE (Logical iOS Forensic Examiner): An Open Source iOS Backup Forensics Examination Tool. In *Proceedings of the*

[6] https://www.forensicinstitute.nl/products-and-services/forensic-products/nfi-memory-toolkit

Conference on Digital Forensics, Security and Law. Association of Digital Forensics, Security and Law, 41.

[2] Noora Al Mutawa, Ibrahim Baggili, and Andrew Marrington. 2012. Forensic analysis of social networking applications on mobile devices. *Digital Investigation* 9 (Aug. 2012), S24–S33. https://doi.org/10.1016/j.diin.2012.05.007

[3] Somaya Ali, Sumaya AlHosani, Farah AlZarooni, and Ibrahim Baggili. 2012. iPad2 Logical Acquisition: Automated or Manual Examination?. In *Proceedings of the Conference on Digital Forensics, Security and Law*. Association of Digital Forensics, Security and Law, 113.

[4] Mona Bader and Ibrahim Baggili. 2010. iPhone 3GS Forensics: Logical Analysis Using Apple iTunes Backup Utility. *Electrical & Computer Engineering and Computer Science Faculty Publications* 4 (Sept. 2010), 16. https://digitalcommons.newhaven.edu/electricalcomputerengineering-facpubs/32

[5] A. Boztas, A. R. J. Riethoven, and M. Roeloffs. 2015. Smart TV forensics: Digital traces on televisions. *Digital Investigation* 12 (March 2015), S72–S80. https://doi.org/10.1016/j.diin.2015.01.012 Proceddings of DFRWS-EU 2015.

[6] Anthony Cuthbertson. 2018. Amazon ordered to give Alexa evidence in double murder case. https://www.independent.co.uk/life-style/gadgets-and-tech/news/amazon-echo-alexa-evidence-murder-case-a8633551.html Section: Lifestyle.

[7] Mousa Al Falayleh. 2013. A Review of Smart TV Forensics: Present State & Future Challenges. In *The International Conference on Digital Information Processing, E-Business and Cloud Computing (DIPECC2013). The Society of Digital Information and Wireless Communication*. 50–55.

[8] Peijun Feng, Qingbao Li, Ping Zhang, and Zhifeng Chen. 2018. Logical acquisition method based on data migration for Android mobile devices. *Digital Investigation* 26 (Sept. 2018), 55–62. https://doi.org/10.1016/j.diin.2018.05.003

[9] ghostlulz. 2019. Hacking Google Chromcast. https://medium.com/@ghostlulzhacks/hacking-google-chromcast-dcdf98392f8f

[10] Cinthya Grajeda, Laura Sanchez, Ibrahim Baggili, Devon Clark, and Frank Breitinger. 2018. Experience constructing the Artifact Genome Project (AGP): Managing the domain's knowledge one artifact at a time. *Digital Investigation* 26 (July 2018), S47–S58. https://doi.org/10.1016/j.diin.2018.04.021 Proceedings of DFRWS-USA 2018.

[11] M. Hadgkiss, S. Morris, and S. Paget. 2019. Sifting through the ashes: Amazon Fire TV stick acquisition and analysis. *Digital Investigation* 28 (March 2019), 112–118. https://doi.org/10.1016/j.diin.2019.01.003

[12] Mohammad Iftekhar Husain, Ibrahim Baggili, and Ramalingam Sridhar. 2010. A simple cost-effective framework for iPhone forensic analysis. In *International Conference on Digital Forensics and Cyber Crime*. Springer, 27–37.

[13] Harish Jonnalagadda. 2017. Google has sold 55 million Chromecasts around the world. https://web.archive.org/web/20171005101127/https://www.androidcentral.com/google-has-sold-55-million-chromecasts-around-world

[14] Austin J. Marck. 2017. *Abusing Android TV Box for Fun and Profit*. Ph.D. Dissertation. University of Cincinnati. https://etd.ohiolink.edu/apexprod/rws_olink/r/1501/10?clear=10&p10_accession_num=ucin1504786962271509

[15] Andrew Marrington, Ibrahim Baggili, George Mohay, and Andrew Clark. 2011. CAT Detect (Computer Activity Timeline Detection): A tool for detecting inconsistency in computer activity timelines. *Digital Investigation* 8 (Aug. 2011), S52–S61. https://doi.org/10.1016/j.diin.2011.05.007

[16] Mohamed Al Marzougy, Ibrahim Baggili, and Andrew Marrington. 2013. BlackBerry PlayBook Backup Forensic Analysis. In *Digital Forensics and Cyber Crime (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering)*, Marcus Rogers and Kathryn C. Seigfried-Spellar (Eds.). Springer, Berlin, Heidelberg, 239–252. https://doi.org/10.1007/978-3-642-39891-9_15

[17] Logan Morrison, Huw Read, Konstantinos Xynos, and Iain Sutherland. 2017. Forensic Evaluation of an Amazon Fire TV Stick, Vol. Advances in Digital Forensics. HAL, 63–79. https://doi.org/10.1007/978-3-319-67208-3_4

[18] A. Tekeoglu and A. Ş Tosun. 2014. Blackbox security evaluation of chromecast network communications. In *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*. IEEE, 1–2. https://doi.org/10.1109/PCCC.2014.7017050 ISSN: 2374-9628.

A LIST OF APPLICATIONS

This section contains a list of all the applications used in this paper as well as their version numbers when applicable.

Entertainment	Tools & Browsers	Sports	News & Fitness	Games	Music
Chromecasting Disney+ (v1.131) Hulu (v3DF2AA36P3.8.26) Peacock (1.4.21-peacocktvGoogle) Tubi (v4.9.0) Pluto TV (v5.4.0-leanback) Apple TV (v1.0) IMDB TV (5.4.7-armv7a) NBC (v7.20.0) The CW (v2.52) XUMO (v1.1) Showtime (v1.11) PBS (v4.10.7) AMC (v2.23.0) ABC (v10.10.0.102) Crackle (v7.10.0.1) Crunchyroll (v2.1.1) FXNOW (v10.11.0.102) YouTube Kids (v1.12.02) PLEX (v8.13.1.23036) OLD MOVIES (v1.12.04) Philo (v4.1.54-15372-google) Redbox (v3.11.0) History (v1.6.2) Vudu (v6.1.1129) Netflix (8.1.2 build 3844) YouTube (v2.12.08) Prime Video (v5.4.7-armv7a)	Downloader (v1.4.2) TVWeb (v1.5.8) Teamviewer (v15.15.46) Anydesk (6.1.8) Jiopages (v3.0.0) Message Writer (v1.8.7)	ESPN (v4.4.0) NBC Sport (v1.0.2012171211) MLB (v6.4.3) FOX Sports (v3.37.0) FOX Sports GO (v4.8.0) NBA (v4.0.29) WWE (v49.0.0) NHL (v3.1.1) Bleacher Report Live (v2.8.9.0) FITE (4.10.1) NFL (v17.16.0) GCN+ (7.2.1)	Peloton (v1.0.320744) Bloomberg Media (v3.0.5) Market Cast (v4.0.4) Fox Nation (v3.33.2) Fox News (v3.17) ABC News Live (v10.11.1.102) CBS News Live (v2.1.2) Newsmax TV (v1.0.1) Haystack News (v4.10) Yahoo! (v1.3.2) Mjunoon.tv (v1005) Aljazera (v1.1.4) NPR One (v1.1.0) Americas Voice (v2.0) The Washington Post (4.13.1) USA Today (v1.4.5)	Orbia (v1.084) Red Ball 4 (v1.4.21) Pacman 256 (v2.0.2) Zen Pinball (v1.47) Fast Like a Fox (v1.4.6) Badland (v3.2.0.45) Into the Dead (v2.6.0)	Amazon Music (v3.3.620.0) Spotify (1.42.0) Pandora (v4.2.1) iHeart (v3.0.7) Tunein (v26.1) Tidal (v2.37.0) Radioline (v2.2.13) Deezer (v3.0.0)
System Applications (35)	android.chromecast android.bluetooth.chromecast android.bluetoothmidiservice android.captiveportallogin cts.ctsshim priv.ctsshim dreams.basic android.externalstorage android.htmlviewer android.managedprovisioning android.modulemetadata android.pacprocessor android.printspooler providers.tv android.proxyhandler settings.intelligence settings.intelligence settings.chromecast android.wallpaperbackup Chromecast Built In (v1.50.228700) Android TV Ambient Mode (v1.0.331776182) tv.netoscope android.chromecastsetupcustomization ext.services android.katniss Google App for Andoid TV (vVaries with device) android.onetimeinitializer overlay.googlewebview Google Play Games (vVaries with device) sss.authbridge tv.bugreportsender tv.frameworkpackagestubs remotecontrol.logging service.chromecast Android System WebView (v88.0.4324.181)				

B LIST OF APPPLICATION ARTIFACTS

Table 2: All of the applications that were installed on the Chromecast TV and the artifacts they contained

Third Party Applications							
Application	**Path**	**Time Based Identifier**	**Logs/ Activity**	**Token/ Cookie**	**Device ID**	**User ID**	**Other**
Amazon Music	WebView	✓			✓		
	...authentication.xml			✓	✓		
Pandora	WebView	✓	✓		✓		
	...Pandora.xml			✓	✓		
Radioline	Firebase	✓		✓			
	Facebook	✓					
	...pref..xml	✓					
	...radioline_favs.db		✓				
iHeart Radio	...tv_preferences.xml		✓				
	...crashlytics-core/ folder	✓					✓
	...tv.db		✓				
TVWeb	WebView	✓	✓	✓			
	...MainActivity.xml		✓				
	...admob.xml		✓				
	...prefs.xml	✓	✓				
Message Writer	...mobile.message.xml		✓				
JioPages	WebView	✓	✓	✓			
	...prefs.xml	✓					
	...admob.xml	✓					
Downloader	WebView	✓	✓	✓			
History Channel	Firebase	✓					
	...analytics.properties	✓					
	...cacert.pem						✓
ABC	...hero_data.xml	✓					
	...APP_MEASUREMENT_CACHE.xml	✓					
	...android.abc.xml	✓					
	...status_repository_file						✓
	...ErrorReport-Tue-0.txt	✓	✓				
	...video_progress_database	✓	✓				
OLD MOVIES	Firebase	✓					
Vudu	Firebase	✓		✓			
	...Tablet_preferences.xml	✓					
	...APP_MEASUREMENT_CACHE.xml	✓					
Crackle	WebView	✓	✓				
	...preferences.xml	✓					
	...lastUser.json					✓	
Crunchyroll	Firebase	✓					
	...AsyncStorage.txt		✓				
	...cookies.dat			✓			
	...cacert.pem						✓
FXNOW	...auth_repo.xml	✓					
	...hero_data.xml	✓					
	...repository_file						✓
	...Tue-0.txt	✓	✓				
	...progress_database	✓	✓				
YouTube	...youtube.xml					✓	
	...identity.db					✓	
YouTube Kids	...storage	✓					
NBC	Firebase	✓					
	Facebook						
	WebView	✓					
	...preferences.xml						✓
	...setting_pref.xml		✓				
	...279920456.xml	✓					
	...7736813566.xml	✓				✓	
	...connectionstatus.xml	✓					
	...measurement.prefs.xml	✓					
	...CACHE.xml	✓					
	...iterableapi.xml					✓	
	...mparticle.db	✓					
PBS	Firebase	✓					
	Facebook						
	WebView	✓		✓			
	...preferences.xml					✓	
Philo	WebView	✓		✓			
Tubi	Firebase	✓					
	Facebook	✓					
	WebView	✓	✓	✓			
	...tubitv_preferences.xml	✓				✓	
	...7866489dff0bd.xml	✓					
Xumo	Firebase	✓					
	WebView	✓					
Pluto TV	Firebase	✓					
	...cache.xml		✓				

Continued on next page

Table 2 – *Continued from previous page*

	..time_sync.xml	✓					
	...app_config.xml				✓		
	...default_user	✓					
NBC Sports	WebView	✓					
	...nbcsports.applications.tv.xml	✓					
	...visitorIDServiceDataStore.xml	✓					
	...AnalyticsDataStorage.xml	✓					
	...AdobeMobile_Lifecycle.xml	✓					
	...cookiefile			✓			
NBA	WebView	✓		✓			
WWE	Firebase	✓					
	WebView	✓		✓			
	...LicenceMetadata.xml	✓					
	...Axis.xml	✓					
	...App.xml			✓			
	...AppGeneral.xml	✓					
NFL	Firebase	✓					
	WebView	✓					
	...CACHE.xml	✓					
	...cacert.pem						✓
FOX Sports GO	WebView	✓		✓			
	...videogo_preferences.xml			✓			
	...CACHE.xml	✓					
	...IMQ0CC7Woj.xml	✓					
	...e03r28t.tmp						✓
	...cookiefile			✓			
FOX Sports	WebView	✓	✓	✓			
Fite	Firebase	✓		✓		✓	
	Facebook	✓					
	WebView	✓		✓			
ESPN	...kosp.xml	✓					
	...cSPrefs.xml	✓					
	...CACHE.xml	✓					
	...one_id.xml	✓		✓		✓	
	...prefs_ip.xml					✓	
	...prefs_ads.xml	✓					
Pacman 256	Facebook	✓					
	...playerprefs.xml	✓					
	...eventCache.json					✓	
Zen Pinball	...appid.xml	✓					
	...UserAccount.xml					✓	
	...prefs.xml	✓					
	...PREFERENCES.xml	✓					
Into The Dead	Firebase	✓					
	Facebook	✓					
	WebView	✓					
	...preferences.xml	✓					
	...cert.pem						✓
	...session_storage.json	✓					
Orbia	Firebase w/ AdMob	✓					
	Facebook	✓					
	WebView	✓	✓	✓			
	...public-data.json	✓					
Red Ball 4	...Cocos2dxPrefsFile.xml	✓					
	...appid.xml	✓					
	...measurement.prefs.xml	✓					
	...analytics.prefs.xml	✓					
Badland	Facebook	✓					
	WebView	✓					
	...supersonic_sdk.db	✓					
	...applovin.sdk.1.xm	✓		✓			
	...mraid_js_store.xml	✓					
	...key_store.xml	✓					
	...config_store.xml	✓					
NPR One	WebView	✓					
	...settings.xml	✓					
	Google Analytics	✓					
	...debug.log	✓	✓				
Aljaseera	...preferences.xml	✓					✓
USA Today	...analytics.properties	✓					
	...workdb	✓					
Haystack News	Firebase	✓				✓	
	Facebook	✓					
	WebView	✓					
	...PREFERENCES.xml	✓					
FOX News	WebView	✓		✓			✓
	...LPYbZYyeKp.xm	✓					
	...CACHE.xml	✓					
	...admob.xml	✓					
	...analytics.properties	✓				✓	
FOX Nation	WebView	✓	✓	✓			
Newsmax TV	Firebase	✓					
	WebView	✓	✓				
	...prefs.xml	✓					

Continued on next page

Table 2 – *Continued from previous page*

CBS News Live	Firebase	✓					
	WebView	✓		✓			
	...CACHE.xml	✓					
	...properties	✓					
Market Cast	WebView	✓		✓			
	Google Analytics					✓	
ABC News Live	WebView	✓		✓			
	...hero_data.xml	✓					
	...CACHE.xml					✓	
	...Repository.xml						✓
	...Tue-0.txt		✓				
	...A46A_aa.6.1.0.1	✓					
	...progress_database		✓				

C LIST OF SYSTEM ARTIFACTS

This section contains a table detailing the types of artifacts collected from the Android system.

Table 3: All of the system applications included on the Chromecast TV and the artifacts they contained

System Applications							
Application	**Path**	**Time Based Identifier**	**Logs/ Activity**	**Token/ Cookie**	**Device ID**	**User ID**	**Other**
Android WebView	WebView	✓					
	...variations_prefs.xml	✓					
Remote Control Logging	...RANDOM_ID.xml				✓		
Android Play Games	...shared-preferences.xml					✓	
	...analytics.prefs.xml	✓					
	...measurement.prefs.xml	✓					
Android Katniss	Firebase	✓					
	WebView	✓					
	...voice.interactor.xml					✓	✓
	...katniss.setting.xml					✓	
	...stats_wrapper.xml					✓	
	...AccountData.pb					✓	
	...db/search.db	✓	✓			✓	
	...behaviorcollector.db	✓		✓			
Dreamx	...StickyAccount.xml					✓	
	...applications.tv.dreamx.pb					✓	
Android Media Shell	...gms.appid.xml	✓					
	...RecommendationsService.xml	✓					
	...leveldb/000003.log		✓				✓
Intelligence Settings	...SuggestionEventStore.xml	✓					
	...suggestions.xml	✓					
Captive Portal Login	WebView	✓		✓	✓	✓	✓
Android Providers	...preferences.xml	✓					