8-30-2018

I Know What You Did Last Summer: Your Smart Home Internet of Things and Your iPhone Forensically Ratting You Out

Gokila Dorai
Florida State University

Shiva Houshmand
Santa Clara University

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

Publisher Citation

Gokila Dorai, Shiva Houshmand, and Ibrahim Baggili. 2018. I Know What You Did Last Summer: Your Smart Home Internet of Things and Your iPhone Forensically Ratting You Out. In ARES 2018: International Conference on Availability, Reliability and Security, August 27–30, 2018, Hamburg, Germany, Gokila Dorai, Shiva Houshmand, and Ibrahim Baggili (Eds.). ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3230833.3232814

Comments

Dr. Baggili was appointed to the University of New Haven's Elder Family Endowed Chair in 2015.

I Know What You Did Last Summer: Your Smart Home Internet of Things and Your iPhone Forensically Ratting You Out

Gokila Dorai
Florida State University
Tallahassee, Florida
dorai@cs.fsu.edu

Shiva Houshmand
Santa Clara University
Santa Clara, California
shoushmand@scu.edu

Ibrahim Baggili
University of New Haven
West Haven, Connecticut
IBaggili@newhaven.edu

ABSTRACT

The adoption of smart home Internet of Things (IoT) devices continues to grow. What if your devices can snitch on you and let us know where you are at any given point in time? In this work we examined the forensic artifacts produced by Nest devices, and in specific, we examined the logical backup structure of an iPhone used to control a Nest thermostat, Nest Indoor Camera and a Nest Outdoor Camera. We also integrated the Google Home Mini as another method of controlling the studied Smart Home devices. Our work is the primary account for the examination of Nest artifacts produced by an iPhone, and is also the first open source research to produce a usable forensics tool we name the Forensic Evidence Acquisition and Analysis System (FEAAS). FEAAS consolidates evidentiary data into a readable report that can infer user events (like entering or leaving a home) and what triggered an event (whether it was the Google Assistant through a voice command, or the use of an iPhone application). Our results are important for the advancement of digital forensics, as there are cases starting to emerge in which smart home IoT devices have already been used as culpatory evidence.

KEYWORDS

Internet of Things Forensics, Mobile device forensics, Nest Device Forensics, IoT Ecosystem Forensics, Digital Evidence Extraction and Analysis, Mobile Forensics

ACM Reference Format:
Gokila Dorai, Shiva Houshmand, and Ibrahim Baggili. 2018. I Know What You Did Last Summer: Your Smart Home Internet of Things and Your iPhone Forensically Ratting You Out. In *ARES 2018: International Conference on Availability, Reliability and Security, August 27–30, 2018, Hamburg, Germany,* Gokila Dorai, Shiva Houshmand, and Ibrahim Baggili (Eds.). ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3230833.3232814

1 INTRODUCTION

More than 20 billion IoT devices will be connected by 2020 according to a prediction by Gartner [13]. In North America, by 2021, more than half of all homes will become smart homes [22]. Another report by Statista [30] shows that about 25 million homes in North America will use smart thermostats by 2019. A recent study by Phadnis [26] showed that on average, there are 10 connected devices per household, including IoT streaming cameras. All these statistics are strong indicators of how IoT is making waves into our lives, and specifically our homes.

IoT devices, although may help secure and monitor homes, can also become rich sources of forensic evidence. This has proven to be the situation in recent court cases. An example is the case of Richard Debate in Connecticut, where evidence from the smart security system of a home, augmented by data from his wife's Fitbit were imperative in building a story that points to him murdering his wife [1]. One IoT device platform that is seeing wide adoption is the Nest platform.

Nest devices have become a household staple. Since 2011 Nest has sold more than 11 million devices with its first product being the thermostat [33], and was later bought by Google in 2014. With Nest devices being placed around a house, investigators can now, through circumstantial digital evidence infer when someone left a home, when they came back and when someone entered and left a room. Due to the popularity of Nest devices, the paradigm shift to a smart-home IoT forensic ecosystem (See Figure 1), and the evidence IoT devices produce, it becomes important to forensically examine IoT devices in home environments.

Our work makes the following contributions:

- To the best of our knowledge, we present the primary account for the forensics of the Nest devices and their companion devices (mobile devices used to control them).
- We built a system using open-source tools to automate the companion device acquisition, IoT device acquisition, and report engine for the Nest ecosystem[1].
- We show how inferences can be made about a user's location using the collected evidence.
- We share our dataset and associated artifacts with the forensics community to stimulate future work in this area[2].

The paper is organized as follows. In Section 2, we share background information and related works. In Section 3, we discuss various challenges in IoT forensics and different factors involved in evidence extraction from a smart-home IoT ecosystem. Section 4 describes the apparatus and software tools used in our experiments. In Section 5, we explain our methodology which consists of various phases such as scenario creation, data acquisition, data analysis, inference engine construction, report generation. Then, we describe our FEAAS automation tool. In Section 6, we discuss our findings and results, and we list details of Nest related artifacts. Finally in

ARES 2018, August 27–30, 2018, Hamburg, Germany
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6448-5/18/08…$15.00
https://doi.org/10.1145/3230833.3232814

[1] The code can be downloaded from https://github.com/gdorai/FEAAS-Tool.git
[2] The dataset can be downloaded from https://agp.newhaven.edu/

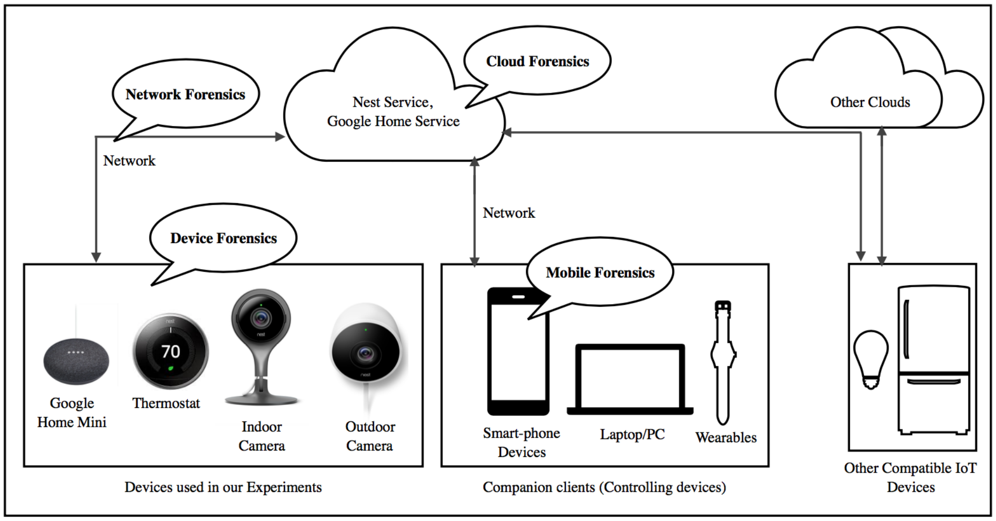


Figure 1: A paradigm for smart-home IoT forensic ecosystem. (Source of device images: *nest.com, store.google.com*)

Section 7 we discuss our work's limitations, followed by conclusion in Section 8.

2 RELATED WORK AND BACKGROUND

In this section we discuss related work in IoT forensic frameworks and digital forensics of IoT devices as well as background information related to iOS forensics.

2.1 IoT Forensics Frameworks

Researchers have designed various frameworks for conducting IoT forensics. One such model is the Forensics-aware IoT (FAIoT) model, which is a high-level model that aims to support forensic investigations by systematically analyzing the challenges and opportunities in IoT forensics [35]. The forensics process including authorization, planning, analysis and storage, has been addressed by another IoT top-down digital forensic investigation methodology [25]. Additionally, a general Digital Forensic Investigation Framework for IoT (DFIF-IoT) was also proposed describing a proactive process which occurs before IoT incident identification, and a reactive process that happens after an IoT incident is identified [18]. Our work is synonymous to the reactive process strategy described by DFIF-IoT [18].

In another effort, Meffert et al. [23] conceived the Forensic State Acquisition from Internet of Things (FSAIoT), a general framework and practical approach for IoT forensics through IoT device state acquisition. In their work they explicated that IoT state data (such as device on, or off), can be gathered and logged in real time through the use of a Forensic State Acquisition Controller (FSAC), where data can be acquired from an IoT device, the cloud, or from another controller. They did indeed leverage the Nest open APIs to pull the state of the Nest thermostat whenever data is exchanged to the cloud. However, accessing historical data and deleted data from the IoT devices was not possible. In their work, they did not focus on companion devices, such as mobile phones used to control the IoT smart home systems. Our proposed research instead focused on the extraction of evidentiary data from the companion device, the iPhone, which was used to control the IoT devices under scrutiny.

2.2 IoT and Digital Forensics

There have been previous efforts aimed at recovering forensic artifacts from IoT devices used by consumers - either from the device or from the cloud. Client-centric and cloud-native artifacts stored within companion clients of the Amazon Echo was studied and analyzed by Chung et.al [6]. Additionally, Rajewski [27] recovered artifacts from Android mobile applications used to control IoT consumer products including the WinkHub, Samsung SmartCam, Amazon Echo and Nest devices. With respect to Nest devices, user account information was recovered from the Android cache files and a cross-platform tool known as "ffmpeg" [3] was used to convert the media files stored in the image manager disk cache to obtain the video files. However, they did not focus on recovering artifacts from iOS devices. While recovering media files may be helpful for an investigation, our work adds to the body of knowledge, and provides a combined report of all the activities recorded through several IoT devices stored by the Nest mobile application.

Other work focused on the network flow of data, especially the work by Copos et al. [7], where they analyzed network traffic generated by the Nest thermostat and smoke detector and showed that traffic analysis enabled them to determine the thermostat transition between Home and Auto Away mode with 88% accuracy. This approach, however, requires setting up the network traffic monitor in advance and may not be helpful to investigators in a postmortem situation.

Central to our work is also the original work on the logical backup acquisition and analysis of iOS devices by Bader and Baggili [4]. In their work they examined the logical backup acquisition

[3] https://www.ffmpeg.org

of the iOS mobile device using the Apple iTunes backup utility. The authors found significant data of forensic value in the logical backup, which we have adopted in our work. Similarly, Husain et al. [14] presented a simple and inexpensive framework for forensic acquisition of iPhone devices. This work also provided useful insights related to the production of a forensically sound report for use in the court. Using their framework *iFF*, they were able to extract plain text passwords, conversation, user accounts, etc., using the iTunes backup utility from non-jailbroken devices.

At the time of writing, there was no published account that examined digital evidence acquired from iOS companion devices used to control Nest IoT platforms, whilst also constructing a tool to automate the forensic analysis process - a gap that this work fills.

3 IOT FORENSIC CHALLENGES

Recent advancements in systems and the emergence of IoT has led to a scientifically unexplored area in digital forensics. In this section, we discuss the prominent challenges involved in IoT forensics which includes the diversity of protocols, volatile nature of IoT data as well as various considerations in smart-home IoT ecosystems.

3.1 Diversity of Protocols

The operating system loaded on Nest devices remains proprietary to Nest Labs, although it is based on Linux. These devices interconnect with each other using a protocol called *Weave* [20]. Weave is a compact message format and communicates only when it needs to. Home devices like Honeywell products use the *Z-Wave* [34] protocol, and other smart-home devices use the *Zigbee* [36] communication protocol. The non-standardization of communication protocols and closed system architectures create unprecedented forensic challenges.

3.2 Data Volatility

In a typical digital forensic investigation, Chain of Custody (CoC) of the digital evidence is followed by preservation, collection, examination, analysis and presentation of evidence. Whereas, in an IoT environment, maintaining CoC of the evidence and preserving it may prove cumbersome. Many IoT devices use Real Time Operating Systems (RTOSs) which do not persist data. Evidence acquisition can be unreliable due to the transient nature of IoT ad-hoc network connectivity. Above all, researchers and forensic experts have recently examined and defined digital forensic investigation frameworks specific to IoT [18, 23, 35] to deal with the volatile nature of the evidence. Until a widely acceptable framework is adopted, IoT investigations will remain time consuming and complex.

3.3 Smart Home IoT Ecosystems

Generally speaking, to recover evidence from a smart home IoT platform, four items should be considered: *(1) Companion-Client Analysis*: This is used to identify client-centric artifacts from mobile devices or Web browsers. Significant data associated with Nest device activities may be available, but consolidating the evidence is challenging. *(2) Cloud Analysis*: This will be helpful in order to analyze artifacts stored in the Cloud by providing valid user credentials for authentication. We observed that it was not possible to alter historical data on *home.nest.com* for thermostat. Whereas, deleting historical video-data is possible.*(3) Network Analysis*: Network monitoring and analysis of network traffic is helpful in reconnaissance, collection of usage statistics, software and system logs which are uploaded to the cloud through the network. *(4) IoT Device Analysis*: Analyzing the memory of IoT devices may help reveal data stored in the device's memory. Since many devices do not have persistent storage, and in specific, Nest devices do not have a memory card, we focused our attention to data that may be acquired from mobile companion devices.

4 TEST ENVIRONMENT

In this section we list the details of the test environment used in our experiments. The apparatus is listed in Table 1. In our work we used the Nest 3rd generation learning thermostat, a Nest indoor camera, a Nest outdoor camera and a Google Home Mini speaker. We note that, we tried to integrate Apple's HomePod [16] in our experiments, but Nest devices were not compatible with HomeKit [15]. Hence, we opted to use the Google Home Mini since it was compatible with Nest devices. The list of tools used in our experiments is shown in Table 2.

Table 1: Test Environment

Item	Description
IoT Devices	(1) Nest Learning Thermostat, 3rd Generation (Model Number: T3007ES), (Quantity: 1) (2) Google Home Mini (Model Number: GA00216-US), (Quantity: 1) (3) Nest Cam Indoor Security Camera (Model Number: NC21102ES), (Quantity: 1) (4) Nest Outdoor Security Camera (Model Number: NC2100ES), (Quantity: 1)
Other Devices/Apps	(1) iPhone 8 (iOS 11.2.6) (2) iPhone 6s (iOS 11.2.5) (3) iPhone 6 (iOS 10.3.3) (4) Mac OS X (v10.12.6) (5) Lenovo Y510P - Ubuntu (v16.04) (6) Nest iOS app (v5.19.1) (7) Google Home iOS app (v1.28.508)
Testing period	2018-03-15 to 2018-04-05

5 METHODOLOGY

We built an open-source forensic tool called Forensic Evidence Acquisition and Analysis System (FEAAS), that consolidates evidentiary data into a readable report that can infer user events. In this section we discuss the methodology used for building this tool in five major phases: scenario creation, data acquisition, data analysis, inference engine construction and report engine construction.

5.1 Scenario Creation

During this study, the Nest IoT devices were controlled using several methods such as through the Nest iOS application and Web application, by voice commands using Google Home Mini, and lastly by manually adjusting the device. Note that in our experiments, we did not have any other human subjects involved other

Table 2: Software Tools Used

Software Tool	Description	Version
DB Browser for SQLite	An open source, freeware visual tool used to design and edit SQLite database files [28]	3.10.0
PLIST Editor	Downloaded from Mac App store. Used to visualize and edit property list (.plist) data	1.6 (8)
iTunes	Utility to create backup of iPhones [2]	12.7.2.58
SQLite3	View SQLite databases [29]	3.20.1
Google Chrome	Used as web browser and analyzing cache files	63.0.3239.84
Epoch Converter	Used to convert Unix Epoch time format into local time [8]	-
ChromagnonCache	Set of tools used for Chrome forensics [5]	-

than the authors. We have created scenarios in order to validate initial findings in SQLite files. The following scenarios were executed to control the thermostat and the camera:

- Manual calibration by dialing the thermostat ring on the Nest thermostat for adjusting the temperature.
- Adjusting the temperature by talking to Google Home Mini.
- Adjusting the temperature by logging into Nest website.
- Adjusting the temperature by using the Nest iOS application.
- Changing the camera's settings by logging into Nest website.
- Changing the camera's settings by using the Nest iOS application.

Note that the Nest camera does not have a physical button to turn it on/off or make any other changes. It is also important to note that even though a Google Home Mini was added to the environment and the Nest application was configured through it, there was no voice command allowed to turn the camera on/off. However, the camera's live video can be streamed on a display using voice commands.

We also list a specific set of experiments conducted on the thermostat and camera for a period of about 20 days. In the items listed below, items 1 through 7 were performed on the thermostat device. Items 8 and 9 are the set of experiments performed on the camera device. These experiments were coined with the central idea of figuring out where the state changes were recorded in the mobile backup based on user's actions and location.

(1) Raise/lower the current temperature to a certain degree.
(2) Change the settings from heat to cool.
(3) Turn Away(Eco) mode ON manually.
(4) User was connected to a Wi-Fi network.
(5) User left the home and returned at a certain time of the day. This was conducted on 03/31/18.
(6) User was driving when the phone is connected to a cellular network.
(7) User was at a location where there was no Internet connection. The cellular data was also not used in this scenario. This was conducted on 03/30/18.
(8) Turning the camera on/off at some point in the day using web application, and the mobile app.
(9) Defining two activity zones for camera to get notification about motions in the zone.

Since the Nest mobile application has the option to track a user's phone location using Geo-fencing [3], we can observe the entry and exit of a user from the fence (200m around the location where the device is pinned). If the user has opted out of this option in the mobile application, then the motion sensor embedded within the IoT device is used to sense the presence of the user. In our work we opted to use the Geo-fence instead of the motion sensor. The Nest application was given permission to use the mobile phone's location services.

5.2 Data Acquisition

In this phase, the user's iPhone device was connected to a forensic workstation using a Universal Serial Bus (USB) cable and a script was run in order to obtain a complete unencrypted backup of the device. The artifacts were extracted using an open source cross-platform software protocol library *libimobiledevice* which can communicate with iOS devices [32]. We then employed a utility called *idevicebackup2* in order to create a backup of the iOS device [31]. The backed up file names are SHA-1 hash values (hexadecimals with 40 characters each). Upon further examination, we note that each file was named as the result of a SHA-1 hash computed on the file name in the format *domain-[subdomain-]fullpath/filename.ext.*

We used a script to automate the process of locating the Nest artifacts (nest.sqlite database file, plist file, GooseEventLogging file) and Google Home Mini artifacts (HomeGraphModel database file) in the iOS backup. Data obtained using the iOS backup utility contained multiple files with information about the thermostat and camera data, Google analytics configuration data, list of connected devices and user preference information, etc.

We located the Nest app user data in the App Group *com.nestlabs.jasper.release* and Google Home app user data in the App Group *com.google.Chromecast* in the backup folder by mapping the equivalent SHA-1 of the file path. Each new version of the Nest or Google Home Mini application may result in database structure variations. For example, new tables may be added to the database. The table names may also vary. Hence, each time these applications are updated by the iTunes utility, we ensure that our FEAAS tool has been updated to produce expected results.

Table 3 lists various files used in our study to recover useful information. UDID represents the *Unique Device Identifier* of the iPhone. We also share the respective SHA-1 values of the files in the backup. As a useful insight to investigators, we note that the path names (except UDID) remain the same when locating artifacts. In our experiments with multiple iOS devices, the backup folder paths remained the same, with the exception of the UDID since it is unique to every iPhone and can be found easily when an iPhone is connected to the iTunes utility [24].

5.3 Data Analysis

The iOS applications used for controlling IoT devices were found to use *SQLite databases*. Most of the evidence data was found in these database files. Property List (Plist) files were analyzed using a Plist Editor. The *Info.plist* file contained the iPhone's build version, International Mobile Equipment Identity (IMEI), phone number,

Table 3: Location of Client-centric Artifacts

Full File Path in Mac-OSX	Filename	Used to obtain
[BaseFolder]/	Info.plist, Status.plist, Manifest.db	iOS Device info and backup status
[BaseFolder]/c5/c5ad63c1c7304bbc53dcd4ac9b7a35060450f8e7	Nest.sqlite	(Nest) Thermostat and camera activities
[BaseFolder]/e1/e1da0e82d74bf22b60dc11a27bedda82f6525590	com.nestlabs.jasper.release.plist	(Nest) Device setup information
[BaseFolder]/47/47df8f986e7ae5ca105bc12f3deb7f2499464f76	GooseEventLogging	(Nest) Network type, geo-fence log
[BaseFolder]/9f/9fce4727c3b25498e410a27309e0de3f7c6affad	HomeGraphModel	(Google Home) Lists connected devices
[BaseFolder]/1d/1d75375510a87429993ca573ad7c883f8195ce72	com.google.Chromecast.plist	(Google Home) Device setup information
/Users/[Username]/Library/Caches/Google/Chrome/Default/Cache/	Chrome cache	Native artifacts in cache data frames

*Note: iOS backup files were located in Mac OSX file path /Users/[Name]/ Library/Application Support/MobileSync/Backup/ followed by the UDID. For presentation in the table, we list this path as the [*BaseFolder*].

last backup date, product version, product type, UDID, sync settings and a list of applications installed on the device. *Status.plist* contained vital information about the backup date and status of the backup such as whether or not a full backup was performed. This information is required to ensure we have the complete set of data needed from the phone. The plist files within the IoT mobile application (see Table 3) were parsed to obtain the version of the mobile application, the date of the device activation, the date of application installation, user account and the device pairing token. In the next section, we list the inference rules constructed into the automation script. In Section 6, we discuss our findings and results of data analysis in detail.

5.4 Inference Engine Construction

In order to produce a report of user's location and a time-line of series of events, we created a set of inference rules. The inference engine is an intermediate step between analyzing the data and creating the report, and can also be viewed as part of the report generation. The rules allow us to create a human-readable report from the raw data. We process the data present in the *GooseEventLogging* file which contains information about the network type of user's iPhone for a certain time-stamp. We specifically parse the file to obtain data about *FenceEvent* and *FenceReport*. We then apply our inference rules on this information to produce the report. Various network types which we identified by parsing the GooseEventLogging file are Cellular, Wi-Fi or No Connection.

Algorithm 1 illustrates the rules as conditional statements. In this pseudocode, we obtain information about the type of network to which the user's iPhone is connected (lines 11-15), and print user's location with respect to where the Nest device is installed and based on the fence events (lines 21-37). We parse the GooseEventLogging file to construct a list of relevant events about network type from the overall FenceReport. Then, we identify whether the FenceEvent type is Enter or Exit. Once the fence event type is identified, we infer whether the user has left home, or has just arrived based on the previous status of a user's location.

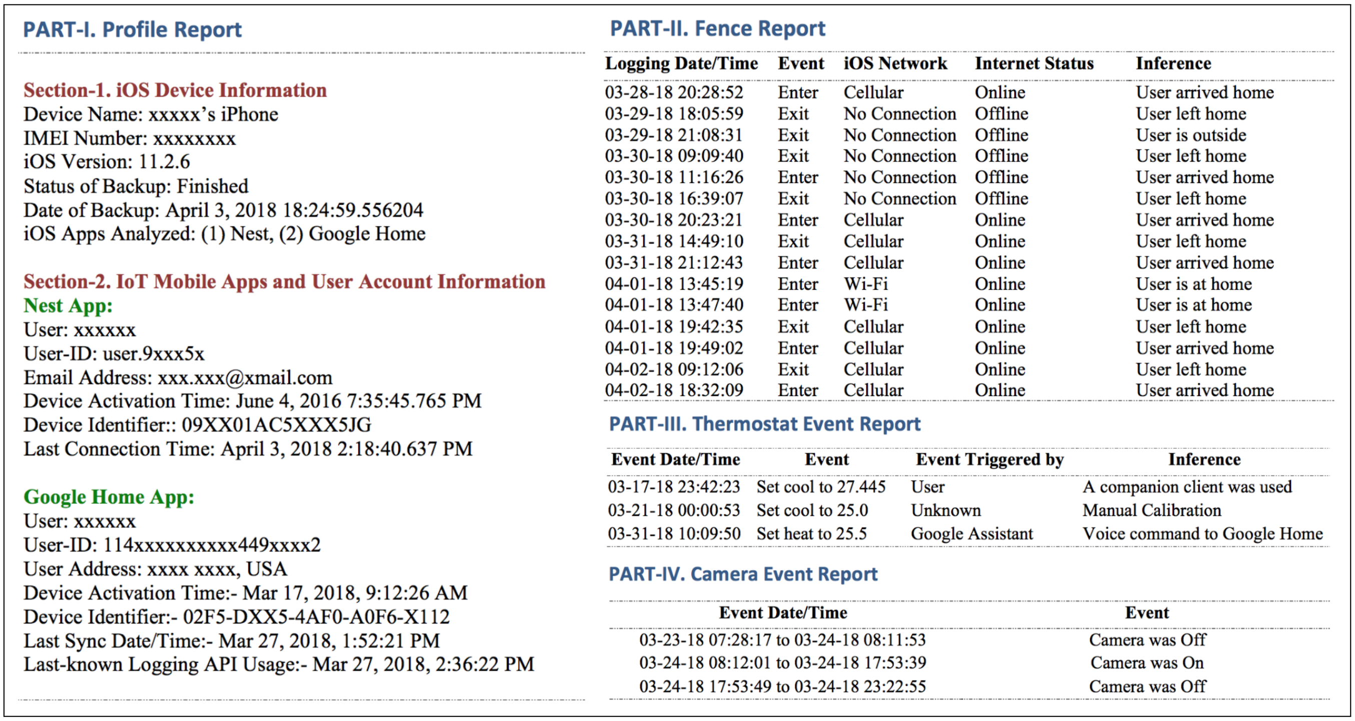
Algorithm 1 Pseudocode for Inference Engine

```
 1: $eventid \leftarrow 0$
 2: $eventIndexList \leftarrow []$
 3: $relevantEvents \leftarrow []$
 4: for all $item$ in $allEvents$ do
 5:   if $item.event$ is $FenceEvent$ then
 6:     $relevantEvents$.append($item$)
 7:     $eventIndexList$.append($eventid$)
 8:     $eventid{+}{+}$
 9:   else if $item.event$ is $FenceReport$ then
10:     for all $i$ in $eventIndexList$ do
11:       if $item.NetworkType$ is $No\ Connection$ then
12:         $relevantEvents[i].InternetStatus \leftarrow Offline$
13:       else
14:         $relevantEvents[i].InternetStatus \leftarrow Online$
15:       end if
16:     end for
17:     $eventIndexList \leftarrow []$
18:   end if
19: end for
20: $atHome \leftarrow False$
21: for all $item$ in $relevantEvents$ do
22:   if $item.type$ is $ENTER$ then
23:     if $atHome$ is $True$ then
24:       $print$ "$User\ is\ at\ home$."
25:     else
26:       $print$ "$User\ arrived\ home$."
27:       $atHome \leftarrow True$
28:     end if
29:   else if $item.type$ is $EXIT$ then
30:     if $atHome$ is $True$ then
31:       $print$ "$User\ left\ home$."
32:       $atHome \leftarrow False$
33:     else
34:       $print$ "$User\ is\ outside$."
35:     end if
36:   end if
37: end for
```

5.5 Report Engine Construction

After parsing the data and applying the inference rules, the report engine produces a PDF report. Figure 2 shows an example of a portion of the report generated using this engine. The following are a few examples of information that can be observed from the report:

- Knowing the fact that Google Home Mini has been configured to control the Nest thermostat and the mobile backup data shows evidence of the thermostat being adjusted using



PART-I. Profile Report

Section-1. iOS Device Information
Device Name: xxxxx's iPhone
IMEI Number: xxxxxxxx
iOS Version: 11.2.6
Status of Backup: Finished
Date of Backup: April 3, 2018 18:24:59.556204
iOS Apps Analyzed: (1) Nest, (2) Google Home

Section-2. IoT Mobile Apps and User Account Information
Nest App:
User: xxxxxx
User-ID: user.9xxx5x
Email Address: xxx.xxx@xmail.com
Device Activation Time: June 4, 2016 7:35:45.765 PM
Device Identifier:: 09XX01AC5XXX5JG
Last Connection Time: April 3, 2018 2:18:40.637 PM

Google Home App:
User: xxxxxx
User-ID: 114xxxxxxxxxx449xxxx2
User Address: xxxx xxxx, USA
Device Activation Time:- Mar 17, 2018, 9:12:26 AM
Device Identifier:- 02F5-DXX5-4AF0-A0F6-X112
Last Sync Date/Time:- Mar 27, 2018, 1:52:21 PM
Last-known Logging API Usage:- Mar 27, 2018, 2:36:22 PM

PART-II. Fence Report

Logging Date/Time	Event	iOS Network	Internet Status	Inference
03-28-18 20:28:52	Enter	Cellular	Online	User arrived home
03-29-18 18:05:59	Exit	No Connection	Offline	User left home
03-29-18 21:08:31	Exit	No Connection	Offline	User is outside
03-30-18 09:09:40	Exit	No Connection	Offline	User left home
03-30-18 11:16:26	Enter	No Connection	Offline	User arrived home
03-30-18 16:39:07	Exit	No Connection	Offline	User left home
03-30-18 20:23:21	Enter	Cellular	Online	User arrived home
03-31-18 14:49:10	Exit	Cellular	Online	User left home
03-31-18 21:12:43	Enter	Cellular	Online	User arrived home
04-01-18 13:45:19	Enter	Wi-Fi	Online	User is at home
04-01-18 13:47:40	Enter	Wi-Fi	Online	User is at home
04-01-18 19:42:35	Exit	Cellular	Online	User left home
04-01-18 19:49:02	Enter	Cellular	Online	User arrived home
04-02-18 09:12:06	Exit	Cellular	Online	User left home
04-02-18 18:32:09	Enter	Cellular	Online	User arrived home

PART-III. Thermostat Event Report

Event Date/Time	Event	Event Triggered by	Inference
03-17-18 23:42:23	Set cool to 27.445	User	A companion client was used
03-21-18 00:00:53	Set cool to 25.0	Unknown	Manual Calibration
03-31-18 10:09:50	Set heat to 25.5	Google Assistant	Voice command to Google Home

PART-IV. Camera Event Report

Event Date/Time	Event
03-23-18 07:28:17 to 03-24-18 08:11:53	Camera was Off
03-24-18 08:12:01 to 03-24-18 17:53:39	Camera was On
03-24-18 17:53:49 to 03-24-18 23:22:55	Camera was Off

Figure 2: A sample FEAAS report

voice commands, implies that someone has been present at home at that specific time (and has adjusted the thermostat).

- The evidence shown in the backup files about HOME/AWAY information recorded by the Nest app, implies whether the user has been in close proximity of the house or not at a specific time.
- The fact that the mobile phone was connected to a Wi-Fi, and the Internet status was *Online* along with fence-event as *Enter* at a specific time implies that the user was at home.

This report contains critical information to aid an investigation along with the timestamps, events, inferences, user information and device information. This report can be used by investigators for reporting CoC and analyzing events. Inference column in the report can provide useful clues to the investigator.

5.6 FEAAS Tool

The FEASS tool we built is a command-line program written using Shell script (version 3.2.57) and Python (version 3.6.2). The automation script is named *AutomationScript.sh* which installs essential utilities and python packages such as libimobiledevice [32], carthage [9], biplist [11] and reportlab [12]. The package Carthage is a dependency manager; Biplist is a Python package which is used to parse binary plist and Reportlab is a Python tookit which is an open-source library for generating PDFs. The AutomationScript.sh then runs another shell script called *DevicePairing.sh* in order to pair the iPhone device with the computer. Once the user establishes trust between the computer and iPhone device by clicking "Trust" button on the iPhone, the device will be paired with the computer. Then, the script will start acquiring an unencrypted backup of the iPhone device.

On successful completion of backup acquisition, a python script will parse the backup files in order to locate and analyze artifacts related to the IoT applications (Nest app and Google Home Mini). *ParserThermostat.py* is a Python script which is used to parse the thermostat-related data whereas *ParserCamera.py* is used parse the camera-related data. The AutomationScript requires one input parameter; a path to the iOS backup files. The output of the tool is a report file containing all the fence events and device events in PDF format (See Figure 2.) The tool also has the feature of producing a ".csv" report file. Fence events and device events are ordered chronologically in the report along with inferences.

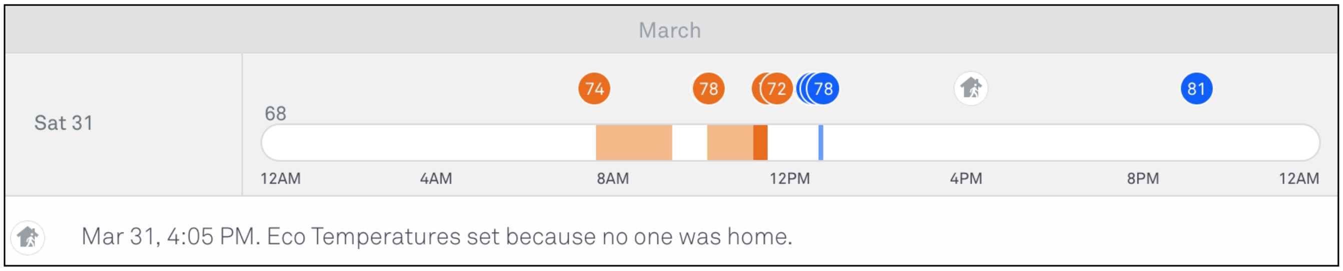
6 DISCUSSIONS AND RESULTS

As mentioned before, the Nest iOS application manages most of the information in a database file called Nest.sqlite. We found that this database contain hundreds of tables. The event-related tables mostly have start date/time and end date/time in Unix epoch format [8]. Significant tables and their relevant fields are listed in Table 4.

There are three preliminary tables such as Z_PRIMARYKEY, Z_METADATA and Z_MODELCACHE. Z_PRIMARYKEY contains the index of database tables. The table Z_METADATA contains hashes of table model version identifiers (*NSStoreModelVersionHashes*, that is, if the backing data store has support for schema migration, the automatically generated model version hashes are stored in the backing store) [17].

Table 4: Overview of Relevant Tables in Nest Database

Table	Relevant Fields
ZCDSTRUCTURE	ZCREATION_TIME - Installation time of the app.
ZCDSTRUCTURE	ZPOSTAL_CODE - Postal code entered during device activation.
ZCDSTRUCTURE	ZCOUNTRY_CODE - Country code.
ZCDSTRUCTURE	ZSTRUCTUREID - Unique identifier of the user account associated with the device.
ZCDGEOFENCE	ZLATITUDE, ZLONGITUDE - Location to which the device was pinned.
ZCDUSERSESSION	ZNAME - Name associated with the user.
ZCDUSERSESSION	ZUSER, ZUSERID - Identifier used for updating user actions on the device.
ZCDUSERSESSION	ZEMAIL - Email id of the user.
ZCDUSERSESSION	ZPRIMARY_PHONE - Primary phone number of the user.
ZCDUSERSESSION	ZPROFILE_IMAGE_URL - Profile picture of the user.
ZCDBASEDEVICE	ZCREATIONTIME - Installation time of the device.
ZCDBASEDEVICE	ZLASTCONNECTIONTIME - Last time when the device communicated with the cloud.
ZCDBASEDEVICE	ZLOCALIPADDRESS - Local IP Address used.
ZCDBASEDEVICE	ZMAC_ADDRESS - MAC address.
ZCDENERGYEVENT	ZTOUCHEDWHEN - Various event time-stamp
ZCDENERGYEVENT	ZCOOLTEMP, ZHEATTEMP - Temperature changes
ZCDENERGYEVENT	ZTOUCHEDID - What caused the adjustment to the device? (User/Google Assistant/Other)
ZCDSCRUBBYCHUNKINFO	ZSTARTDATE, ZENDDATE - Start/End time of the chunk of camera video data.
ZCDSCHEDULESETPOINT	ZTOUCHEDAT, ZTEMPERATURE - An overview of temperature schedule by (epoch) date.



Mar 31, 4:05 PM. Eco Temperatures set because no one was home.

Figure 3: Nest website snapshot for experiment #5

The ZCDSTRUCTURE table reveals the postal code, time zone, city and the date of device activation. In order to reveal the details of a user, it is sufficient to examine ZCDUSERSESSION where the e-mail of the user and the profile images are saved in columns ZEMAIL and ZPROFILE_IMAGE_URL respectively. The URLs used for uploading logs and transporting objects are found in table ZCDUSERSESSIONURLS. The table ZCDWHERE has entries to map location names such as Bedroom, Entryway, Family Room, Front Yard, etc. to a WHERE_ID, which is used in the ZCDBASEDEVICE table to indicate where the device is installed. This helps us identify the specific location of an activity.

The table ZCDGEOFENCE stores the physical location in terms of latitude/longitude where a device is pinned. The temperature adjustments to the thermostat are recorded in the ZCDENERGYEVENT table along with timestamps. Suppose, if multiple thermostats are used in a smart home, then, ZCDENERGYEVENT also relates temperature adjustment to thermostats based on locations defined by the user (for example, upstairs/downstairs). ZCDSCHEDULE contains information about the current schedule mode set in the respective thermostats (For example: Schedule mode can be Cool/Heat). The table ZCDSCHEDULESETPOINT lists the temperature adjustments by userID. For example, user identified by user-id (such as “user.9****55”) performed a temperature adjustment and a history of maximum/minimum temperatures.

The table ZCDSCRUBBYCHUNKINFO contains information about video frames transferred to the cloud at certain time intervals and can especially be useful for investigations. The table contains ZCHUNKIDENTIFIER which is an epoch time indicating the start date of the chunk, ZSTARTDATE, and ZENDDATE. Each row indicates a chunk of 20 frames. Analyzing the difference between the end date and the start date shows that if the camera is ON, the difference for each row is about 19-21 minutes. In case the camera is OFF at some intervals, this difference is much larger than 21 minutes, since during that period the data is not transferred to the cloud. The large time difference can be an indication to the investigator that the camera has been turned off during that period.

In order to validate the results obtained from our Report Engine, we kept detailed journal notes in which we had recorded all user actions as part of the experiments. We can view the device events by accessing the historical time-line information from the cloud using the Nest Home website. The history (available for the last 10 days) clearly shows whether the thermostat temperature schedule was set by Auto-schedule, manual calibration, Nest companion applications or Eco temperature was set because no one was home.

Here we discuss a few examples of the experiments we did to verify the report that was generated by our tool with information available on the mobile app, the cloud and our detailed journal notes which contains the ground truth. In experiment #5 (as described in Section 5.1), where the user left the home at a certain time in a day and arrived later at a certain time was correctly acquired from the cloud since the user's phone location was ON and the phone was using cellular data. This is recorded by the mobile application as well as the cloud. Our parsing tool was also able to obtain this data. Figure 3 shows a snapshot of the Nest thermostat website where we can observe that *no one was home* where a gray home symbol inside a circle is found between the time 1PM and 9PM. The same can be observed in the report (Figure 2 Part-II) where a fence event "Exit" happened on 03-31-18 14:49:10, and a fence event "Enter" happened at 21:12:43.

The *GooseEventLogging* file has record of up to 10-days of data and the *nest.sqlite* database has all the data starting from the installation date of the app. In experiment #7 (described in Section 5.1), where the user was in a location away from home with iPhone not connected to either Wi-Fi or Cellular network, the phone's location could not be uploaded to the cloud and hence, AWAY mode was not triggered automatically. We can observe this in the report where the entry "No Connection" can be found on 03-30-18 09:09:40. Also, whenever the user's iPhone could not upload data to the cloud right-away, a log was found in *GooseEventLogging* file. This data appears to be a queued-up data for the *key: FenceReport* in *GooseEventLogging* file.

When the iPhone is connected to the internet, this data queue gets uploaded to the cloud and leaves a trace in the FenceReport. Experiment #7 is an interesting example where we can infer that the user was not home (by looking into the mobile backup data) but this could not be observed in the cloud right away. Therefore, in a way, the mobile data can provide more insight in an abnormal scenario where the user's mobile application was running in the background in order to log the phone's location but could not upload it to the cloud immediately. Figure 4 shows a snapshot of this scenario where the thermostat had been running throughout the day without automatically setting "Eco" temperature in AWAY mode. The results in the report (see Figure 2 Part-II,-III,-IV) matched the data stored in the cloud. This shows that we were able to recover most of the artifacts from the mobile backup in normal operation mode.

The list of devices that are configured to be controlled by Google Home Mini (voice commands) can be identified by parsing the *HomeGraphModel* database. The table ZDEVICE has information about the connected IoT devices *(nest-home-assistant-prod)* in ZAGENTID field. In the same table, the field ZDEVICETYPE lists the role of connected devices such as *action.devices.types.THERMOSTAT and SPEAKER.* The table ZTRAIT helps identify the role of these connected devices in the field ZIDENTIFICATION such as *Assistant* and *TemperatureSetting*. The user and device profile information was recovered using the *com.google.Chromecast.plist* file.

It is essential to show that the data collected from the mobile backup is associated with the evidential IoT device. Hence, we mounted the thermostat and the camera on a Linux file system (Ubuntu 16.04) by connecting the device to the computer using a USB cable. We recovered a file called *Technical Info.plist* and *Info.plist* from the thermostat and the camera accordingly. The table ZCDBASEDEVICE has data in the columns such as ZIDENTIFIER, ZDIAMONDBACKPLATE matched the data we recovered from the *Techinal Info.plist* file. ZIDENTIFIER is the serial number of the display and ZDIAMONDBACKPLATE is the serial number of the base of the thermostat. The MAC address retrieved from the Technical Info.plist file and the table ZCDBASEDEVICE was found to be the same. This is helpful in tying an evidential device to a phone.

6.1 Camera Artifacts Recovered from Chrome Cache

Since our approach focused on the logical acquisition of data which does not include deleted data, we turned our focus to the potential of reconstructing deleted artifacts from Chrome cache. As mentioned earlier, Nest users can access their devices through the *home.nest.com* website to view videos recorded by Nest camera. When a user uses a browser like Google Chrome for this purpose, website and cookie data (such as JavaScript scripts, graphic images, html files, multimedia contents), will be available in cache. The results described in this section are not a part of our FEAAS tool nor the inference engine, since the tool mainly focuses on analyzing the artifacts from iOS devices. To be able to analyze artifacts from Chrome Cache, the investigator needs to have access to the computer (in this case, a MacBook) used by the Nest user. However, the results in this section could be of importance, especially in cases when some video clips have been deleted from the account, but when Chrome cache preserved the data. This technique can then be used to recover these deleted artifacts.

We examined Google Chrome cache to obtain such artifacts. As per the description of Chrome Disk Cache Format [10], cache files include the index file (which is a hash table of Uniform Resource Locater (URL) of objects stored in it), data block files, and separate data stream files. Data block files with file names data_# store small objects. Separate data streams are contained inside the files named f_######. We were also able to find compressed JSON strings (content-encoding was in gzip format and content-type was JSON format).

We employed an open source tool called Chromagnon[4] in order to parse Chrome cache files stored on Mac OSX. The following command *$ python2 chromagnonCache.py /User/[Username]/Library /Caches/Google/Chrome/Default/Cache -o ParsedCache/* was used in the terminal for parsing cache files from the default location. The expected lifetime of these artifacts in normal operational mode is until the user clears the browsing history and cache files.

Using the aforementioned command, the Chromagnon parser [5] creates a directory called *ParsedCache* where we can find cache artifacts. This directory contains an index.html file and several other files. The index.html file contain links to all cached data (such as text files, image files, event clips, profile picture of the user, etc). To specifically locate artifacts related to event clips and images, we filtered the html links to find the term "get_event_clip" and "get_image" respectively. By clicking on the filtered link, we were able to locate the file containing the link to the preview of the video clip (with 9 frames in each). Figure 5 is a snapshot of

[4] https://github.com/JRBANCEL/Chromagnon

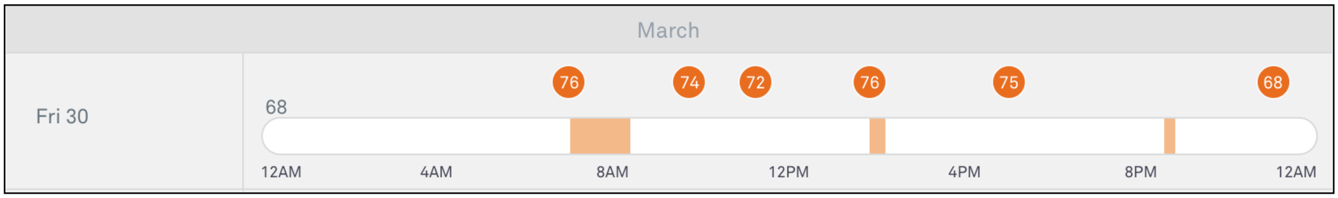
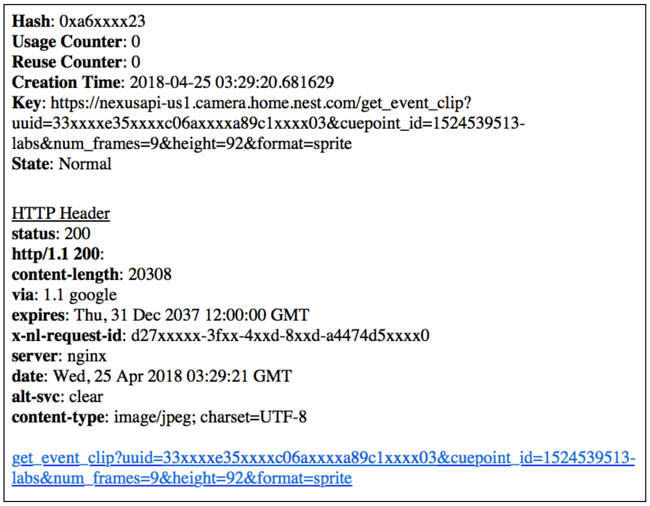


Figure 4: Nest website snapshot for experiment #7



```
Hash: 0xa6xxxx23
Usage Counter: 0
Reuse Counter: 0
Creation Time: 2018-04-25 03:29:20.681629
Key: https://nexusapi-us1.camera.home.nest.com/get_event_clip?
uuid=33xxxxe35xxxxc06axxxxa89c1xxxx03&cuepoint_id=1524539513-
labs&num_frames=9&height=92&format=sprite
State: Normal


HTTP Header
status: 200
http/1.1 200:
content-length: 20308
via: 1.1 google
expires: Thu, 31 Dec 2037 12:00:00 GMT
x-nl-request-id: d27xxxxx-3fxx-4xxd-8xxd-a4474d5xxxx0
server: nginx
date: Wed, 25 Apr 2018 03:29:21 GMT
alt-svc: clear
content-type: image/jpeg; charset=UTF-8

get_event_clip?uuid=33xxxxe35xxxxc06axxxxa89c1xxxx03&cuepoint_id=1524539513-
labs&num_frames=9&height=92&format=sprite
```

Figure 5: Snapshot of a Cache-file

a file present in *ParsedCache* (note that we masked some of the characters in the snapshot). We can observe that this file contains some critical details like when the video was captured in epoch time(*cuepoint_id* in "Key") and also when the video data was last accessed ("Creation Time") in GMT. By looking into these files we were able to recover images and event clips stored in the cache. Figure 6 shows an example of an image of the lab recorded through the Nest Indoor Camera which we were able to obtain from the cache.

This technique would be very useful to investigators for recovering artifacts post-deletion since the camera's video history and snapshots are unrecoverable (even from the cloud) once they are deleted [19].

7 LIMITATIONS

This work has several limitations. The first is that if different versions of the iPhone operating system and future versions of the Nest applications are used, our results may vary. The second being that our study was limited to iPhones. Lastly, our work is focused on data that is logically acquired from the mobile device, which means that it will only work if data has not been deleted from the phone under examination.

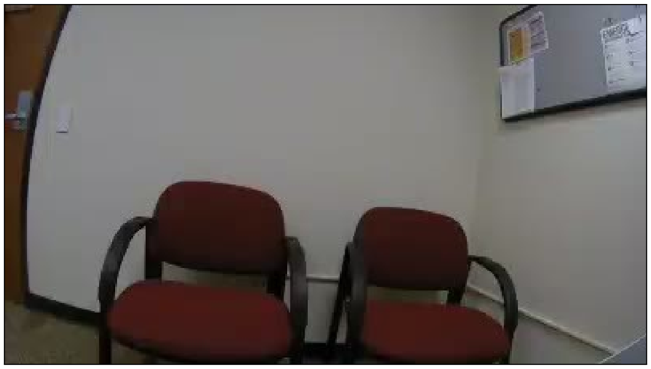


Figure 6: Recovered Chrome Cache Artifact

8 CONCLUSION AND FUTURE WORK

People are starting to adopt smart home technologies. Almost all smart home devices are connected to an application on the mobile device for ease of use and control, therefore, automating the forensic analysis of the mobile devices in the IoT ecosystem becomes crucial. We documented our approach for recovering artifacts from the mobile device. We were able to find relevant data from databases recovered from the mobile device's backup that show information such as the date and time at which certain changes were made to the Nest devices. This includes whether anyone manually calibrated the thermostat, whether the user was present at home during a certain time, whether the user spoke to the Google Home Mini device to make any adjustments to the thermostat, and whether the camera was intentionally turned off at a certain time.

In the future, we would like to focus more on the cloud side of the analysis, especially for the Google Home Mini product. Most of the data is more easily available on the cloud, but accessing cloud requires a *Subpoena*. We also would like to analyze IoT mobile applications in order to obtain the alert notifications received by a user (such as, a sound heard, or a movement on a specific zone detected, or someone at the door, etc). We aim to conduct this study on a jail-broken device. We also plan to study other Nest devices such as the Nest Doorbell and Alarm System in order to acquire more data that can be inferred as a result of the *interaction* of all the IoT devices in a smart home. Also, in our future work we will conduct experiments using the HomeBridge Nest plugin [21] in order to study the exchange of information between HomePod and connected devices.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Sudhir Aggarwal for his valuable suggestions and insightful comments on this work, and Manuel Hernandez for helping in the development of the tool.

REFERENCES

[1] Dave Altimari. 2018. More Evidence Turned Over In Fitbit Murder Case. (Jan 2018). http://www.courant.com/news/connecticut/hc-news-dabate-fit-bit-murder-continued-20180119-story.html

[2] Apple. 2017. iTunes. https://www.apple.com/itunes/. (2017).

[3] Apple. Accessed: 03-07-2018. Obtain the geographic location and orientation of a device. https://developer.apple.com/documentation/corelocation. (Accessed: 03-07-2018).

[4] Mona Bader and Ibrahim Baggili. 2010. iPhone 3GS forensics: Logical analysis using apple iTunes backup utility. (2010).

[5] Jean-Remy Bancel. 2018. Chrome Chromium Forensic Tool : Parses History, Visited Links, Downloaded Files and Cache. (2018). Retrieved April 21, 2018 from https://github.com/JRBANCEL/Chromagnon

[6] Hyunji Chung, Jungheum Park, and Sangjin Lee. 2017. Digital forensic approaches for Amazon Alexa ecosystem. *Digital Investigation* 22 (2017), S15–S25.

[7] Bogdan Copos, Karl Levitt, Matt Bishop, and Jeff Rowe. 2016. Is Anybody Home? Inferring Activity From Smart Home Network Traffic. In *Security and Privacy Workshops (SPW), 2016 IEEE.* IEEE, 245–251.

[8] Epochconverter.org. 2017. Epoch Converter - Unix Timestamp Converter. https://www.epochconverter.com. (2017).

[9] Matt Diephouse et.al. 2018. A simple, decentralized dependency manager. (2018). Retrieved April 01, 2018 from https://github.com/Carthage/Carthage

[10] ForensicsWiki. Accessed: 01-05-2018. Chrome Disk Cache Format. http://www.forensicswiki.org/wiki/Chrome_Disk_Cache_Format. (Accessed: 01-05-2018).

[11] Python Software Foundation. 2018. Biplist. (2018). Retrieved April 01, 2018 from https://pypi.org/project/biplist/

[12] Python Software Foundation. 2018. Reportlab. (2018). Retrieved April 01, 2018 from https://pypi.org/project/reportlab/

[13] Gartner. 2017. 8.4 Billion Connected Things. https://www.gartner.com/newsroom/id/3598917. (2017). Accessed: 03-04-2018.

[14] 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.

[15] Apple Inc. 2018. HomeKit. (2018). Retrieved February 2, 2018 from https://developer.apple.com/homekit/

[16] Apple Inc. 2018. HomePod. (2018). Retrieved February 2, 2018 from https://www.apple.com/homepod/

[17] Apple Inc. 2018. Implementation Strategy. (2018). Retrieved March 22, 2018 from https://developer.apple.com/library/content/documentation/DataManagement/Conceptual/IncrementalStorePG/ImplementationStrategy/ImplementationStrategy.html

[18] Victor R Kebande and Indrakshi Ray. 2016. A generic digital forensic investigation framework for internet of things (iot). In *Future Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on.* IEEE, 356–362.

[19] Nest Labs. 2018. How to delete your camera video history and snapshots. (2018). Retrieved April 21, 2018 from https://nest.com/support/article/Can-I-delete-my-Nest-Cam-Video-History

[20] Nest Labs. 2018. OpenWeave. (2018). Retrieved March 21, 2018 from https://github.com/openweave/openweave-core

[21] Kraig M. 2018. homebridge-nest. (2018). Retrieved February 3, 2018 from https://github.com/KraigM/homebridge-nest

[22] Chase Martin. 2017. 73 Million Smart Homes In North America Projected. https://www.mediapost.com/publications/article/304304. (2017).

[23] Christopher Meffert, Devon Clark, Ibrahim Baggili, and Frank Breitinger. 2017. Forensic State Acquisition from Internet of Things (FSAIoT): A general framework and practical approach for IoT forensics through IoT device state acquisition. In *Proceedings of the 12th International Conference on Availability, Reliability and Security.* ACM, 56.

[24] What's my UDID? Accessed: 03-06-2018. So an iOS dev asked you for your UDID. http://whatsmyudid.com. (Accessed: 03-06-2018).

[25] Sundresan Perumal, Norita Md Norwawi, and Valliappan Raman. 2015. Internet of Things (IoT) digital forensic investigation model: Top-down forensic approach methodology. In *Digital Information Processing and Communications (ICDIPC), 2015 Fifth International Conference on.* IEEE, 19–23.

[26] Shilpa Phadnis. 2016. Households have 10 connected devices now, will rise to 50 by 2020. (2016). Retrieved April 3, 2018 from https://cio.economictimes.indiatimes.com/news/internet-of-things/households-have-10-connected-devices-now-will-rise-to-50-by-2020/53765773

[27] J.T. Rajewski. 2016. Internet of things forensics at Enfuse 2016. A Presentation at Enfuse. (2016).

[28] Sqlitebrowser.org. 2017. DB Browser for SQLite. http://sqlitebrowser.org. (2017).

[29] Sqlite.org. 2017. Sqlite database. https://www.sqlite.org. (2017).

[30] Statista. 2016. Homes with smart thermostats in the North America 2014-2020. (2016). Retrieved March 28, 2018 from https://www.statista.com/statistics/625868/homes-with-smart-thermostats-in-north-america/

[31] Martin Szulecki. 2018. idevicebackup2. (2018). Retrieved February 13, 2018 from http://manpages.ubuntu.com/manpages/artful/man1/idevicebackup2.1.html

[32] Martin Szulecki. 2018. libimobiledevice. (2018). Retrieved February 12, 2018 from http://www.libimobiledevice.org

[33] Megan Wollerton. 2018. Nest says it has sold over 11 million devices since 2011. (2018). Retrieved March 28, 2018 from https://www.cnet.com/news/nest-says-it-has-sold-over-11-million-devices-since-2011/

[34] Z-Wave. 2018. Safer, smarter homes start with Z-Wave. (2018). Retrieved March 28, 2018 from http://www.z-wave.com

[35] Shams Zawoad and Ragib Hasan. 2015. FAIoT: Towards Building a Forensics Aware Eco System for the Internet of Things. In *Services Computing (SCC), 2015 IEEE International Conference on.* IEEE, 279–284.

[36] Zigbee. 2017. What is Zigbee. (2017). Retrieved March 28, 2018 from http://www.zigbee.org