IoT Digital Forensics Readiness in the Edge: A Roadmap for Acquiring Digital Evidences from Intelligent Smart Applications

Andrii Shalaginov[1], Asif Iqbal[2], and Johannes Olegård[2]

[1] Norwegian University of Science and Technology, Gjøvik, Norway
`andrii.shalaginov@ntnu.no`
[2] KTH Royal Institute of Technology, Stockholm, Sweden
`asif.iqbal@ee.kth.se,jolegard@kth.se`

Abstract. Entering the era of the Internet of Things, the traditional Computer Forensics is no longer as trivial as decades ago with a rather limited pool of possible computer components. It has been demonstrated recently how the complexity and advancement of IoT are being used by malicious actors attack digital and physical infrastructures and systems. The investigative methodology, therefore, faces multiple challenges related to the fact that billions of interconnected devices generate tiny pieces of data that easily comprehend the Big Data paradigm. As a result, Computer Forensics is no longer a simple methodology of the straightforward process. In this paper, we study the complexity and readiness of community-accepted devices in a smart application towards assistance in criminal investigations. In particular, we present a clear methodology and involved tools related to Smart Applications. Relevant artefacts are discussed and analysed using the prism of the Digital Forensics Process. This research contributes towards increased awareness of the IoT Forensics in the Edge, corresponding challenges and opportunities.

1 Introduction

Internet of Things (IoT) has brought virtually unlimited possibilities for the development of smart applications that target everyday's life improvement. With broadening horizons of such architecture's flexibility and automation, one can create a nearly autonomous system on private, corporate and national scales. While, undoubtedly, it is a positive development, there are few growing concerns from the perspective of security and safety. Diversity of used technologies a novel cyber threat landscape with a high chance of zero-day vulnerabilities and novel attack scenarios (e.g., Mirai botnet [15,4]). Moreover, it creates new challenges as well as opportunities for Crime Investigations involving the IoT ecosystem.

Despite the fact that each individual IoT Edge device is generally simpler than laptops or large-scale servers, there exist multiple limitations and challenges. Speaking of Digital Forensics Process [9,18,13], *Identification* phase might hit the wall when trying to identify the relevant information, where it stored originally in the IoT ecosystem and to whom it belongs, *Preservation* of the data might not be easy due to the fact that pieces of data are spread across multiple instances that do not relate to timestamps or data custodians in explicit ways, *Aggregation* of multiple pieces of data can easily end up being Big Data paradigm with the relevance and "needle in a haystack" issue, *Analysis* can yield irrelevant results, result from the lack of understanding of used IoT ecosystem, legally-binding agreement and personal data protection regulations. Finally, Smart Applications might introduce a layer of Machine Learning (ML) / Artificial Intelligence (AI) models, which require knowledge expert to interpret, possibly reverse-engineer and provide explainable answers on why the data have been processed and the automated decision been made [26,27].

Unlike other research papers in the area, this contribution seeks an understanding of the forensics readiness of interconnected devices in the Edge, rather than of a stand-alone device. The focus is on the open-source platforms to create a corresponding road map for the investigative process along with recommendations. In particular, it was (i) created a Smart Application (environment sensors control) scenario with Edge devices involving IoT hub (Raspberry Pi) and IoT end-point devices (Arduino Uno and ESP8266), (ii) performed step-by-step collection of the artefacts in accordance to Digital Forensics Process and (iii) analysis of the acquired digital evidences from interconnected Edge devices using open-source existing tools. Finally, a special focus will be given to understanding and reverse-engineering ML/AI software found along with raw data on IoT devices' storage, considering omnipresence of such software in IoT appliances. The paper is organized as follows. The Section 2 presents an overview of the IoT forensics approaches and known artefacts that can be found across various types of devices. Further, Section 3 gives an overview of the suggested use case and particular methodological steps on Edge devices, while Section 4 included results of the Digital Forensics Process steps with corresponding artefacts from Smart Applications. Finally, Section 5 and 6 discusses IoT forensics readiness and overall preparedness of the Edge ecosystem for investigations.

2 Related Works

Due to widespread demand, cheap hardware, large community support and available plug-in components, the amount of IoT devices is growing exponentially, reaching 18 bln devices by 2022 according to Ericsson market study [8]. Out of those, 16 bln devices will be attributed to short-range devices. It means that rapid development of 4G+ (and 5G) connectivity and low-power wide range communication protocols (BLE - Bluetooth Low Energy, LoRa, IQRF and ZigBee) make geographically distributed communication very easy both in economic sense and deployment costs [22]. Therefore, it is important to keep in mind that IoT forensics is not only device-focused investigation, yet rather network- and cloud-oriented, as in the case with Edge devices.

"*Edge devices*" or "*Edge computing*" transformed the way how the data are handled in the distributed sensors networks. This technology brought disruption into data processing in a way that the data can be processed in the network while being transmitted from end-node IoT devices to data processing hubs and "*data lakes*" [24]. With all possible open-source platforms, sensors and actuators, the Edge devices are placed on the crossroads of all data streams flowing up and down the IoT ecosystem. The data that is transmitted and processed are ranging from the trivial regular (M2M) communication containing sensor measurements to user-related sensitive data containing credentials and personal information [23].

2.1 IoT Forensics: State of the Art

It is important to understand that the requirements for Digital Forensics methodologies that were used in the 1990s or even 1980s were completely different from what they are now, i.e. completely different from what is required in 2020s [20]. It is mainly due to the fact that cheap and easy-to-use technologies made it possible to generate an extreme amount of data, creating a paradigm of so-called Big Data. The amount of data will be grown bringing more challenges to police investigating cases, leaving the only way out is to adopt new technologies capable of processing and extracting real-world data.

Digital Forensics over the course of last decade became multi-faceted science targeting all possible aspects of discovering digital traces in a large set of data found at or attributed to a crime scene [17]. Speaking of Smart Applications from the infrastructure level, we can say that there are multiple components, resembling general IoT ecosystem: *Cloud data storage, IoT gateways, IoT nodes, sensors* and *actuators* dealing with tasks such as *data acquisition, aggregation, processing, analysis* and *data-driven decision making* [6]. Moreover, it is important to understand the challenges in Digital Forensics related to such versatile, multi-platform and cross-domain infrastructure. There are considerable differences in the computational and data processing capabilities of both IoT nodes, also called micro-controllers (Micro Controller Units, or MCU) and IoT gateways, also called micro-computers (System on a Chip, or SoC). Even though the size of the stored data is relatively small on each device, it can easily enter the Big Data paradigm once the Smart infrastructure has multiple nodes in the network. In addition, IoT brings challenges to digital pieces of evidence preservation attributed to boundary-less networks distributed over large regions. Therefore, *1-2-3 zone model* [18] of digital forensics investigation was suggested and covers all digital data found in fog, cloud, routers and gateway servers. This model differentiates between internal, middle and outer networks, however, still being complex. To facilitate flexibility, there was suggested a paradigm of "Fragile Evidence Zone" and [19] as a prerequisite for the data accumulation platform.

2.2 IoT Forensics: Order of Volatility and Data Preservation

Edge Computing is the intrinsically agile environment and most of the data can be considered as dynamic, bringing more constraints on the data identification and collection during digital forensics process. From the perspective of Digital Forensics, this can be a challenge due to the fact that the Order of Volatility needs to be maintained in a general way: Random Access Memory (RAM), network traffic, disk storage, etc [5]. However, one of the difficulties is related to the fact that the size of the storage in the Edge devices makes it impossible to store all the data. Therefore, only a few data pieces can be stored for a longer period, while others are gone. Another difficulty is that the crucial for forensics "*timestamps*" are not available on the devices like micro-controllers due to limited computational and storage capacity [14]. Even though, there exist timekeeping functionalities like Time Library for Arduino[3], those do not commonly use to avoid unnecessary delays and computing overhead. As a result, the timestamps might be inevitably lost even if the files exist. At the same time, micro-computer platforms like Orange Pi or Raspberry Pi provide full support to maintain current and updated in a real-time manned time and data, leaving IoT gateway/hub tiers as the last with reasonably trustworthy timestamps. The amount of data available in each component differs a lot: from 32 KBytes of flash and 2KBytes of RAM (Arduino) to 256 Gbytes of SD card and 4Gbytes of RAM (Raspberry Pi). To the authors knowledge, there have not been explored enough analysis of digital pieces of evidence in a set of Smart Applications in Edge with respect to different types of memory defined in order of volatility. One of the community-accepted tools that can be used on microcomputers to facilitate the preservation of Order of volatility is "Forensics Mode" in Kali Linux[4].

3 Use Case and Methodology

The main goal of this paper is to demonstrate possible ways of extracting relevant digital pieces of evidence from MCU and SoC devices found in the Edge. There will be given a general road map for such data acquisition on Smart Applications in the Edge, also taking in mind that ML / AI is one of the intelligent components of such an interconnected network. The peculiarity of DF in Smart Applications that we want to specifically highlight in this paper is the presence of intelligent models that were trained from the data. Currently, there exist a large number of intelligent ML models that can be used for *clustering, classification and* [16] such as *Artificial Neural Network (ANN), Support Vector Machines (SVM), Bayesian Network (BN), Hidden Markov Models (HMM).* Technically speaking, ML can be trained from any kind of real-world data to be able to give a prediction regarding new previously unseen data sample. Moreover, according to Deloitte [25], 80% of enterprise IoT solutions will include ML / AI components. Therefore, this section presents the way how ML models can be used in the Smart Application and how those can be integrated into the DF process as a *digital evidence*.

3.1 Smart Applications: Edge Devices

A scenario that we tailored is smart application closed-loop controller (IoT hub) that handles communication with IoT end-node devices harvesting environmen-

[3] `https://playground.arduino.cc/Code/Time/`
[4] `https://docs.kali.org/general-use/kali-linux-forensics-mode`

tal parameters such as temperature and humidity. Such scenario generally consist of the following components on a high level: *sensor* - a simple passive circuit that "senses" physical world measurements and converts to electrical signal, *controller* - devices that make decision based on the input sensor measurements and *actuator* - a simple active device that can do physical actions upon controlling electrical signal. This interconnected network also provides a *feedback loop*, while actuator performing actions based on the measurements from sensors [7]. Basic communication in such scenarios is usually organized via Message Queuing Telemetry Transport (MQTT) protocol as a lightweight and reliable solution for transmission of sensor data and actuators commands. An example of Smart Application-based systems is shown in the Figure 1.

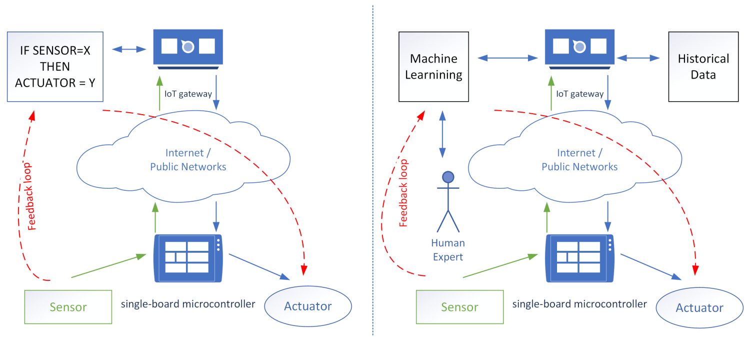

IF SENSOR=X THEN ACTUATOR = Y
IoT gateway
Internet / Public Networks
Feedback loop
Sensor
single-board microcontroller
Actuator
Machine Learning
Historical Data
IoT gateway
Internet / Public Networks
Feedback loop
Human Expert
Sensor
single-board microcontroller
Actuator

Fig. 1. Regular rule-based (*left*) and intelligent Machine Learning-based (*right*) implementation of the feedback loop in Smart Applications

3.2 Experimental Environment

For the demonstration purpose, we model a core architecture of the *Smart Home*, also found in other domains - intelligent monitoring with a *feedback loop*. An imaginary scenario was developed (inspired by [28]), where the IoT node uses the ML model trained on the IoT gateway to protect against cyber attacks. The diagram of the experimental installation shown in the Figure 2. As mentioned earlier, this includes the following components with computational capabilities:

IoT hub/gateway is implemented using *Raspberry Pi 3 Model B* (1.2 GHz Quad-Core CPU, 1GB RAM, 16GB MicroSD card, Ethernet). It is one of the most lightweight and low-end SoC with minimal system components required to run OS. It has Raspbian 4.19 (Debian Buster 4.19) installed. To demonstrate also intelligent application, we have ported *ArduinoANN* project[5] from AVR to ARM to specifically simulate ANN training step using Mosquittopp v.1.5.7-1 (MQTT version 3.1/3.1.1 client C++ library and corresponding Broker), Boost v.1.67.0-13 for uBLAS vector storage and JSON serialization, g++ v. 4:8.3.0-1 as a compiler. This node was assigned IP address 192.168.0.200. SoC needs a reliable power supply and, in some cases, can run from the battery.

[5] `http://robotics.hobbizine.com/arduinoann.html`

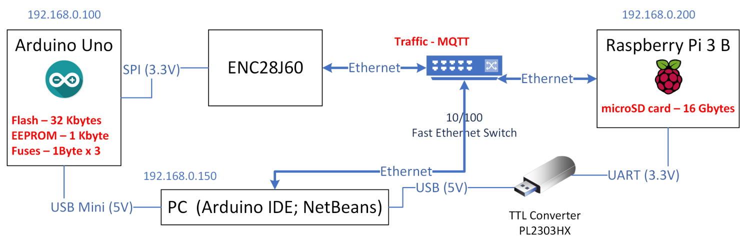


Fig. 2. Experimental setup used in this paper, including IoT node and IoT gateway. Types of the media with corresponding size are denoted in *red font*

IoT node is implemented using *Arduino Uno rev. 3* (16MHz CPU, 2KBytes RAM, 32KBytes, ENC28j60 Ethernet board). ArduinoIDE 1.8.9 was]used to program Arduino UNO. The communication protocol implementation was done using the following libraries: UIPEthernet 2.0.7 to work with ENC28J60 Ethernet controller, PubSubClient 2.7.0 and ArduinoJson 6.11.5, while the testing phase was inspired by *ArduinoANN* project. To reduce the size of the code and accommodate larger packers, the configuration of the first and second libraries were changed to UID_CONF_UDP_CONNS =1 and MQTT_MAX_PACKET_SIZE = 256 respectively. This node was assigned to address 192.168.0.100. It was used Ethernet protocol instead of WiFi on Arduino to set up the baseline. It has faster initialization, lower price and simpler connection routine for the experiment phase. MCU can easily run from battery or solar power.

Stand-alone IoT device without ML. To contrast the rest of the paper, we will study a stand-alone IoT device that does not use ML. The diagram of the experimental installation shown in the Figure 3. The device is connected to a WiFi network and sends data directly (i.e. without an intermediate edge-device) to a remote server. The device has a ESP32-WROOM-32D [10] (2.4GHz and 520KiB RAM) processor and a 16 MiB flash memory. It does not run a full operating system. This makes the device more powerful than the Arduino and but less powerful than the raspberry pi. It behaves like a MCU but has capabilities closer to a SoC. The device is used to measure temperature and humidity. The device has an Universal Asynchronous Receiver/Transmitter (UART) interface.

3.3 Digital Forensics Process in the Internet of Things

IoT Digital Forensics is a relatively new field were two important issues have to be counted in: *firstly*, limited computational capabilities does not allow to implement security mechanisms exposing more data to forensics investigators, *secondly*, previously unseen and undocumented proprietary technologies will delay data analysis requiring an additional level of reverse-engineering. Due to connectivity and versatility of the devices, it becomes a real challenge to fix the baseline in the so-called Digital Forensics process [12]. Despite the lack of standardization, improper pieces of evidence handing and challenging chain of custody in IoT ecosystem, the authors emphasized that *pre-investigation readiness*

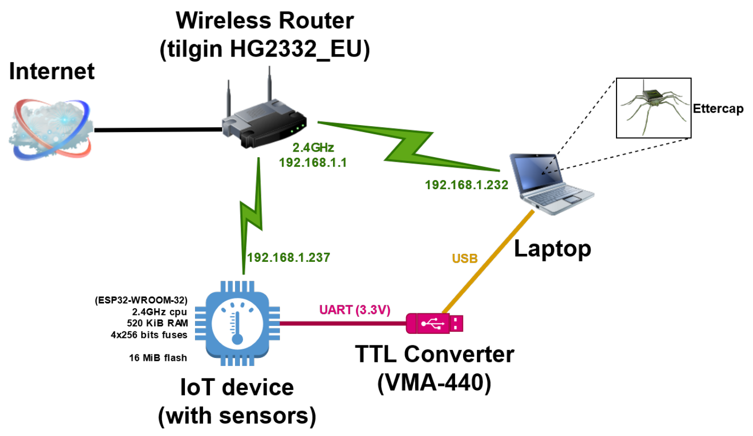


Fig. 3. Experimental setup for the stand-alone IoT device without ML

and *real-time integration* will be a key to successful digital forensics investigation in IoT in future [31]. From the perspective of the Digital Forensics, there can be highlighted multiple phases suggested over the last decades [30]. However, from the approach strategy, one needs to define the following stages:

Field work and Acquisition. This stage includes *Identification, Preservation and Collection* of the Smart Devices from the crime scene, identifying their relevance and storing according to predefined Chain of Custody.

Lab work and Analysis. During this stage *Examination, Analysis and Presentation* are performed, when a Forensics Investigator extracted data from Smart Devices and tries to link them together attributing found pieces of evidence to a crime scene.

4 Computer Forensics Investigation in the Edge

This section will explain step-by-step digital pieces of evidence analysis and roadmap for extracting data in the Edge. As was mentioned before, we emphasize the importance of understanding how the ML models work and their particular place in the IoT. While speaking about IoT, there can be seen three following areas where data normally exist: *cloud, network and devices*, while IoT digital forensics readiness is still a challenge as explained by Alenezi et al. [3]. Particularly, there is a need to maintain log files, transmit relevant data and store timestamps, that might require modification of the IoT infrastructure. Therefore, we will be going step by step in this section over the approaches to acquire and then analyze any relevant data from Smart Applications.

4.1 Field work and Acquisition

As also mentioned by Goudbeek et al. for the Smart Home case [11], there has to be followed a routine for proper analysis of the digital pieces of evidence. Every investigation starts with the preparation and analysis of the smart infrastructure, identifying components, preparing necessary hardware and software tools.

Components identification and attribution. A person, who deals with the Smart Application setup, needs to clearly identify used components, sensors, actuators, connectivity and possible information flow and anticipated logic of the system. The picture of the aforementioned setup is shown in the Figure 4. In this case, all the devices have labels and distinct logos, making them easy to identify and reveal capabilities and technical characteristics.

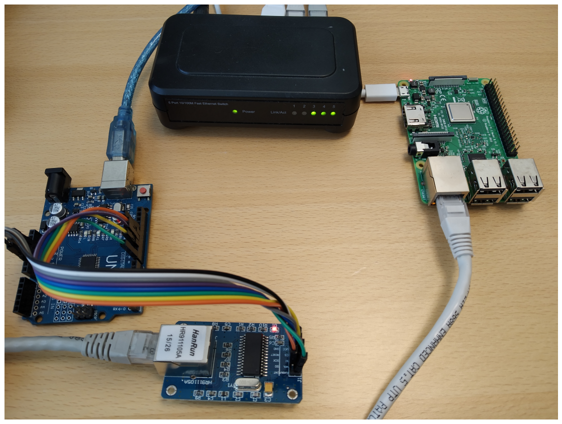
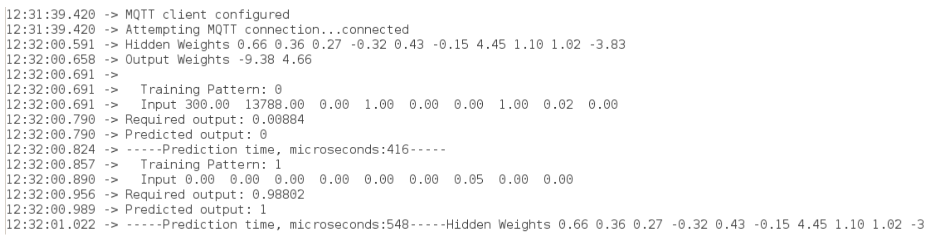


Fig. 4. Photo of our experimental setup

Live device on-chip access via JTAG[6]*/ISP*[7]*/TTL*[8]. The next step is to assess whether data can be retrieved when the devices are connected and in *live* mode. It is of crucial importance to follow the *Order of Volatility* as defined by Hegarty et al. [13] to ensure the preservation of all digital data that might be stored either in the permanent memory like flash/SD card or in memory, while will be available only when the device is powered one. Since Arduino has serial communication available, it is possible to connect to the terminal and observe actual communication that is shown in the Figure 5. From this, it is understandable that MCU is trying to establish MQTT communication and receives data from another component, also successfully identifying some given input pattern. Further, Raspberry Pi board has UART communication, which can be accessed through USB to TTL adapter by connecting TX/RX/GND pins to board pins 6,8 and 10 respectively. Access SoC via UART is given via Linux terminal: *sudo chmod 666 /dev/ttyUSB1; screen /dev/ttyUSB1 115200*

This process requires an understanding of what kind of boards and technologies are used. Therefore, there can be accessed documentation and discussion

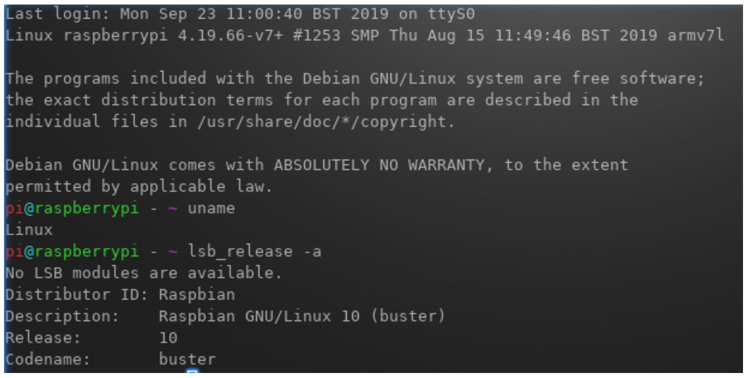
[6] Joint Test Action Group standard
[7] In System Programmer
[8] Universal Asynchronous Receiver/Transmitter serial convertor



```
12:31:39.420 -> MQTT client configured
12:31:39.420 -> Attempting MQTT connection...connected
12:32:00.591 -> Hidden Weights 0.66 0.36 0.27 -0.32 0.43 -0.15 4.45 1.10 1.02 -3.83
12:32:00.658 -> Output Weights -9.38 4.66
12:32:00.691 ->
12:32:00.691 ->   Training Pattern: 0
12:32:00.691 ->   Input 300.00  13788.00  0.00  1.00  0.00  0.00  1.00  0.02  0.00
12:32:00.790 -> Required output: 0.00884
12:32:00.790 -> Predicted output: 0
12:32:00.824 -> -----Prediction time, microseconds:416-----
12:32:00.857 ->   Training Pattern: 1
12:32:00.890 ->   Input 0.00  0.00  0.00  0.00  0.00  0.00  0.05  0.00  0.00
12:32:00.956 -> Required output: 0.98802
12:32:00.989 -> Predicted output: 1
12:32:01.022 -> -----Prediction time, microseconds:548-----Hidden Weights 0.66 0.36 0.27 -0.32 0.43 -0.15 4.45 1.10 1.02 -3
```

Fig. 5. Serial monitor from Arduino IDE showing output of live MCU

forums for each particular component. In our case, by using standard password/login it was possible to log in to the system and check OS information as shown in the Figure 6.



```
Last login: Mon Sep 23 11:00:40 BST 2019 on ttyS0
Linux raspberrypi 4.19.66-v7+ #1253 SMP Thu Aug 15 11:49:46 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
pi@raspberrypi - ~ uname
Linux
pi@raspberrypi - ~ lsb_release -a
No LSB modules are available.
Distributor ID: Raspbian
Description:    Raspbian GNU/Linux 10 (buster)
Release:        10
Codename:       buster
```

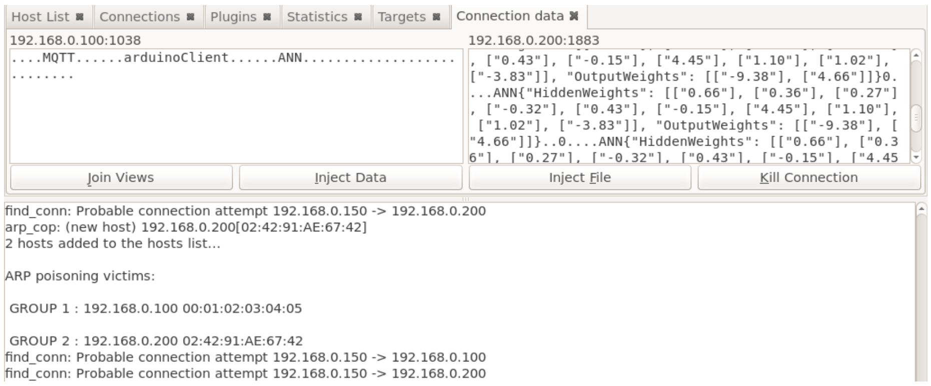
Fig. 6. UART communication with SoC displaying access terminal and OS information

Live acquisition of the network traffic. Finally, one of the important sources for digital evidential data is network communication within the given IoT infrastructure. To acquire necessary data and technical details, we used Address Resolution Protocol (ARP) Spoofing approach with a help of EtterCAP[9] tool that allows sniffing of a stitched network. The result of the program execution is shown in the Figure 7.

From EtterCAP we note two important issues: there are two devices with IP addressed *192.168.0.100, 192.168.0.200* that have regular MQTT communication transferring weights (parameters) of the ANN.

Stand-alone IoT device without ML. We use a TTL-to-USB adapter to connect to the UART interface as shown in the Figure 8. Using the "screen" com-

[9] `https://www.sans.org/reading-room/whitepapers/tools/ettercap-primer-1406`



```
192.168.0.100:1038
....MQTT......arduinoClient......ANN...................
........
```

```
192.168.0.200:1883
, ["0.43"], ["-0.15"], ["4.45"], ["1.10"], ["1.02"],
["-3.83"]], "OutputWeights": [["-9.38"], ["4.66"]]}0.
...ANN{"HiddenWeights": [["0.66"], ["0.36"], ["0.27"]
, ["-0.32"], ["0.43"], ["-0.15"], ["4.45"], ["1.10"],
 ["1.02"], ["-3.83"]], "OutputWeights": [["-9.38"], [
"4.66"]]}..0....ANN{"HiddenWeights": [["0.66"], ["0.3
6"], ["0.27"], ["-0.32"], ["0.43"], ["-0.15"], ["4.45
```

```
find_conn: Probable connection attempt 192.168.0.150 -> 192.168.0.200
arp_cop: (new host) 192.168.0.200[02:42:91:AE:67:42]
2 hosts added to the hosts list...

ARP poisoning victims:

 GROUP 1 : 192.168.0.100 00:01:02:03:04:05

 GROUP 2 : 192.168.0.200 02:42:91:AE:67:42
find_conn: Probable connection attempt 192.168.0.150 -> 192.168.0.100
find_conn: Probable connection attempt 192.168.0.150 -> 192.168.0.200
```

Fig. 7. EtterCAP: intercepted communication between IoT node and IoT gateway

mand we were able to observe sensor output, including timestamps. We connected a laptop to the wifi network. We used ettercap ARP poisoning to capture and record (using tcpdump) the traffic between the router and the device. We were able to capture encrypted UDP-packets between the device and a remote server. Using esptool [2] we were able to read the flash memory over the UART interface. Esptool will inject code into the device to achieve this and reboot the device before and after each operation. It is therefore important to respect the order of volatility. Esptool can also read the virtual memory of the device, but will overwrite some of it in the process.

```
$ screen /dev/ttyUSB0 115200
$ esptool -p /dev/ttyUSB0 dump_mem 0x3ff8_0000 0x142000 esp32_mem1.hex
$ esptool -p /dev/ttyUSB0 dump_mem 0x5000_0000 0x2000 esp32_mem2.hex
$ esptool -p /dev/ttyUSB0 read_flash 0 0x1000000 esp32_flash.hex
$ espefuse -p /dev/ttyUSB0 summary > esp32_efuses.txt
```

Listing 1.1. Retrieving dump of the flash memory from ESP32-WROOM-32D using *esptool*

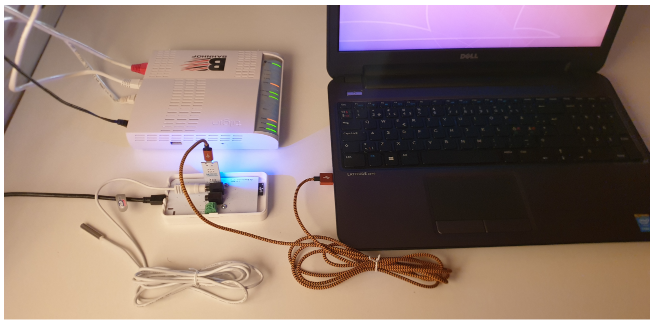


Fig. 8. Extracting data from stand-alone IoT device without ML component

4.2 Lab Work and Analysis

Once *field* work has been finished, the DF process continues with the retrieving and analysis of digital data on each device separately.

IoT node artifacts MCU is, often, custom-made hardware components with proprietary software that will certainly delay the analysis phase. However, there exist common approaches to dump the chip firmware and analyse available data. In our setup, we utilize Avrdude[10] tool with Arduino as AVR in-system programming technique (ISP) to retrieve the data to dump the content of the memory on Arduino Uno as suggested by *Arduino Forensics* [1].

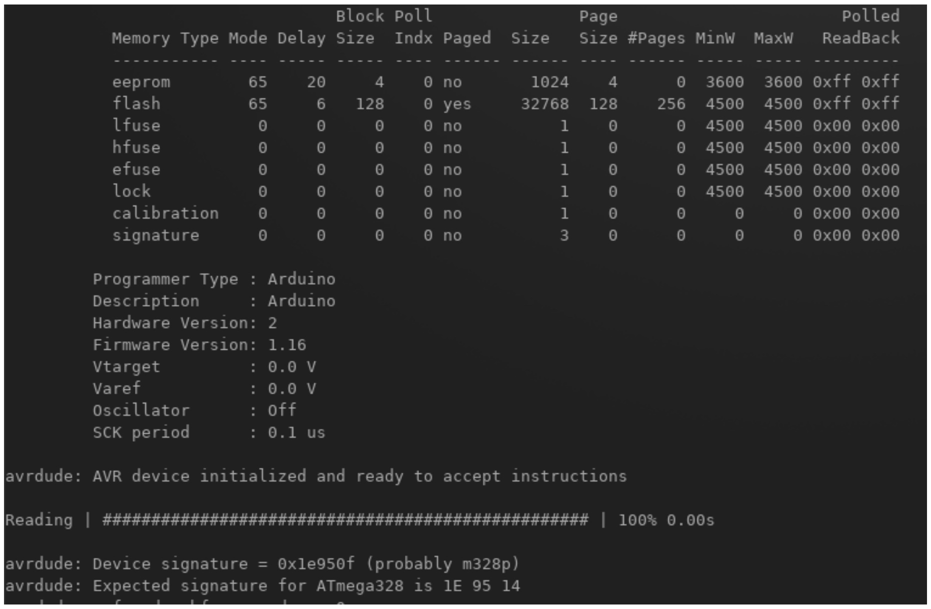
```
$ avrdude -P /dev/ttyUSB0 -F -v -v -c arduino -pm328 -D -Uflash:r:
    ↪ arduino_dump.hex:r
```

Listing 1.2. Retrieving dump of the flash memory from Arduino Uno using *avrdude*

```
$ md5sum: 219de396b61bd3feefc064295fa53828  arduino_dump.hex
$ ls -la: -rw-rw-r-- 1  32652 nov.  21 12:17 arduino_dump.hex
```

Listing 1.3. Characteristics of retrieved Arduino flash dump file

The *avrdude* software gives an overview of available memory sections (including flash, EEPROM, fuses, etc) and corresponding hardware device signatures as shown in the Figure 9.

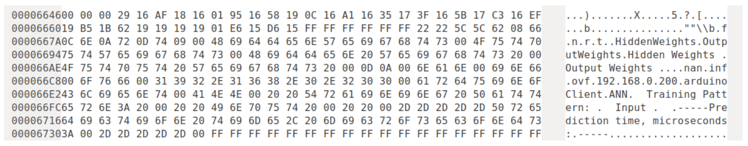


```
                               Block Poll               Page                       Polled
           Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  MaxW   ReadBack
           ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- ----- ---------
           eeprom        65    20     4    0 no       1024    4      0  3600  3600 0xff 0xff
           flash         65     6   128    0 yes     32768  128    256  4500  4500 0xff 0xff
           lfuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           hfuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           efuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           lock           0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           calibration    0     0     0    0 no          1    0      0     0     0 0x00 0x00
           signature      0     0     0    0 no          3    0      0     0     0 0x00 0x00

         Programmer Type : Arduino
         Description     : Arduino
         Hardware Version: 2
         Firmware Version: 1.16
         Vtarget         : 0.0 V
         Varef           : 0.0 V
         Oscillator      : Off
         SCK period      : 0.1 us

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: Expected signature for ATmega328 is 1E 95 14
```

Fig. 9. Output of the *Avrdude* software verbose output

[10] https://www.nongnu.org/avrdude/

Once the data has been dumped, we use GHex to analyse the content. The first thing that comes to our attention - hard-coded comments about MQTT and IP address as shown in the Figure 10. Despite the fact that we are not able to retrieve any kind of log files or timestamps, available data can give a clear picture of the device's functionality. Further reverse-engineering and analysis of code will help the investigation, however, takes much more time.



```
0000664600 00 00 29 16 AF 18 16 01 95 16 58 19 0C 16 A1 16 35 17 3F 16 5B 17 C3 16 EF    ...).......X.....5.?.[....
0000666019 B5 1B 62 19 19 19 19 01 E6 15 D6 15 FF FF FF FF FF FF 22 22 5C 5C 62 08 66    ...b..............""\\b.f
0000667A0C 6E 0A 72 0D 74 09 00 48 69 64 64 65 6E 57 65 69 67 68 74 73 00 4F 75 74 70    .n.r.t..HiddenWeights.Outp
0000669475 74 57 65 69 67 68 74 73 00 48 69 64 64 65 6E 20 57 65 69 67 68 74 73 20 00    utWeights.Hidden Weights .
000066AE4F 75 74 70 75 74 20 57 65 69 67 68 74 73 20 00 0D 0A 00 6E 61 6E 00 69 6E 66    Output Weights ....nan.inf
000066C800 6F 76 66 00 31 39 32 2E 31 36 38 2E 30 2E 32 30 30 00 61 72 64 75 69 6E 6F    .ovf.192.168.0.200.arduino
000066E243 6C 69 65 6E 74 00 41 4E 4E 00 20 20 54 72 61 69 6E 69 6E 67 20 50 61 74 74    Client.ANN.  Training Patt
000066FC65 72 6E 3A 20 00 20 20 49 6E 70 75 74 20 00 20 20 00 2D 2D 2D 2D 2D 50 72 65    ern: .  Input .  .-----Pre
0000671664 69 63 74 69 6F 6E 20 74 69 6D 65 2C 20 6D 69 63 72 6F 73 65 63 6F 6E 64 73    diction time, microseconds
000067303A 00 2D 2D 2D 2D 2D 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF    :.-----..................
```

Fig. 10. GHex editor used to analyse flash dump from MCU

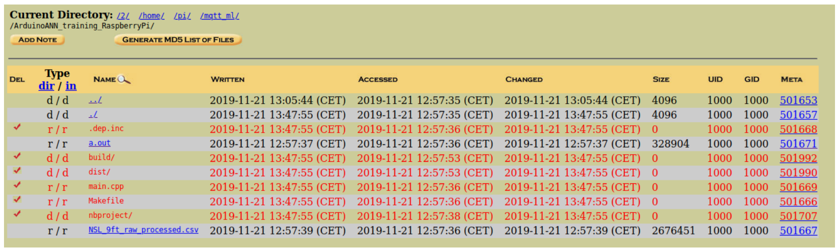
IoT hub/gateway artifacts Since Raspberry Pi has Debian installed, that we can refer to general guidelines for *Linux Forensics*, which has broader number of tools and approaches available then Arduino [21,29]. To acquire read-only image copy of the microSD card, we performed following basic operations:

```
$ fdisk -l
Disk /dev/mmcblk0: 14,9 GiB, 15931539456 bytes, 31116288 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x41503d89

Device          Boot   Start      End  Sectors  Size Id Type
/dev/mmcblk0p1          8192   532480   524289  256M  c W95 FAT32 (LBA)
/dev/mmcblk0p2        540672 31116287 30575616 14,6G 83 Linux

$ sudo umount /dev/mmcblk0
$ sudo dd if=/dev/mmcblk0 of=~/sd-card-copy.img
31116288+0 oppfoeringer inn
31116288+0 oppfoeringer ut
15931539456 byte (16 GB), 15 GiB kopiert, 247,108 s, 64,5 MB/s
$ md5sum sd-card-copy.img
93aef0ff0432a512e498153576221ccf  sd-card-copy.img
```

Listing 1.4. Retrieving dump of the microSD card dump from Raspberri PI

Once a read-only copy of the memory card, the following step includes analysis with a widely used software SleuthKit & Autopsy[11]. An example of the folder content, various dates and deleted files are shown in the Figure 11. The content of the folder may be indicated what kind of software is used, programming languages (C++ in this case) and many other relevant artefacts. Moreover, used dataset (NSL-KDD Cup 99) is also present in the folder, indicating a relation to the network data analysis too. The only executable that is in the folder is *a.out* that requires an additional round of analysis, reverse-engineering, per se.

[11] `https://www.sleuthkit.org/`


Current Directory: /2/ /home/ /pi/ /mqtt_ml/
/ArduinoANN_training_RaspberryPi/

Del	Type dir / in	Name	Written	Accessed	Changed	Size	UID	GID	Meta
	d / d	../	2019-11-21 13:05:44 (CET)	2019-11-21 12:57:35 (CET)	2019-11-21 13:05:44 (CET)	4096	1000	1000	501653
	d / d	./	2019-11-21 13:47:55 (CET)	2019-11-21 12:57:35 (CET)	2019-11-21 13:47:55 (CET)	4096	1000	1000	501657
✓	r / r	.dep.inc	2019-11-21 13:47:55 (CET)	2019-11-21 12:57:36 (CET)	2019-11-21 13:47:55 (CET)	0	1000	1000	501668
	r / r	a.out	2019-11-21 12:57:37 (CET)	2019-11-21 12:57:36 (CET)	2019-11-21 12:57:37 (CET)	328904	1000	1000	501671
✓	d / d	build/	2019-11-21 13:47:55 (CET)	2019-11-21 12:57:53 (CET)	2019-11-21 13:47:55 (CET)	0	1000	1000	501992
✓	d / d	dist/	2019-11-21 13:47:55 (CET)	2019-11-21 12:57:53 (CET)	2019-11-21 13:47:55 (CET)	0	1000	1000	501990
✓	r / r	main.cpp	2019-11-21 13:47:55 (CET)	2019-11-21 12:57:36 (CET)	2019-11-21 13:47:55 (CET)	0	1000	1000	501669
✓	r / r	Makefile	2019-11-21 13:47:55 (CET)	2019-11-21 12:57:36 (CET)	2019-11-21 13:47:55 (CET)	0	1000	1000	501666
✓	d / d	nbproject/	2019-11-21 13:47:55 (CET)	2019-11-21 12:57:38 (CET)	2019-11-21 13:47:55 (CET)	0	1000	1000	501707
	r / r	NSL_9ft_raw_processed.csv	2019-11-21 12:57:39 (CET)	2019-11-21 12:57:36 (CET)	2019-11-21 12:57:39 (CET)	2676451	1000	1000	501667

Fig. 11. SleuthKit report from imported IoT gateway microSD card image

Machine Learning Component Analysis Once the software component allegedly attributed to ML is located, the forensics analyst has two challenges: *reverse-engineer any binaries* and *discover application's functionality relevant for the crime investigation.* As mentioned before, any ML application program use *data* to train a specific *model* use for further decision making in Smart Applications. The report is extracted using SleuthKit and shown in the Listing 1.5.

```
$ file /2/home/pi/mqtt_ml/ArduinoANN_training_RaspberryPi/a.out
File Type: ELF 32-bit LSB shared object, ARM, EABI5 version 1 (GNU/Linux
    ↪ ), dynamically linked, interpreter /lib/ld-, for GNU/Linux 3.2.0,
    ↪  BuildID[sha1]=d70d380be66d6e2b6544802dd745707db2834430, not
    ↪ stripped
```

Listing 1.5. Retrieving dump of the microSD card dump from Raspberri PI

The easiest way to understand the functionality of a piece of software is to analyze the source code, which will most likely contain readable variables and function names and self-explainable comments. However, in most cases, the forensics investigators are left only with a compiled binary file. To understand the functionality we can either use commercial IDA Pro or free cross-platform tool Radare2, *Portable reverse-engineering framework*[12] as depicted in Listing 1.6.

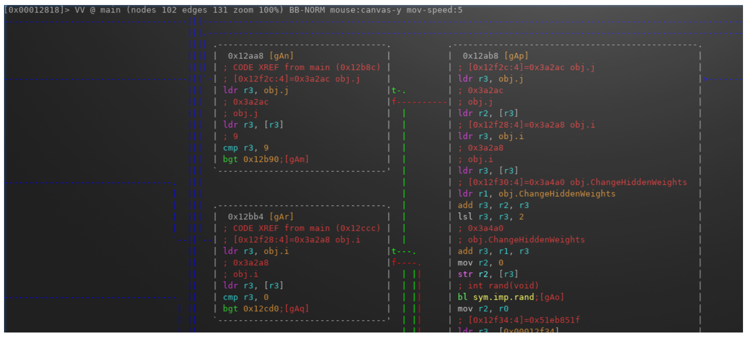
```
$ radare2 -aarm ./a.out
> aaa
> s main
> VVV
```

Listing 1.6. Reserse engineering of the *a.out* file

Found artefacts can be linked to the content of the MQTT communication in JSON format that was intercepted earlier using EtterCAP software. Moreover, building function calls graphs, we can get an overview of what kind of operations are performed. Disassembling of ANN training functional routine is shown in the Figure 12 with the variables names consistent with earlier found digital data.

IoT device without ML - artifacts The memory and flash were analyzed using GHex. Sensor data was visible in plain text (JSON) and included timestamps. The firmware image can be extracted and analyzed using radare2. The

[12] https://rada.re/n/



```
[0x00012818]> VV @ main (nodes 102 edges 131 zoom 100%) BB-NORM mouse:canvas-y mov-speed:5

 0x12aa8 [gAn]
; CODE XREF from main (0x12b8c)
; [0x12f2c:4]=0x3a2ac obj.j
ldr r3, obj.j
; 0x3a2ac
; obj.j
ldr r3, [r3]
; 9
cmp r3, 9
bgt 0x12b90;[gAm]

 0x12bb4 [gAr]
; CODE XREF from main (0x12ccc)
; [0x12f28:4]=0x3a2a8 obj.i
ldr r3, obj.i
; 0x3a2a8
; obj.i
ldr r3, [r3]
cmp r3, 0
bgt 0x12cd0;[gAq]

 0x12ab8 [gAp]
; [0x12f2c:4]=0x3a2ac obj.j
ldr r3, obj.j
; 0x3a2ac
; obj.j
ldr r2, [r3]
; [0x12f28:4]=0x3a2a8 obj.i
ldr r3, obj.i
; 0x3a2a8
; obj.i
ldr r3, [r3]
; [0x12f30:4]=0x3a4a0 obj.ChangeHiddenWeights
ldr r1, obj.ChangeHiddenWeights
add r3, r2, r3
lsl r3, r3, 2
; 0x3a4a0
; obj.ChangeHiddenWeights
add r3, r1, r3
mov r2, 0
str r2, [r3]
; int rand(void)
bl sym.imp.rand;[gAo]
mov r2, r0
; [0x12f34:4]=0x51eb851f
```

Fig. 12. ARM call function graph disassembly of the binary using Radare2

ESP32 uses a specialized firmware image format, which complicates the reverse-engineering process. Wireshark was used to analyze recorded network packets.

Without an edge device there are fewer available artifacts, and every artifact had its own challenges in terms of extraction, preservation and analysis. The device uses Wifi and the traffic could be captured, but the packets were encrypted and the data is stored in a proprietary cloud. The device had an easily accessible UART interface, but a bootloader must be uploaded to extract artifacts, damaging some artifacts in the process. Esptool is open source, but is platform-specific. It also lacks the functionality of converting firmware images to a standardized format, like ELF32. The flash was not encrypted, but requires reverse-engineering to fully interpret. Furthermore, due to the limited space the data only remains on the device for a limited time before being overwritten by newer data.

5 Digital Forensics Readiness and Cyber-Physical Incident Preparedness

Digital Forensics Readiness is defined as a certain level of readiness of an organization to preserve, collect and analyze any digital data citekarie2017digital. Such data are treated as digital pieces of evidence in any legal or court-related matter. In majority cases, this issue is related to the fact that data have to be stored in an appropriate way. This paper provides an example of a roadmap that can be used when analyzing digital pieces of evidence across any Smart Applications with IoT-based distributed infrastructure. Data retention policies are established in the organization to follow a common practice of preservation data that further can be used to speed up any involved forensics investigations or incident response. Since IoT-bases systems have naturally distributed versatile resources, one needs to ensure *pre-investigation readiness* and *real-time integration* as a key to successful digital forensics investigation in IoT in future [31].

The *core difference* of Smart Applications from other IoT applications is the existence of the ML processing mechanism, which implies that the decisions made by separate components based on the data might not be straight-forward. From the previous section, we can reconstruct the following scenario. The data are being processed on IoT gateways using the ML model called ANN. Further, the trained model (weights of the neuron connections) is being transmitted via Ethernet connection to IoT nodes with the help of the MQTT protocol. The protocol was not encrypted, so we used the ARP spoofing attack to intercept traffic between those two components. IoT node has open serial communication that gives a hint about ML model usage, which is also confirmed by analysis of string comments in dumped flash memory with the help of Arduino Forensics. Moreover, IoT gateway has Debian OS leaving many opportunities for Linux Forensics, such as file system analysis. After a closer look at the content of the folders, a network intrusion data set was found along with the binary file compiled for ARM. Reverse-engineering and disassembly of the binary file will reveal internal logic and functions used for ANN training. While understanding of hardware and software technologies is a crucial part of any modern investigation, there is also a need for cross-disciplinary awareness and expertise exchange to ensure a correct understanding of Smart Applications. Subsequently, it reflects aspects of their possible involvement in any criminal activities. Our belief is that technology awareness and expert knowledge will help to move from *Reactive* Digital Crime Investigation to *Proactive* Crime Prevention and Incident Response when it comes to any illegal activities in Smart Applications.

6 Conclusions and Discussions

This paper presents the thorny path of the Digital Forensics expert handling Smart Devices and Smart Applications. It is clear that the traditional understanding of Computer Forensics is under constant evolvement. Static data and well-known technologies are no longer a State of the Art. With the advancement of Smart Devices, IoT and later Smart Applications, a forensics expert is facing Big Data paradigm, proprietary software, previously unseen hardware components and Cloud Computing involvement. However, the growing demand for intelligent data analytics brings Machine Learning models in every component of the IoT ecosystem. Despite the complexity of such data analytics, one will need to utilize insights into how the data are processed and what kind of decisions are made. This will ensure timely response to incidents and proactive crime prevention in modern societies living in Smart Cities. This paper presented an example of a Digital Forensics Investigation roadmap in a Smart Application, explaining a whole range of forensics activities needed for a clearer understanding of the key value of any investigation - data, or digital pieces of evidence.

Even in an ecosystem without edge-devices and machine learning, the process is similar and has similar challenges. Especially the lack of standardization – every IoT platform (and cloud platform) requires specialized tools and knowledge to reverse-engineer the hardware and software. We expect cooperation between the forensic investigator and various actors, including the manufacturer, the developer and the cloud provider, to become a crucial part of forensic investigation.

7 Acknowledgement

Authors would like to thank to the Department of Information Security and Communication Technology (IIK) at the Norwegian University of Science and Technology for support and funding of this contribution. Moreover, this research has received funding from the Swedish Civil Contingencies Agency (MSB) through the research center Resilient Information and Control Systems (RICS).

References

1. The application of reverse engineering techniques against the arduino microcontroller to acquire uploaded applications (2014), accessed: 19.11.2019
2. Ahlberg, F.: esptool (2020), `https://github.com/espressif/esptool`, access date: 29.05.2020
3. Alenezi, A., Atlam, H., Alsagri, R., Alassafi, M., Wills, G.: Iot forensics: A state-of-the-art review, challenges and future directions. In: Proceedings of the 4th International Conference on Complexity, Future Information Systems and Risk (2019)
4. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., et al.: Understanding the mirai botnet. In: 26th {USENIX} Security Symposium. pp. 1093–1110 (2017)
5. Damshenas, M., Dehghantanha, A., Mahmoud, R., bin Shamsuddin, S.: Forensics investigation challenges in cloud computing environments. In: Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec). pp. 190–194. IEEE (2012)
6. Delicato, F.C., Pires, P.F., Batista, T., Cavalcante, E., Costa, B., Barros, T.: Towards an iot ecosystem. In: Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems. pp. 25–28. ACM (2013)
7. Dengler, S., Awad, A., Dressler, F.: Sensor/actuator networks in smart homes for supporting elderly and handicapped people. In: 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07). vol. 2, pp. 863–868. IEEE (2007)
8. Ericsson: Internet of things forecast (2019), `https://www.ericsson.com/en/mobility-report/internet-of-things-forecast`, accessed: 04.10.2019
9. Esposito, C., Castiglione, A., Pop, F., Choo, K.K.R.: Challenges of connecting edge and cloud computing: A security and forensic perspective. IEEE Cloud Computing **4**(2), 13–17 (2017)
10. Espressif: Esp32-wroom-32d (2019), `https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf`, access date: 29.05.2020
11. Goudbeek, A., Choo, K.K.R., Le-Khac, N.A.: A forensic investigation framework for smart home environment. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). pp. 1446–1451. IEEE (2018)
12. Grance, T., Chevalier, S., Scarfone, K.K., Dang, H.: Guide to integrating forensic techniques into incident response. Tech. rep. (2006)
13. Hegarty, R., Lamb, D.J., Attwood, A.: Digital evidence challenges in the internet of things. In: INC. pp. 163–172 (2014)

14. Koen, R., Olivier, M.S.: The use of file timestamps in digital forensics. In: ISSA. pp. 1–16. Citeseer (2008)
15. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: Ddos in the iot: Mirai and other botnets. Computer **50**(7), 80–84 (2017)
16. Kononenko, I., Kukar, M.: Machine Learning and Data Mining: Introduction to Principles and Algorithms. Horwood Publishing Limited (2007)
17. Lillis, D., Becker, B., O'Sullivan, T., Scanlon, M.: Current challenges and future research areas for digital forensic investigation. arXiv (2016)
18. Oriwoh, E., Jazani, D., Epiphaniou, G., Sant, P.: Internet of things forensics: Challenges and approaches. In: 9th IEEE International Conference on Collaborative computing: networking, Applications and Worksharing. pp. 608–615. IEEE (2013)
19. Perumal, S., Norwawi, N.M., Raman, V.: Internet of things (iot) digital forensic investigation model: Top-down forensic approach methodology. In: 2015 Fifth International Conference on Digital Information Processing and Communications (ICDIPC). pp. 19–23. IEEE (2015)
20. Pollitt, M.: A history of digital forensics. In: IFIP International Conference on Digital Forensics. pp. 3–15. Springer (2010)
21. Pomeranz, H.: Linux forensics (for non-linux folks), `http://www.deer-run.com/~hal/LinuxForensicsForNon-LinuxFolks.pdf`, accessed: 21.11.2019
22. Postscapes: Iot standards and protocols (2019), `https://www.postscapes.com/internet-of-things-protocols/`, accessed: 04.10.2019
23. Sadeghi, A., Wachsmann, C., Waidner, M.: Security and privacy challenges in industrial internet of things. In: 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC). pp. 1–6 (June 2015)
24. Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017)
25. Schatsky, D., Kumar, N., Bumb, S.: Intelligent iot: Bringing the power of ai to the internet of things (2017)
26. Shalaginov, A.: Soft computing and hybrid intelligence for decision support in forensics science. In: IEEE Intelligence and Security Informatics. pp. 304–309 (2016)
27. Shalaginov, A.: Advancing Neuro-Fuzzy Algorithm for Automated Classification in Largescale Forensic and Cybercrime Investigations: Adaptive Machine Learning for Big Data Forensic. Ph.D. thesis, Norwegian University of Science and Technology (2018)
28. Shalaginov, A., Semeniuta, O., Alazab, M.: Meml: Resource-aware mqtt-based machine learning for network attacks detection on iot edge devices. In: Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion. pp. 123–128. ACM (2019)
29. Willis, C.: Forensics with linux 101 or how to do forensics for free (2003), `https://www.blackhat.com/presentations/bh-usa-03/bh-us-03-willis-c/bh-us-03-willis.pdf`, accessed: 21.11.2019
30. Yusoff, Y., Ismail, R., Hassan, Z.: Common phases of computer forensics investigation models. International Journal of Computer Science & Information Technology **3**(3), 17–31 (2011)
31. Zulkipli, N.H.N., Alenezi, A., Wills, G.B.: Iot forensic: Bridging the challenges in digital forensic and the internet of things. In: International Conference on Internet of Things, Big Data and Security. vol. 2, pp. 315–324. SCITEPRESS (2017)