Can you call the software in your device firmware?

Rashedul Hassan, Konstantinos Markantonakis and Raja Naeem Akram
Information Security Group
Royal Holloway, University of London
Surrey, Egham TW20 0EX
mdrashedul.hassan.2014@live.rhul.ac.uk, k.markantonakis@rhul.ac.uk,
rajanaeem.akram.2008@live.rhul.ac.uk

Abstract— **In addition to storing a plethora of code and functionality, devices also possess a certain set of data and commands, known as firmware. The ability of devices to perform specific tasks relies on the firmware. In order to understand the scope of firmware we conducted a qualitative analysis of its properties, functionalities and security. We examined the use of the terminology over a period of years and found that firmware is used in numerous situations in a range of contexts. In this paper, we propose a taxonomy of firmware to classify the field and understand it better.**

I. Introduction

Electronic devices are being used every day for a variety of purposes. These devices range from mobile phones, to laptops, microwaves, electric shavers and many others. The physical device itself is known as the hardware and the programs that run within the device are referred to as software [1]. Firmware is described loosely as a coupled mixture of both hardware and software.

"Hardware" usually refers to a combination of multiple components with a physical existence, which will make a noise when dropped on the floor [2]. Any device with this property is known as hardware; for example, a processor, flash drive, or microwave. However, such devices are unable to function on their own and they require certain levels of "instructions" [3] or "firmware" to perform their tasks. Instructions can be defined as set of programmed routines that handle the different components of the hardware, initiating it to carry out some activity. The American Heritage Dictionary [4] defines software as written or printed data such as programs, routines, and symbolic languages, essential to the operation of computers.

"Firmware" is vaguely referred to as the interface between software and hardware [5]. Firmware can be stored in any storage media; however, it is typically stored in Read Only Memory (ROM) chips [6] where the program is stored and requires no power source to maintain the data. A common example of a piece of firmware is the Basic Input Output System (BIOS) chip on a computer. The BIOS chip contains permanent instructions about the computer and is activated when the computer is started.

Despite the firmware being permanently stored on ROM chips, it can also be replaced or re-written to enable updates or make changes [7]. The older types of firmware were generally stored in ROM chips and the only way to update would be to replace them with new ROM chips. The storage mechanism for firmware has evolved beyond ROM chips and now it is possible to [8] re-write firmware and store it in erasable media such as an Erasable Programmable ROM (EPROM), or a flash drive.

If firmware is stored in a non-writable store, an upgrade of the firmware would mean replacement of the complete unit. In the past, when the BIOS chip had to be updated, the whole motherboard unit of the computer had to be replaced to accomplish that task. Even though the firmware was the instruction set or the software content within the BIOS chip, the "chip" was considered to be the firmware.

BIOS is a particular type of Personal Computer (PC) firmware. Usually, any IBM PC will carry a BIOS implementation or another implementation known as Unified Extensible Firmware Interface (UEFI) [9]. UEFI is generic and resides mostly in systems that are not IBM PC compatible. There are other firmware in use other than UEFI and BIOS, e.g. CoreBoot [10], and OpenBIOS [11].

The example of BIOS illustrates the different ways firmware is described. As a whole, the BIOS can be referred to as firmware, but if only the "chip" or the physical existence is considered, it can be termed "hardware". In addition, the program part within the chip can also be regarded as "software".

The IEEE Standard Glossary of Software Engineering Terminology [12] extended the definition of "firmware" as follows: "the combination of a hardware device and computer instructions or computer data that reside as read-only software on the hardware device". However, studies by Barsamian et al. [6], [13], [14] described "firmware" as the intermediate layer between hardware and software. There has been no conclusive study done to distinguish firmware from software or hardware. The scope of firmware needs to be studied and defined. The number of devices that have firmware is continually increasing. Every day approximately 2 million devices [15] are being activated, yet there is no clear definition of firmware in the literature.

This paper is structured as follows: Section 2 discusses related work. It consists of a brief overview of the trends in firmware. Section 3 deals with the taxonomy. It is followed by our rationale and then a brief discussion of each element together with descriptions of some attacks and vulnerabilities that apply to firmware. The final section presents our conclusions and possibilities for future research.

II. Related Works

In this section we discuss previous work on firmware and trends over the years. We start by discussing the changes over the years. More discussion on trends follow this section. We continue the discussion of trends towards storage trends and end this section with discussions related to current trends.

The usage of the word began in 1967. It was first mentioned by Opler [3]. Work done by Opler discusses about the software portion that is saved in a ROM providing specific guidelines on how the software should behave. From 1967 till 1971 there were no significant work done on firmware. In 1971 Barsamian and Dec mentioned firmware as programmed instructions stored in special ROM or Read-Write (RW) control stores [13].

There were no major update from the year 1971 till 1993. In 1993, Mange et.al used the term firmware to visualise the concept of transforming software logic into hardware instruction set [16]. The idea was used to provide a bridge between the hardware and software and vice versa. The software part exhibited the idea of driving the hardware to do a specific task. The author did not refer the software itself to be the firmware, however he pointed the transformation process to be the firmware since it was bridging the gap for establishing a communication platform.

According to work done by Chen [17], the software itself has been called a firmware. From 1993 till 2012, many work has been done [5], [18]–[24] on firmware, however the idea was addressed according to the context in discussion. It ranged from an "in-between" layer to "software", nobody highlighted its position.

Early storage of firmware was done in a Read Only Memory (ROM) or permanent storage media. The practice of writing firmware in the ROM continues today. In 1971, firmware was quoted as "micro-instructions that could be saved in the ROM mostly to be used for read-only purposes" [13]. However, the convention did not remain stagnant and limited to ROM. 1990s and beyond introduced the rise of Integrated Circuitry (IC). Instruction sets in IC were hard-coded and installed in devices. Flash storage was introduced simultaneously. This introduction provided more speed and convenience for the developers and manufacturers. Re-writing the flash storage became easier. Unlike the IC's, which had to be manually removed to update the programs, flash storage did not require any removal and it was reused to overwrite existing firmware.

Upon introduction of flash storage, the year 1993 referred firmware as "a way to convert hardware instructions into software instructions" [16]. As discussed in the previous section, the idea of bridge was being highlighted. It has also been referred as a type of a software loosely referring as "a form of software" known as firmware [6].

The idea was not limited to software alone, but also focused on the area of hardware. Work done by Schubert [25] described firmware as "chip-set". The connection or the "layer" that resides between software and hardware is typically referred to as "firmware", as identified in 2009 as the interface between the two [5]. Beyond 1990s flash storage was widely used but the usage of ROM was still in existence. In 2010, the firmware stored in ROM was referred to as a defined "functionality" that accomplishes a set of function or job [20]. Following the work, further research in 2012 on firmware redefined it as being software stored in certain memories [21].

Firmware has evolved to represent any programmable content of a hardware device; it did not only represent software or binary code. It also exemplified machine code for a processor, but also configurations and data for Application-Specific Integrated Circuits (ASICs), programmable logic devices and others. On the domain of processors, the firmware is being referred to as being "complicated logic which is stored as micro-instructions" [13]. In an embedded system, [6] described firmware as programs that are stored in hardware memories such as ROMs. The work deems the firmware to be just "hardware".

In work carried out in aviation and dealing with software engineering challenges, it was quoted "It is sometimes not clear what constitutes a processor, for example, because so much specialized electronics is involved. Similarly, software is sometimes in read-only memories called firmware rather than software" [18].

It can be clearly seen the understanding varied over the years, where it was referred to as dedicated software and/or hardware specific instruction set. The ways the term has been defined was in context to the application or device under discussion.

III. The Taxonomy and its justification

In this section we will expand on the landscape of firmware. We start by proposing a taxonomy structure of firmware followed by a subsequent section briefly discussing the rationale behind the categorisation. The diagram of our taxonomy is illustrated in Figure 1.

A. Framework

The first family is named as "Framework" that highlights the basic structure and characteristics of a firmware. This family tree has been further dissected into three branches namely: "Core", "Protection" and "Interface".

1) Core: We bisect the firmware into multiple parts, the rationale behind naming this section "Core" is to represent the major functionalities of the firmware. This branch consists of three more sub-children that are highlighted in figure 1.

a) Hardware: We categorise "hardware" where information to control a device runs on a printed circuit board or a signal converter. An example of this classification would be the microchip inside a watch, table lamp, toy cars [26]. The circuitry uses electricity to convert electrical energy to different forms of energy. In case of a watch, it is converted to mechanical and kinetic energy which helps with the movement of dials. In case of a lamp, the electrical energy is converted to light and heat energy. The range of attacks within this domain are limited since it involves manipulating

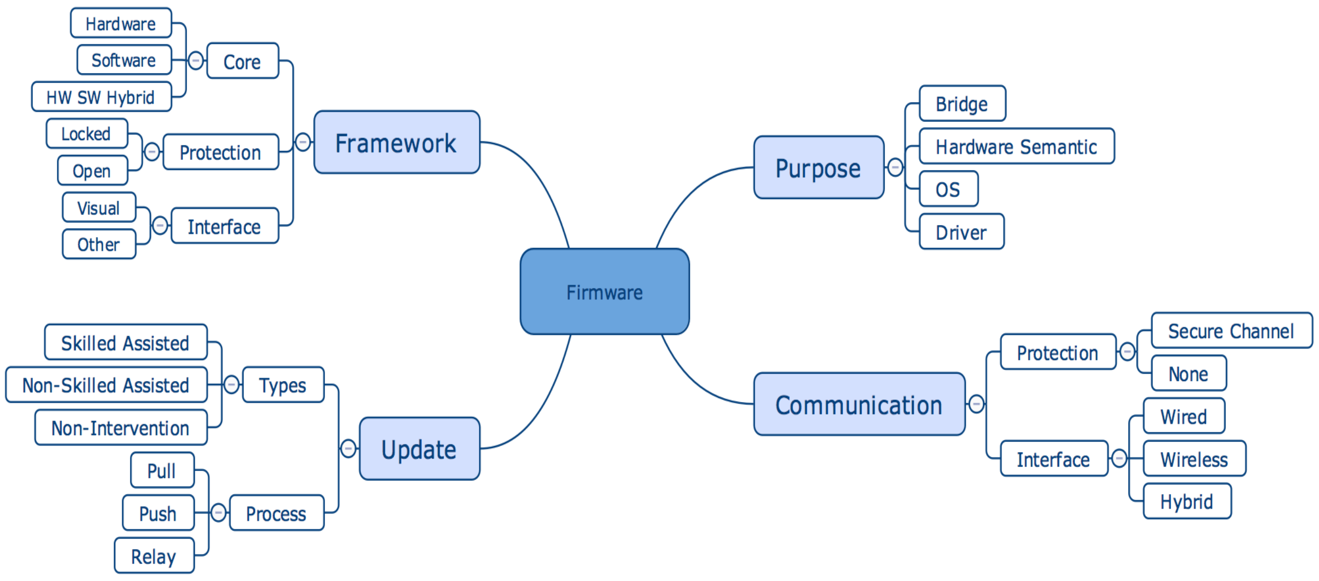

Fig. 1. Taxonomy of firmware

the hardware components and is usually referred to as hacks rather than attacks [27].

b) Software: We classify "software" to the kind of firmware where the programs are stored programmatically inside a storage media e.g. ROM, EPROM, EEPROM etc. These are programs that have their existence in a digital format rather than a series of electrical signals. Examples would include, drivers for a printer [8], visual firmware for customising a mouse [28], etc. In work done by Maskiewicz et al. [29] a Logitech G600 mouse was programmed by writing a custom software to achieve a file transfer task.

Attacks in this domain usually involves manipulating the code to achieve extra benefits. It has been shown in work done by Yanlin et. al [30] it is possible to reverse engineer the firmware of a network card and manipulate it. Using a package known as Interactive Dis-Assembler (IDA) it is possible to alter the function calls within a software program.

c) HW SW Hybrid: The device family which involves both characteristic of hardware and software are enlisted in this category. As referred to our example of BIOS in Section 1, the chip-set itself has a physical existence whereas, it cannot be called hardware because there is a software program within the chip. On the contrary, labelling it as mere software is not applicable since, it is a chip which has physical existence and does not require power to store the information. Other example for this criteria include the set top TV boxes. They convert direct button press signals to machine readable signals [19]. Furthermore, they have their storage installed within their device that stores the firmware content and updating that content can be done by connecting a cable and flashing it [31].

There are attacks targeted to BIOS. Work done by Wojtczuk and Alexander [32] showed the attack of BIOS by manipulating several checksums. They used a BIOS building tool followed by patching to compensate the checksum. Other work has shown how data can be hidden in BIOS chips. BIOS has a capacity of storing 128K to 512K of data and this storage can be used by criminals [33]. Miscreants could use this feature to manipulate a target computer to cause damage or create data-loss in an organization or a specific person. Fortunately, the modern implementations of BIOS [34] have changed and stronger security primitives within BIOS are introduced with secure boot and UEFI (Unified Extensible Firmware Interface) [9].

2) Protection: This sub-tree discusses the security mechanism that is available in any kind of firmware. It has been further elaborated into two more sub-children namely:

a) Locked: When describing firmware, how the firmware itself is utilised or stored in a media is a very important aspect. A firmware can be stored in a ROM [35], but not always; since it can also be stored in a hard disk drive, EPROM flash drive or Solid State Drive (SSD). The protection of the storage is classified as an attribute. Some firmware has protective mechanism to restrain unauthorised modification or alteration whereas other firmware does not have any protection. By "Locked" we refer to the family of firmware which are digitally signed by manufacturer [36] and has their default administrator login credentials encrypted.

Attacks involved in this domain includes the successful "jailbreaking" [37] of multiple iOS versions. iOS jailbreaking removes the restrictions of the Apple's iOS operating system using software exploits. This allows the locked restriction of downloading additional applications, music and themes that are not available through the official AppStore [38].

b) Open: In situations where the username and password to the administrator access portal of the firmware is not cryptographically protected or written in plain text falls into our branch of "Open". It has been discovered manufacturers [22] do not take necessary precaution to encrypt the default administrator login credentials within the firmware. Rather they use specific company-title [22] and other fixed place decimal value data linked with the firmware that can be easily brute-forced to retrieve the credentials. These type of devices are termed as "Open" in our taxonomy. For example, in a TP-Link TD-W8951ND V4 ADSL router it was observed that the username and password were stored in plain text when a simple man in the middle attack was performed using Cain [39]. Furthermore, it has also been reported in specific software packages e.g. Deploy Studio [40], which is used to image and deploy Mac computers to manage workstations and servers, that the firmware passwords are stored in the log in plain texts. In addition to these, the telecommunication giant EE's BrightBox routers login page stores their customers credentials in a plain text format [41].

3) Interface: The interface of a firmware is placed as a part of the firmware's framework capabilities. It represents two different types of firmware that consists of an interface to allow configuration; whereas there is another set of firmware that comes with predefined characteristic and functionality that does not allow any customisation and thus only behaves according to the manufacturers pre-set instruction sets.

a) Visual: Ideally the interface makes way for configuring the firmware. It also allows the control of the device. The BIOS of a computer machine has been considered as an example. The BIOS is a type of firmware that gives the user an option to customise the settings of a computer [42]. The interface used is a Graphical User Interface (GUI) which allows navigation via a keyboard that would allow the user to tweak settings e.g. primary boot device, secondary boot device.

Another example would include when a user of a computer would want to enable a Firmware password to restrict the machine from booting from another hard drive. In an Apple Macintosh where there is a specific Graphical User Interface (GUI) that enables the usage of a firmware password [43]. The presence of an interface enables the option to set up a password. The interface allows the possibility to set up a password, without it the user will not be able to set it up. Only the manufacturer would have been able to do that.

The visual category has been established to also represent messaging output too. In a telephone handset, the display that shows the digits inputted into the set can also be referred as the visual interface. More examples would include tuning the thermostat in a home where the current temperature and other settings are displayed as a message or information to the user.

b) None: To the family of firmware that does not exhibit a configurable interface is termed as to be residing in this section. The example clarifies the members of this sub-children. The hardware content within a CPU consists of processors [44], RAM [45] and hard disk drive [46] to name a few. All of these devices have firmware within them that provides them with instruction on how to operate specific mechanical parts of that device. For example, the firmware within hard disk [47] drive controls the disk rotation and read-write speed. It also contains information regarding how the allocation tables are managed. Configuring the firmware is not possible since the data is already set up by the manufacturer and there is no interactive interface to tweak and configure the settings.

There have been attacks on some of these components even though they lack a visual interface. Kaspersky researchers discovered that the hard drive firmware could be compromised [46]. They showed, subverting the firmware allows the attacker to create invisible storage space to hide data from the system. The data is not erased even when the hard drive is being formatted. This allows the attacker to be able to retrieve the data at a later date.

The above section of our taxonomy discussed the basic structure and characteristics of a firmware. Within the limited scope of this paper, we tried to explain the different arrays of firmware that exists in the industry and highlight their respective positions.

B. Purpose

This category has been formed to fulfil our goal of categorising different firmware according to their functional capabilities.

1) Bridge: It is known as an intermediary that allows the connection from the product itself to its operator. An example has been used to depict the usability of bridge.

For example, the firmware of a battery, when the operating system of the computer or a phone would ask for the percentage left or the life cycle or capacity of the battery, the firmware within the battery would reply with those information [48]. In this situation, the firmware is acting as a bridge providing a connection medium to establish workflow between the software and the hardware counterpart. Another example would include the mechanism in a touch screen device in a supermarket till. The firmware within the touch screen converts the mechanical touch inputs into electrical output to be read by the system [49]. The software or "firmware" here aids as a bridge connecting the input to convert into outputs.

2) Hardware Semantic: A module of hardware that has micro-instructions [50] embedded into the chipset falls into this category. It is a block originally created by the manufacturer and has the logic within it to make the hardware work accordingly. Example would include an electric razor by Philips SensoTouch. The power button is synchronised with the motor in a chipset. When the power button is pressed, the logical instructions within the semiconductor chip instructs the motor head to start rotating hence achieving the task and making the device useful [51]. Similar example would be the usage of a kitchen Toaster. It has the same architecture, where the button is programmed to push the bread and initiate the heating of the filament. This branch has overlapping with our "HW SW Hybrid branch" within "Framework". However, we have still placed "Hardware Semantic" here since it is a specific quality of the firmware and cannot be ignored when it comes to categorise a purpose of the firmware.

3) Operating System: The operating system, e.g. iOS, Android, Windows Embedded acts as a firmware to operate the embedded device or a phone or a Kiosk. The firmware allows the device's hardware to be used to convenient output. i.e. processing a touch or processing a swipe of the card in a kiosk. Operating system is closely related to a bridge. The difference comes in its behaviour, the operating system provides an user interface to interact with the phone: making the buttons on the phone do specific task, making the touch inputs process relevant output, ability to respond to force-touch [52]. When an iPhone's software fails to launch, the device needs to be reset to factory settings. The process is known as recovery mode [53] however the technical terminology is referred to as Device Firmware Upgrade (DFU) [54]. To update the operating system of iPhone, an iPod software firmware file (IPSW) is required [55]. This can be obtained manually and injected to iTunes or can be automatically downloaded by iTunes itself. Once the software binary is acquired, iTunes can initiate installation of the latest firmware on to the device.

4) Driver: When there is a hardware, there needs to be specific type of instruction set provided by the manufacturer to make the device work desirably into another environment. For example: the printer, HP LaserJet 4200, to be identified properly by a computer or a network, there is a need of a driver program that would help accomplish the effective communication [56]. In addition to that, when an Apple computer is used to run a different operating system, the hardware should be given proper instruction to behave accordingly to the new environment. An example of this is the Bootcamp software from Apple which makes the Macintosh hardware compatible with the Microsoft Windows operating system allowing all the peripherals to work perfectly [57].

C. Communication

The firmware requires transfer of data within other devices and with the outside world. We have referred the process of data transfer to be called "Communication". This branch consists of two more sub-children that entail the functionality of our communication.

1) Protection: We refer to the ways the communication is monitored or allowed. Some communications are followed with cryptographic security mechanisms whilst others are not. Thus the sub-children of this branch represents:

a) Secure Channel: The firmware in a Logitech G600 mouse uses RSA encryption [29] to communicate with the device. Gathering data from the mouse and transferring the data to the operating system of the computer is done via a secure channel communication. The data communicated is not in plain text. This allows a secured medium for communication that prohibits interception of data and adds more security.

The importance of security varies from use cases. Different firmware have different level of security. When the application is a sensitive issue; in any medical equipment that has the risk of the well-being of the patient, security is a very important criteria. For example, the pace-maker within a patient's heart; the firmware of any planted device should have security since it is vulnerable [23] to malicious attacks [58] which could compromise a human life.

There has been attacks reported that falls into our categorisation within a secured channel. Maskiewicz et. al [29] reported in their work, the firmware of the mouse can be compromised to write custom malicious code executing other tasks. They have written their own code to copy a file while the mouse was in work. The work depicts, the firmware in peripheral devices are not secure. In another work done by [59], they discussed the manipulation of a network card which was using a flawed firmware that an attacker may subvert remotely by sending packets on the network to the adapter.

b) Other: However in cases like Nikon camera D3100 or other entertainment devices e.g. music players, where the firmware could be easily manipulated to enable hidden features are termed as "none" or "other" in the protection category. The Canon IXUS30 (SD200) camera's firmware can be easily modified [60] to attain pictures of higher resolution, taking RAW images, shooting self-capture images, time-lapse, etc. The firmware modification allows all these, and the user may not have to buy a later model to get those features. The tweak has more usage. In an experiment to capture images of the Earth the camera was modified to capture pictures throughout the whole duration of the experiment [61]. In addition to this, previous work [58], [62], [63] also discusses about lacking proper user authentication.

2) Interface: The branch has been dissected further to categorise different medium of communication that takes place within a firmware. Typically three interfaces are proposed.

a) Wired: Firmware updates in automotive industry is carried out using a wired interface [64]. It can also be used to connect to the diagnostics port to retrieve information about the vehicle as well. The firmware diagnosis and update process is done off-board, by connecting (hardwired) a diagnosis tool with the on-board network and performing firmware updates [65].

b) Wireless: Other ways of communication include performing the whole data transfer over a wireless connection e.g. Wi-Fi, 4G, Bluetooth, infrared, etc. For example, in automotive the update to the firmware can be patched via over the air [66], [67]. The wireless mechanism is not only limited to automotive domain, in a mobile phone Over The Air (OTA) updates of firmware of a mobile telephone handset can also be done too [68].

c) Hybrid: However, some devices are not only limited to these two types of communication. It is not possible to categorise them into the aforementioned Wired and Wireless category. So we open up a new category named "Hybrid". A smart-card is an example of this category. The smart-card could operate two ways, as a chip-and-pin mechanism and also in a contactless mechanism too [69]. The example of smart-card can not necessarily be categorised as having a "wired" interface nor it can be categorised as having a "wireless" interface. It has the ability of performing both the interfaces and thus we have our another sub-children: "hybrid".

D. Update

This branch consists of two more sub-children that represent the ways the update of the firmware is involved.

As discussed earlier when firmware used to reside on a ROM, it was difficult to apply updates. But with recent improvements in technology, as of 2013, most firmware can be updated [70]. But still there are risks. Upon a failed update, the whole device runs the risk of being termed as "bricked" where the update procedure has deemed to have destroyed the device [71].

Applying firmware updates on critical embedded systems can be cumbersome and daunting [24], however there are different mechanisms that accommodates for varying update mechanisms and techniques. As depicted in Figure 1, the "Update" branch is dissected into two further sub-children: "Types" - discusses the security mechanism of the updates and "Process" - the ways the update takes place.

1) Types: Types refer to the possible ways the update of a firmware could be performed. Types has been broken down into three simplistic branches that discusses the possible ways a user can have their firmware updated to get a newer version or revert back to a different version.

a) Skilled: This section is referred to the installation of a firmware that a normal user would not be able to perform by himself i.e. to perform an update of the BMW 6 series auto-mobile's dashboard navigation system, the user need to bring the car to BMW's dealer to apply an update. Here, the technician will use one of the communication mechanism as discussed earlier to perform the update.

b) Non-Skilled: Any sort of update that does not require the special assistance or unavailable toolset is referred to as non-skilled update. If we consider the situation where the phone's firmware needs to be updated, it can be easily done via the operating system's interface. The phone manufacturers have readily made the resource available to everyone for them to be able to perform it without any skilled assistance. For example: any "Android" user could update their phone software by themselves clicking appropriate buttons from the user interface.

c) Non-intervention: This type of update is referred to those planned updates that are done automatically on a specific date, time or a period. As an example, when a new software update is available in a Windows machine, if the user has agreed to apply the updates, they will be downloaded and installed automatically to the system [72]. Devices today are increasingly equipped with WiFi components and the availability of WiFi hotspots has improved. This means that devices can take advantage of free and fast connections to download Web content, e.g. prefetch Webpages for offline use, update RSS feeds, or download podcasts, new e-mails or updates for Web widgets [72].

For example: when a user has set up an "auto-update" in his or her machine, the update shall take place without any further input. It should carry on at a prescribed time. The situation is also true when a user turns on the auto-update feature, e.g. Microsoft Windows Update.

2) Process: When an update is due, the manufacturer may choose to distribute the latest update to all the devices under its ecosystem. The update option could be prompted or if previously recorded it can initiate on its own. Consecutively in situation where the update has failed, it may automatically request the update to be performed again [73].

a) Pull: To categorise the situation when updates are sent or received, we classified a "Pull" sub-children. The purpose of this sub-children is to fulfil the kind of updates that takes place upon a request. When a new version of the firmware is being released by the manufacturer the consumer might not be notified straight away. It could be due to design issues of the manufacturer or server traffic once a new version is release. To obtain this type of update materials, a user would have to query the network asking for new update. If an update is available, the firmware would then be downloaded or installed. For example: when a firmware update of nVidia GTX 650M graphics card driver is released, the notification does not pop up reminding the user to update it. However, when the user would voluntarily prompt for "Check for Updates", the new firmware version would show up [74]. Obtaining this kind of updates is referred as the "Pull" mechanism in our tree.

b) Push: In situations when new information or data needs to be sent to the subscribers or customers, one of the ways used by companies or manufacturers is to dispatch or release the data. The data from the central network is intended to reach the product as a part of an update [75]. For example: in a Programmable Logic Controller (PLC) in an industry, when it is needed to provide changes to the current instruction set [76], the whole firmware could be patched in via a push update.

c) Relay: In a very specific case, a resource does not necessarily have to be obtained via a push or pull mechanism. A resource can sometimes be available via a third party. We have termed this type of procedures as "Relay". The source is obtained from multiple media e.g. a ROM, DVD or flash storage and then applied to the target media. For example: when a user would attempt to update the firmware of an Apple Watch Sport, the user would need to obtain the firmware update via the iPhone rather than the Apple Watch itself [77]. The Apple Watch does not have the mechanism or capability to download the new firmware and relies on a another device to apply the update. To make this update possible, the iPhone has to transfer or "relay" the data via Bluetooth to complete the update [78].

IV. Conclusion

The landscape of firmware is enormous and categorising in a compact fashion was the main concern behind this paper.

According to the best of the authors knowledge, there is no definitive explanation of firmware in the literature. So, we took the first initiative to design an easier image of firmware that would help in understanding the field better. We have specifically focused in making the field compact and have minimal overlapping between themselves. Our design would help other researchers and manufacturers to get better clarity and understanding on firmware. When there would be a requirement to describe firmware they would be able to relate to this paper to comprehend fine detail in their explanation. The work would also allow researchers to short-list the aspects of firmware they are interested. This could potentially help others eliminate the indefinite understanding of firmware and help them focus better on a topic.

By focusing on consumer electronics and everyday appliances we were able to categorise majority of the devices within our tree. We have also provided justification and examples for classifying firmware in this way. The design choices to reach to this stage has been analysed and logically rationalised before placing them in the tree. Our future work will focus on different sections of the taxonomy tree in greater detail and also investigate the vulnerable security elements of firmware.

References

[1] D. A. Patterson, *Computer organization and design: the hardware/software interface*, D. A. Patterson and J. L. Hennessy, Eds. Newnes, 1994.

[2] A. Clements, *Principles of computer hardware*. Oxford University Press, 2006.

[3] H. W. Lawson Jr, "Programming-language-oriented instruction streams," *Computers, IEEE Transactions on*, vol. 100, no. 5, pp. 476–485, 1968.

[4] P. Davies, *The American heritage dictionary of the English language*. Dell Pub Co, 1976.

[5] T. Eisenbarth, R. Koschke, and D. Simon, "Incremental location of combined features for large-scale programs," in *Software Maintenance, 2002. Proceedings. International Conference on*. IEEE, 2002, pp. 273–282.

[6] W. O. Cesário, D. Lyonnard, G. Nicolescu, Y. Paviot, S. Yoo, A. A. Jerraya, L. Gauthier, and M. Diaz-Nava, "Multiprocessor SoC platforms: a component-based design approach," *IEEE Design & Test of Computers*, no. 6, pp. 52–63, 2002.

[7] P. Dice, *Quick boot: a guide for embedded firmware developers*. Intel Press, 2013.

[8] A. Cui, M. Costello, and S. J. Stolfo, "When firmware modifications attack: A case study of embedded exploitation." in *NDSS*, 2013.

[9] *UEFI Secure Boot in Modern Computer Security Solutions*, 09 2013. [Online]. Available: http://www.uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2013.pdf

[10] *coreboot*, 02 2016. [Online]. Available: https://blogs.coreboot.org/about/

[11] *OpenBIOS*, 03 2016. [Online]. Available: http://www.openfirmware.info/Welcome_to_OpenBIOS

[12] September, "IEEE Standard Glossary of Software Engineering terminology," *Office*, vol. 121990, no. 1, pp. 1–84, 1990. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=159342

[13] H. Barsamian and A. DeCegama, "Evaluation of hardware-firmware-software trade-offs with mathematical modeling," in *Proceedings of the May 18-20, 1971, spring joint computer conference*. ACM, 1971, pp. 151–161.

[14] D. Davidson, B. Moench, T. Ristenpart, and S. Jha, "FIE on Firmware: Finding Vulnerabilities in Embedded Systems Using Symbolic Execution." in *USENIX Security*. Citeseer, 2013, pp. 463–478.

[15] "Google I/O 2013 Keynote," 2013. [Online]. Available: https://plus.google.com/app/basic/stream/z133ddfxxoz5e1iog04cj35obprltdqifo40k

[16] D. Mange, "Teaching firmware as a bridge between hardware and software," *Education, IEEE Transactions on*, vol. 36, no. 1, pp. 152–157, 1993.

[17] S.-C. Chen, P. O'neill, P. L. Sotos, J. M. Lim, and S. A. Jacobi, "Initialization and update of software and/or firmware in electronic devices," Aug. 5 2008, uS Patent 7,409,685.

[18] J. C. Knight, "Software challenges in aviation systems," in *Computer Safety, Reliability and Security*. Springer, 2002, pp. 106–112.

[19] S. Dutta, R. Jensen, and A. Rieckmann, "Viper: A multiprocessor SOC for advanced set-top box and digital TV systems," *IEEE Design & Test of Computers*, no. 5, pp. 21–31, 2001.

[20] K. Kursawe and D. Schellekens, "Flexible μtpms through disembedding," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*. ACM, 2009, pp. 116–124.

[21] L. McMinn and J. Butts, "A firmware verification tool for programmable logic controllers," in *Critical Infrastructure Protection VI*. Springer, 2012, pp. 59–69.

[22] G. Ramesh and R. Umarani, "Data Security in Local Area Network Based on Fast Encryption Algorithm," in *Recent Trends in Network Security and Applications*. Springer, 2010, pp. 11–26.

[23] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008, pp. 129–142.

[24] A. Bellissimo, J. Burgess, and K. Fu, "Secure Software Updates: Disappointments and New Challenges." in *HotSec 06: 1st USENIX Workshop on Hot Topics in Security*, 2006.

[25] K.-D. Schubert, "Improvements in functional simulation addressing challenges in large, distributed industry projects," in *Design Automation Conference, 2003. Proceedings*. IEEE, 2003, pp. 11–14.

[26] *Quartz clocks and watches*. [Online]. Available: http://www.explainthatstuff.com/quartzclockwatch.html

[27] "Developer hacks Apple Watch to boot and run Windows 95." [Online]. Available: http://9to5mac.com/2016/04/30/developer-hacks-apple-watch-to-run-windows-95/

[28] R. R. Plant, N. Hammond, and T. Whitehouse, "How choice of mouse may affect response timing in psychological studies," *Behavior Research Methods, Instruments, & Computers*, vol. 35, no. 2, pp. 276–284, 2003.

[29] J. Maskiewicz, B. Ellis, J. Mouradian, and H. Shacham, "Mouse trap: Exploiting firmware updates in usb peripherals," in *8th USENIX Workshop on Offensive Technologies (WOOT 14)*, 2014.

[30] Y. Li, J. M. McCune, and A. Perrig, "VIPER: verifying the integrity of PERipherals' firmware," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 3–16.

[31] T. corp., *How to upgrade the new firmware.*, Topfield. [Online]. Available: http://www.toppy.org.uk/downloads/firmware.php

[32] R. Wojtczuk and A. Tereshkin, "Attacking intel BIOS," *BlackHat, Las Vegas, USA*, 2009.

[33] P. Gershteyn, M. Davis, and S. Shenoi, "Forensic analysis of BIOS chips," in *Advances in Digital Forensics II*. Springer, 2006, pp. 301–314.

[34] R. Wilkins and B. Richardson, "UEFI Secure Boot in Modern Computer Security Solutions," UEFI forum Tech. report, Tech. Rep., 2013. [Online]. Available: http://www.uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2013.pdf

[35] L. Zhang, S.-g. Hao, J. Zheng, Y.-a. Tan, Q.-x. Zhang, and Y.-z. Li, "Descrambling data on solid-state disks by reverse-engineering the firmware," *Digital Investigation*, vol. 12, pp. 77–87, 2015.

[36] A. iOS team, "iOS Security-white paper," Apple Inc., Tech. Rep., 09 2015. [Online]. Available: https://www.apple.com/business/docs/iOS_Security_Guide.pdf

[37] "iOS 9 Cydia." [Online]. Available: http://www.ios9cydia.com/ios-9-3-jailbreak.html

[38] "App Store - Apple Developer." [Online]. Available: https://developer.apple.com/app-store/

[39] "TP-Link router sending username and passwords in plain text." [Online]. Available: https://www.reddit.com/r/AskNetsec/comments/2vmikg/tplink_router_sending_password_in_plain_text_plus/

[40] "Deploy studio Forum firmware passwords in plain text." [Online]. Available: http://www.deploystudio.com/Forums/viewtopic.php?id=4974

[41] "EE BrightBox router hacked." [Online]. Available: https://scotthelme.co.uk/ee-brightbox-router-hacked/

[42] M. Predko and M. Predko, *PC PhD: Inside PC Interfacing*. McGraw-Hill Professional, 1999.

[43] "How to Set a Firmware Password on a Mac." [Online]. Available: http://osxdaily.com/2014/01/06/set-firmware-password-mac/

[44] D. Becker, R. K. Singh, and S. G. Tell, "An engineering environment for hardware/software co-simulation," in *Proceedings of the 29th ACM/IEEE Design Automation Conference*. IEEE Computer Society Press, 1992, pp. 129–134.

[45] C. Bernard and F. Clermidy, "A low-power VLIW processor for 3GPP-LTE complex numbers processing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*. IEEE, 2011, pp. 1–6.

[46] *How the NSAs Firmware Hacking Works and Why Its So Unsettling*. [Online]. Available: https://www.wired.com/2015/02/nsa-firmware-hacking/

[47] I. Sutherland, G. Davies, and A. Blyth, "Malware and steganography in hard disk firmware," *Journal in computer virology*, vol. 7, no. 3, pp. 215–219, 2011.

[48] C. Miller, "Battery firmware hacking," *Black Hat USA*, pp. 3–4, 2011.

[49] G. Barrett and R. Omote, "Projected-capacitive touch technology," *Information Display*, vol. 26, no. 3, pp. 16–21, 2010.

[50] H. McGraw, *Dictionary of Scientific and Technical Terms*, S. P. Parker, Ed. McGraw-Hill Professional, 2003.

[51] Philips, "User Manual of Philips Sensotouch 3D," 2014. [Online]. Available: http://download.p4c.philips.com/files/r/rq1280_22/rq1280_22_dfu_eng.pdf

[52] Using a Force Touch trackpad. [Online]. Available: https://support.apple.com/en-gb/HT204352/

[53] If you cant update or restore your iPhone, iPad, or iPod touch. [Online]. Available: https://support.apple.com/en-us/HT201263

[54] iPhone DFU mode explained, and how to enter DFU mode on your iPhone. [Online]. Available: http://osxdaily.com/2010/06/24/iphone-dfu-mode-explained-and-how-to-enter-dfu-mode-on-your-iphone/

[55] E. Sadun, M. Grothaus, and S. Sande, "Putting Your Data and Media on Your iPad," in *Taking Your iPad 2 to the Max*. Springer, 2011, pp. 19–49.

[56] *Hewlett-Packard LaserJet 4200/4300 Series Printers - Firmware Update/Download Release/Installation Notes*. [Online]. Available: http://whp-hou4.cold.extweb.hp.com/pub/printers/software/lj4200lbreadmefw.txt

[57] C. Seibold, *Big Book of Apple Hacks: Tips & Tools for unlocking the power of your Apple devices*. " O'Reilly Media, Inc.", 2008.

[58] S. Hanna, R. Rolles, A. Molina-Markham, P. Poosankam, J. Blocki, K. Fu, and D. Song, "Take Two Software Updates and See Me in the Morning: The Case for Software Security Evaluations of Medical Devices." in *HealthSec*, 2011.

[59] L. Duflot, Y.-A. Perez, and B. Morin, "What if you cant trust your network card?" in *Recent Advances in Intrusion Detection*. Springer, 2011, pp. 378–397.

[60] Talking to the Masterminds Behind the Nikon Hacker Project. [Online]. Available: http://www.thephoblographer.com/2013/08/04/talking-to-the-masterminds-behind-the-nikon-hacker-project/#.VwRX38cglmk

[61] "Lego Man in Space." [Online]. Available: http://www.legomaninspace.com//tmagin/FPGAs.html

[62] A. Aviv, P. Cerny, S. Clark, E. Cronin, G. Shah, M. Sherr, and M. Blaze, "Security evaluation of es&s voting machines and election management system," *USENIX*, 2008.

[63] A. Cui and S. J. Stolfo, "A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan," in *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 2010, pp. 97–106.

[64] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Wey, "Security requirements for automotive on-board networks," in *Intelligent Transport Systems Telecommunications,(ITST), 2009 9th International Conference on*. IEEE, 2009, pp. 641–646.

[65] M. S. Idrees and Y. Roudier, "Computer Aided Design of a Firmware Flashing Protocol for Vehicular On-Board Networks," Eurecom, Tech. Rep., 2009. [Online]. Available: http://www.eurecom.fr/en/publication/2899/download/rs-publi-2899.pdf

[66] D. K. Nilsson and U. E. Larson, "Secure firmware updates over the air in intelligent vehicles," in *Communications Workshops, 2008. ICC Workshops' 08. IEEE International Conference on*. IEEE, 2008, pp. 380–384.

[67] M. Shavit, A. Gryc, and R. Miucic, "Firmware update over the air (FOTA) for automotive industry," SAE Technical Paper, Tech. Rep., 2007. [Online]. Available: http://papers.sae.org/2007-01-3523/

[68] M. Guven and M. Lorang, "Automated over-the-air firmware update for a wireless phone," Sep. 18 2007, uS Patent App. 11/857,090.

[69] K. Markantonakis *et al.*, *Smart cards, tokens, security and applications*. Springer Science & Business Media, 2007.

[70] M. E. Soper, D. L. Prowse, and S. Mueller, *CompTIA A+ 220-701 and 220-702 Cert Guide*. Pearson Education, 2011.

[71] H. Mansor, K. Markantonakis, R. N. Akram, and K. Mayes, "Don't Brick Your Car: Firmware Confidentiality and Rollback for Vehicles," in *Availability, Reliability and Security (ARES), 2015 10th International Conference on*. IEEE, 2015, pp. 139–148.

[72] E. Vartiainen, V. Roto, and A. Popescu, "Auto-update: a concept for automatic downloading of web content to a mobile device," in *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*. ACM, 2007, pp. 683–689.

[73] O. M. Alliance, "Firmware Update Management Object," *Open Mobile Alliance Ltd., Version*, pp. 1–0, 2006.

[74] "How to update Android smartphone." [Online]. Available: http://www.pcadvisor.co.uk/how-to/google-android/how-update-android-smartphone-or-tablet-summary-3614890/

[75] N. Trigoni, Y. Yao, A. J. Demers, J. Gehrke, and R. Rajaraman, "Hybrid Push-Pull Query Processing for Sensor Networks." in *GI Jahrestagung (2)*, 2004, pp. 370–374.

[76] Z. Basnight, J. Butts, J. Lopez, and T. Dube, "Firmware modification attacks on programmable logic controllers," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 76–84, 2013.

[77] "Update the software on your Apple watch." [Online]. Available: https://support.apple.com/en-gb/HT204641

[78] "The Story Behind Bluetooth Technology." [Online]. Available: https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth