# Security management on Arduino-based electronic devices

Jorge Sainz Raso, Sergio Martín*, Gabriel Diaz,
*Electrical and Computer Engineering Dep.*
*UNED-Spanish University for Distance Education*
Madrid, Spain
Email: jorgesr86@gmail.com, smartin@ieec.uned.es, gdiaz@ieec.uned.es

***Abstract*—Arduino has arisen as a very popular element among the devices, platforms and communication protocols that make up the Internet of Things (IoT). This popularity has grown because it is low-cost and flexible device but with a huge potential for home-made or educational electronic projects. However, due to the low-cost requirement design, this device has some hardware limitations and vulnerabilities that must be carefully studied for each project. This article analyzes different software and hardware vulnerabilities that can be found in different version of Arduino boards. Finally, some good practices and recommendations are presented in order to mitigate presented vulnerabilities.**

## I. Introduction

Embedded devices are systems with computer-hardware and embedded software, designed to do a specific task. It can be an independent system or be part of bigger systems like SCADA networks, cars or domotic sensors [1]. Currently, embedded devices are a fundamental part of the Internet of Things (IoT).

These embedded devices, or the systems they are part of, may be exposed to the internet, what involves a risk to any element connected inside the organization. If a device is vulnerable, either by its hardware, software or its non-encrypted communications, it will turn it into unauthorized entry point to any the network.

Arduino is a modular and easy-to-use low cost embedded platform used in thousands of projects of any kind [3]. This device can interact with multiple sensors and communicate through divers protocols, and it is extensible with other modules called *shields*, if any other functionality is required. One of the latest released Arduino board, the Arduino YUN, can run the OpenWrt-Yun Linux distribution. In this board, Linux and Arduino applications run on two well-defined hardware device sections and communication between them uses the Bridge library as figure 1 shows.

Arduino vulnerabilities and weaknesses must be well known in production environments in order to minimize risks on production environments. Bad settings in both of hardware and software will lead to high security breaches that give an attacker free way to get unauthorized data of the organization [9].

At the time this article is being written, there is a gap on the literature concerning specific vulnerabilities on Arduino. Some few articles can be found regarding IoT and embedded devices security but almost none are completely focused in Arduino hardware.

Some authors like [22] focus on embedded system hardware cloning what doesn't impact Arduino devices. Currently, all Arduino hardware design is public and has free rights so anyone can use this design as base and create new devices. Then, there's no interest on protecting an Arduino device against cloning.

Other authors like Qifeng Chen et al. [21] focus on the generic impact of IoT devices security on the network. In this case, the article checks the number of lost packages on the network under a DoS attack. These kind of generic-attack articles do not study on the impact on the device as it is done in this article but general consquences of the attack on IoT.

Other authors like Audrey Ann Gendreau give a list of generic attack patterns analysis on IoT and impact measurement using a single version of an Arduino board as a case of study [10]. Carlos Alberca et al [5] focuses on the Yun Arduino but the article does not go further on vulnerability study for other boards.

Arduino boards and required software are open-source what means that anybody can design its own Arduino-based board. As a result, it is possible to find on the market derivative or cloned boards with original or modified Arduino software so users must be careful when using an Arduino board if it is not certified.

The goal of this article is to analyze hardware and firmware security aspects on Arduino boards, with and without Operating System (OS). Conclusions of this paper highlight main Arduino security issues and provide some good practices that can be used in order to limit presented risks.

## II. Metodology

This work involves vulnerabilities analysis in hardware, software and communication of Arduino devices with and without OS. Currently, a vast number of Arduino devices can be found on the market so, in order to keep a focused goal, the text focuses on a limited number of Arduino devices. Regarding devices with OS, vulnerabilities on Arduino YUN has been analyzed. For classic Arduino boards, Mega, Uno and MKR 1010 have been used. Also, an Arduino clone, Wemos D1R2 was a target of our tests.

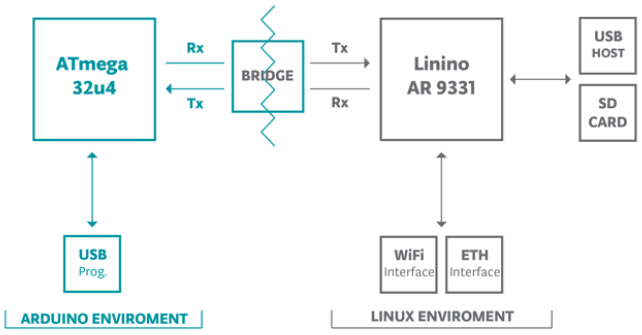


Fig. 1. Communication between Arduino and Linux using *Bridge* library

Described Arduino analysis starts with a reading of the documentation, including design and connectors. The Arduino hardware vulnerabilities presented are only theoretically described from an analysis of the schemas and description of the boards Due to the high risk of board damage, applying these attacks on our Arduino cards would certainly damage the boards. For firmware analysis, Arduino IDE 1.0.5 has been used to install sketches.

Yun hardware contains two hardware modules that communicates with each other so analysis goes further than for simple Arduino boards. This study implies vulnerabilities analysis of hardware and software on the microprocessor module. Also, communication betwen both modules may expose vulnerabilities so this communication has been analyzed too.

This text also describes found network communication vulnerabilities in the previously presented Arduino boards. A dennial-of-services (DoS) attack has been performed on the presented classic Arduino boards without OS and in YUN in order to compare behavior. Used tools for DoS attacks have been DoS.Linux.SSPing.10 script, Hulk and Hping3.

Finally, after vulnerabilities analysis, a set of good practices is proposed.

## III. Arduino

### A. Hardware vulnerabilities

The hardware of Arduino devices has been designed to perform low-cost fast-prototyping but not to keep a high security or safety level [3]. Some shown vulnerabilities are part of the decision of creating a low-cost device and this must be taken into account when this device is used.

Firstly, none of the Arduino devices are protected to work in hard environments where dust or humidity are present. Some industrial conditions require devices to be resilient to these elements. However, Arduino devices will require of suitable cases and adapted connectors in order to work under that hard conditions.

Arduino is not protected against overvoltage or overcurrent on the source or I/O pins. Shortcutting any I/O pin or applying more than the expected voltage will damage the board. While doing tests, a beginner user in electronics may perform actions

| | Recommended Input source (V) | Limit input Source (V) | DC current per Pin | DC current for 3.3V pin |
|---|---|---|---|---|
| Micro | 7-12 | 20 | 20mA | 50mA |
| MEGA 2560 REV3 | 7-12 | 20 | 20mA | 50mA |
| Leonardo With headers | 7-12 | 20 | 40mA | 50mA |
| Uno rev3 | 7-12 | 20 | 20mA | 50mA |
| Duemilanove | 7-12 | 20 | 40mA | 50mA |
| Yun rev 2 | 5 | 5 | 40mA | 50mA |

Fig. 2. Arduino boards, source requirements, source limits and DC pins requirements

that has no turning back like connecting a LED with no resistor. If the LED has no internal resistance, it will create a shortcut between pins that will damage the board.

Users also must take care of pin voltage when connecting external devices to the pins. As example, Arduino devices can be connected using a serial port using the TX/RX pins. However, they cannot be connected directly to a RS232 interface. RS232 operates with +/- 12V but Arduino uses 5V or 3.3V on their pins. Then, a logic level converter should be used.

User must also take care of using supported and stable voltage and current on the Arduino boards and connected devices. Figure 2 shows a summary of source limitations and current limitations on some Arduino boards. Users must also be careful with the power source used as low-quality power supply may harm the board.

Some Arduino devices have a socket to connect its microcontroller like Arduino UNO, what make easier for an attacker to extract the microcontroller and analyze its behavior. More recent Arduino boards keep microcontroller welded on the board, which make extraction more complicated. In this case, the microcontroller must be unwelded and there is a high risk of damage.

Other hardware attacks less invasive than unweld chips are based on making the microcontroller to work erroneously by forcing it to work in undervoltage or overvoltage conditions. When working under these conditions, some microcontrollers just do not react to signal transits. Then, in some processors, some fast signal may deactivate security without destroying the protected data [2].

However, hardware analysis oriented to open-design microcontrollers has no sense. Indeed, Arduino design is open and free to be used in other designs so anybody may know how internals works. This attack is used on close-design microcontrollers or microprocessors.

### B. Firmware vulnerabilities

Most Arduino systems run without operating systems. Arduino boards without operating system are programmed by writing a set of instructions in its microcontroller [3] These instructions can be written using Arduino programming language or Arduino software and uploaded to the board.

Arduino boards include an already-to-use installed bootloader which may be a door to a malicious third party if he has access to the hardware [8]. This default bootloader helps developers to quickly install a program using a UART-USB

| Arduino | Microcontroller | Speed | Memory | | | |
|---|---|---|---|---|---|---|
| | | | flash | SRAM | EEProm | ROM |
| Micro | ATmega32U4 | 16 MHz | 32 KB | 2.5 KB | 1 KB | |
| MEGA 2560 REV3 | ATmega2560 | 16 MHz | 256 KB | 8 KB | 4 KB | |
| Due | AT91SAM3X8E | 84 MHz | 2 x 256 KB | 64 + 32 KB | - | 16 KB |
| Leonardo | ATmega32U4 | 16 MHz | 32 KB | 2.5 KB | 1 KB | |
| Uno rev3 | ATmega328P | 20 MHz | 32 KB | 1 KB | 1 KB | |
| Duemilanove | ATmega168 | 20 MHz | 16 KB | 1 KB | 512 B | |
| Yun rev 2 | ATmega32U4 | 16 MHz | 32 KB | 2.5 KB | 1 KB | |

Fig. 3. Arduino boards, microcontroller speed and memory capacity

wire directly connected to the Arduino board. If a malicious third party has usb-hardware access to the board, he will be able to easily upload a new firmware.

Arduino default bootloader ads some delay on booting. This delay is between around 1,5 seconds on the latest versions to 10 seconds on the oldest ones. In some environments, like drone controllers, this booting delay can lead to dangerous situations.

Luckily, default bootloader can be rewritten or completely deleted using an external programmer [7]. If bootloader is removed, firmware reprogramming can only be done using the external programmer. This requires a new piece of hardware and higher skills to reprogram the hardware.

If an attacker has an external programmer and access to the boards, it will be possible to him to extract the firmware in binary format [6] as flash memory data is not encrypted. The downloaded code can be afterwards disassembled and converted in human-readable code using tools like HexRays.

In order to prevent a third unauthorized actor to access the firmware, the security bit of the microcontroller can be activated. However, this bit does not prevent a program on the bootloader from reading the flash memory. Then, in order to secure completely the flash memory, the bootloader must be deleted.

As any other software, Arduino firmware can be attacked exploiting buffers overflow bugs in order to get access to unauthorized data. This attack modifies normal application flow by writing data in unexpected memory addresses [4]. Even though this attack is complex to perform on Arduino microcontrollers due to its stack operation, it is not impossible. This kind of attack can be really dangerous as an attacker could even execute bootloader code to overwrite the firmware.

Arduino microcontrollers does only have little memory size so bad designed software will end up in fatal states if memory is full. Full SRAM avoids the sketch to run as expected and there is no easy way to detect this problem [14]. Arduino development and testing require attention and good practices to avoid full-memory problems. A summary with memory capacities can be seen in figure 3.

When developing on Arduino, stack vulnerabilities must be avoided during development as running-time protections are not available. Protection techniques like Address Space Layout Randomization (ASLR) or Stack Canaries that can avoid some stack attacks are not supported for AVR devices. If we try to compile any application using stack protection options, the compiler will complain about it as we can see on figure 4.

When working with calculations, developers have to take

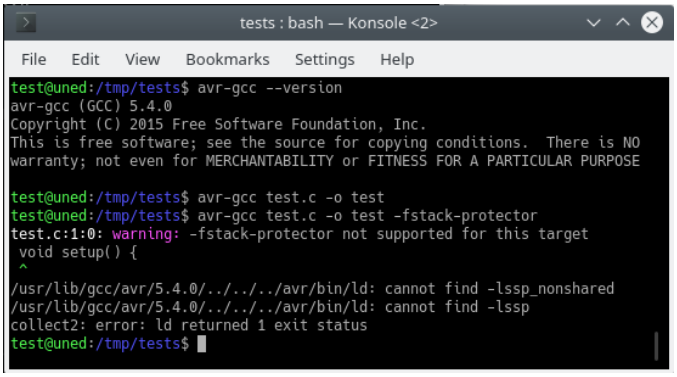


Fig. 4. Cross-compiler avr-gcc complains about stack protector option

care about the different data type range value on each board [16]. Depending on the Arduino board, different data types use different stored bits. For example, integer uses two bytes except for Arduino Due and SAMD based boards like MKR1000 and Zero, where 4 bytes are used. Double data uses 8 bytes for the Due board and 4 bytes for the rest of boards. If this is not taken into account when developing, calculations end up using numeric values outside the rang, what leads to erroneous results.

Communication stack implementations are also vulnerable to attacks. In this article, in order to compare resilience to DoS attacks on the WiFi stack, this attack is performed against the Arduino MEGA with a WiFly shield card, Arduino MEGA with an ethernet shield, Arduino YUN, MKR 1010 and Arduino UNO with an ethernet shield. A test on a Wemos D1R2 card was also done. In order to perform these tests, an sketch of a simply web server was uploaded to Arduino. For example, for Arduino mega, the SparkFun Wifly WebServer was used. Chosen DoS tools have been DoS.Linux.SSPing.10 script, Hulk and Hping3

Arduino YUN board on Wifi, Arduino MEGA with ethernet connection, MKR 1010 on wifi and Arduino UNO with ethernet shield, tolerate all a SYN flooding attack and a HTTP DoS attack. While attacks are being performed, the access to the servers becomes almost impossible. However, once attacks stop, server becomes accessible again with no need to restart the board.

Arduino MEGA with the Wifi shield loses connection after only 100 ICMP packets sent on few seconds. After the attack, the MEGA board does not answer to any more ping request neither to web requests as it can be seen on figure 5. In order to recover the connection on the board, a full reboot is required. Figure 6 shows the YUN board response time to pings while performing the DoS. Chart shows that for some seconds, response time increases but finally Linux can handle it and recovers normal response timing.

Wemos D1R2 board has stability problems as soon as a DoS attack starts. This boards constantly reboots because of an exception as it can be seen on image 7. This happens in both attacks: SYN flooding and HTTP DoS.

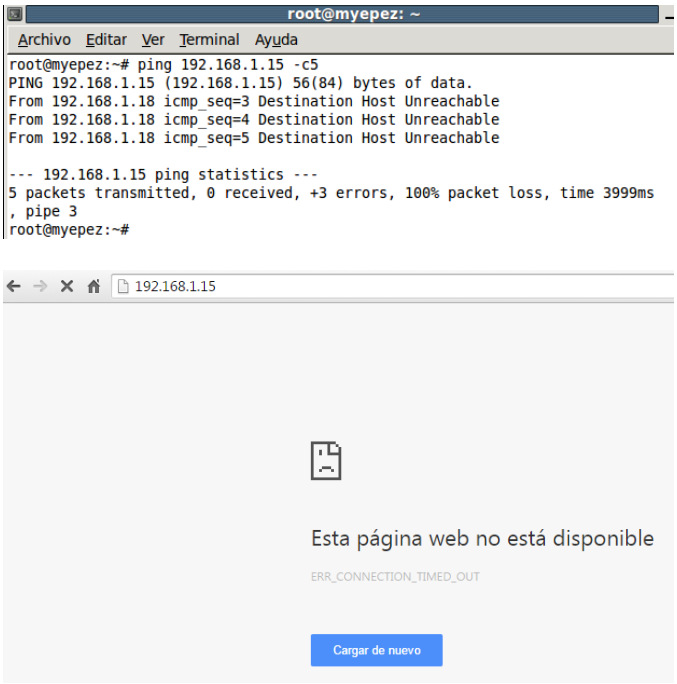


Fig. 5. Arduino Mega not responding to neither pings (up) nor web request (down) after an attack of 100 ICMP packets

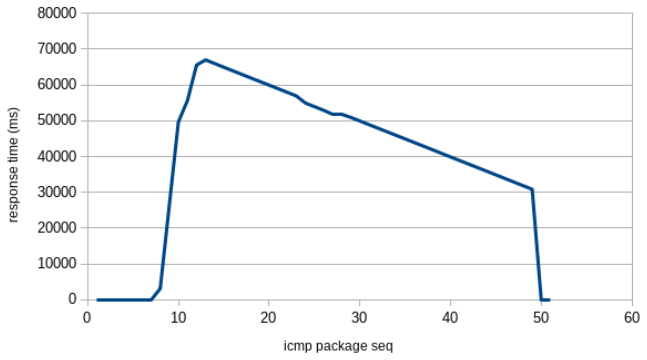


Fig. 6. Response time of ping command while performing a DoS attack

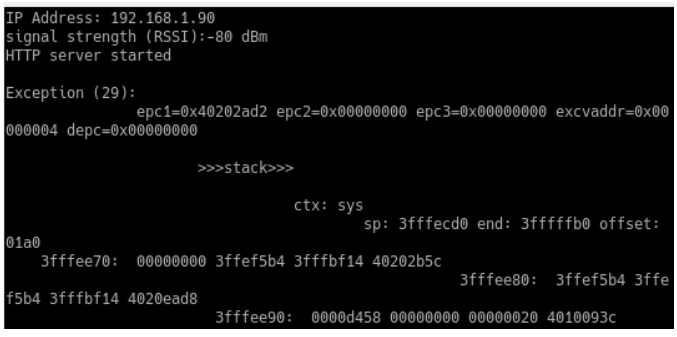


Fig. 7. Wemos D1-R2 crash under DoS attack

## IV. YUN VULNERABILITIES

Yun Arduino board is an interesting board to analyze due to its architecture and operating mode. Yun is composed of two well defined modules: AVR ATmega32U4 microcontroller and a MIPS Atheros AR9331 microprocessor. The microcontroller runs Arduino code as it is already known. The processor runs a customized Linux distribution based on OpenWrt called OpenWrt-Yun.

### A. Hardware

On the Arduino YUN, evil parties with hardware access could reset the board to its factory values, what may lead to default configuration vulnerabilities. Holding down the Arduino's WiFi reset button for more than 30 seconds will reset it to factory conditions. If pressing down between 5 and 30 seconds, it will only reset the WiFi networking settings.

Arduino Yun board does not have an integrated voltage regulator unlike other Arduino. Thanks to the integrated voltage regulator, Arduino boards support a little range of power input as can be seen in figure 2 However, Arduino Yun must be supplied with a regulated 5V DC power and any higher voltage will damage the board. Moreover, some parts of the Arduino Yun board, like the Wifi module, are quite sensitive to unstable DC current. Unstable current may reset or destroy the board [13].

It is important for developers to know storage limitations. Because of the limited number of writes, programmers should not use the built-in non-volatile memory. If persistent data must be stored, an external storage device should be used.

Arduino devices can communicate with other devices using the serial port. Arduino boards have at least one serial interface and Yun board has two: Serial and Serial1 [18]. Serial is used to communicate with an external device and Serial1 communicates Linux microprocessor and Arduino microcontroller.

Even Serial1 interface is reserved to board internal communication, it has the 0 (RX) and 1(TX) external pins, where any person with physical access to the devices could connect. However, OpenWrt sets the communication baudrate to 250000 [19] and this is not commonly supported by serial clients because of the difficulty to handle errors using that speed. If an attacker had access to the Arduino devices and were able to read on 250 Kbauds, he could use two serial cables to read from RX and TX and see data exchange between microprocessor and microcontroller.

### B. Software

On the Yun board, communication between both modules -Arduino and Linux- is done using the Bridge library which can also be exploited. An attacker can take advantage of buffer vulnerabilities on any of the parts to use the Bridge library [5]. Thus, an attack coming from Arduino microcontroller can impact on the Linux section and the other way around.

On some old Yun boards are delivered with the old Lilino Linux version, which contains too many issues to move it to a production system [13]. Some issues that a user may experiment with this old version can include: liability of bridge

library, WiFi stability, undevelopped functions (fileio), stability when reading big files. Reported issues was fixed on the OpenWrt version.

At the time of writing this article, included Python version is 2.7, which end of life date is April 2020 [17] This means that current version of OpenWrt contains an almost-outdated version of Python that must be updated in short time. Developers should foresee this update and code accordingly to this situation.

OpenWrt has security breaches, including passwords in clear text, unpatched daemons, vulnerable kernel or lack of integrity checks on critical files [5]. Keeping default configuration for production devices may imply security issues too.

Default configuration is a common vulnerability existing in devices with operating system, what also apply to the Arduino YUN. Some of this default configuration involves default password and non-secure HTTP requests. Moreover, this vulnerability may even impact on uploaded sketches.

As example, default root password, which is used to connect to the web CLI or through ssh, is *Arduino*. This password can be changed through the CLI but as it is served using non-secured connection, any attempt to change the password may be sniffed. HTTP API requests are done, on default configuration of Arduino, using a non-secure channel. HTTP server used is uhttpd and its configuration file can be found in /etc/httpd.conf [11].

An attacker that knows the Arduino Yun Linux password and with access to the same network of the device will be able to upload any sketch. Arduino software -online or local one- detects if an Arduino Yun is connected to the same network and allows the user to upload the sketch through Linux using the network. Arduino password will be asked so, as said before, this password must be kept safe.

Unlike the Arduino unit, the Linux part can run applications with stack protections. Even existing, risks of stack vulnerabilities existing on applications running on OpenWRT can be minimized on production systems using protection techniques. These protection techniques like like Address Space Layout Randomization (ASLR) or Stack Canaries will force the application stop if expected flow is not followed.

As it can be seen on figure 1, network devices on YUN boards are handled by Linux, what means that there are potentially more hardware resources and powerful software to perform network management. As any other Linux OS, lots of tools can be installed in order to manage network as iptables and snort. IPTables is generally used as firewall as it provides packet filtering, network address translation (NAT) and other packet advance management. Snort is real-time traffic analysis and packet logger that can be used as Intrusion Prevention System (IDS).

### C. Asynchronous booting

The two Arduino Yun modules boot processes are completely independent and with no synchronization. On one side, the AVR ATmega32U4 microcontroller runs an uploaded sketch as any other Arduino board. On the other side, the MIPS Atheros AR9331 microprocessor boots the OpenWrt-Yun Linux distribution.

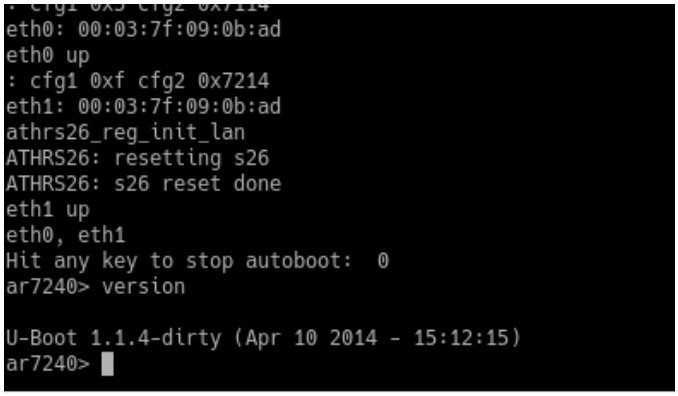


Fig. 8. U-Boot on Arduino Yun OpenWrt-Yun booting process

The two Arduino Yun modules can be also reset independently using the corresponding reset buttons. This means that if the microcontroller is reset, the Linux system keeps running. It also works the other way around: if the microprocessor is reset, only Linux will reboot and the microcontroller sketch continues running.

Developers must take care of this asynchronous booting when communication is done between systems as asynchronous booting and reset can easily lead to race conditions. When the Linux part is booting it sends and reads information from the serial port /dev/ttyATH0. This port is the Serial1 and it is used to communicate the microprocessor and the microcontroller. This means that a sketch can potentially send or read data while Linux is booting.

Linux boot process can be interrupted on u-boot. As it can be seen on image 8, any key press information received by the microprocessor will stop the booting process. A bad-coded sketch can completely stop the Linux booting process or worst, override the booting system with incorrect information. A malicious third party can take advantage of this and upload a sketch that reflash the Yun with a modified OpenWrt system [20].

When OpenWrt-Yun offers the user to perform a failsafe boot, which will not initialize the WiFi connection. In order to enter this mode, the user is asked to press a 'f' on OpenWrt-Yun boot as it can be seen on image 9. Even if this is a low-probability case, a sketch could send a 'f' character through the serial port, forcing Linux to boot in failsafe mode.

Unexpected rebooting can also impact on sketches that reads information from Linux.

## V. Discussion and conclusions

This article has analyzed vulnerabilities and weaknesses of some Arduino boards. This analysis include Arduino board with and without OS. A summary of the Arduino vulnerabilities can be found on figure 10. Regarding the vulnerabilities found on Yun, the Arduino board with OS, a summary can be found on figure 11.

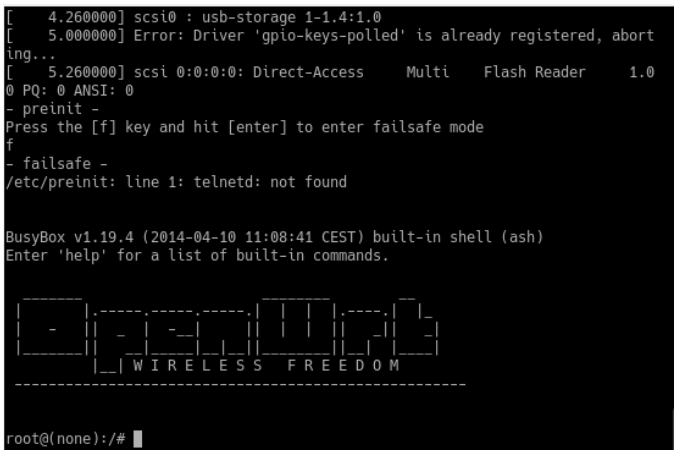


Fig. 9. Failsafe boot option choose on Arduino Yun OpenWrt-Yun booting process

| Arduino |
|---|
| **Hardware** |
| No hard environments protection (dust, humidity…) |
| No Overvoltage or overcurrent protection on I/O pins |
| No Input overvoltage protection |
| No unstable voltage or current protection |
| |
| **Firmware** |
| Easy booloader override if physical access |
| Easy sketch upload if physical access |
| Booting delay: between 1.5 & 10 seconds |
| Easy sketch download for later analysis |
| No stack protection on running sketchs |
| |
| **Resiliance to DoS attacks** |
| Arduino Mega lose connection |
| Wemos D1R2 reboots |

Fig. 10. Summary of Arduino boards vulnerabilities

Users must take care of all these vulnerabilities because even if this product is only used on testing environments, it can be cracked over the network. This unauthorized access will give an attacker free way on the organization network.

Hardware analysis shows that the Arduino design is more focused on decreasing costs than on security. This can be seen, for example, in the lack of overvoltage or overcurrent on the source or I/O pins. There is neither an in-built tamper protection against malicious hardware access.

Regarding the hardware, there are some projects trying to create harder boards, resilient to user mistakes and hard environments. As example, RC Rugged Circuits takes Arduino boards design as a base, and they improve the hardware. However, they do not propose an improved version of all available Arduino boards and hardware is much more expensive.

Arduino users must keep Arduino low-cost goal design in their minds and should invest some extra effort on securing it. Physical access must be avoided for non-authorized parties. If

| Yun |
|---|
| **Hardware** |
| Limited numbers of write on memory |
| Communication between Microcontroller and Microprocessor can be read on I/O pins |
| |
| **Software** |
| Buffer overflow on Arduino part may impact on Linux module |
| Python 2.7 (end of life in april 2020) |
| Default configuration on OpenWRT |
| No encryption on configuration server |
| |
| **Communication Arduino-Linux** |
| Asynchronous booting – complicated synchronization |

Fig. 11. Summary of Yun vulnerabilities

untrustworthy people have access to the physical system, they can manipulate the environment easily: shutdown the system, change network connections or even connect to the system using its ports.

By having physically connection, an attacker can also extract and replace stored firmware. Unauthorized access can sometimes be avoided by activating the security bit of the microcontroller. This bit prevents the flash memory from being read from external sources.

Arduino low-cost design implies low-memory availability what requires from a developer a good memory-use sight. Full memory problems are not easy to detect as the sketch continue running. However, this sketch will behave unexpectedly and can corrupt the memory.

As other software, Arduino firmware can be attacked using buffer overflow vulnerabilities. These vulnerabilities imply memory corruption and may let an attacker take full control of the device. During the development, the number of these bugs can be minimized by using good development practices.

Memory corruption could be avoided using techniques like ASLR or Stack Canaries but Arduino modules do not implement these techniques. However, stack smashing protection are used by default on Arduino Yun when applications run on the Linux module [15].

Regarding Arduino YUN, it runs OpenWrt-Yun Linux distribution, what potentially contains same bugs on the modules it is composed off. Found security breaches include passwords in clear text, unpatched daemons, vulnerable kernel or lack of integrity checks on critical files. Keeping default configuration is also dangerous so the user should change it as soon as possible.

As any other IoT device, Arduino boards are exposed to DoS attacks. Results from this article show that Arduino MEGA, with no operating system, required to reboot in order to recover from a DoS attack on Wifi. However, other Arduino boards was able to survive to the performed DoS attack.

Arduino Yun is a much more powerful system that gives the user more resources in hardware and software by including a Linux segment on the board. Using an operating system

gives the user powerful tools like firewalls or IDSs to manage network communications. These tools are not available in any other Arduino board but in Yun.

However, even if Arduino Yun seems to be a great board to use in any environment, independent boot of their two modules makes development more complex. This complexity comes from the need of handling possible reset of one of the modules while the other keeps working. Applications must take care of the correctness of the sent and received data as any unexpected data can make the system to stop working.

In conclusion, due to its price but the lack of security on its design, Arduino devices are interesting to learn or prototype environnements, but they should not be used in critical environments with no surveillance. Good practices are required in development and deployment of the systems. However, Arduino company knows that no board or software is vulnerability-free so a vulnerability disclosure policy was created in order to researches to feel at ease when reporting vulnerabilities. This guideline is present on its web as Coordinated Vulnerability Disclosure (CVD) Policy [12].

## REFERENCES


[1] Raj Kamal. "Embedded Systems: Architecture, Programming and Design", McGraw-Hill 2007

[2] Copy Protection in Modern Microcontrollers. https://www.cl.cam.ac.uk/~sps32/mcu_lock.html. Last access: 24/11/2019

[3] What is Arduino?. https://www.arduino.cc/en/Guide/Introduction. Last access: 24/11/2019

[4] Stack Exchange Electrial Engineering: Exploiting stack buffer overflows on an Arduino. https://electronics.stackexchange.com/questions/78880/exploiting-stack-buffer-overflows-on-an-Arduino. Last access: 24/11/2019

[5] Carlos Alberca, Guillermo Suarez-Tangil, Sergio Pastrana and Paolo Palmieri. "Security Analysis and Exploitation of Arduino devices in the Internet of Things". https://cosec.inf.uc3m.es/docs/papers/2016mal-iot.pdf. Last access: 22/11/2019

[6] Arduino Forum: Extracting Code from an Arduino https://forum.arduino.cc/index.php?topic=403201.0. Last access: 22/11/2019

[7] Arduino: Bootloader Development https://www.arduino.cc/en/Hacking/Bootloader. Last access: 22/11/2019

[8] Arduino: Bootload the Arduino Mini https://www.arduino.cc/en/Hacking/MiniBootloader. Last access: 22/11/2019

[9] J. Sainz-Raso and S. Martin and G. Diaz and M. Castro. "Security Vulnerabilities in Raspberry Pi–Analysis of the System Weaknesses", IEEE Consumer Electronics Magazine, Volume 8, Number 6, Year 2019, Month November, Pages 47-52

[10] Internet of Things: Arduino Vulnerability Analysis Audrey Gendreau https://www.researchgate.net/publication/305731153_Internet_of_Things_Arduino_Vulnerability_Analysis. Last access: 25/11/2019

[11] Change Yun REST API URL https://forum.arduino.cc/index.php?topic=309960.0. Last access: 25/11/2019

[12] Coordinated Vulnerability Disclosure (CVD) Policy https://www.arduino.cc/en/Main/Security. Last access: 26/11/2019

[13] Arduino Yun Playground https://playground.arduino.cc/Hardware/Yun. Last access: 09/12/2019

[14] Arduino Memory https://www.arduino.cc/en/tutorial/memory. Last access: 10/12/2019

[15] OpenWrt security features https://openwrt.org/docs/guide-user/security/security-features. Last access: 10/12/2019

[16] Arduino data type https://www.arduino.cc/reference/en/#variables. Last access: 16/12/2019

[17] End of Support of Python 2020 https://www.python.org/psf/press-release/pr20191220/. Last access: 25/12/2019

[18] Arduino serial communication https://www.arduino.cc/reference/en/language/functions/communication/serial/. Last access: 09/01/2019S

[19] Arduino (gnu) stty limitation? https://forum.arduino.cc/index.php?topic=269120.msg1897208#msg1897208. Last access: 09/01/2019

[20] Reflashing the OpenWrt-Yun image on the Yún https://www.arduino.cc/en/Tutorial/YunUBootReflash. Last access: 15/01/2019

[21] Q. Chen, H. Chen, Y. Cai, Y. Zhang and X. Huang. "Denial of Service Attack on IoT System", 2018 9th International Conference on Information Technology in Medicine and Education (ITME), Hangzhou, 2018, pp. 755-758.

[22] Protecting embedded systems against Class I & Class II cloning attacks using Arduino boards. B. K. Mathew. 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), Kottayam, India, 2018, pp. 1-4