Hardware-based capture-the-flag challenges

Hardware-based Capture-The-Flag Challenges

Paolo PRINETTO
Politecnico di Torino
CINI Cybersecurity National Lab.
Turin, Italy
paolo.prinetto@polito.it

Gianluca ROASCIO
Politecnico di Torino
CINI Cybersecurity National Lab.
Turin, Italy
gianluca.roascio@polito.it

Antonio VARRIALE
Blu5 Labs Ltd.
Ta' Xbiex, Malta
av@blu5labs.eu

***Abstract*—In a world where cybersecurity is becoming increasingly important and where the lack of workforce is estimated in terms of millions of people, gamification is getting a more and more significant role in leading to excellent results in terms of both training and recruitment.**

Within cybersecurity gamification, the so-called Capture-The-Flag (CTF) challenges are definitely the corner stones, as proved by the high number of events, competitions, and training courses that rely on them. In these events, the participants are confronted directly with games and riddles related to practical problems of hacking, cyber-attack, and cyber-defense.

Although hardware security and hardware-based security already play a key role in the cybersecurity arena, in the worldwide panorama of CTF events hardware-based challenges are unfortunately still very marginal.

In the present paper, we focus on hardware-based challenges, providing first a formal definition and then proposing, for the first time, a comprehensive taxonomy. We eventually share experiences gathered in preparing and delivering several hardware-based challenges in significant events and training courses that involved hundreds of attendees.

***Index Terms*—cybersecurity, education, gamification, capture-the-flag, challenges, hardware, hardware security.**

I. Introduction

In recent times, the world is experiencing a digital revolution that leaves no aspect of our life uncovered. Our job, the management and purchase of good and services, and even the organisation of our free time inevitably rely on constantly-connected digital devices. As a consequence, the issue of data security can no longer be ignored and it must be addressed at every level, from the awareness rising of any citizen, up to the massive investments by public and private sectors in order to increase the number of available experts in cybersecurity. Today, we see a very strong push towards hiring Research and Development specialists in cybersecurity, in a plenty of fields and domains. The number of projected unfilled jobs worldwide in cybersecurity has recently been estimated in 3.5 M by 2021 [18]. Such a huge number definitely requires a significant push by institutional education centers such as schools and universities. Nevertheless, even if the number of dedicated academic courses, BS and MS curricula is growing, they still lack a *practical imprint*, on which the attackers are instead very well prepared [27]. Without a significant effort in this direction, presenting cybersecurity from theory-oriented point of view is likely to appear as boring and therefore not attractive to many students.

An important different paradigm in cybersecurity teaching is the one heavily exploiting *gamification* [30] [35]: students are asked to directly face security problems by solving riddles and challenges related to the breakdown or the decryption of software, communication systems or devices, or by implementing countermeasures to prevent attacks by opposing teams. Within cybersecurity gamification, the so-called *Capture-The-Flag (CTF) challenges* are definitely the corner stones, as proved by the high number of events, competitions, and training courses that rely on them. In these events, the participants are confronted directly with games and riddles related to practical problems of hacking, cyber-attack, and cyber-defense. CTF challenges are the basis of the so-called *Capture-The-Flag (CTF) competitions*, where the aim is in fact to extract from the challenge a unique string, the *flag*, which certifies the success.

The results of the CTF in terms of education and creation of new practical knowledge have been recognised by various studies [29] [43] [39], also in relation to their adoption in the context of university courses on computer security [34]. CTF competitions are therefore a valid approach to try to fill the workforce gap, mainly thanks to their attractive power among new generations and to their ability to make participants develop their *adversarial thinking*, which has proved to be essential in defending infrastructures from malicious cyber-attacks [42].

On the basis of these considerations, the Italian CINI Cybersecurity National Laboratory[1] has been developing the *CyberChallenge.IT*[2] program since 2017. CyberChallenge.IT is, in fact, the main Italian initiative aiming at identifying, attracting, recruiting, and placing the next generation of IT security professionals, thus seeking to reduce the lack of IT workforce at the national level. Its target are young talents (aged 16-23) and the 2020 edition has involved more than 4,400 of the best students who live and study in Italy. To create and grow such a community of young cyber-defenders, the program offers training opportunities to stimulate interest in STEM disciplines and, in particular, in information and computer security. Participants also have the opportunity to get in direct contact with IT companies working in the field, which actively contribute to their orientation and professional training. The program combines traditional training activities

[1] https://cybersecnatlab.it/
[2] https://www.cyberchallenge.it/

with a gamification-oriented approach which requires the participation in on-line competitions where different scenarios of networks and real work environments are simulated. The model is unique on the international scene; in fact, not only it exploits gaming as an instrument for attracting young people, but it also offers a multidisciplinary training.

The training process ends with two final competitions, organized at the training node level and at the national level, respectively. The former is a Jeopardy style CTF (see Section II) run concurrently by all the attendees of all the training nodes. The latter, which is in fact the Italian CTF championship in Cybersecurity, is an Attack-Defense style CTF (see Section II), attended by teams of 6 members each, one per training node, and planned each year in a specific location. In 2020, both the competitions have been organized remotely, due to Covid-19 restrictions. Both the competitions exploit infrastructures in terms of servers and software applications, completely developed in-house and managed by the Cybersecurity National Laboratory. Similarly, all the challenges used in both the Jeopardy and the Attack-Defense competitions are brand new and developed in-house. Since the 2020 edition, students experience hardware security techniques and then *hardware-based challenges* during the final competition.

CTFs usually focus on "mainstream" aspects of cybersecurity, such as challenges based on web exploits in which, for example, it is possible to exploit SQL injection [16] or Cross-Site Scripting (XSS) [15] to retrieve the flag, or a step-based challenge where the interaction with a command line is offered by a vulnerable system that hides the flag, for example, in the home folder of some user whose login needs to be cracked, or in some software to be attacked through code injection [17]. This prevalence is caused by the fact that these issues are very popular, and related problems with possible defense techniques have been studied more in depth and for a longer time. However, it is to be pointed out that all layers of an IT system can be subject to threats, from the highest application layers down to the hardware level. Hardware components are subject to intrinsic vulnerabilities and are exposed to particular attacks [45], which can determine an even more marked danger. In fact, hardware is not patchable as a piece of software, and if present, the vulnerability remains until the component is active. Furthermore, hardware is *the root* of systems: if hardware is compromised, all upper layers could be compromised as well, even if protected against web or software attacks [26]. The theme should not be underestimated or downgraded to a *niche* theme, inaccessible to most: it should instead be included in the ecosystem of cybersecurity education and training, included CTF competitions as well.

The present paper focuses on hardware-based challenges, providing first a formal definition and then proposing, for the first time, a comprehensive taxonomy. Some examples of real challenges are then presented, each classified according to the proposed taxonomy. We eventually share experiences gathered in preparing and delivering several hardware-based challenges in different environments. The sequel of the paper is organized as follows. Section II provides a general background on CTF competitions; Section III details hardware-based challenges and proposes a new taxonomy, as well as an overview of the hardware-based challenges proposed in some famous CTF competitions around the world; Section IV reports some examples of hardware-based challenges implemented this year during the CyberChallenge.IT event; Section V concludes the paper.

II. Background on CTF Competitions

A Capture-The-Flag challenge is a game in which the goal is breaching into one or more vulnerable IT assets (websites, files, databases, network devices, hardware devices, and so on) to guess or get a *flag* [32]. The flag is a unique string, decided by the organizers and formatted in a competition-specific manner, which certifies the success in the challenge. Individuals or teams participating to CTF competitions get points for each correct flag submitted to the competition organisers, and the winner is usually the individual or team owning the highest number of points at the end of a given predefined time slot.

Three main different CTF challenge types exits, based on execution modalities and involved actors:

- *Jeopardy*: Participants are asked to face a vulnerable system which hides the flag. The flag can be "captured" by exploiting the vulnerabilities that have been artificially inserted into the system by the competition organizers. Participants may be grouped in teams, but there is no interaction among the teams. The only opponent is the challenge itself.
- *Attack/Defense*: Participants are grouped in teams and each team is given an instance of a system injected with several vulnerabilities. All the instances get the same vulnerabilities and are connected to a same network. The competition includes two phases: for a first period of time (e.g., one hour), each team can access its instance, only, and, during this slot, the team should identify and fix the vulnerabilities on its own instance. In this way they can prevent other teams from capturing their flag exploiting these vulnerabilities during the next phase. In a second phase, connection is opened and each team is free to access the instances of the opponent teams and capturing their flags if the vulnerabilities present in their instances have not been properly patched during the first phase. Points are awarded based on three factors: (i) the number of flags captured on the instances of other teams (*attack points*), (ii) the number of flags stolen by other teams from your instance (*defense points*), (iii) the percentage of time the services remain up and work properly (*SLA points*). With respect to Jeopardy-style, these challenges allow participants to gain experience on both offensive and defensive skills.
- *King of the Hill*: It is a slight variant of Attack/Defense CTF, in which participants are usually grouped in teams, and the goal is taking and holding control of a machine or a network. The challenge last for a given period of time,

and at the end, the team that held the system longest is the winner.

CTFs are usually clustered according to the topics they deal with, as:

- *Binary*: This category includes all those challenges that require the exploitation of a vulnerable software application. The name stems from the abstraction level exploited during the attack, i.e., the machine binary code, often resorting to disassembly and debugging tools. This class can be further split into:
  - *Reversing*: The challenges are based on the backwards reconstruction of the behavior of the application, in order to allow, for instance, a particular interaction to retrieve the flag. Beyond the knowledge of programming languages, a good familiarity with static code analysis tools such as decompilers and disassemblers is often required.
  - *Pwn*: These are the challenges that most closely resemble hackers' activities in the collective imagination, e.g., breaking a remote vulnerable service on a server. The exploit can be carried out by injecting binary instructions into the application's memory through a breach opened by a vulnerability, or by hijacking the execution towards blocks of hidden code or spread bytes that were not originally intended to be executed in that order. For these challenges, the vulnerable binaries can be presented either as *white-box* if source files are made available, or *black-box* if no file is attached.
- *Web*: This category includes all the challenges dealing with vulnerable web services, susceptible to attacks based on command or code injections, which allow to retrieve information that is originally not accessible, including the flag. Examples include challenges based on web login crack, malicious SQL query injections, tampering with cookies, etc.
- *Crypto*: The challenges consist in breaking an encryption scheme to decipher a message that directly or indirectly contains the flag. The encryption scheme can be either a classic one but implemented in a vulnerable way, or a brand new one to be reversed. These are usually the longest challenges in terms of time, because they may require an automatic breaking phase due, for example, to the execution of an *ad hoc* script written by participants. Mathematical knowledge of combinatorics, prime numbers, modular arithmetic are usually very helpful.
- *Forensics*: This category of challenges takes its name from the fact that the techniques used to capture the flag mimic the typical forensic approaches adopted by law enforcement and investigation agencies. They very often exploit steganography, including, for instance, malformed files, packet captures, .jpg or .png files modified to hide texts or executable pieces of code. By digging into these files with scripts and tools, participants can extract data (that are often encrypted) to recover the flag.
- *Networking*: In these challenges actions typical of the network domain (such as: breaking firewalls, deceiving access policies, attempting spoofing attacks and poisoning of network protocols, or reconstructing a message from individual packets) must be exploited to capture the flag.
- *Miscellaneous*: The challenges typically span several non-technical topics and their resolution usually requires just basic logic and/or reasoning efforts, thus to make them beginner-friendly.
- *Hardware*: These challenges will be extensively discussed in the next Section.

III. Hardware CTF

A. A Look Around

As already mentioned, in the context of the CTF competitions organized around the world, the topic of hardware security today still plays a very marginal role, when not present at all. This is mainly due to the relative novelty of the topic, which has begun to spread only in recent years. Hence follows a low amount of specific skills, compared with the much greater amount of experts in the fields of cryptography, reversing, software security, among the organizers of the competitions as well. Therefore, a state-of-the-art of the topic limits to an overview of the few events that worldwide include hardware-related challenges.

The *Hardwear.io* platform [23], which includes hardware security researchers from all over the world and organizes courses, conferences and webinars on the topic, hosts a hardware-oriented CTF competition since 2017. The proposed challenges typically cover various themes, such as RFID, Bluetooth, automotive components, side-channel analysis, (de)soldering and radio. Proper sets of physical tools needed for the challenges and a guidance on how to use them are usually provided to the participants.

Riscure [25], an important security evaluation laboratory specialized in embedded systems and IoT security, organized RHme (Riscure Hack me) from 2015 to 2018: a CTF event mainly oriented to safety in the automotive environment and based on the use of Arduino™ products [10] for the implementation of the challenges [3] [4] [8]. The LiveOverflow channel maintains a collection of videos regarding the challenges of the event and their solutions [5]. Although very innovative and well implemented, many of these challenges are based on attacks on cryptographic protocols (e.g., a length-extension attacks to a SHA implementation) or communication protocols (e.g., UART), which do not require any specific knowledge of the hardware domain. In these challenges, the physical boards just play the role of a mere support for the execution of the challenge, in a manner no different from that of PCs, servers, virtual machines, switches and all the other devices used in challenges of any type. As we shall point out in the sequel of paper, we do not consider the above challenges as "true" hardware-based CTF challenges.

The *Hack@DAC* [21] hardware security contest has been held within the Design Automation Conference (DAC) [19]

since 2017. It is a competition focused on the topic of microarchitectural and side-channel flaws in chips [33] [41] [36]. Participating teams (students and industrial teams as well) are given a design of a vulnerable chip to be studied before the competition. The aim is to identify the greatest number of security problems. The winners of this first phase then participate in the CTF competition held live at the conference: here, the teams are assigned a new design of a vulnerable SoC, and must take advantage of their previous experience to find as many vulnerabilities as possible in a given time slot. At the end, the winner is the team that has submitted, in the format of flags, the greatest number of problems in the design.

Some general CTF events tried to incorporate hardware-based challenges into their programs. *Chujowy CTF* [11] introduced challenges aimed at finding vulnerabilities within the Verilog code of an automotive processor based on RISC-V [13] [12]. The *Google Capture The Flag* event [20] introduced some hardware-oriented challenges as well. In the 2017 edition, a challenge which consisted in cracking a slot machine, required to physically connect to the pins of the Arduino™ board which controlled the machine in order to extract the flag [6]. Other challenges that required to reverse HDL code or schematic hardware components were included in the 2018, 2019, and 2020 editions [7] [9] [14].

B. Definition and Taxonomy

The purpose of this subsection is twofold: we first provide a definition of *hardware-based CTF challenge* and then propose a brand-new taxonomy of hardware-based challenges. At the authors' best knowledge, both the definition and the dimensions of the taxonomy are introduced here for the first time and they both stem from the experiences authors collected while preparing and delivering several hardware-based challenges in significant competitions, talent scouting programs, training activities, and BS and MS level courses that globally involved hundreds of attendees.

A *hardware-based CTF challenge* is a challenge in which the challenger must exploit her/his knowledge about digital hardware (including methodologies and technologies related to design, validation, verification, testing, maintenance, etc., at all the abstraction levels), in order to capture a flag consisting in identifying, remediating, or exploiting vulnerabilities [45] artificially introduced either in the design or in the actual implementation of the hardware structure of a digital system.

The above definition has some relevant practical implications, among which we would like to point out the following ones:

1) The fact that a challenge simply "rely" on a hardware device does not imply that the challenge is a hardware-based CTF challenge. At the ultimate end, each program runs on hardware, so "running on a hardware device" cannot be a sufficient condition;
2) In a true hardware-based CTF challenge, capturing the flag must require a significant knowledge about digital hardware and cannot be successfully solved exploiting just other lateral knowledge;
3) In a true hardware-based CTF challenge, the flag could not be captured by just exploiting some vulnerabilities artificially introduced either in the software applications that runs on the hardware device, or in the communication or security protocols that are adopted by that device;
4) A true hardware-based CTF challenge can be implemented and proposed in real competitions without resorting to any "physical" hardware device, since it can rely on some particular features or aspects of the design of the device, which can be provided to participant via description files or proper EDA environments and tools.

Starting from the above definition, let's now propose a new taxonomy for hardware-based CTF challenges. It relies on five orthogonal dimensions: (i) the challenge purpose, (ii) its difficulty, (iii) its execution mode, (iv) its topic, and (v) the hardware device description. Let's analyse each dimension in details:

1) **Challenge Purpose**: challenges have to be planned differently according to their ultimate purpose, distinguishing among:
   - *Training*: in this case the challenge should be organized in such a way to smoothly drive the students through the different learning steps, usually characterised by an increasing complexity;
   - *Competition in Jeopardy style*: (see Section II): these challenges are usually more complex and hard-to-solve versions of the challenges used for training, where additional tricks are intentionally inserted according to the difficulty level of the competition;
   - *Competition in Attack/Defense style*: (see Section II): these are definitely the most complex hardware-based challenges, since they pose a lot of severe and hard constraints, including, among the others, the fact that any team must get a copy of the instance and that, during the second phase of the competition, each instance can be concurrently accessed by all the teams and by the game server. This practically means that, when a physical hardware device is involved, each instance must be equipped by a custom (software) wrapper in order to properly queue, manage, and serve all the incoming concurrent requests.
2) **Challenge Difficulty**: each challenge should be characterized by a proper ranking of its difficulty. To our best knowledge, unfortunately no consolidated official ranking schemes exist today. Consequently, it is usually up to the challenge's authors to provide a reasonable ranking, based on their experiences in training and gaming. We usually adopt a 5-value ranking, 1 being the easiest and 5 the hardest.
3) **Challenge Execution Mode**: it mainly deals with the tools provided to the attendees to solve the challenge. Three possibilities are usually exploited:
   - *By-hand*: Participants are given a design representation of a digital hardware (usually HDL code

or schematics), and the challenge can be solved manually just carefully analyzing the provided design description, without the need to resort to any specific tool. This kind of challenges are definitely the easiest and cheapest to implement and they just require expertise and knowledge in hardware design and test to be solved.
- *EDA-tool-based*: To solve the challenge, the participants have to resort to the facilities offered by a specific EDA platform (typically simulators and/or automated synthesis tools), made (fully or partially) available during the competition. Participants can exploit the provided tool to access a "model" of the hardware device, in which the vulnerabilities have been inserted. Note that, in this case, an instance of the selected platform must be made available to each participant and, in some cases, custom "wrappers" have to be designed in order to prevent participants from using the whole set of capabilities offered by the platform, since they could exploit some of these facilities to find a fastest and trivial ways to capture the flag.
- *Hardware-device-based*: In this case each participant must face a real hardware device (typically a small system, a PCB, or a development kit) that somewhere and somehow stores the flag to be captured. In some cases, an FPGA-based implementation/emulation of the target hardware device can be profitably exploited. Note that this case poses some severe issues in term of scalability, since during the competition each participant (or team of participants) must be given a different instance of the hardware device, regardless the competition type. Practically, it can be effectively adopted in the training phases and in teaching courses, where the hardware resources can be effectively shared in time among the attendees.

It is worth mentioning that, from a conceptual point of view, a fourth alternative, completely based on a *pure software emulation* of the hardware device could theoretically be adopted. In our experience, such a solution is mostly ineffective, due to practical difficulties in completely emulating via software, at the same time: (i) the expected hardware behavior, (ii) the set of vulnerabilities to be inserted, and (iii) their possible remediations.

4) **Challenge Topic**: several topics can be covered, including, among the others, the following ones:
   - *Hardware Trojans*: Participants are provided with a digital hardware into which a hardware trojan [46] has been artificially inserted. The identification and/or the exploitation of the trojan lead the participants to capture the flag.
   - *Unprotected test infrastructures*: In these challenges, the flag can be obtained by a clever exploitation of a test infrastructure available in the hardware device. These can range from the IEEE standard 1500 - Standard for Embedded Core Test [1] to the 1149.1-2013 - IEEE Standard for Test Access Port and Boundary-Scan Architecture [2] and to simple scan chains [28], all left accessible.
   - *Undocumented functions and features*: In additions to the hardware descriptions or implementations, participants are provided with a related documentation in which some peculiar features are deliberately omitted. These may include, for instance, machine instructions, components or undocumented side effects of the joint use of multiple documented components [31]. The flag can be captured only by exploiting (one of) these hidden features.
   - *Design bugs and flaws*: The hardware that participants have to deal with includes some design bugs or flaws [45] that introduce a vulnerability, through which the flag can be reached. Examples include, among the others, incorrectly-implemented machine instructions, internal race condition for which sensitive information can be released, etc.
   - *Side-channel Attacks*: Participants are given a hardware device vulnerable by side-channel attacks [40], such as timing or power attacks [38] [37]. Participants must be equipped with a set of tools that allow them to perform the attack and capture the flag.
   - *Weak implementations of hardware-based security modules*: The hardware delivered to the participants belongs to one of the families of modules used for security (e.g., hardware ciphers, random number generators, authenticators, etc.), but that has been designed and implemented introducing some weakness or vulnerabilities that can be exploited to get to the flag.

5) **Hardware Device Description**: when a description of the hardware design has to be provided, two orthogonal additional dimensions have to be considered, the *Abstraction Level* and the *Representation Domain*:
   - **Abstraction Level**: it identifies the level of details provided in the system description: it typically ranges from *System* to *Register-Transfer (RT)* to *Logic* level. Very seldom lowest abstraction levels are used.
   - **Representation Domain**: the provided descriptions can belong to the *Behavioral* or *Structural* domain. In the former case, the *behavior* of the target hardware system is provided in terms of properties (both functional and non-functional ones), which define what the system does and the circumstance under which it operates; in the latter case the *structure*, (i.e., the topology) of the target system is provided in terms of a set of functional building blocks, properly interconnected.

In conclusion, it is worth pointing out that hardware-based challenges involving *invasive attacks* [45] are usually not implemented, since they require the availability of advanced (and often expensive) tools and equipment that, in turns, require a very high degree of expertise to be properly and safely used and exploited.

IV. Our Experience

In this Section, we briefly introduce some hardware-based challenges that we prepared and delivered in different environments, including, among the others: (i) the training of TeamItaly (the Italian Team of cyber-defender that got the silver medal at the last European Championship in Bucharest, on November 2019), (ii) significant competitions, (iii) CTF training courses that involved hundreds of attendees with different backgrounds, and (iv) University courses in Cybersecurity. In particular, we shall focus on 4 different challenges. Readers interested in additional technical details are kindly invited to directly contact the paper's authors.

A. TIGER21X

- *Challenge Purpose*: Training
- *Challenge Difficulty*: 5/5 (hard)
- *Challenge Execution Mode*: By-hand
- *Challenge Topic*: Undocumented functions and features
- *Hardware Device Description*: RT-level structural.

Challenge Description: participants are faced with a simple custom processor, implemented as a reduced variant of the RISC DLX ISA [44]. In particular, they are provided with (i) RT-level structural description of the device, (ii) VHDL behavioral description of its control unit and (iii) device technical documentation. The processor implements an undocumented machine-level instruction, and namely an *indirect jump*, i.e., a jump instruction whose destination address is stored into one of the user-accessible general-purpose register. The flag to be captured is the label of the undocumented machine instruction.

In order to successfully solve the challenge, participants have first to find the proper mapping between documented machine instructions and their actual implementation, by reversing the control bits that the execution of each instruction activates in the data path. Of course, opcodes present in the control unit VHDL code have been labeled with non-speaking names not to make trivial the mapping. The additional undocumented instruction was properly hidden among the others, and to identify it participants must understand the meaning of each control signal issued by the control unit and to note the strange behavior of the processor when the undocumented machine instruction is executed.

B. AUTH_98X276YC

- *Challenge Purpose*: Jeopardy Competition
- *Challenge Difficulty*: 4/5 (medium-hard)
- *Challenge Execution Mode*: By-hand
- *Challenge Topic*: Hardware Trojan
- *Hardware Device Description*: Logic-level structural.

Challenge Description: participants are faced with a hardware device implementing an access enabler which grants access when a user-provided key matches the one previously stored inside the device, by asserting a PASS_FAIL output signal. The device includes a hardware trojan which, when activated, serially outputs the content of the stored key. Participants are given a file containing the behavioral specifications of the circuit and its *netlist*, i.e., a structural description at the logic abstraction level. The flag to be captured is the sequence of values to be assigned to input signals to get the key stored inside the device. The solution can be reached first by noticing that the internal registers are implemented in such a way that they could behave as shift registers, and then identifying the trojan activations sequence; this requires a detailed analysis of the provided netlist.

C. BROK_11491

- *Challenge Purpose*: Jeopardy Competition
- *Challenge Difficulty*: 3/5 (medium)
- *Challenge Execution Mode*: EDA-tool-based
- *Challenge Topic*: Unprotected test infrastructures
- *Hardware Device Description*: RT-level structural.

Challenge Description: participants are asked to capture a flag consisting in the value stored into the Device Identification Register of a simple hardware device designed to be compliant with the IEEE 1149.1 standard. To get it, they are given: (i) the device data sheet, which includes all the details about the TAP implementation, (ii) the RT-level structural VHDL description of the device, (iii) the possibility of using a simulator, properly wrapped in order to allow the users just to force the device's primary inputs, to read its primary outputs, and to run simulation campaigns.

A variation of the challenge can be proposed, in which the simulator is replaced by an actual implementation of the hardware device, typically resorting to a FPGA. In this case, an additional environment that allows participants to interact with the device must be provided.

D. CrashCube

- *Challenge Purpose*: Jeopardy Competition
- *Challenge Difficulty*: 3/5 (medium)
- *Challenge Execution Mode*: EDA-tool-based
- *Challenge Topic*: Weak implementations of hardware-based security modules
- *Hardware Device Description*: The physical device is provided, along with System-level documentation.

Challenge Description: participants are asked to investigate how extracting sensible data and keys from the secure flash embedded in the USB cryptographic token emulated by the SEcube™ development kit [24]. The secrets can be extracted exploiting either a hardware (semi-permanent) vulnerability of the chip, based on the IEEE 1149.1 standard, or a firmware vulnerability based on a mismanagement of the internal flash segments. To extract the secrets, they are given: (i) device data sheet, including the JTAG semi-permanent and permanent

states' description, (ii) partial info on the serial communication protocol, (iii) JTAG programmer and flash programming tool.

A variation of the challenge can be proposed, in which the SEcube™ exposes its internal bus between the embedded application secure processor and the embedded smart card. In this case, participants may benefit from extra information retrieved through a probe on the exposed bus, as it happened in many real cases where the CPU and the smart card are separate devices mounted on a PCB (e.g., Ledger cryptocurrency hardware wallet [22]).

V. Conclusions

In this paper, leveraging the experiences we gathered form different training opportunities and official competitions, we defined the concept of *hardware-based CTF challenge*, i.e., a challenge based on hardware-related security issues. In addition, for the first time, we proposed a taxonomy and presented some challenges we adopted in different situations.

Although hardware security is getting increasing interest within the cybersecurity community, its role in international competition is still marginal. In Italy, the 2020 edition of the *CyberChallenge.IT*[3] program for the first time included a complete week devoted to hardware security and in the final national competition, in Jeopardy style, attended by 400+ participants, we proposed a set of three hardware-based CTF challenges.

A plenty of work still needs to be done, especially in the direction of defining some shared and agreed metadata for the challenges, on their classifications, and on their sharing. Additional issues concern identifying and experiencing some viable solutions for properly and effectively including hardware-based CTF challenges within different training environments, including, among the other, from the one hand, professional hybrid Cyber Ranges and, from the other, university BS and MS courses.

Authors are interested and available to share experiences on the various issues outlined in the present paper.

VI. Acknowledgments

The activities presented in the present paper are partially supported by: (i) the European Union's Horizon 2020 research and innovation programme, under grant agreement No. 830892, project SPARTA, (ii) the Italian CINI Cybersecurity National Lab. via the program *CyberChallenge.IT*, and (iii) Blu5 Labs in Malta.

References

[1] IEEE 1500 Standard for Embedded Core Test (SECT). http://grouper.ieee.org/groups/1500/index.html, 2005. [Online; accessed 03-August-2020].

[2] 1149.1-2013 - IEEE Standard for Test Access Port and Boundary-Scan Architecture. https://standards.ieee.org/standard/1149_1-2013.html, 2013. [Online; accessed 24-July-2020].

[3] GitHub - Riscure/RHme-2015: RHme+ 2015 challenge. https://github.com/Riscure/RHme-2015, 2016. [Online; accessed 23-July-2020].

[4] GitHub - Riscure/RHme-2016: RHme2 challenge (2016). https://github.com/Riscure/RHme-2016, 2017. [Online; accessed 23-July-2020].

[5] Riscure Embedded Hardware CTF - LiveOverflow. https://old.liveoverflow.com/rhme/index.html, 2017. [Online; accessed 23-July-2020].

[6] slot machine write-up (Google CTF 2017 Finals). https://blog.bushwhackers.ru/slot-machine-write-up-google-ctf-2017-finals/, 2017. [Online; accessed 24-July-2020].

[7] 2018-06-23-Google-CTF-Quals - 220 Misc / Wired CSV. https://github.com/EmpireCTF/empirectf/blob/master/writeups/2018-06-23-Google-CTF-Quals/README.md#220-misc--wired-csv, 2018. [Online; accessed 24-July-2020].

[8] GitHub - Riscure/RHme-2017: Riscure Hack Me embedded hardware CTF 2017-2018. https://github.com/Riscure/RHme-2017, 2018. [Online; accessed 23-July-2020].

[9] CTFTime.org / Google Capture The Flag 2019 (Quals) / flagrom / Writeup. https://ctftime.org/writeup/15870, 2019. [Online; accessed 24-July-2020].

[10] Arduino - Home. https://www.arduino.cc/, 2020. [Online; accessed 03-August-2020].

[11] Chujowy CTF. https://chujowyc.tf/, 2020. [Online; accessed 24-July-2020].

[12] ChujowyCTF - Ford CPU — Ethan Wu. https://ethanwu.dev/blog/2020/07/16/chujowy-ctf-ford-cpu/, 2020. [Online; accessed 24-July-2020].

[13] ChujowyCTF 2020 - Ford CPU I & II - Daniel Brodsky. https://www.danbrodsky.me/writeups/chujowyctf2020-fordcpu/, 2020. [Online; accessed 24-July-2020].

[14] CTFTime.org / Google Capture The Flag 2020 / basics / Writeup. https://ctftime.org/writeup/23035, 2020. [Online; accessed 26-August-2020].

[15] CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting'). https://cwe.mitre.org/data/definitions/79.html, 2020. [Online; accessed 21-July-2020].

[16] CWE-89: Neutralization of Special Elements used in an SQL Command ('SQL Injection'). https://cwe.mitre.org/data/definitions/89.html, 2020. [Online; accessed 21-July-2020].

[17] CWE-94: Improper Control of Generation of Code ('Code Injection'). https://cwe.mitre.org/data/definitions/94.html, 2020. [Online; accessed 21-July-2020].

[18] Cyber NYC. https://cyber-nyc.com/, 2020. [Online; accessed 03-August-2020].

[19] Design Automation Conference. https://www.dac.com/, 2020. [Online; accessed 24-July-2020].

[20] Google CTF - Build your future with Google. https://buildyourfuture.withgoogle.com/events/ctf/, 2020. [Online; accessed 24-July-2020].

[21] Hack@DAC2020. https://hackat.events/dac20/, 2020. [Online; accessed 24-July-2020].

[22] Hardware Wallet - State-of-the-art security of crypto assets — Ledger. https://www.ledger.com/, 2020. [Online; accessed 03-August-2020].

[23] hardwear.io — Hardware Security Conference & Training - Netherlands, Germany & USA. https://hardwear.io/, 2020. [Online; accessed 23-July-2020].

[24] Multiple reconfigurable silicon in a single package. https://www.secube.eu, 2020. [Online; accessed 03-August-2020].

[25] Oustanding security diagnostic and support - Riscure. https://www.riscure.com/, 2020. [Online; accessed 23-July-2020].

[26] R. Baldoni, R. De Nicola, and P. Prinetto. *Il Futuro della Cybersecurity in Italia: Ambiti Progettuali Strategici*, chapter 4, pages 80–86. Consorzio Interuniversitario Nazionale per l'Informatica - CINI, 2018. ISBN: 9788894137330.

[27] S. Bratus. What hackers learn that the rest of us don't: Notes on hacker curriculum. *IEEE Security Privacy*, 5(4):72–75, 2007.

[28] M. Bushnell and V. Agrawal. *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*, chapter 14. Springer Science & Business Media, 2013.

[29] R. S Cheung, J. P Cohen, H. Z Lo, and F. Elia. Challenge based learning in cybersecurity education. In *Proceedings of the International Conference on Security and Management (SAM)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2011.

[30] S. Deterding, D. Dixon, R. Khaled, and L. Nacke. From game design elements to gamefulness: defining" gamification". In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pages 9–15, 2011.

[31] C. Domas. Hardware backdoors in x86 cpus, 2018.

[32] C. Eagle. Computer security competitions: Expanding educational outcomes. *IEEE Security Privacy*, 11(4):69–71, 2013.

[3] https://www.cyberchallenge.it/

[33] Q. Ge, Y. Yarom, D. Cock, and G. Heiser. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *Journal of Cryptographic Engineering*, 8(1):1–27, 2018.

[34] J. Hamari, J. Koivisto, and H. Sarsa. Does gamification work? – a literature review of empirical studies on gamification. In *2014 47th Hawaii International Conference on System Sciences*, pages 3025–3034, 2014.

[35] K. Huotari and J. Hamari. Defining gamification: a service marketing perspective. In *Proceeding of the 16th international academic MindTrek conference*, pages 17–22, 2012.

[36] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom. Spectre attacks: Exploiting speculative execution. In *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[37] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Annual International Cryptology Conference*, pages 388–397. Springer, 1999.

[38] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

[39] K. Leune and S. J Petrilli Jr. Using capture-the-flag to enhance the effectiveness of cybersecurity education. In *Proceedings of the 18th Annual Conference on Information Technology Education*, pages 47–52, 2017.

[40] Y. Li, M. Chen, and J. Wang. Introduction to side-channel attacks and fault attacks. In *2016 Asia-Pacific International Symposium on Electromagnetic Compatibility (APEMC)*, volume 01, pages 573–575, May 2016.

[41] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg. Meltdown: Reading kernel memory from user space. In *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[42] J. Mirkovic and P. AH Peterson. Class capture-the-flag exercises. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*, 2014.

[43] C. I. Muntean. Raising engagement in e-learning through gamification. In *Proc. 6th international conference on virtual learning ICVL*, volume 1, pages 323–329, 2011.

[44] D. Patterson, J. Hennessy, and D. Goldberg. *Computer architecture: a quantitative approach*, volume 2. Morgan Kaufmann San Mateo, CA, 1990.

[45] P. Prinetto and G. Roascio. Hardware security, vulnerabilities, and attacks: A comprehensive taxonomy. In *ITASEC*, pages 177–189, 2020.

[46] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor. Hardware trojans: Lessons learned after one decade of research. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 22(1):6, 2016.