Remotizing and Virtualizing Chips and Circuits for Hardware-based Capture-the-Flag Challenges

Remotizing and Virtualizing Chips and Circuits for Hardware-based Capture-the-Flag Challenges

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

Samuele Yves Cerini
CINI Cybersecurity National Lab.
samuele.cerini@consorzio-cini.it

Paolo Prinetto
Politecnico di Torino, Italy
IMT Lucca, Italy
CINI Cybersecurity National Lab.
paolo.prinetto@polito.it

***Abstract*—In the very rapid digital revolution we are experiencing, the availability of cybersecurity experts becomes critical in every organization and at multiple levels. However, classical and theory-oriented training seems to lack effectiveness and power of attraction, while professional selection and training processes based on cybersecurity gamification are being successfully experimented, among which Capture-the-Flag (CTF) competitions certainly stand out.**

Nevertheless, careful analysis reveals that such initiatives have a major shortcoming in addressing security issues when training people to tackle hardware-related security issues. Several motivations can be identified, including the inadequate technical knowledge of the White Teams charged of the challenges preparations, and the evident logistic problems posed by the availability of real hardware devices when the numbers of trainees significantly scales up.

This paper presents a platform able to provide as a service hardware-based CTF challenges and exercises, involving circuits and chips that can be physically connected to a server or simulated, to deal with topics such as hardware bugs, flaws and backdoors, vulnerabilities in test infrastructures, and side-channel attacks. The platform is presented from a technical perspective, and data for deducting related efficiency, stability and scalability are offered.

***Index Terms*—cybersecurity, education, training, gamification, capture-the-flag, challenge, hardware, hardware security**

1. Introduction

The process of continuous digitization that the world is experiencing touches so many areas of our societies that it becomes impossible not to address the security aspects related to the data produced and exchanged by this huge multitude of devices. The topic of cybersecurity has become crucial in every discussion, and at every level of organizations. As a result, in both the public and private sector, a significant gap has arisen between the demand for the workforce as security experts and its supply. The Cybersecurity Workforce Study of 2021, provided by the (ISC)^2 Association, estimates the shortage of cybersecurity professionals to be 2.72 million globally [1]. On the other hand, according to a study conducted by the ISACA Institute, about 50% of human resources managers say they do not believe that people hired in cybersecurity are truly qualified for their role, and an additional 16% even believe they are unable to accurately assess it [2].

This alarming picture is very complex to justify in terms of causes on a worldwide level, but surely it must take into account the difficulty with which training courses, at university level and not, adapt to the demand in such a dynamic scenario. Especially, even when they try to decrease the deficit, they often do it in an inadequate manner, by adopting a prevalently theoretical and notional training. More than concepts, to cover the demand it would be necessary to have people skilled from a practical point of view: in front of the dynamic nature, the adaptability, and the ability to think “out of the box” of the attackers, it would be necessary to oppose *white hackers* with the same characteristics [3]. In this regard, the good potential of the *education through gamification* [4] has been recently demonstrated: students are pushed to acquire practical knowledge through riddles and challenges that trace real and daily problems in the field of cybersecurity. Among the most famous examples of cybersecurity gamification, there are undoubtedly *Capture-the-Flag* (CTF) challenges, for which prestigious competitions are organized at a global level. In these challenges, participants earn points by decrypting ciphers, reversing binary code, or configuring network for access protection. The mix of learned practical knowledge and acquired *adversarial thinking* ultimately ensures greater efficiency than more classic face-to-face education [5] [6].

In the global panorama, the CTFs have plenty of different participation and development modes, as well as focus on the most different topics, ranging from cryptography to software reversing and cracking, from web security to analysis and filtering of network packets. On the other hand, challenges that deal with security problems linked to the lowest level of computing systems, i.e., the hardware, tend to lack. The research carried out for this study shows how hardware-based challenges are mainly relegated to specific events on the subject, and aimed at people who are often already operating in the sector. As for the CTF platforms open to ethical hacker teams from all over the world, the label “hardware” is often used to classify challenges that, for example, are instead related to software that runs on embedded platforms, or do not require any knowledge of digital hardware or CPU architectures. Such a poor offer is also witnessed by exponents of leading companies in the hardware sector [7], who recognize gamification as an important contribution to the training of professionals prepared also from the point of view of hardware security.

Among the causes of this imbalance, it is wrong to include a supposed lower risk related to this layer, as very powerful hardware vulnerabilities and attacks, capable of severely endangering the entire architecture of an IT system, are known by the community [8]. More reasonable causes are probably the recent widespread awareness of the problem as a result of very famous hardware vulnerabilities disclosure (such as Meltdown [9] and Spectre [10] of Intel processors), and the shortage of personnel with the necessary knowledge to give practical sessions on the subject, and even less to set up challenges.

Moreover, objective logistical difficulties in organizing challenges based on physical devices can be added. If in the context of a competition localized in time and space the problem can be limited, this can considerably widen, for example, in the case of:

- offering these challenges in the context of courses and training phases, with hundreds or thousands of people located in the most disparate places, to which it is impossible to provide all the necessary material;
- variety of devices that need to be purchased, programmed individually, checked against any possible misuse or irreparable damage to be used again;
- remote participation: Covid-19 pandemic has imposed these needs as well in the last two years;

and so on.

Thanks to the work of the CyberChallenge.IT project[1] in Italy, for two years the diffusion of this type of challenges has been encouraged in the audience of the participants (aged from 16 to 24), and a development track has started up to the implementation of a platform to host training and competitions in the CTF format, based on emulated or physical hardware devices. These are reached remotely via containerized TCP services, designed to offer the challenges to a large number of contemporary participants, despite the limited resources both on the servers and in relation to the number of physical devices available.

This paper aims to present the three main environments of this platform, describing them from the technical point of view and the context in which they operate in Sections 3 and 4. Before this, Section 2 gives an overview of the concept of CTF challenge and hardware-based challenge that drives this work. Finally, Section 5 presents the data related to the technical measurements made on the environments, and Section 6 offers a glimpse of the state of the art of the CTF events, their involvement in this topic, and the methods of offering the related challenges. Section 7 concludes the paper.

2. Background

Capture-the-Flag (CTF) is the name given to a game that consists of breaching one or more IT assets (e.g., algorithms, applications, files, devices), which are artificially made vulnerable so that players can exploit these vulnerabilities with the hacking techniques and tools they know within a given timeframe. In the game, the correct discovery and exploitation of such flaws guarantees the player the acquisition of a *flag* [11], a unique string forged by the game organizers, which, if submitted to a challenge service, increases the score. The CTF challenges are of multiple types, of which the two most famous are:

- *Jeopardy*, in which participants stand individually or in groups in front of the vulnerable asset, which is the only opponent in the challenge, and must try to find the entry point for the flag;
- *Attack/Defense* (A/D), in which the individual or grouped participants are given control of a vulnerable asset (*vulnbox*), identical for each participant, which must be defended (via patch or filter) from the attacks of the other participants, and at the same time, vulnbox instances of all the other opponents must be attacked.

The challenges are then grouped according to the topic they deal with. Usually, they can be clustered in these categories:

- *Binary*: related to software applications that contain vulnerabilities exploitable at the binary code level. These can be further classified in two categories, not necessarily separated:
  - *Reversing*: based on the reconstruction of the application behavior through static analysis of the binary code;
  - *Pwn*: based on the actual exploitation of vulnerabilities on a running application instance;
- *Web*: related to common website vulnerabilities, such as command and code injection, malicious SQL queries, malicious use of cookies, etc.;
- *Crypto*: related to vulnerable encryption schemes;
- *Network*: related to typical network attacks such as bypassing firewalls and access policies, spoofing and poisoning attacks, etc.;
- *Miscellaneous*: not very specific to a topic and/or mainly aimed at stimulating the logic/reasoning skills of the participants;
- *Hardware*: for this category of challenge, the community does not seem to have a universal definition, also for the reasons listed in the previous Section. In some major world competitions, like *Insomni'Hack* or *Cyber Apocalypse*, challenges have been labeled as "hardware" even when simply involving the presence of a physical board, on which firmware reversing or pwning was necessary, or based on the reconstruction of a message exchanged through a communication protocol usually adopted between hardware modules, such as I^{2}C or SPI, as it is shown later. The definition accepted for this paper is the one expressed in [12], according to which a *hardware-based CTF challenge* is a challenge that must have vulnerabilities residing in the hardware structure of a digital system, and whose resolution requires specific knowledge of the methodologies and techniques of hardware design, validation and verification.

As well, from [12] derives the taxonomy accepted by this work for this last type of challenge, which may deal with different sub-topics, including:

- *Hardware trojans* [13];
- *Unprotected test infrastructures* [14];
- *Undocumented functions and features* [15];
- *Design bugs and flaws* [8];
- *Side-channel and fault injection attacks*: [16] [17] [18];
- *Weak implementations of hardware-based security modules*: such as buggy implementations (or anyway

[1] https://cyberchallenge.it/

exploitable) for hardware ciphers, random number generators, authenticators, etc.

The taxonomy also defines 3 different execution modes:

- *Inspection-based*: the challenge target a description of a digital hardware (usually schematics and/or Hardware Description Language (HDL) files, related to devices described at the gate or register-transfer level, but also reports from its synthesis or operation). The challenge is solvable through static analysis of these files, and/or possibly using local emulation tools on participants' PC if needed. Usually, the flag is the encoding of the answer to a question related to the target (e.g., the input pattern that triggers a certain condition, or a unique code that can only be recovered from the exact location of the vulnerability, etc.);
- *Simulation-based*: here, the target is a device simulated through a hardware logic simulator, offering a "live" interface with the participants, that in most of cases can only write input and read outputs, and possibly control internal signals depending on the challenge difficulty. Participants are given a description of the target (e.g., HDL, schematics or other), which they use for preliminary understanding and possibly local simulations before sending an exploit to the service hosting the running target. Here, the use of a remote service is essential to prevent access to a hard-coded flag within the design: the participant is obviously provided with a sanitized copy, i.e., without the flag;
- *Device-based*: here, participants face a real hardware device, containing a vulnerability that leads to the flag capture. In this mode, the interaction with the device is always initiated by an application that has the primary task of reading the inputs and transmitting them to the device, and allowing the exit of the outputs. A copy of the device can be physically delivered into the hands of the participants, or be connected to a remote service. In the first case, the interaction application must be installed on the participants' machine, while in the second case, it can run behind a TCP socket accessible to participants. In both cases, as for the previous mode, participants are provided with a sanitized representation of the target.

As mentioned in the previous Section, the Italian project CyberChallenge.IT has been playing a key role in the dissemination of this category of CTF. CyberChallenge.IT is an initiative of the Italian Cybersecurity National Laboratory[2], and represents the main Italian track for cybersecurity education of young people, aged from 16 to 24 and coming from Italian high schools and universities. The course includes a selection and an initial training phase, which combines more classical teaching with training on a very extensive set of CTF Jeopardy, uploaded on a unique platform and authored by the community of experts throughout Italy. At this stage, the training is organized by *venues*, spread across the country, and each venue selects a team of about 20 participants. At the end of the course, there are two final events: a local Jeopardy competition among the members of a venue, and a national A/D competition in which the local teams compete against each other to elect the winning venue. The 2022 edition has had 5344 participation requests from 34 different venues, including 2 military academies.

Starting from the 2020 edition, the first inspection-based challenges have been introduced. The flag was in some cases the activation sequence of a trojan internal to the device, or the operational code of an undocumented instruction. In the local finals of the 2021 edition, simulation-based challenges, remotely accessible via the competition portal, were submitted to participants. The Italian CTF competition m0leCon 2021, held in Turin in December, hosted a device-based challenge, implemented on the SE*cube*™ board[3] by the Maltese Blu5 Group, that also includes an FPGA on the chip. The FPGA was programmed with the challenge's vulnerable target. Teams who were present were given a copy of the device, while remote teams were provided with a virtual machine reachable through the competition's VPN and physically connected to the device.

The efforts made to include within these competitions some challenges based on digital hardware, emulated and physical, have led to the consolidation of a first implementation of the environments described in the next Sections of this paper, and their subsequent expansion to the current point. The work was made possible by the inclusion of these environments within PAIDEUSIS [19], the hybrid cyber range of the Cybersecurity National Laboratory.

3. CTF Infrastructure for Hardware Flaws and Backdoors

The platform presented in this Section is designed to offer as a service hardware-based Jeopardy challenges that deal with issues related to security flaws in digital circuits, or related to the presence of backdoors and trojans inside them. The platform therefore allows to simulate a circuit description in an HDL, or to execute it in a physical way on a reprogrammable device (FPGA), and to offer the participant a "live" interface with it. The service is a classic character-based TCP service, architected to read commands in a specific encoding, allowing basic operations on the target hardware, such as: (i) setting inputs, (ii) observing outputs (and possibly also internal states) and (iii) advancing time (in case of simulation only).

In the following Subsections, details about the environment based on the circuit simulation are explained first, and following, those about the environment based on their physical execution on FPGAs.

3.1. Simulation-based Environment

This environment relies on the use of ModelSim[4], the most popular framework in the hardware design community for simulating logic circuits described in many different HDLs. Inside, the entire toolchain for HDL compilation and simulation is present, as well as a set of commands already predefined for the operations described

[2] https://cybersecnatlab.it/

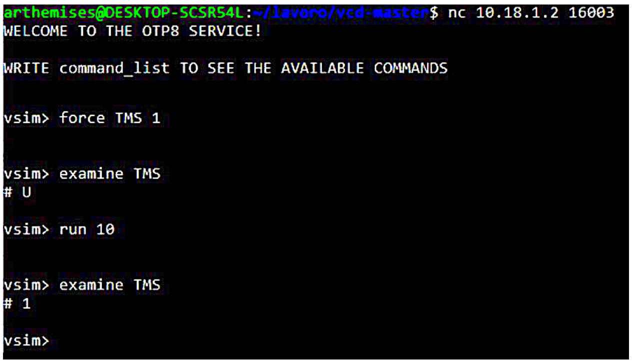
[3] https://www.secube.blu5group.com/

[4] https://eda.sw.siemens.com/en-US/ic/modelsim/

above. Therefore, for the setup of this environment, it has been necessary to enclose ModelSim behind a *wrapper* based on Python, allowing to filter commands and signal names as needed.

The wrapper is hosted though the Linux tool `socat`, able to accept connections from the internet. It launches a ModelSim instance for every user that connects to the service. Sharing a single instance between participants has been avoided, not to introduce a software scheduler and wait time for interacting with the target. The instance starts the circuit simulation from the initial time, i.e., from the beginning of the main stimulation process of the *testbench* entity. In fact, the target is not directly simulated as a top entity, but is enclosed in a portless top entity (precisely, a testbench), which initializes the inputs, resets the circuit, and automatically starts the clock where present. From this point on, the player interacts with the circuit through the following commands (Figure 1):

- `force <signal_name> <value>`, used to set the value of a signal;
- `examine <signal_name>`, displaying the value of a signal;
- `run <time>`, used to advance the simulation.



```
arthemises@DESKTOP-SCSR54L:~/lavoro/vcd-master$ nc 10.18.1.2 16003
WELCOME TO THE OTP8 SERVICE!

WRITE command_list TO SEE THE AVAILABLE COMMANDS


vsim> force TMS 1



vsim> examine TMS
# U

vsim> run 10



vsim> examine TMS
# 1

vsim>
```

Figure 1: User interface of the simulation-based environment.

The last command is particularly important because the player, for practical reasons, is not usually allowed to directly control the clock signal, which is automatically managed by the testbench. Therefore, the time advancement determines the advancement of a precise number of clock cycles, on the basis of a known operating frequency of the circuit.

In order to analyze and process the user commands, a filter is needed between the socket and the ModelSim instance. The filter analyzes the user inputs and discards the forbidden ones, resorting to some *configuration files* provided by the challenge designers. In particular, the power of the `force` command must be carefully limited, as it can be used to set the value of internal signals, trying to escape from some circuit logic, and find the flag in an easier way. The system acknowledges the presence of three configuration files:

- `whitelist.conf`: contains the list of allowed ModelSim commands. Any other command is rejected through notification to user;
- `blacklist.conf`: contains the set of words that must not appear in the input string sent by the challenge participant. For instance, the name of the HDL signal or constant storing the flag must be surely blacklisted. Usually, the same applies for all internal signals and variables, unless an internal state is needed for the challenge resolution. As well, it contains all forbidden punctuation symbols that may allow to gain access outside the interface or crash the environment;
- `blacklist-force.conf`: is a special blacklist for the `force` command only, containing the set of signals and variables that the participant must not be able to directly set.

It is important to note here that any challenge designer, as long as they provide a testbench in accordance with the features described above, and this triple of files, can include their challenge in the environment, whatever are the features of their target.

When a disconnection from the service is sensed, the software cleans temporary files and folders created by the simulation instance, and kills it in order to free memory and CPU occupation. Alternatively, as it is reasonable to assume, the service also features a timeout of a few seconds to prevent congestion. In fact, it is assumed that the resolution of the challenge is done by automating the sequence of commands.

The entire service is encapsulated into a Docker[5] container. Docker allows to build lightweight Linux images that bundle all the necessary dependencies to run a specific application, massively improving reproducibility. The result is a fast and stable deployment of applications and isolated environments with respect to the hosting machine.

3.2. FPGA-based Environment

This environment leverages the use of the SE*cube*™ chip by Blu5 Group (Figure 2). SE*cube*™ is a 3D System-in-Package designed for security applications, that contains an STM32F4 microcontroller with an ARM Cortex-M4 core, a Lattice MachX02 FPGA, and an Infineon smart card. For the purposes of this project, the FPGA is used to host the challenge target, and the MCU to handle communication between it and the outside world. Using an FPGA versus a simulation certainly brings more effectiveness to the challenge, which can fit into a more realistic storytelling, where there is really a vulnerable hardware device to interact with, even if remote.

In this environment, the role previously played by the wrapper is entrusted to a *host application*, which physically connects to the board via USB. In addition to providing the interface through `socat`, the host application is tasked to filter the commands and send them to the firmware running on the MCU of the board. The firmware plays the same role previously assigned to the testbench: since the MCU has a GPIO interface with the FPGA, it provides a continuous clock to the target, applies the inputs and reads the outputs, sending them back to the host application.

Also in this case, any target described in HDL can be hosted by the board, and challenge designers are only asked to provide their HDL files, while they do not need to deal with the communication part managed by the firmware. As for the host application, since the interface

[5] https://www.docker.com/

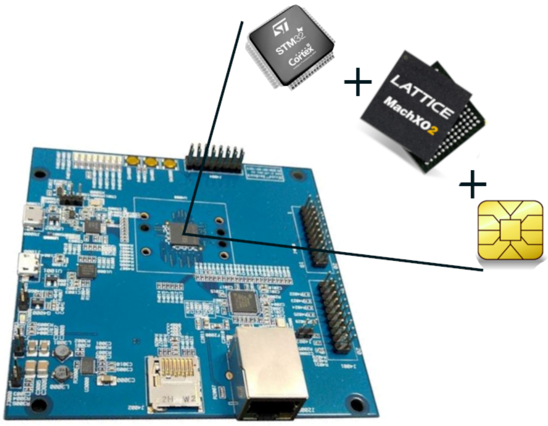


Figure 2: The SE*cube*™ Development Kit and its components.

with the target is not defined by a standard as it was for ModelSim commands, the challenge designer can program it using a preferred scheme. The particular setup of this environment can also allow to host challenges where the interaction with the hardware takes place at a higher level. For example, thanks to the USB connection, the target on the FPGA could play the role of a vulnerable coprocessor or peripheral, which could be passed a sequence of instructions and data instead of native inputs.

With respect to the simulation-based environment, several different aspects must be considered server-side, relating to the containerization of the service and the enqueuing of the requests. In fact, when a user connects, it is not just a matter of launching an execution instance, but the connection with a real hardware is required, and this hardware must be ready and not busy. To this purpose, the host machine enumerates all the connected devices, and for each of these, a Docker container is created with the host application described above. This application is executed on a socket that is initially not accessible from the outside. To coordinate the access to the challenge, during the setup of the environment, the HAProxy[6] service is executed: this remaps all sockets on a single TCP port accessible on the internet, enqueues all the requests, and correctly balances the load on the containers that host the copies of the challenge.

At this point, a punctual connection is established between a user and a board. This is maintaned until the user quits, or until a timeout expires, set with similar constraints to the simulation-based environment.

4. CTF Infrastructure for Side-Channel Attacks

This environment is aimed at training on hardware security issues intrinsically linked to the physicality of the device, and offers the possibility to conduct real power side-channel attacks [17] on real chips. The environment makes use of the ChipWhisperer platform[7], from the company NewAE Technology (Figure 3).

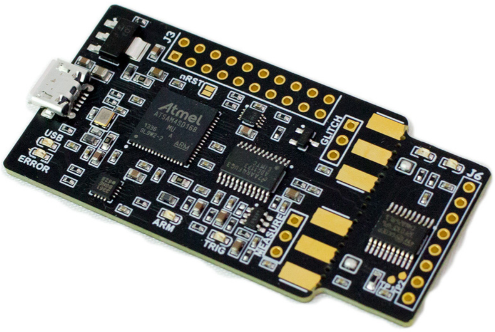


Figure 3: The ChipWhisperer.

As you can also see from the Figure, the platform consists of two parts: an STM32F0 microcontroller that acts as a “victim”, on which firmware executing cryptographic algorithms is loaded. The fundamental pins of this chip, such as the clock and the power supply, are connected to a second “attacker” system, with a more complex Atmel processor that manages the interaction with the user, the implementation of side-channel and fault injection attacks, and the collection of data from the victim for analysis. The platform is entirely based on Python libraries, which allow an easy and abstract use compared to the complexity of the attack operations.

The environment involves a dispatcher service on the hosting machine to process requests, and the development of a Docker container to be downloaded and launched by the users in order to set a local environment. In fact, processing and visualization of the capture data is done locally by the users, and the remote service is only responsible for serving the capture requests.

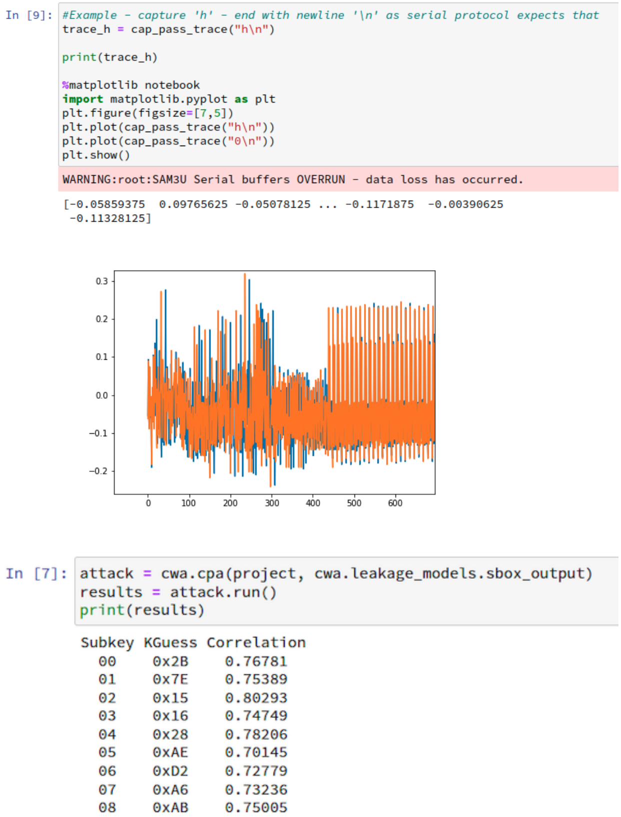
From the user point of view, each challenge consists of a Jupyter Notebook file[8]. In short, a Jupyter Notebook file can be seen as a Markdown file enriched with interactive graphs and executable code snippets. The file is rendered to the users as a webpage, and the code snippets execution is handled by an underlying Python kernel. The webpage has editable code boxes, allowing the users to test and reiterate multiple attack mechanisms, and obtain the related results in both graphical and numerical representations (Figure 4). Inside the container that users download, all the tools needed to run and display the Notebook are already present (i.e., Jupyter engine, ChipWhisperer library, Python’s `matplotlib`, etc.)

The remote service is responsible of handling all the incoming requests of capture operation sent by the users. Each request includes a set of configuration data such as the one reserved to the ChipWhisperer scope board and the one related to the capture configuration. The service has then to bind the incoming request to the correct ChipWhisperer, and uses a “direct mapping”-like approach: given m the number of boards available, all the requests are equally partitioned on m queues. Conversely to the FPGA-based environment, here all the boards only respond to a single

[6] http://www.haproxy.org/

[7] https://rtfm.newae.com/Capture/ChipWhisperer-Nano/

[8] https://docs.jupyter.org/en/latest/



```
In [9]: #Example - capture 'h' - end with newline '\n' as serial protocol expects that
        trace_h = cap_pass_trace("h\n")

        print(trace_h)

        %matplotlib notebook
        import matplotlib.pyplot as plt
        plt.figure(figsize=[7,5])
        plt.plot(cap_pass_trace("h\n"))
        plt.plot(cap_pass_trace("0\n"))
        plt.show()

        WARNING:root:SAM3U Serial buffers OVERRUN - data loss has occurred.

        [-0.05859375  0.09765625 -0.05078125 ... -0.1171875  -0.00390625
         -0.11328125]
```

```
In [7]: attack = cwa.cpa(project, cwa.leakage_models.sbox_output)
        results = attack.run()
        print(results)

        Subkey KGuess Correlation
          00    0x2B    0.76781
          01    0x7E    0.75389
          02    0x15    0.80293
          03    0x16    0.74749
          04    0x28    0.78206
          05    0xAE    0.70145
          06    0xD2    0.72779
          07    0xA6    0.73236
          08    0xAB    0.75005
```

Figure 4: Screenshots from Jupyter Notebooks performing offline side-channel analysis.

capture request, after which the connection is immediately closed. The rest of the computation necessary for the advancement of the exercise is accomplished within the users' local container.

The various backend components are all encapsulated within a Docker container as well. Each of these is able to write useful logs, and also to report the most critical errors as Telegram notifications to the system maintainer, allowing for a timely problem resolution when necessary.

5. Evaluation

Still in the absence of comprehensive data evaluating the reception and effectiveness of these environments within the audience, for the purposes of this article and in the intention of the authors, it is meaningful to offer data that especially assess:

- the impact of the services on the host machines in terms of computational cost;
- the user's perception of the service, measured by waiting times to receive an answer (net of the user's geographical location and possible connection issues or delays).

The environments are currently hosted on two of the servers in the PAIDEUSIS hybrid cyber range [19], maintained by the Cybersecurity National Laboratory. One server is allocated for the two environments described in Section 3, and another for the one described in Section 4. The machines have 40 Intel Xeon E5-2650 CPUs and 128 GB of RAM, and run Ubuntu Server 21.10. It is to be noted that each service application is enclosed within a container - which means every instance for the simulation-based environment, and both the dispatcher and all the device-specific container for FPGA-based and ChipWhisperer-based environment. All containers are based on Ubuntu Focal 20.04. The bandwidth assigned to services is 1 Gbps symmetrical. The current sizing is based on the expected use of events like CyberChallenge.IT, where this type of challenges represents about 10% of the total offer in both phases (the training one of 4 months, and the competition one of 8 hours at the end) and where the number of participants is about 700.

The simulation-based environment is totally independent from additional hardware, and therefore it is more similar to a classic TCP service. In the tests conducted, each instance never exceeded 50% occupancy of a physical core, and 70 MB of RAM. The time between receiving a command and executing it is less than 10 ms as long as resources are available.

The FPGA-based environment is much lighter from the point of view of server computation, since the task of the host application is minimal and only related to the correct passing of commands to the board. Both from the HAProxy and from the single container points of view, for each board, the impact on RAM is negligible and also the impact on CPU. The bottleneck is obviously related to the presence of the physical boards, which mainly results in a dispatching time of the request towards a free board. Given t the connection timeout (which starts when the service starts listening to a user), n the number of connected users, and m the number of present boards, the maximum response time T is obtained as

$$T = \begin{cases} 0\,\text{s} & \text{if } n <= m \\ \lceil \frac{n-m}{m} \rceil * t\ \text{s} & \text{if} \, n > m \end{cases} \qquad (1)$$

For example, if the timeout is 3 seconds, the physical boards are 40, and there are simultaneously 100 users requesting access to the service (which is very unlikely in competitions with characteristics such as those described), the maximum experienced wait time is 6 seconds. The processing time of the request by the board is negligible compared to this time.

In the last environment presented, the one for the side-channel analysis, the same considerations cannot be applied: although the computationally-intensive analysis is done in local, there is still the need to account a certain computation effort to capture traces and pack data. Figure 5 shows the latency in the response to the user as a function of the number of traces whose capture is requested. There is a roughly linear progression, going from 8.14 seconds for capturing 50 tracks to 39.64 seconds for capturing 800 tracks. Also for this environment, there is a certain additional latency due to the queue of users, and the times can scale up to a nx factor with respect to those presented in the worst case, if n is the number of users queued on a single ChipWhisperer. The length of this queue is obviously decreased in a natural way as the number of present devices increases.

As far as resource occupation is concerned, the tests showed that a single container for a physical device never exceeds 60% occupation of a physical core, and 120 MB of memory. Since all services are containerized, all the

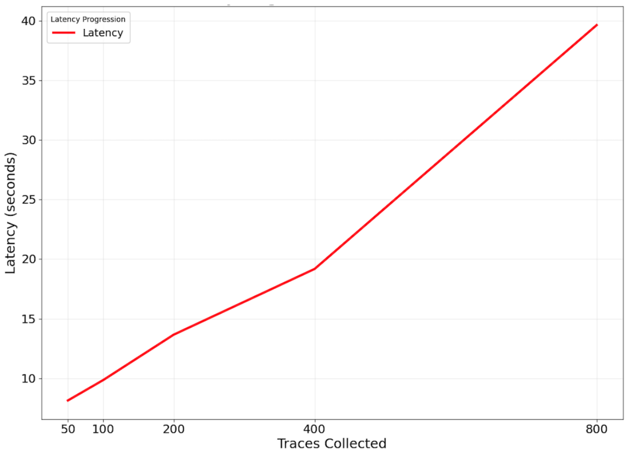


Figure 5: Latency of a request vs. number of requested traces in the ChipWhisperer-based environment.

results presented here have been obtained using `docker stats`.

6. Related Work

Given the premises stated in the introductory part of this paper, the placement of this work within the current landscape is far from conventional. For example, a recent work by Kucek *et al.* [20] surveys very well 19 CTF platforms, of which 8 are open-source (*CTFd*, *pbctf*, *HackTheArch*, *Mellivora*, *Pedagogic-CTF*, *PicoCTF*, *RootTheBox*, *WrathCTF*), analyzing them from the point of view of the tools used and required, the proposed challenges, and the scoring system. The work performs an evaluation for the open platforms by creating a set of test challenges, based on mainstream topics in CTFs, such as cryptography or web-based code injection attacks. No mention is made for challenges that are in any way hardware-related, and none of these platforms seem to be able to support them. Very similar work has been done in [21].

As already mentioned, when looking for references to challenges with type "hardware" within CTF platforms and events, two types of instances are mainly encountered:

1) the specific ones, which propose challenges that can be reasonably defined as advanced, and target an audience that is already mostly expert in the field;
2) the generic ones, that since some time have begun to introduce challenges that deal with or involve hardware, but often the knowledge about security issues related to the electronic domain is not really stimulated.

HackAT events[9], organized since 2018 as part of major hardware design and test conferences such as DAC (Design Automation Conference)[10], USENIX Security Symposium[11], and CHES (Conference on Cryptographic Hardware and Embedded Systems)[12], involve a specific competition held in two phases: in the first, prior to the conference, participants are given a description of a vulnerable SoC of which they must be able to discover as many vulnerabilities as possible, using any tool. In the second phase, teams are provided with a second vulnerable SoC, and the finalists compete in a live, time-limited CTF, which always consists of accumulating points by submitting identified vulnerabilities.

The Hardwear.io platform[13] also runs a CTF competition in the format focusing on radio protocols, automotive, side-channel analysis, and even physical invasive attacks on chips. Its special Capture-the-Signal (CTS) [14] internal track is exclusively based on reverse engineering of radio signals: participants are put in front of Jeopardy challenges of increasing difficulty, ranging from simple reading to signal demodulation and decryption.

Another example is the CTF RHme event[15], organized from 2015 to 2018 by the company Riscure, which operates in the field of automotive hardware security. In this event, there is a multi-level challenge based on the Arduino platform[16]: participants are given a firmware to flash (or already flashed) on the board, and the score is increased by carrying out exploits involving binary reversing and pwning, but also related to cryptography, side-channel analysis and fault injection attacks.

As for the most famous CTF competitions worldwide, we conducted a survey by collecting a sample of 40 challenges tagged as "hardware", and offered in the editions of the last 3 years (2019, 2020, 2021) of 18 major competitions:

- *Aero CTF*
- *ALLES! CTF*
- *Chujowy CTF*
- *CONFidence CTF*
- *CTF Internacional MetaRed*
- *Cyber Apocalypse*
- *Google Capture The Flag*
- *HTB Uni CTF*
- *ICHSA CTF*
- *Insomni'hack*
- *KnightCTF*
- *Ledger Donjon CTF*
- *pbctf*
- *ph0wn*
- *Pwn2Win CTF*
- *UIUCTF*
- *WPICTF*
- *X-MAS CTF*

The research has been conducted through the use of the *CTFTime.org* portal[17], which is the main CTF collection portal, continuously uploaded by teams from all over the world. For reasons of space, we do not report here all the names of the individual challenges analyzed, which in any case can be found by simply connecting to the portal, searching challenges for the "hardware" tag only, and discarding the results prior to 2019.

5 challenges out of 40 involve the use of physical hardware (local in 2 cases and remote in 3 cases), of the ARM or Atmel family. All of these challenges are actually

[9] https://hackatevent.org/
[10] https://www.dac.com/
[11] https://www.usenix.org/conference/usenixsecurity22
[12] https://ches.iacr.org/
[13] https://hardwear.io/
[14] https://hardwear.io/usa-2021/ctf.php
[15] https://www.riscure.com/challenge
[16] https://www.arduino.cc/
[17] https://ctftime.org/

focused on binary reversing and pwning activities over the software running on the board, and none seem to fall within the definition given in [12]: no knowledge of the CPU architecture is required, or no schematic or descriptive code of the hardware is given to the participants.

Another 9 out of 40 are based on remote hardware emulation. A known processor architecture (RISCV, Intel or Atmel) is emulated in 8 cases out of 9, while only in one case the service is provided by a program written in C++ which interacts with a module written in SystemVerilog and "executed" through Verilator[18]. In the other 8 mentioned, no detail is given on how the hardware is simulated, but it is easy to assume that this is done via a fully-software instruction set simulator (such as Unicorn[19]), not involving any simulation of HDL code. In fact, there would be no need, as the vulnerabilities lie in modules external to the processor and connected to it.

Only in 5 of these 9 challenges, some HDL code is provided to the participants, and the flag is reached by executing a code that triggers the vulnerability in the external module. We can therefore categorize these challenges as a mix between binary and hardware. In the remaining 4 challenges of this subset, the only activities are reversing and pwning on the binary code. The challenge with the module described in SystemVerilog is therefore the only one that purely demands knowledge of digital hardware.

In total, the challenges where the HDL code is involved are 7 out of 40, and another 8 out of 40 require the presence of schematics, for a total of 15 challenges out of the 40 analyzed. Beyond the 5 described above that involve remote emulation, the other 10 can be all categorized as inspection-based (see Section 2). 40% of the remaining 25 challenges of the set are based on binary reversing and pwning activities: even if they involve the presence of local or remote hardware, or a remote emulation service, the knowledge about their hardware details is however irrelevant for the resolution. The other 60% is composed by challenges requiring the extrapolation of information from a capture of waveforms, relating to communications on physical protocols (such as CAN, I2C, SPI), for which no specific knowledge of hardware is actually required, and for which the probable best categorization is as miscellaneous challenges (see Section 2).

7. Conclusions

This paper presented a platform capable of offering CTF challenges focused on hardware security aspects, through remote interaction with physical or simulated circuits using standard tools from the hardware designers community. The challenges that can be proposed through the platform stimulate knowledge about flaws or backdoors inserted in the design of the chips, vulnerabilities in their test infrastructures, and attacks based on power side-channel analysis. Required and used tools were listed, and the technical characteristics and performance of the platform were described in a quantitative manner, highlighting data to assess efficiency, security, usability and portability.

The platform is already currently in use on the training portal of the CyberChallenge.IT program, but work is underway to refine and expand the platform, in relation to all the environments presented, for the final events of this and other initiatives scheduled for 2022. In particular, the simulation-based environment is moving from ModelSim to GHDL[20], an open framework much lighter than the previous one in terms of resource consumption, and also allowing more features like waveform plotting. As for the FPGA-based environment, it is migrating to the Zilinx Pynq platform[21], which allows for larger circuit sizes, faster reconfiguration times, and easier and more customizable user interaction based on Jupyter Notebooks, already employed for the ChipWhisperer environment.

Moreover, the platform currently supports Jeopardy-type challenges only. A new system to host Attack/Defense challenges has already been architected, with the main purpose to be stable and to allow an agile reconfigurability of hardware-based *vulnbox* services without this significantly impacting game time.

Certainly, subject of next studies will be the collection of data on the reception of these challenges by the diversified audience that will use the platform in the coming months, on the criticalities encountered, and on a definitive systematization of the topic of hardware-based challenges, taking into account the experience in the field.

8. Acknowledgments

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

References

[1] I. C. W. Study, "A resilient cybersecurity profession charts the path forward," https://www.isc2.org//-/media/ISC2/Research/2021/ISC2-Cybersecurity-Workforce-Study-2021.ashx, 2021, [Online; Accessed 2022, March 2nd].

[2] ISACA, "State of cybersecurity 2021 - part 1: Global update on workforce efforts, resources and budgets," https://securitydelta.nl/media/com_hsd/report/424/document/state-of-cybersecurity-2021-part-1.pdf, 2021, [Online; Accessed 2022, March 2nd].

[3] S. Bratus, "What hackers learn that the rest of us don't: Notes on hacker curriculum," *IEEE Security & Privacy*, vol. 5, no. 4, pp. 72–75, 2007.

[4] A. Anil Yasin and A. Abbas, "Role of gamification in engineering education: A systematic literature review," in *2021 IEEE Global Engineering Education Conference (EDUCON)*, 2021, pp. 210–213.

[5] 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*, 2017, pp. 47–52.

[6] J. Vykopal, V. Švábenskỳ, and E.-C. Chang, "Benefits and pitfalls of using capture the flag games in university courses," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020, pp. 752–758.

[18] https://www.veripool.org/verilator/

[19] https://www.unicorn-engine.org/

[20] http://ghdl.free.fr/

[21] http://www.pynq.io/

[7] J. M. Fung, “Capture-the-flag competitions need to include hardware,” https://www.eetimes.com/capture-the-flag-competitions-need-to-include-hardware/, 2020, [Online; Accessed 2022, March 9th].

[8] P. Prinetto and G. Roascio, “Hardware security, vulnerabilities, and attacks: A comprehensive taxonomy,” in *ITASEC*, 2020, pp. 177–189.

[9] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin *et al.*, “Meltdown: Reading kernel memory from user space,” in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 973–990.

[10] 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 *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1–19.

[11] C. Eagle, “Computer security competitions: Expanding educational outcomes,” *IEEE Security & Privacy*, vol. 11, no. 4, pp. 69–71, 2013.

[12] P. Prinetto, G. Roascio, and A. Varriale, “Hardware-based capture-the-flag challenges,” in *2020 IEEE East-West Design & Test Symposium (EWDTS)*, 2020, pp. 1–8.

[13] 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)*, vol. 22, no. 1, p. 6, 2016.

[14] A. Cui, Y. Luo, and C.-H. Chang, “Static and dynamic obfuscations of scan data against scan-based side-channel attacks,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 2, pp. 363–376, 2017.

[15] C. Domas, “Hardware backdoors in x86 cpus,” 2018.

[16] Y. Lyu and P. Mishra, “A survey of side-channel attacks on caches and countermeasures,” *Journal of Hardware and Systems Security*, vol. 2, no. 1, pp. 33–50, 2018.

[17] M. Randolph and W. Diehl, “Power side-channel attack analysis: A review of 20 years of study for the layman,” *Cryptography*, vol. 4, no. 2, p. 15, 2020.

[18] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, “Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures,” *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[19] G. Berra, G. Ferraro, M. Fornero, N. Maunero, P. Prinetto, and G. Roascio, “Paideusis: A remote hybrid cyber range for hardware, network, and iot security training,” in *ITASEC*, 2021, pp. 284–297.

[20] S. Kucek and M. Leitner, “An empirical survey of functions and configurations of open-source capture the flag (ctf) environments,” *Journal of Network and Computer Applications*, vol. 151, p. 102470, 2020.

[21] M. Swann, J. Rose, G. Bendiab, S. Shiaeles, and F. Li, “Open source and commercial capture the flag cyber security learning platforms - a case study,” in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2021, pp. 198–205.