CERBERE: Cybersecurity Exercise for Red and Blue team Entertainment, REproducibility

Pierre-Victor Besson, Romain Brisse, Hélène Orsini, Natan Talon, Jean-François Lalande, Frédéric Majorczyk, Alexandre Sanchez, Valérie Viet Triem Tong

CERBERE: Cybersecurity Exercise for Red and Blue team Entertainment, REproducibility

Pierre-Victor Besson
CentraleSupélec, Inria, Univ. Rennes, CNRS, IRISA
Rennes, France
pierre-victor.besson@inria.fr

Romain Brisse
Malizen, CentraleSupélec, Inria, Univ. Rennes, CNRS, IRISA
Rennes, France
romain@malizen.com

Hélène Orsini
CentraleSupélec, Inria, Univ. Rennes, CNRS, IRISA
Rennes, France
helene.orsini@irisa.fr

Natan Talon
Hackuity, CentraleSupélec, Inria, Univ. Rennes, CNRS, IRISA
Rennes, France
ntalon@hackuity.io

Jean-François Lalande
CentraleSupélec, Inria, Univ. Rennes, CNRS, IRISA
Rennes, France
jean-francois.lalande@irisa.fr

Frédéric Majorczyk
DGA, CentraleSupélec, Inria, Univ. Rennes, CNRS, IRISA
Rennes, France
frederic.majorczyk@intradef.gouv.fr

Alexandre Sanchez
CentraleSupélec, Inria, Univ. Rennes, CNRS, IRISA
Rennes, France
alexandre.sanchez@inria.fr

Valérie Viet Triem Tong
CentraleSupélec, Inria, Univ. Rennes, CNRS, IRISA
Rennes, France
valerie.viettriemtong@centralesupelec.fr

***Abstract*—Experimenting in cybersecurity requires manipulating reliable and realistic data. In particular, labelled data derived from the observation of a complete campaign is rarely available, due to its high sensitivity and the difficulty of accurately labelling datasets. This situation harms the reproducibility of research results and therefore to their impact. In this article, we present the CERBERE project that addresses this issue through a reproducible attack-defense exercise and a labelled dataset usable for research purposes. The attack-defense exercise is first composed of an exercise for red teamers automatically deployed with variable attack scenarios. Second, an exercise for blue teamers can be operated using the system and network logs generated during the attack phase. We provide with this article, the software to rebuild the infrastructure for red teamers. We share a labelled dataset where we spot the ground truth, i.e. the log lines that have been involved in the attacker's actions.**

***Index Terms*—cybersecurity, red / blue teams, cyber training**

I. Introduction

Cybersecurity includes a vast array of specialized fields, ranging from intrusion detection and penetration testing to remediation, mitigation, and digital forensics. These diverse facets of cybersecurity operate in synergy. At the core of this discipline lies the concept of information systems security. Information systems security, encompasses networks, databases, and applications. It aims to identify attack surface and create a resilient defence against diverse threats, making it an essential aspect of cybersecurity strategy. Despite its importance, the pursuit of realistic information systems security solutions remains a challenge. The scarcity of authentic architectures is still an obstacle for researchers and practitioners: mimicking real infrastructures can notably disclose sensitive data, increasing the scale of cyberranges is costly, simulating legitimate user behaviors is difficult in randomly generated infrastructures [1].

One of the main academic issues is the lack of data [2]. Cybersecurity suffers from data scarsity for good reasons such as the sensitivity of security data, the high cost and low reliability of data anonymization, the difficulty of capturing, storing and releasing data to the public [3]. As a result, many different initiatives have been taken to produce the data needed to evaluate their work. Other researchers even tried to produce data for everyone to use such as the datasets produced by the Canadian Institute for Cybersecurity [4], SOCBED [5], or SecGen [6]. However, despite the merits of all these previous approaches, they present important flaws [7] due to the complexity of the task. First, the presented architectures are rigid and difficult to change. Then, the management of the attack surface is complex, and must be handled with care because of potential side effects. Indeed, implementing a vulnerability often implies installing older package versions or modifying rights, creating a significant probability to open more than one way for attackers to get inside the system. Finally and most of all, all previously presented architectures

979-8-3503-2445-7/23/$31.00 ©2023 IEEE

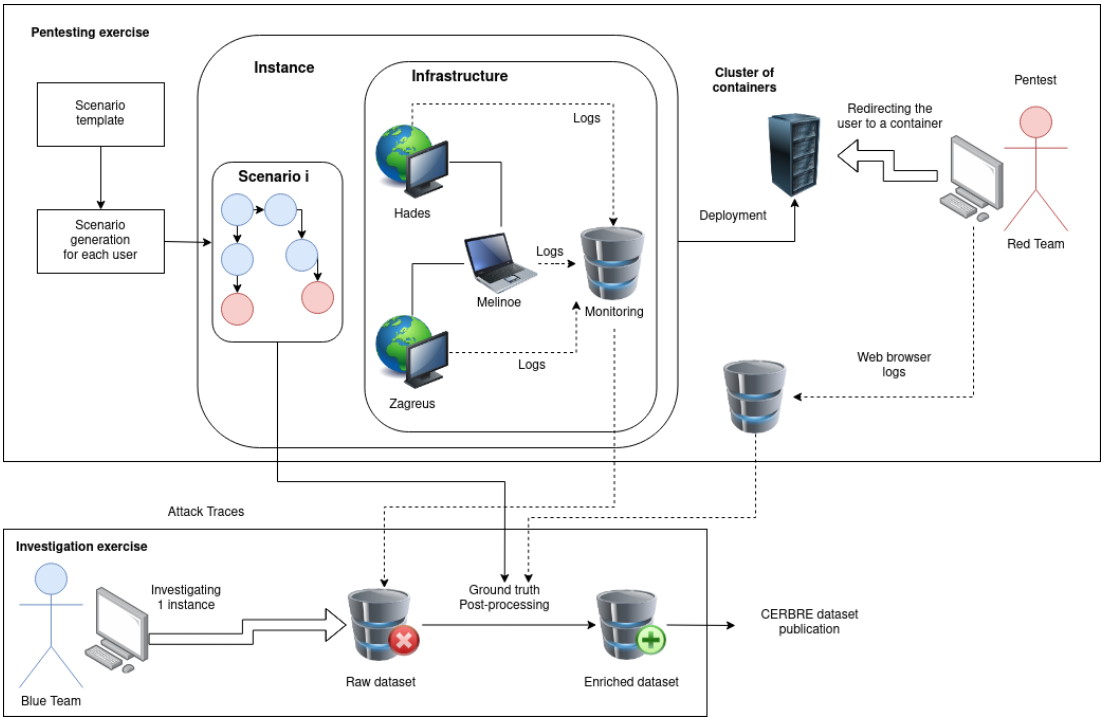

Fig. 1. Overview of CERBERE architecture

rarely promote more than one side of cybersecurity when in reality red teams, blue teams are all intricately linked.

In this article, we describe the CERBERE project that stands for Cybersecurity Exercise for Red and Blue team Entertainment, REproducibility. It is more than just a traditional *Capture-the-Flag (CTF)* challenge conceived to train security teams attacking an IT infrastructure. The CERBERE exercise is composed of two parts: a first CTF-type exercise in which a player must attack an infrastructure that has been generated in order to guarantee its uniqueness. During this exercise, player's actions are monitored from three angles: their pentesting activities performed in their web browser, their network activity, and their activity on host operating systems. This data allows a second hunting phase in which a player must reconstruct all the stages of an attack scenario by exploring the logs. In order to adapt to the skills of the players, the CERBERE exercise is derived into several instances for which we control the difficulty.

Additionally, this article presents the CERBERE dataset containing all the logs produced during the CERBERE exercise. The dataset has been generated with the participation of 22 players: 13 red players and 9 blue players with three levels of difficulty. We explain how this dataset was labelled, enabling us to reconstruct the evolution of the players.

Several outputs are distributed with this publication [1]:

- The software for generating the infrastructure and executing these instances on a regular Linux operating system. The software for deployment on a cluster is not given as it is too specific.
- The raw dataset of the logs of each red team player that can be used for.
- The enriched dataset containing labels for the logs to highlight the evidences of intrusion.

In the following sections, we go into further detail regarding CERBERE. Section II presents the methodology employed for building our red/blue team exercise. In particular, we give details about the pentesting scenario and the information collected during a pentest. Section III presents how we conducted red team and blue team experiments with the players. Finally, Section IV details the labelling of the logs to form the CERBERE dataset. Section V concludes the article.

II. The CERBERE Project

CERBERE is designed as a cybersecurity exercise in two parts where we successively conduct two phases: pentesting a group of vulnerable infrastructures using a similar attack scenario and investigating the logs of said pentest. Figure 1 summarizes the global architecture of the project.

A. Overview

The first phase of the CERBERE exercise is the attack phase. Each player (represented in red) obtains his or her own target instance: a virtual infrastructure hosting the 3 machines to be attacked. All the instances are automatically deployed starting from a generic description of the attack scenario and closely monitored in order to gather data. Each instance deploys a variation of the generic scenario: instances differ

[1] https://gitlab.inria.fr/cidre-public/cerbere-dataset/

TABLE I
LIST OF PROCEDURES AVAILABLE FOR EACH TECHNIQUE IN CERBERE.

Technique	Procedures	Detection rules
$\tau_{0,3}$ = T1190	π_1: Website with command injection (easy) π_2: Website with command injection (medium) π_3: Django directory traversal rewarding ssh key	Access to alice's home repository Chmod in images directory, commands executing .jpg files Not detectable through system logs
τ_1= T1068	π_4: Vulnerable sudo version (CVE-2019-14287) π_5: Vulnerable pkexec process	Command containing "-u#-1" Command executing the pkexec process directly
$\tau_{2,6}$ = T1552	π_6: Passwords in .bash_history π_7: Password in .txt file	Command accessing the .bash_history file Command accessing the important.txt file
$\tau_{4,6}$ = T1021	π_9: SSH Access from key π_{10}: SQL server rewarding a flag	SSHD user_login Command related to the psql executable

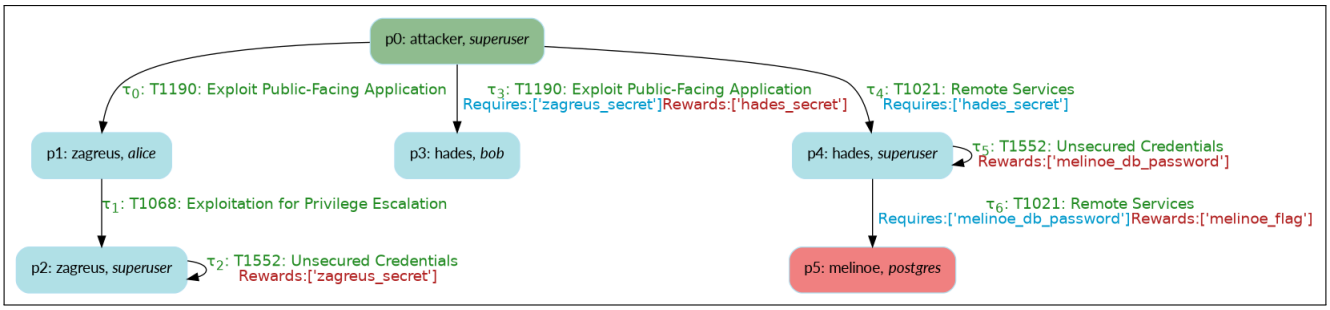

Fig. 2. CERBERE scenario

from each other by the involved operating systems, installed software and their vulnerabilities. Each instance in the CERBERE exercise was unique to each participant allowing him to discover its own way of performing the attack campaign. These aspects are presented in more details in Section II-B. To these instances, two more machines implementing the GHOSTS framework [2] were deployed in order to generate fake life on the infrastructure. However, we don't represent as they are not relevant to the attack scenario.

In the second phase of the CERBERE exercise, new players (the blue team) receives the logs associated to a pentest session of the first phase, except browser logs that would not be available during a real investigation. The logs have been acquired from several monitors, as described later in Section II-C. Blue teamers investigate the logs in a platform developped by Malizen[3] and are expected to flag assets corresponding to attacker actions.

Finally, once the exercise has been played, we gather all the generated data and perform some post-processing actions described in Section IV in order to detect and label the log lines which identify the attacks.

B. Variation of attack scenario for each player

Each instance of this scenario is generated from a template scenario. For this experiment, the CERBERE template scenario consists of three hosts named *zagreus*, *hades* and *melinoe* inside a local network. This scenario is meant to be played linearly: information stored on *zagreus* is required to progress to *hades* for instance. *Zagreus* and *hades* are both vulnerable to procedures linked to misconfigured websites, while *melinoe* hosts a PostgreSQL database. The goal of the attacker is to reach this database, which hosts a flag.

A graphical representation of the overall scenario is available in the Figure 2. Attacker positions are represented as tuples of *(host, user)*, and transitions are labelled with the required MITRE ATT&CK technique[4]. As previously mentioned, instances of this scenario will differ on a procedural level. Table I details available procedures for each technique. All credentials were also randomized for each instance. The second part of the table, detection rules, will be detailed later in Section III-D.

This experiment was designed to be played by a group of participants with heterogeneous attacker skill levels and in a short period of time (two hours). In order to make it possible for every participant to make some progress while maintaining some depth for the more skilled participants we chose to make CERBERE a network of a few hosts with varied challenges, as opposed to a single but more resilient host. Furthermore, in order to increase the range of difficulty offered by CERBERE, instances of the scenario will differ from each others. Indeed, each procedure may have several variations of different difficulties. For instance, in order to perform privilege escalation on a host, one instance might present a vulnerability in its sudo package [8], while the other is vulnerable to the pwnkit exploit [9]. It is to be noted however that the overall path of attackers, *i.e* which sessions they log in and how they progress through the network, remains

[2]https://github.com/cmu-sei/GHOSTS
[3]https://www.malizen.com/
[4]https://attack.mitre.org/

consistent throughout all instances. An additional benefit is that participants cannot easily copycat actions of others. To implement these variations, we used the URSID automatic vulnerable architecture generation tool [10], which aims to convert high-level scenario descriptions into low level virtual host configuration files.

Finally, we required a combination of network and system activities in order to get valuable logs for the blue team. This remark led to the design of an attack scenario where attackers gain shell accesses by exploiting website vulnerabilities but also use system vulnerabilities to get new credentials or roles.

Finally, we describe globally the expected solution of this scenario:. The red player should:

- Exploit the website hosted on *zagreus* by performing a command injection in a poorly configured search bar. Two levels of difficulty are available: some instances limit how many characters could be inputted in the search bar, leading to attackers having to use the upload function of the website to execute their script instead of typing it directly. This exploit leads them to obtain a shell as user *alice*.
- Perform a privilege escalation by exploiting one of the two vulnerabilities [8], [9] (randomly selected).
- Harvest credentials present in the *superuser* directory (which requires root access). Those credentials hint that they are linked to the host *hades*.
- Use those credentials to log as an admin on a Django website hosted on *hades*. Once this is achieved, acquire an SSH key stored in the /notes directory of the website, which hints that this key is linked to a user named *superuser*.
- Use this key to log in through SSH as *superuser*. The IP could be found by installing network scanning tools on the host *zagreus* using sudo privileges.
- Harvest credentials again, which hints that they are linked to the host *melinoe*.
- Use those credentials to access the PostgreSQL instance on *melinoe* and recover the flag, thus ending the scenario.

C. Exercise monitoring at multi-level

During the CERBERE exercise, the activities of the red teamers are monitored at three different levels. First, all their actions at the browser level are recorded such as filling and sending forms, clicking buttons, checking cookies values. Second, we monitor their actions at operating system level, especially actions occurring in a shell session. Finally, network traffic is recorded in the virtual environment.

a) Browser monitoring: To monitor the attacker activity at browser level, we took advantage of the Chrome Devtools Protocol (CDP). This protocol is an interface that allows to instrument, inspect, debug and profile Blink-based browsers such as Chromium and Google Chrome. We developed a tool to connect with the CDP and a script that we injected directly in the browser to record what CDP did not provide us. More precisely, we recorded the Page and Network domain of the CDP to gather navigation information and pages events induced by pentester actions. Then, the injected script captures information that help us differentiate those events, that is which button was clicked or which form was submitted and with which inputs, as inputs may be transformed by Javascript before being sent. The script was also responsible for tracking cookie changes manually done by the pentester as no event from the CDP allows it. All gathered information was processed and recorded in a Mongodb database as specific actions of the attacker: accessing a web page, sending an HTTP form, setting cookies, managing alerts, clicking elements and the different API calls that may come with it.

b) System-level monitoring: We used the auditd tool for monitoring the three GNU/Linux hosts. The logged data includes sensitive file access permissions (read, write, execute), user commands that modify critical system components, services related to time management, automation, and connections. Additionally, we incorporated information monitoring redundancy by logging all system calls of specific types, such as read/write operations and memory allocations for all the processes already monitored. That way, even if an attacker messes with the logging functionality of a process at the application level, the information will still be logged at the system level.

c) Network monitoring: We used Suricata to collect the network traffic. This open-source software enables us to collect the data by adding alerts, statistics, and choices on storage. We gather data in pcap format, recording all the exchanges, and then we add a step to transform data in the netflow format to have a condensed version of the data. For devops reasons, we have such monitoring in all machines in the scenario.

III. Experiments with Cerbere

15 instances were generated on a single server of 36 cores and 80 GB of RAM. Each instance had five hosts in total: *zagreus*, *hades*, *melinoe*, the gateway host and the Suricata host. For the first phase of the exercise, there were 13 participants. They were given roughly one hour to succeed is as many exploits as possible due to logistics constraints, however, since the instances were available through the Internet we let them continue the attacks for as long as they wanted. For the blue team phase, 9 participants investigated the logs for one hour each as well. It is interesting to note that red team exercise being more common than blue team exercises, we can observe more appeal for the first phase.

A. Red team experiment

Since participants required remote access to the instances - as they were all working on their personal computer - all hosts were made available through the Internet. Their entry point was however only accessible through specific URLs with custom port, in order to avoid the instances being compromised by attackers outside the experiment connecting to the port 80. This was achieved by carefully configuring network rules, a custom DNS server and a gateway host used by participants to be connected to the Internet.

The first phase of the exercise was the red team exercise. We requested that the participants come with Docker installed on their computer so that we can record actions coming from the Chrome web browser. Then each participant was given a URL pointing to a specific instance to attack. This way, the attacks performed by one participant could not disturb others. Participants started the exercise with very limited knowledge of the architecture they were attacking.

Participants were told that three hosts are involved in total. We gave them two URLs: one pointing to the website hosted by their specific *zagreus* host, and one for *hades*. They were instructed to first start with *zagreus*, and only move on to *hades* when they find direct clues about how to attack it. This was done due to our short time constraints for the experiment (2 hours), in order to avoid participants spending too much time trying to brute-force *hades*. Hints were occasionally provided to struggling participants in order to improve their learning experience.

B. Blue team experiment

Following the first phase of the exercise, we organized a second phase consisting of a blue team exercise. For this part of the project, each participant was given access to a log investigation platform. This platform allows the visualization of the data from the first phase. It aggregates the fields contained inside each line of log in order to visualize them using various charts. It helps in getting a better idea of the data while allowing the analyst to easily spot the abnormal values. Once the platform as well as the context of the exercise were presented to the participants, they were each given one hour to investigate and find as many steps of the scenario as possible.

The result of these investigations are stored under the shape of user action and associated context. Because this data is very specific to the graphical tool, it would be useless to publish the raw data associated to user's actions. Nine investigations were conducted, for a total of 2706 user actions recorded, allowing us to map exactly the progress of every analyst during the blue team part of the exercise. This exercise helps to evaluate the quality of the data recorded from a security standpoint.

C. Data quality

We manually investigated the data collected during the two exercises for all participants. Our goal is to be able to post-process the logs to quantify 1) how many steps of attacks have been achieved by a red teamer; 2) how many steps of attacks have been correctly investigated by blue teamers. This post-processing is presented in the next section and requires that the relevant information have been correctly logged.

When building our post-processing program, we observed the following points about the data. No logs are missing for one of the steps of attacks, whatever the variation is. Nevertheless, the nature of the log is heterogeneous and contains format variations. Despite monitoring systems or networks with only one probe each, we gather data that is very different and difficult to mix. The system data coming from auditd was notably complicated to explore and reuse as is due to a lack of standardization in the data. Some log lines were recorded over multiple lines some not. Sometimes certain fields seemed to contain one nature of data but depending on the captured log line, this nature could change. However, all the important information to detect and understand the attack is present, it is even redundant and can be followed through the flow of the attack.

TABLE II
SUCCESSFUL ATTACKS (RED TEAM) AND DISCOVERIES (BLUE TEAM)

		Red team exploitation	**Blue team discovery**
Total nb players		13	7
Scenario step	**Mitre ATT&CK Technique**		
T_0	T1190	7	5
T_1	T1068	7	5
T_2	T1155	7	3
T_3	T1190	4	2
T_4	T1021	3	2
T_5	T1552	3	2
T_6	T1021	3	0

D. Discussion

This section discusses how players behaved during the red and blue team exercises. In particular, we explain the reasons behind some players failing to complete the whole red team scenario. As presented in Figure 2, the scenario is composed of 7 steps and 6 different attack techniques. In Table II we report how players progressed in both parts of the exercise and we explain below the reasons why some red and blue teamers could not succeed in completing their variation of the scenario.

The first two columns of Table II shows which step and technique is related to the scenario for further technical references. The third column shows how many red teamers have managed to successfully conduct each step of the scenario. The fourth column shows how many blue teamers uncovered this step of the attack during the investigation.

Regarding the red teamers, it appears that the first exploit (finding a command injection within a website in order to create a reverse shell) was a roadblock for almost half of the participants. This does track with the expected difficulty of the scenario, as this website had a decent amount of functionalities and pages not relevant for the intended exploit, which could act as diversions for the participants. Once the attackers found the location of the command injection, they still had to properly initiate a reverse shell. Depending on the scenario variation, the difficulty of this step may vary because some variations required to upload and update the execution permission for their payload. The combination of (relative) difficulty and multistep nature of the exploit may thus explain this result, and it appears that attackers who managed to pass this step had no issues exploring the rest of the *zagreus* host.

The next roadblock appears to be accessing the second host, *hades*. Logging as *superuser* is not trivial because it required several steps. First attackers had to properly reuse the

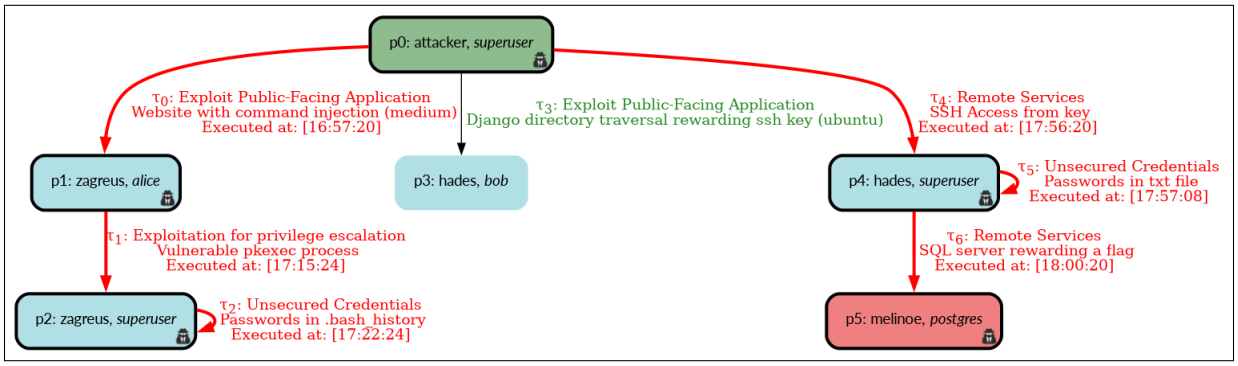

Fig. 3. Attack path through instance 6. Red transitions were detected using Zircolite rules. Transition 3 for attacker position (hades, *bob*) is not detectable through auditd logs, but the linear structure of the scenario ensures that it was compromised as well.

credentials acquired in *zagreus* on the website hosted on *hades*, successfully find an SSH key hidden in one of the pages, format it properly, then use it from their reverse shell acquired in the host *zagreus*. Time constraints also started ticking at this step for participants who spent a long time on the first host. In particular, we observed in the logs that the volatile and primitive nature of the shell acquired on *zagreus* (through the combination of a reverse shell + exploit) meant attackers sometime accidentally closed it and had to redo the previous steps. Finally, attackers who managed this step appeared to have no trouble with the last step, as it simply required to gather credentials similarly to previous steps and to connect to a Psql service, whose IP is found by scanning the network.

We manually analyzed the logs to understand the reasons behind each missing step of an attack. 2 transitions (both for the initial website exploitation $\tau 0$) were undetected as the attackers used unorthodox methods. For instance for procedure π_2, we expected attackers to upload a reverse-shell script using the website image upload function and execute it. However, one of the participants instead recognized that the host naturally had python installed and used that to directly download and execute a script instead. In these two cases the red teamer found an alternative method to the one we prepared for in order to exploit the vulnerability. If we refer to Table I, it means the detection rules corresponding to these exploits were not triggered and the attackers evaded our monitoring. A most interesting result concerning the management of attack surface during the exercise. Nonetheless, in these cases, we assumed that the attacker achieved initial access anyway when the next steps were found in the logs: for instance acquiring credentials as *(zagreus, superuser)* requires the attacker to have accessed machine *zagreus* in the first place. The Figure 3 showcases one of those results for the instance number 6. Complete results for all participants are available in the dataset, as presented later in Section IV.

Regarding the blue teamers, the first thing we noted during the analysis of the data is that not all blue teamers reported on their results as was advised during the presentation of the tool. Unfortunately, two out of the nine participants did not register any findings and thus could not be included in these results. However, we get very interesting feedbacks from the seven remaining participants. First, we note that the web part of the scenario is the most commonly discovered step of the scenario. This can be explained because such attacks produce a lot more noise in the logs and because analysts are used to search for attackers coming from the Internet and are likely to search that part of the dataset first. Another interesting observation we made is that the participants discover groups of steps rather than a single one. Almost all the participants that have made a first discovery were able to discover directly related steps. Directly related steps are the main parts of the scenario: T_0, T_1, T_2, T_3, and T_4, T_5, T_6. The difficulty lied in jumping from phase to phase since hopping from one host to another was not directly linked in the scenario. A final remark is that no participant was able to find the last exploit: the database access. This can be explained by the fact that it was done *legitimately* since the attackers had legitimate credentials at that point. Indeed, the last access to the database on the third machine (τ_6) is done using legitimate credentials found during the previous step. Added to the time constraint we put on the exercise, no participant reached this last discovery.

IV. CERBERE Dataset

With this article, we provide the CERBERE dataset[5]. The dataset contains the raw logs from red teamers and the ground truth about these logs: we automatically labelled the logs to spot the lines that are related to successful attacks of the red teamers. Our goal is to make available a dataset that can be reused in new works related to attack detection. Additionally, the CERBERE dataset also contains the scenario for ensuring the reproducibility of the exercise using the URSID generation tool [10]. The monitoring configuration is also available, enabling future work to extract logs similarly to what has been done during our experiments. As presented in Table I, we designed detection rules to decide if a procedure was successfully employed by an attacker. Depending on the procedure, a detection rule may be implemented at different log levels: system, network, browser. This section discusses the design of these rules to obtain ground truth labels.

[5]https://gitlab.inria.fr/cidre-public/cerbere-dataset/

A. System logs labelling

System logs created by auditd were extracted from every instance at the end of the exercise. Detection rules have been designed using Zircolite [11], a python SIGMA-based detection tool. These rules were manually crafted based on our personal experience and log results from playing the scenario and overall rely on the detection of specific commands or access to files corresponding to the evolution of the attacker in the scenario. For instance, exploiting CVE-2019-14287 [8] requires the attacker to use the command sudo with either the flags *-u#-1* or *-u#4294967295*. Since our auditd was configured to register this kind of command executed by users, we can detect the exploitation of this specific CVE with the following rule :

```
        SELECT * FROM logs WHERE ((type = 'EXECVE' )) AND
↪   (a1 LIKE '-u#-1%' OR a1 LIKE '-u#4294967295')
```

This rule is similar to the ones present in professional security projects such as SIGMA [12]. Every possible procedure - with the exception of the ones associated to τ_4 as it was not detectable through system logs.

These rules were then used on all extracted system logs and correlated with the description of each scenario in order to evaluate the participant's path through each of them. In total, out of the 15 instances planned for this exercise, 7 show at least a sign of compromise and 3 were entirely played out, meaning that the attacker reached the last host. Among these instances, a total of 28 transitions were detected using our rules, out of the 30 we expected to recognize based on attacker performance at the end of the exercise. The 2 missing were due to unexpected attacker actions as discussed before in Section III-D.

B. Network logs labelling

Network traffic recorded by Suricata was extracted in pcap and netflow format to give two detailed traffic levels. The pcap format provides a complete version of the traffic within packets. In contrast, the netflow structure offers a condensed version of traffic gathered in communications. These communications contain only meta-data information (without the payload). This format is studied in the intrusion detection field as it can reduce the quantity of data and eliminate the payload, which could be nonsensical (if encrypted, for example).

We provide ground-truth labels for each packet/communication. Each packet/communication that belongs to a successful attack is labelled as an attack, with the corresponding MITRE ATT&CK techniques (Table I). The other packets/communications that do not participate in a successful attack in a scenario are considered as attempts. For each instance, the packets/communications of a successful attack are identified with the attack path's timestamps and a baseline. The timestamp comes from the previous labelling process of system logs (Section IV-A) and indicates where to look in the network logs. The baseline is a pcap/netflow file that recorded a perfect player playing the instance, thus generating the minimal number of network lines for completing the scenario. The baseline helps us to review manually and decide if the found network logs should be labelled or not.

For example, let us focus on instance 7 and machine melinoe. We want to identify the access to the SQL server of the procedure T1021. The technique for procedure T1021 occurs at τ_6=15:45:03 for the baseline. For instance 7, as seen in Figure 5, we deduce from the previous analysis of system logs that it happened at 17:33:36. With theses timestamps, we compare the two lines for pcap files as represented in Figure 4. We also can compare netflows as represented in Listing 1 and Listing 2. The attack label is finally manually added in the netflow of Figure 4 because ports and protocols obviously match, IP source and destination are the same and the length of the flow is comparable.


```
{
  "StartTime": "15:45:03.794101",
  "Dur": 0.009609,
  "Proto": "tcp",
  "SrcAddr": "192.168.56.3",
  "Sport": 39486,
  "Dir": "->",
  "DstAddr": "192.168.56.4",
  "Dport": 5432,
  "State": "RST",
  "sTos": 0,
  "dTos": 0,
  "TotPkts": 30,
  "TotBytes": 4014,
  "SrcBytes": 1651,

}
```


Listing 1: Example of netflow from baseline


```
{
  "StartTime": "17:33:36.832586",
  "Dur": 0.02053,
  "Proto": "tcp",
  "SrcAddr": "10.35.56.12",
  "Sport": 55976,
  "Dir": "->",
  "DstAddr": "10.35.56.13",
  "Dport": 5432,
  "State": "RST",
  "sTos": 0,
  "dTos": 0,
  "TotPkts": 27,
  "TotBytes": 4239,
  "SrcBytes": 1875,
   "Label": "Attack",
   "Label description": "T1021"
}
```


Listing 2: Example of netflow for the instance 7

C. Browser logs labelling

Attacker's actions were recorded by the browser monitoring tool into a Mongo database. We recorded the actions performed in the web browser by players and the visited web pages. Nevertheless, the labelling uses only the recorded actions. In this labelling process, some attack steps can be labelled and others cannot. Indeed, if the attacker setups a reverse shell even not ciphered, the new established connection is external to the web browser. As a consequence, only the command injection of step T0.

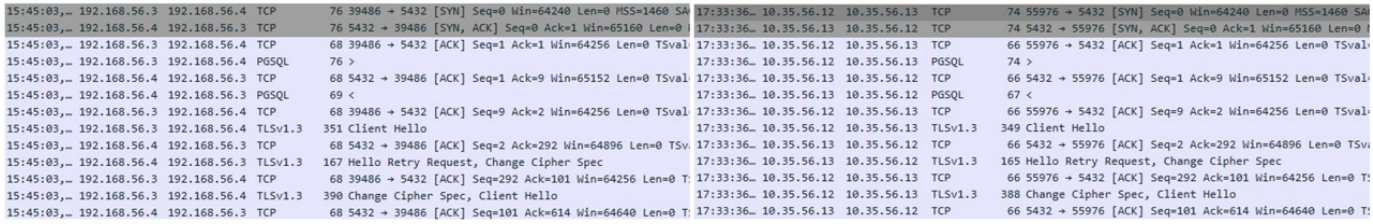

Fig. 4. Wireshark screenshot of melinoe pcap (left: baseline, right: instance 7) In instance 7: IP 10.35.56.13 is melinoe, IP 10.35.56.12 is Hades. In baseline: IP 192.168.56.4 is melinoe, IP 192.168.56.3 is Hades.

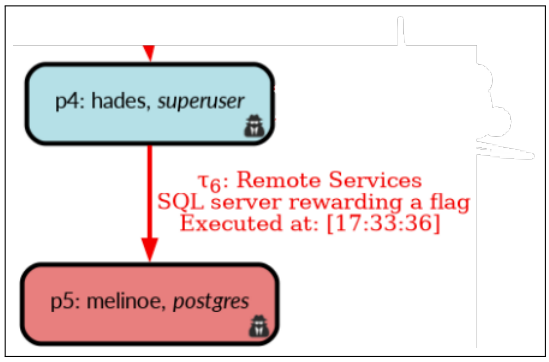

Fig. 5. Attack path extract from instance 7.

Similarly to network logs, we consider malicious the succeeding actions that lead to the exploitation of the command injection. The other lines of the logs were labelled as attempts because it corresponds to research of vulnerabilities, attempts of attacks or documentation. Depending on the scenario, the command injection was either directly exploitable or needed to be prepared by uploading a binary and executing it. As a consequence, depending on the variation of scenario, one or two lines are labelled as a successful attack.

A labelled action is a JSON object that contains the type of action, the argument of the action and our label. For example in Listing 3, the JSON object shows the executing of an injection command using a call to the python interpreter. We can observe the script used by the attacker in the argument of the command.

V. Conclusion

Throughout this paper we have presented the CERBERE cybersecurity exercise and its resulting dataset. CERBERE has the particularity of offering both a red team and a blue team exercise with a variety of difficulty levels. The scenario and architecture of this exercise are given and can be modified for generating a new playground. Three types of logs can be automatically extracted from a played exercise which is particularly for intrusion detection evaluations.

Along with the exercise itself, we provide the dataset resulting from the first edition of the exercise. This dataset contains the logs of the 13 red teamers for the three types of captured logs. Additionally, we labelled the dataset after the

```
{
  "at_time_stamp": "1681310750.539",
  "by_user": "649984c67f6052c4c05c9486",
  "from_page": "649984c07f6052c4c05c9484",
  "type": "SendHTTPForm",
  "args": [
  {
    "name": "inputs",
    "value": {
      "q": "hop | python3 -c \"import socket,os,pty;
            s=socket.socket(socket.AF_INET,socket.
            SOCK_STREAM);s.connect((\\\"7.tcp.eu.
            ngrok.io\\\",11162));os.dup2(s.fileno(),0);
            os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);
            pty.spawn(\\\"/bin/sh\\\")\""
    }
  },
  {
    "name": "xpath",
    "value": "/html/body/main/div/div/div[2]/div[1]
              /div[2]/form"
  }
  ],
  "Label": "Attack"
}
```

Listing 3: Example of web action for a command injection

exercise in order to spot the ground truth about line logs that are successful attacks and make it as usable as possible for future research, regardless of the application.

In the future, we wish to work on one particular area that would help us make the exercise even more realistic: fake life. We have implemented basic behaviours for this edition, but it was easy to spot and get around.

References

[1] S. Floyd and V. Paxson, "Difficulties in simulating the internet," *IEEE/ACm Transactions on Networking*, vol. 9, no. 4, pp. 392–403, 2001.

[2] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, 2018.

[3] M. Lanvin, P.-F. Gimenez, Y. Han, F. Majorczyk, L. Mé, and É. Totel, "Errors in the cicids2017 dataset and the significant differences in detection performances it makes," in *Risks and Security of Internet and Systems*, S. Kallel, M. Jmaiel, M. Zulkernine, A. Hadj Kacem, F. Cuppens, and N. Cuppens, Eds. Cham: Springer Nature Switzerland, 2023, pp. 18–33.

[4] Canadian Institute for Cybersecurity, "Canadian institute for cybersecurity datasets," 2023. [Online]. Available: https://www.unb.ca/cic/datasets/index.html

[5] R. Uetz, C. Hemminghaus, L. Hackländer, P. Schlipper, and M. Henze, "Reproducible and adaptable log data generation for sound cybersecurity experiments," in *Annual Computer Security Applications Conference*, 2021, pp. 690–705.

[6] Z. C. Schreuders, T. Shaw, M. Shan-A-Khuda, G. Ravichandran, J. Keighley, and M. Ordean, "Security scenario generator (SecGen): A framework for generating randomly vulnerable rich-scenario VMs for learning computer security and hosting CTF events," in *2017 USENIX Workshop on Advances in Security Education (ASE 17)*. Vancouver, BC: USENIX Association, Aug. 2017. [Online]. Available: https://www.usenix.org/conference/ase17/workshop-program/presentation/schreuders

[7] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: the CICIDS2017 case study," in *SPW*, 2021, pp. 7–12.

[8] N. I. of Standards and Technology. (2019) Cve-2019-14287. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-14287

[9] ——. (2021) Cve-2021-4034. [Online]. Available: https://nvd.nist.gov/vuln/detail/cve-2021-4034

[10] P.-V. Besson, V. Viet Triem Tong, G. Guette, G. Piolle, and E. Abgrall, "Ursid: Using formalism to refine attack scenarios for vulnerable infrastructure deployment," 2023. [Online]. Available: https://arxiv.org/pdf/2303.17373.pdf

[11] wagga40. (2021) Zircolite. [Online]. Available: https://github.com/wagga40/Zircolite

[12] F. Roth. (2019) Sudo privilege escalation cve-2019-14287. [Online]. Available: https://github.com/SigmaHQ/sigma/blob/master/rules/linux/process_creation/proc_creation_lnx_sudo_cve_2019_14287.yml