# Methodology for Automating Attacking Agents in Cyber Range Training Platforms

Pablo Martínez Sánchez[1] [0009-0001-0044-1338], Pantaleone Nespoli ✉[1,2] [0000-0002-4041-1205], Joaquín García Alfaro[2] [0000-0002-7453-4393], and Félix Gómez Mármol[1] [0000-0002-6567-5012]

[1] Department of Information and Communications Engineering, University of Murcia, 30100, Murcia, Spain {pablo.martinezs2,pantaleone.nespoli,felixgm}@um.es

[2] SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, 19 place Marguerite Perey, 91120 Palaiseau, France {pantaleone.nespoli, joaquin.garcia_alfaro}@telecom-sudparis.eu

**Abstract.** The world faces cyberattacks daily and the targets of these attacks are often critical infrastructure, including the healthcare sector. In addition, more than half of cybersecurity professionals lack the necessary knowledge to deploy the relevant countermeasures to these attacks. In this regard, there is no doubt that education and training in cybersecurity are essential to defend technological assets. That is why, in this context, it is easy to understand that Cyber Ranges play a crucial role since these tools provide the user with a hyper-realistic experience for quality training. Thanks to attack simulators, commonly Advanced Persistent Threats (APT) generators, those realistic defensive cyberexercises can be performed. To implement these components, a behavioral matrix is needed, marking the different stages used by a cybersecurity expert during an attack, e.g. reconnaissance, explotation, data exfiltration, etc. Since bringing the current methodologies to a hyper-realistic production environment is an inordinate challenge, a novel matrix will be designed from simulation environments for training. This new methodology will compact dependent phases and simplify similar stages to automatically. Furthermore, the contribution contains a logic that increases the reality of the attacks. Finally, a proof of concept is made to evaluate the purposes the contribution purses.

**Keywords:** Attack methodology · Advanced Persistent Threat · Cyber Range · Cybersecurity · Critical Infrastructure

## 1 Introduction

Investment in cybersecurity increases every year despite not seeing a robust result against cyberattacks since, as studies published by CSIS (Center for Strategic and International Studies) state, this investment in security and research is still not sufficient [1]. Specifically, in the previous report, one can observe the evolution of the United States and China in this field, attributed to their investment to innovation and self-reliance, independent of other powers.

The healthcare sector, a critical pillar of modern society, is alarmingly vulnerable regarding cybersecurity [2]. This vulnerability is not just a theoretical concern; it has been well-documented and widely acknowledged by industry experts and professionals, as reported in [3,4]. The complex network of interconnected devices, medical records, and patient data forms a repository of sensitive information that is highly attractive for potential exploitation because this software often does not count with continuous updates, like Internet of Things (IoT) devices [5]. Cyber terrorists, recognizing this weakness, frequently target healthcare institutions, not just for the valuable data they hold but also to exploit the life-saving nature of these institutions. A successful cyberattack can disrupt medical services, putting countless lives at risk [6]. As such, there is an urgent need for the healthcare sector to bolster its defenses and prioritize cybersecurity, ensuring the safety and well-being of its patients.

To study the impact of attacks suffered, techniques have been used to quantify the different security postures of networks by simulating cyberattacks. Some of these tests are i) Traditional Explotation tests, which focus on testing the robustness, security, and integrity of the network, and ii) Red Team exercises, where a group of experts simulate a realistic attack where they try to compromise the network by imitating the attacks carried out by cybercriminals [7]. As it can be seen, both are carried out by teams of people, which means that these tests: i) are not completely reliable, given the human error rate caused by, for example, fatigue, and ii) are costly, due to the large outlay involved in hiring an experienced professional. For this reason, the great challenge of automating the actions performed by experts and cybercriminals is born. In particular, the main objective is to achieve: i) robustness, to ensure determinism; ii) scalability, to be able to replicate the tests as many times as necessary; iii) and low cost of the results, compared to a realistic simulation of an attack [8].

One of the most powerful and largely-used tools for training cybersecurity professionals is the Cyber Range platform, where gamification, team confrontations, etc. can be implemented. These platforms are virtualization environments designed to simulate or emulate hyper-realistic environments where activities can be carried out, such as defensive and evasive actions, explotation, data exfiltration, etc [9]. Thanks to this type of tool, nowadays, users of a Cyber Range can acquire highly demanded skills in cybersecurity. Furthermore, according to the World Economic Forum report [10], 62% of cybersecurity professionals do not yet have the necessary skills to be able to respond to a cyberattack [11], including defending against novel and disruptive attacks, such as zero-day attacks.

In this context, it is worth remarking that simulating a realistic attack from which the users of a Cyber Range must defend themselves and/or the simulated infrastructure is a highly complex challenge. These simulations are designed to imitate actual cyber threats and, therefore, require a deep understanding of current cybersecurity threats, tactics, and techniques. The complexity arises not only from the need to create a believable and relevant attack scenario but also from the necessity to adapt and evolve these scenarios in real-time in response to the actions and decisions of the trainees.

This dynamic interplay ensures that the Cyber Range provides an immersive and educational experience, pushing users to apply their knowledge and skills in a high-stakes, controlled environment. The ultimate goal is to prepare cybersecurity professionals for the unpredictable and ever-evolving nature of real-world cyber threats. Thus, the design and execution of these simulations represent a crucial tasks.

The intricacies of real-world scenarios make it a demanding task to replicate accurately. That is why the methodology employed for conducting the attack should be comprehensively and inclusively defined to enable its adaptation to scenarios not previously encountered by the tool. Despite the existence of attack methodologies that define this casuistry, they do not necessarily meet these requirements, given the passage of time and the increasing complexity of new attack vectors. Therefore, this paper aims to answer the following research questions:

1. Can APT simulators follow realistic attack matrices?
2. Are APT simulators sufficiently autonomous to perform an attack?
3. Can APT simulators be useful to Cyber Range users to improve their cybersecurity skills?

To solve these questions, a novel attack methodology has been proposed that compacts, automates, and intercommunicates the different phases presented in the existing attack methodologies. In this sense, such a matrix responds to the abovementioned question 1. The new attack matrix has been trialed using a proof of concept in a Cyber Range setting. This was made possible through a novel framework and various design patterns and application programming interfaces allowing the attack simulator to use any offensive tool and make decisions with an environment it gradually knows, answering the abovementioned question 2. The improvement in the authenticity and adaptability of the attacker will directly impact the defensive maneuvers of the users, which in turn will improve the skills needed to respond effectively to a cyberattack as an answer to question 3.

The structure of the article is as follows. Section 2 analyzes the existing literature on attack methodologies. Then, Section 3 proposes the developed methodology that solves the raised problems. Section 4 exposes a tool-independent architecture using design patterns. Next, Section 5 contains the tool's poof of concept based on the proposed architecture. Finally, Section 6 summarizes the conclusions of the research presented and shows possible future lines of research.

## 2 State of the art

Despite the vast increment in cybersecurity investment, the problem of finding a realistic attack simulator remains unresolved [12]. Presently, prominent national enterprises maintain teams of cybersecurity experts engaged in Red Team exercises. Each team member specializes in specific attack domains such as web services, mail services, firewalls, and more. Given the complexity of today's attacks, these experts follow their methodology based on their personal experience. In turn, they use tools that automate mechanical and repetitive actions, saving time and increasing the reliability of attacks. Examples of such tools are Nmap, for device enumeration; Hashcat, for checking insecure passwords; Burpsuite, for automating actions on web interfaces; among others [12].

Nowadays, systems have more protection measures, which means an increase in the complexity of attacks and tools that automate the exploitation of vulnerabilities to infect a system in a controlled or malicious way. This is why defining a methodology that standardizes all attack techniques requires much knowledge and is highly complex. Over time, several attack matrices have appeared, although there are now globally recognized classifications, such as MITRE ATT&CK [13] or Cyber Kill Chain (CKC) [14].

Specifically, MITRE ATT&CK is a globally accessible knowledge base on tactics (i.e., thematic sets of techniques) and techniques (i.e., attack sets based on real-world observations). The MITRE ATT&CK knowledge base is a foundation for developing specific threat models and methodologies in the private sector, government, and the cybersecurity products and services community. The matrix is created from a theoretical point of view and brings together the most commonly used attacks in real environments. The repository par excellence, which contains the implementation of attacks organized under the MITRE ATT&CK hierarchy, is the Atomic Red Team[3]. In particular, the matrix comprises the phases described and its unique identifier, as described in Table 1.

**Table 1.** MITRE ATT&CK's phases

| Phase | ID | Description |
|---|---|---|
| Reconnaissance | TA0043 | Gathering information |
| Resource Development | TA0042 | Creation of support resources |
| Initial Access | TA0001 | Network information gathering |
| Execution | TA0002 | Vulnerability exploitation |
| Persistence | TA0003 | Establishing persistence |
| Privilege Escalation | TA0004 | Obtaining administrator rights |
| Defense Evasion | TA0005 | Maneuvers to avoid detection |
| Credential Access | TA0006 | Theft of users and passwords |
| Discovery | TA0007 | Discovery of the internal environment |
| Lateral Movement | TA0008 | Moving around the internal environment |
| Collection | TA0009 | Gathering information of interest |
| Command and Control | TA0011 | Communicating with compromised systems to control them |
| Exfiltration | TA0010 | Theft of information |
| Impact | TA0040 | Manipulation, disruption, and destruction |

On the other hand, the CKC model of cybersecurity explains the typical procedure cybercriminals follow to complete a successful cyberattack. Concretely, it is a framework developed by Lockheed Martin, derived from military attack models, and translated to the digital world to help teams understand, detect, and prevent persistent cyber threats. It is comprised of the phases reported in Table 2.

Apart from the previous theoretical methodologies, practical approaches, such as the one provided by the expert Carlos Polop, i.e., HackTricks [15], can also be found. This methodology has been developed based on extensive experience on offensive security. In conjunction with this, one can discover a comprehensive description of the implementation of each attack vector. In particular, he explains to the user the reasoning behind each decision taken in the methodology and describes the concepts, steps to

[3] `https://atomicredteam.io/`

**Table 2.** Cyber Kill Chain's phases

| Phase | Description |
|---|---|
| Recognition | Collection of information from open sources |
| Preparation | Selection and exploitation of attack vectors |
| Distribution | Distribution of malicious payload across systems |
| Exploitation | Exploitation of distributed malware |
| Installation | Establishment of persistence |
| Command & Control | Communication with compromised computers |
| Actions on Objective | Monitoring and post-exploitation |

follow, or configurations required for the tools involved in cybersecurity [15]. It comprises the phases reported in Table 3.

**Table 3.** HackTricks's phases

| Phase | Description |
|---|---|
| Physical Attacks | Physical attacks on equipment |
| Discovering | Discovery of access routes and equipment |
| Discovering internal | Capturing sensitive information once inside the network |
| Port scan | Search for vulnerable services on computers |
| Searching service version exploits | Search for known vulnerabilities in services |
| Pentesting Services - Automatic Tools | Exploitation of vulnerabilities with automatic tools |
| Pentesting Services - Brute-Forcing services | Exploitation using brute force attacks |
| Phishing | If the previous steps did not work, phishing |
| Getting Shell | Obtaining means for the execution of remote commands |
| Inside | Execution of remote actions on victim machines |
| Exfiltration | Extracting and inserting files on the victim |
| Privilege Escalation - Local Privesc | Obtaining privileges on the victim machine |
| Privilege Escalation - Domain Privesc | Obtaining privileges in the organization |
| POST - Looting | Obtaining critical or confidential data |
| POST - Persistence | Establishing persistence |
| Pivoting | Infecting other computers connected to the victim |

The above matrices contain valuable knowledge that should be taken into account when it comes to describe the steps that an attacker would make in a realistic situation. The problem with them is that the knowledge is too segmented, causing the software to fall into very rigid and sequential implementations. Despite being able to catalog different attacks accurately, MITRE ATT&CK's fragmentation complicates implementation. While the database's methodology of storing attacks is generic, which is not inherently negative, applying these attacks to a specific use case becomes more complex, as they are not designed for any particular environment. On the other hand, CKC has no mechanism that allows flexibility between the different phases. The design of this matrix is simple, yet it is closely interconnected in a way that can result in repetitive, predictable, and identifiable attacks, ultimately making it challenging to attach tools. Finally, Hack-Trick was created to introduce and teach people about pentesting, making it the most practical option. Nevertheless, it has drawbacks, too. Specifically, it is focused on the educational environment and not on the automation of attacks, making it very difficult to develop a tool. Moreover, the tool would have, again, the same problem: it would not be flexible enough to implement in a Cyber Range environment.

As mentioned above, the existing matrices have an informative approach and are far from being able to carry out hyper-realistic implementations in a Cyber Range environment. This is because fragmenting the stages with so much detail increases the complexity of automating them. Solving fragmentation would lead to software solutions that would very accurately simulate the activities performed by a human attacker, answering the question 1 posed in the introduction.

## 3 Methodology

The challenge that this paper aims to solve is to develop a methodology that is generic and flexible enough to be able to automate all kinds of realistic attacks within a training platform (e.g., Cyber Range platforms) where cybersecurity capabilities can be developed while answering questions mentioned above.

### 3.1 Matrix

The methodologies analyzed above do not represent a good fit with Cyber Ranges environments for the reasons given above. Despite this, given that they contain mature, widely used, and recognized knowledge, we will later contrast what knowledge is contained in the novel methodology with MITRE ATT&CK 1, CKC 2 and HackTricks 3.

This matrix is created with the motivation of compacting the highly dependent phases. In addition, the possible coupling of tools used by attackers is considered. The phases resulting from the exhaustive study conducted are reported in Table 4.

**Table 4.** Phases of the proposed novel matrix and their description

| Phase | Description |
|---|---|
| Discovery | Discovery of information, equipment, and vulnerabilities |
| Explotation | Exploitation of vulnerabilities |
| Inside | Exfiltration, infiltration, and information gathering |
| Pivoting | Provisioning and infection of other computers connected to the victim |

The *Discovery* phase includes everything related to the discovery of organizational information: finger/footprint (data collection in the public domain), the discovery of equipment, services, and vulnerabilities in the organization, the relationship between services and Common Vulnerabilities and Exposures (CVE), etc.

Next, *Explotation* encompasses all types of vulnerability exploitation: exploitation of CVEs, brute force and Denial of Service (DoS) attacks and remote code execution, among others.

Once the control of the computer has been taken, the next phase is the *Inside*, where all the actions that will occur inside the victim machine are carried out: data exfiltration, persistence, privilege escalation, just to cite some examples.

Finally, in certain situations, it is necessary to establish communication channels for lateral movement between different machines. This phase is called *Pivoting*.

All phases are encompassed by reasoning that relates and manages the dependencies between the abovementioned phases. Certainly, it is thanks to *Logic*, which will be explained in Section 3.2. In the case of a Cyber Range instructor, a skilled professional responsible for guiding, monitoring, and managing cyberexercises within the platform, this role enables the configuration and adaptation of the different stages to simulate an attack effectively.

The formulation of the suggested matrix arises from observing that various stages transpire concurrently during a specific attack phase and the pronounced interdependence of certain elements. The main objective is the attacker simulation is to achieve a target. Generally, such a target can have various forms in a training environment, e.g., retrieve a configuration or file, gaining access to an equip, denying a service, or getting root privileges, among others. After capturing a target, several actions are executed, allowing them to occur within the same phase. Similarly, some phases have a tight connection to the prior stage, paving the way for integration. Consequently, even though compressing the phases might lead to a marginal rise in complexity, it does not result in information loss. On the contrary, this condensation can enhance the automation of implementations using this approach.

Once the different stages have been defined, the reliability and completeness of the proposed methodology must be supported. To do so, comparing it with those previously mentioned is mandatory. That is to say, Table 5 compares our proposal with MITRE ATT&CK, while Table 6 contrasts it with CKC, and finally Table 7 compares it with HackTricks.

**Table 5.** Comparison of our proposal with MITRE ATT&CK

| | Discovery | Exploitation | Inside | Pivoting | Logic |
|---|---|---|---|---|---|
| TA0043 | X | | | | |
| TA0042 | | X | | | |
| TA0001 | X | | | | |
| TA0002 | | X | | | |
| TA0003 | | | X | | |
| TA0004 | | X | | | |
| TA0005 | | | | | X |
| TA0006 | | | X | | |
| TA0007 | X | | | X | |
| TA0008 | | | | X | X |
| TA0009 | | | X | | |
| TA0011 | | | | | X |
| TA0010 | | | X | | |
| TA0040 | | X | X | | |

**Table 6.** Comparison of our proposal with CKC

| | Discovery | Exploitation | Inside | Pivoting | Logic |
|---|---|---|---|---|---|
| Reconnaissance | X | | | | |
| Weaponization | | | | | X |
| Delivery | | X | X | X | X |
| Exploitation | | X | | | X |
| Installation | | | X | | |
| Command and Control | | | | | X |
| Actions on Objective | | | X | | |

**Table 7.** Comparison of our proposal with HackTricks

| | Discovery | Exploitation | Inside | Pivoting | Logic |
|---|---|---|---|---|---|
| Discovering | X | | | | |
| Internal Test | X | | | | |
| Port scan | X | | | | |
| Searching service | X | | | | |
| Pentesting service | | X | | | |
| Phishing | | X | | | |
| Getting shell | | X | | | |
| Inside | | | X | | |
| Exfiltration | | | X | | |
| Privilege escalation | | | X | | |
| Post | | | X | | |
| Pivoting | | | | X | |

As depicted in these tables, for every stage of the matrix being compared, there is always one of the columns marked, i.e., all phases are included in the designed methodology, thus indicating that the novel matrix is complete. This means that sections of previous methodologies are compacted and oriented to automation, and the information in the other matrices is maintained, converging in a realistic attack methodology. That is, using this approach provides a solution to the question 1.

### 3.2 Logic phase

The phases described above are correctly measured and compared with the other methodologies, demonstrating the capabilities of the Logic phase. In particular, the proposed methodology need a logic that controls and coheres the different phases for realistic attacks. It is worth remarking that we were inspired by the Observe, Orient, Decide, Act (OODA) loop [16]. The logic followed to concatenate the different phases is depicted in Figure 1.

The methodology followed starts with a “target”. This is the application’s objective and will determine the means of one of the two stop conditions. Once the agent starts,

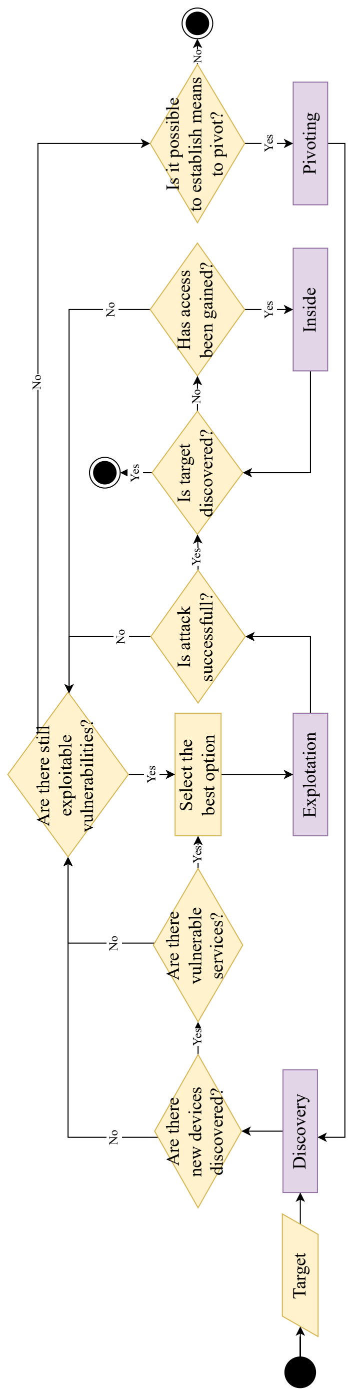

**Fig. 1.** Attack flowchart representing the phases of the proposed methodology within the Logic phase

it tries to discover vulnerable equipment and their respective services, which implies the *Discovery* phase and use of tools like Nmap, an open source utility for network discovery and security auditing [17]. Next, it selects the best of the attack vectors, as explained in Section 3.2, and exploits them, which implies the *Exploitation* phase and uses tools like Metasploit, a penetration testing framework [18]. If the attack is successful, we will check if we have reached the target. If the target has been reached, the program terminates. Otherwise, we check if the attack gains access, a new agent will be deployed, and all the information of the new user machine is exfiltrated, which implies the *Inside* phase and use of its tool, like PEASS-ng, a privilege escalation tool and exfiltration [19]. Then it is checked again whether the target has been reached. If it has not been reached, we will check if ”Has access been gained?” forcing the answer to be negative. Finally, the logic checks if there are additional vulnerabilities to exploit and repeats the process.

If, in any of the above decisions, we are unable to continue, we will go on to evaluate if there are any vulnerabilities left to exploit. Note that if we do not find any device after running the phase *Discovery* for the first time, we proceed to search for any remaining vulnerabilities. This decision is made because the agent has found vulnerabilities within the equipment where it is located.

Finally, the agents will perform the lateral movement if no vulnerabilities are left to exploit. This action involves the *Pivoting* phase and its tools. This phase is the last of the possibilities because this action implies a huge interaction with the network and is potentially detectable by Intrusion Detection Systems (IDS). If it is impossible to perform lateral movement on any of the computers, the attack will end.

The decisions made by the proposed logic also consider factors when the different phases take place, as we will see next.

**Topology exploration** At the time of the initial discovery of devices and vulnerabilities, or after the lateral movement, agents must consider which devices to analyze or attack. When it comes to decision-making in terms of cybersecurity, the experts in the field will be the ones to provide the basis. Depending on the objective the attacker is trying to achieve, we can categorize the attacks into various types, such as taking control of the machine, performing a denial of service on the victim organization, or stealing critical data, among others. Choosing one machine or another as a target in the “Discovery” phase will be determined by the initialization data, located in “Target”.

To ensure a comprehensive network exploration, finding an algorithm that can achieve this goal is crucial. We have chosen the Breadth First Search (BFS) approach for our proposal, as it offers distinct advantages [20]. One of the significant benefits is that APTs can conduct this exploration in parallel, as illustrated in Figure 2.

By distributing the exploration of the devices, the algorithm can scan the topology more efficiently, reducing the total search time. In addition, the parallel version of BFS maintains its sequential counterpart’s completeness and optimality properties, guaranteeing that if there is a path from the outside to the target, BFS will find it because it is complete if the branching factor is finite. Since the topology is a graph of equally weighted edges, it will find the shortest path eventually because if every node has the

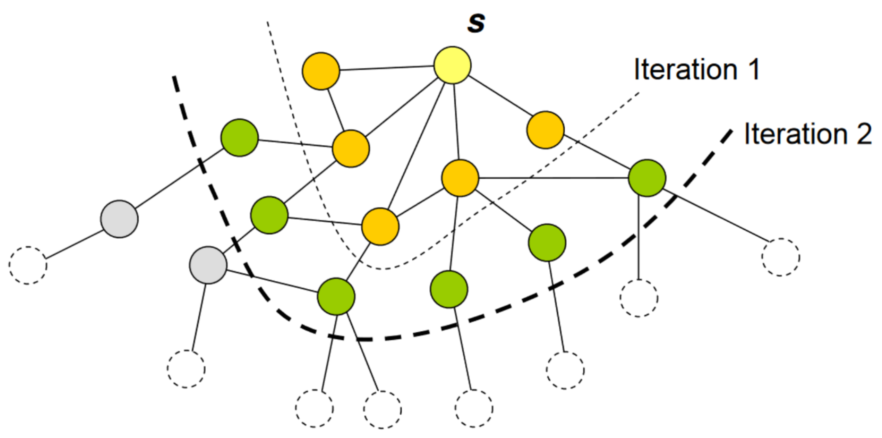

**Fig. 2.** Parallel breadth-first search of APTs for Discovery and Pivoting phases

same cost, BFS is optimal. However, BSF's parallel feature must be used thoroughly to enhance IDS detection.

**Select the best option** The *Explotation* phase is very extensive, since there are many and very heterogeneous vulnerabilities and possible offensive tools. Thanks to the architecture proposed in Section 4, the implementation of the phase will be able to couple tools used by professionals to the proposed methodology. The critical point in the proposed flow comes when it is necessary to decide which of the available attacks given by the supported tools will be executed. To solve this decision-making, a knowledge base is proposed where the "score" of the attack is stored and updated for each implemented tool. This option has been chosen for the following reasons: i) vulnerabilities are not always exploitable, ii) different interface implementations exist, depending on the tools used, and iii) the criticality is not always sufficient; priority should be given to attacks that enable the takeover of the equipment. In addition, exploitation depends on temporal, contextual, and even external factors, such as the stability of the network, among others. In Figure 3, the knowledge base is denoted with red color, and is referred to as *Attack score*. After implementing the methodology and providing sufficient training time, the accuracy of vector exploitation will be improved. This will help in avoiding patched attack vectors. The equation 1 updates and stores the results, which helps in learning the best vectors to exploit.

$$score = (old\_score * n + CVSS * success\_lvl)/(n + 1) \quad (1)$$

Where the variables take values depending on the following:

– *score*: The calculated gravity score. It will be stored in the knowledge base.

- *old_score*: The average score previously saved in the knowledge base and based on which it will be decided whether to use that attack vector.
- *n*: The number of times the average has been calculated.
- *CVSS*: Common Vulnerability Scoring System (CVSS) or criticality level of the detected vulnerability as determined by Forum of Incident Response and Security Teams.
- *success_lvl*: A variable that can take the values:
  - 0.2, being unable to exploit the vulnerability.
  - 1, when it is possible to exploit it and achieve the desired target or Remote Code Execution with a user who does not have root permissions.
  - 2, when exploit achieves Remote Code Execution with a user with root permissions.

The values used in the success level variable are a first approximation since they have desirable characteristics. For vulnerabilities patched to have a lower value, a value of 0.2 has been assigned to penalize this type of event. On the other hand, if the vulnerability has been successfully exploited, then we will speak of a factor of 1 since, as this value is the idempotence of the multiplication, it is equivalent not to punishing or rewarding the score. Finally, suppose that vulnerability returns a better result than expected, such as a user with administrator privileges. In that case, we will discuss a factor of 2 to promote using that vulnerability in future cases.

After an attempt to breach a system, the result will be stored, and the following data will be updated in the knowledge base: *score*, *glscvss*, and *n*. In case that attack has not obtained any result, the value of the success level variable will be one-fifth of the criticality value or *CVSS*, it will be registered in *Logic* and the next most promising one for a certain machine will be tested. The attacks start with the valuation established in *CVSS*, so if there is no record in the knowledge base, this value will be taken to make the comparison with the other available attacks.

The CVSS provides a standardized measure of the inherent severity of computer system vulnerabilities. However, it falls short as a sole metric for assessing their criticality due to its static nature and lack of contextual consideration. CVSS scores do not usually account for real-world factors such as the effectiveness of patches, mitigations, or the specific environmental conditions that affect exploitability, which can be added only with a quite deep knowledge of the network assets. This limitation underscores the need for a more comprehensive approach integrating dynamic, context-specific factors alongside CVSS scores. This formula aims to fix the problems that CVSS scores may expose in our context (i.e., simulation of APTs in a CR ecosystem), rewarding and punishing the attack vectors by experimenting with them.

By implementing a logical framework, we ensure that each process phase is distinct and interconnected, creating a coherent and comprehensive system. This approach, driven by logic, prevents stagnation from overly formulaic processes. Instead of adhering to a rigid structure, the system dynamically adjusts to various scenarios, maintaining unpredictability and adaptability. This adaptability is crucial for maintaining a sense of realism, as it mirrors the complexity and variability of real-world situations. The process remains relevant, engaging, and effective, avoiding over-simplification and embracing complexity. Furthermore, this logical integration promotes a deeper understanding and a more nuanced approach to problem-solving, as it requires and fosters an environment where critical thinking and innovation are paramount. In essence, logic is not just a tool for coherence but a catalyst for creativity and realism needed in a Cyber Range environment. Therefore, the training of people is effectively improved with automated network agents, and the propose stands that question 2 has been resolved.

## 4 Architecture

In this section, an architecture is proposed for carrying out the methodology. As previously mentioned, APT generators are proposed in this case, given their great power and importance in Cyber Ranges. The primary reason for selecting APTs is that they are globally regarded as the most challenging and relentless attacks. An additional abstraction layer is necessary for the proposal to implement the proposed matrix correctly. That is why we have resorted to creating an Application Programming Interface (API), which achieves homogeneity using different APT generators. Such an API interface will allow the methodology implementation to easily couple the tools used to perform real attacks, thus providing the methodology with the necessary realism in Cyber Range environments.

Furthermore, an API has been proposed for each stage, as various automation tools are available for exploiting vulnerabilities or security flaws. Within each API, the option to launch commands manually is also included because an instructor could control phases manually if needed. Streamlined integration is achieved through the use of it, facilitating seamless communication between various software applications and enhancing interoperability. Additionally, APIs allow systems to scale and adapt to new features or changes without requiring extensive modifications. The architecture is shown below in Figure 3.

Specifically, differentiation between various components is based on their respective colors in figure above: i) *purple*, which houses the logic of each phase, intercommunicating the different tools and functionalities available; ii) *light blue*, which represents the interfaces corresponding to each stage of the methodology; iii) *orange*, which refers to the implementation of the interface for each available tool and a manual command launcher; iv) *gray*, which highlights each of the available tools; v) *red*, which represents where the knowledge base of the attacks is located; vi) and *green*, which houses how the API communicates with the APTs controller, specifically with the *Logic* class.

To run a complete implementation of the architecture, it will be necessary to have at least one tool implementing each interface. Otherwise, it will not be possible to loop over the methodology. An example of tool per phase would be: i) Nmap, a penetration testing framework as a *Discovery_tool* [17]; ii) Metasploit, a open source utility for network discovery and security auditing as an *Exploitation_tool* [18]; iii) PEASS-ng, a privilege escalation tool and exfiltration as an *Inside_tool* [19]; iv) SSHUTTLE, a VPN simulator over SSH connections as a *Pivoting_tool* [21]; v) CALDERA, an automated adversary emulation platform manager as *APT_controller* [22].

The solution's foundation lies in the interface *APT_controller*, which communicates the APTs of an existing tool with the rest of the architecture. In addition, *Logic* is in

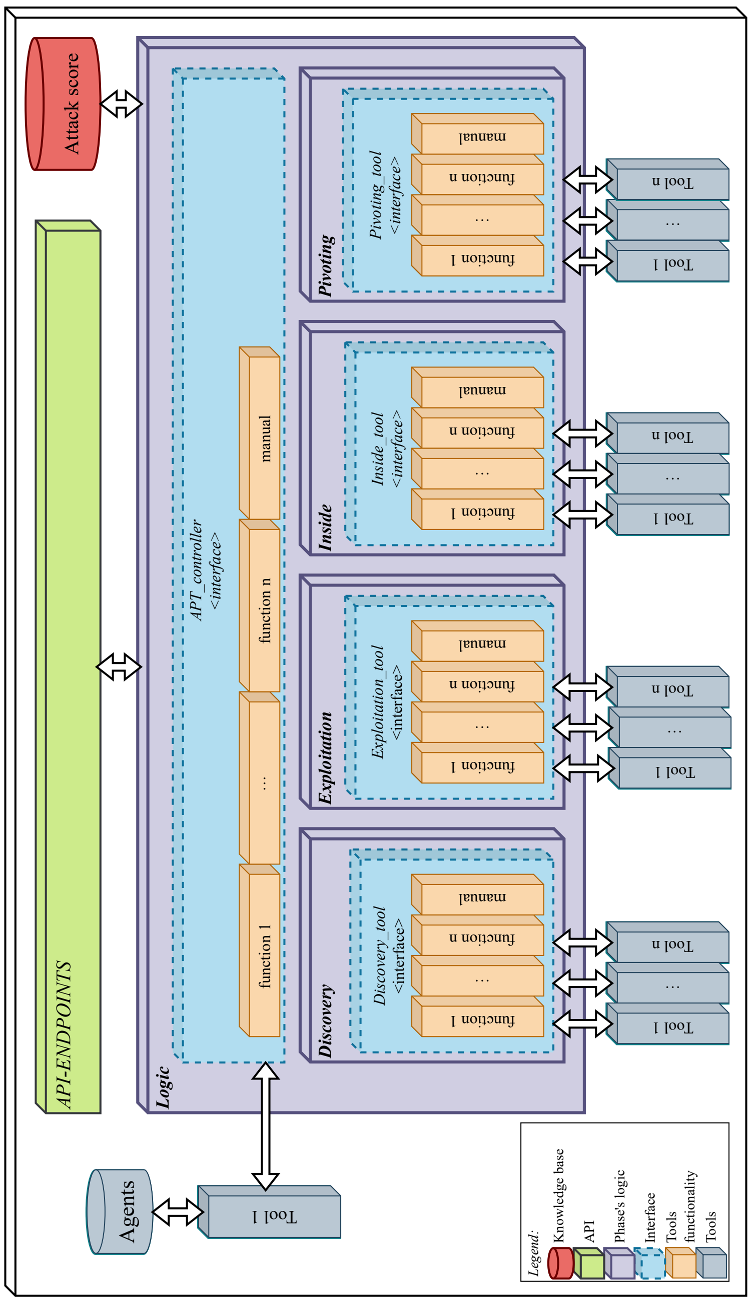

**Fig. 3.** An abstract overview of the proposed framework, highlighting the main components and phases

charge of intercommunicating and coordinating the rest of the phases, which is why the rest of the stages are contained in it.

The architecture's modular design offers numerous benefits. Each tool functions as a separate unit, minimizing the risk of systemic failure due to a single tool malfunction. This separation enhances maintainability, as updates or fixes can be applied to individual tools without impacting the entire system. Furthermore, the API-centric approach facilitates integration with external systems and enables seamless data exchange and interoperability. The architecture's scalability ensures it can adapt to evolving requirements and accommodate new tools or attacks as they emerge. By embracing a user-centric design, the architecture can be customized to meet specific user needs, providing a more intuitive and efficient user experience. This approach balances robustness, flexibility, and user-friendliness, making it an ideal solution for complex and dynamic environments.

The combination of all the elements proposed allows us to have a more complex and complete educational environment where the architecture will able to launch realistic and automatic attacks. Thus, we consider the question 2 to be solved.

## 5 Implementation

The implementation of the proposed framework has been developed in Python, and, as defined in the Section 4, it has different APIs accompanied by the adapter pattern for each phase. This pattern will play an important role, allowing collaboration between objects with incompatible interfaces. Before continuing, it is crucial to comment on the terminology that is used during the implementation of the solution:

- **Agent**: APT deposited on the victim.
- **Abilities**: Actions that the *Agent* can execute.
- **Operation**: Execution of textitAbilities on *Agent*.

All the APIs mentioned in the architecture have been defined to develop the presented solution. Next, the adapter pattern and the API created to control the APT generators or attack simulators are explained. The API's implementation is the most complex and important since it is the basis for executing the created matrix. It must be carefully designed as any constraints in this segment will be passed on to the remainder of the solution.

In the code of the adapter pattern created, the functions of creation, reading, update, and deletion (Create, Read, Update, Delete, CRUD) have been implemented for the operations and skills, as can be seen in listing 1.1:

**Listing 1.1.** Implementation of the interface of APT_controller

```
class APT_controller(Singleton):

    #AGENTS
    interface retrieve_agents()
    interface retrieve_agent_detail(id)
    interface delete_agent(id)

    #ABILITIES
    interface retrieve_abilities()
    interface retrieve_abilities_by_tactic(id)
    interface retrieve_abilities_by_tactic_string(id)
    interface retrieve_ability_details(id)
    interface add_ability(ability)
    interface delete_abilities(ids)

    #OPERATIONS
    interface retrieve_operations()
    interface retrieve_operation_details(id)
    interface add_operation(operation)
    interface delete_operation(id)
    interface add_action_to_operation(ability)
```

The interfaces define methods that will later be redefined by the child classes, where the specific functionality of each tool will be implemented. In addition, the class uses the Singleton pattern to ensure that a class has only one instance, while providing a global access point to this instance.

Since the tool aims to extend and concatenate tools and functionalities, part of the implementation of the APT_controller's interface will developed to control, for example, Caldera. It will be shown below, specifically, the *Agents* and *Abilities* units as shown in listing 1.2.

**Listing 1.2.** Implementation of the Caldera API 1/2

```
class Caldera(APT_controller):
    def __init__(self, network_address):
        self.api = Caldera_API(network_address)

    #AGENTS
    def retrieve_agents(self):
        return self.api.get_agents()

    def retrieve_agent_detail(self, id):
        return self.api.get_agent_detail(id)

    def delete_agent(self, ids):
        return self.api.delete_agents(ids)

    #ABILITIES
    def retrieve_abilities(self):
        return self.api.get_abilities()

    def retrieve_abilities_by_tactic(self, id):
        return self.format(self.api.get_abilities_by_tactic(id))

    def retrieve_abilities_by_tactic_string(self, id):
        return self.format(self.api.
            get_abilities_by_tactic_string(id))

    def retrieve_ability_details(self, id):
        return self.api.get_ability_detail(id)

    def add_ability(self, ability):
        return self.api.add_ability(ability)
    def delete_abilities(self, ids):

        return self.api.delete_abilities(ids)
```

Some tools have their APIs, which operate through HTTP requests. It might be a good decision to create an additional class that encapsulates the “raw” API calls, like Caldera. This decision should be based on the repeated inconsistencies contained in the native API to solve future changes easily. Additionally, the control of the different responses and errors can be seen in listing 1.3.

**Listing 1.3.** Implementation of the Caldera API 2/2

```
class Tool_API():
    TOKEN = 'caldera_api_token'

    Caldera_API(addr):
        API_HTTP = connect(addr, token)

    #AGENTS
    function get_agents():
        return API_HTTP.get('api/v2/agents')

    funcion get_agents(agent):
        return API_HTTP.get('api/v2/agents', agent)

    funcion remove_agents(ids):
        return API_HTTP.post('api/v2/agents', ids)

    #ABILITIES
    funcion get_abilities():
        return API_HTTP.get('api/v2/abilities')

    funcion get_abilities(id):
        return API_HTTP.get('api/v2/abilities', id)

    funcion add_ability(ability):
        return API_HTTP.put('api/v2/abilities', ability)

    funcion delete_abilitiy(id):
        return API_HTTP.delete('api/v2/abilities', id)
```

In this last case, the utilization of CRUD methods is observed: i) creation, where objects are created (set); ii) reading, where objects are retrieved (get), iii) update, where objects are overwritten; iv) and delete, where objects are deleted (delete). In order to access the tool API, we use a token and the methods described in the documentation.

It is thanks to the union of both design decisions that the proposed solution can be independent of tools, avoid the drawbacks that it contains, and automate the attack methodology. Thus, we affirm that an APT simulator can be sufficiently autonomous and, above all, realistic. It would greatly enrich the experience of the student and will be able to better develop potential skills. Thus, the question 3 is considered solved.

## 6 Conclusions

Cybersecurity in healthcare safeguards sensitive patient data prevents identity theft, and ensures uninterrupted medical services. Breaches can compromise patient safety, disrupt operations, and erode trust in healthcare institutions. Robust security is vital for patient protection and maintaining healthcare integrity. To this end, it is essential to train users of computer systems with offensive and defensive skills. Cyber Ranges have proven to be a crucial tool for the latter, hosting realistic attack agents to train users automatically.

To better argue on the current and future capabilities of the automatic attack frameworks within the Cyber Range ecosystem, we have raised three research questions in this article, which we have demonstrated and answered affirmatively throughout the text. In particular, we believe that APT simulators can follow realistic attack matrices thanks to the proposed matrix. Then, the proposed matrix is sufficiently autonomous to perform an attack due to the methodology proposed. Finally, because of realism, APT simulators are useful to Cyber Range users to improve their cybersecurity skills.

In this context, our proposal's primary motivation stems from current attack matrices' inability to be oriented towards adaptive and realistic automation of attacks in Cyber Ranges environments. This is why a matrix was developed, generalizing enough to encompass all the necessary concepts and avoid losing information or utility. From it, we have developed an architecture where tools are implemented per stage that, in a modular way, can host the functionality required by the methodology. APIs enable seamless integration and communication between diverse tool implementations, promoting scalability, efficiency, and innovation. They facilitate rapid development, enhance security through controlled access, and extend reach and flexibility in the landscape.

Although the contribution has a huge potential, it requires further refinement for upcoming endeavors. We will continue to investigate different options and the challenges that lie ahead in Cyber Ranges environments. Ideally, a micro-service tailored to utilize within a Cyber Range should be implemented to test compatibility with other modules. Moreover, we aim to explore techniques suggesting realistic waiting periods shaped by fluctuating challenges while ensuring student engagement. Alongside this, we are looking into the infusion of artificial intelligence to supplant some functionality located on *Logic*, as the selection of the best option exploiting vulnerabilities, ushering in a deeper layer of authenticity.

## Acknowledgments

This work has been partially funded by the Spanish Ministry of Universities linked to the European Union through the NextGenerationEU programme, from the postdoctoral grant Margarita Salas (172/MSJD/22), and from the strategic project CDL-TALENTUM from the Spanish National Institute of Cybersecurity (INCIBE) and by the Recovery, Transformation and Resilience Plan, NextGenerationEU. Authors acknowledge as well support from the CybAlliance project (Grant no. 337316).

## References

1. "Viewpoint: For stronger tech, Europe must spend more on defence and research," https://sciencebusiness.net/viewpoint/Sovereignty/stronger-tech-europe-must-spend-more-defence-and-research, accessed: April 20, 2023.
2. A. López Martínez, M. Gil Pérez, and A. Ruiz-Martínez, "A comprehensive review of the state-of-the-art on security and privacy issues in healthcare," *ACM Comput. Surv.*, vol. 55, no. 12, mar 2023. [Online]. Available: https://doi.org/10.1145/3571156
3. G. A. Cavaliere, R. Alfalasi, G. N. Jasani, G. R. Ciottone, and B. J. Lawner, "Terrorist attacks against healthcare facilities: a review," *Health security*, vol. 19, no. 5, pp. 546–550, 2021.
4. A. I. Newaz, A. K. Sikder, M. A. Rahman, and A. S. Uluagac, "A survey on security and privacy issues in modern healthcare systems: Attacks and defenses," *ACM Transactions on Computing for Healthcare*, vol. 2, no. 3, pp. 1–44, 2021.
5. S. Vishnu, S. J. Ramson, and R. Jegan, "Internet of medical things (iomt) - an overview," in *2020 5th International Conference on Devices, Circuits and Systems (ICDCS)*, 2020, pp. 101–104.
6. A. Razaque, F. Amsaad, M. Jaro Khan, S. Hariri, S. Chen, C. Siting, and X. Ji, "Survey: Cybersecurity vulnerabilities, attacks and solutions in the medical domain," *IEEE Access*, vol. 7, pp. 168 774–168 797, 2019.
7. A. Fatima, T. A. Khan, T. M. Abdellatif, S. Zulfiqar, M. Asif, W. Safi, H. A. Hamadi, and A. H. Al-Kassem, "Impact and research challenges of penetrating testing and vulnerability assessment on network threat," in *2023 International Conference on Business Analytics for Technology and Security (ICBATS)*, 2023, pp. 1–8.
8. P. Vats, M. Mandot, and A. Gosain, "A comprehensive literature review of penetration testing & its applications," in *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2020, pp. 674–680.
9. P. Nespoli, M. Albaladejo-González, J. A. Pastor Valera, J. A. Ruipérez-Valiente, and F. Gómez Mármol, "Capacidades avanzadas de simulación y evaluación con elementos de gamificación," in *VII Jornadas Nacionales de Investigación en Ciberseguridad (JNIC '22)*, 2022, pp. 55–62.
10. World Economic Forum, "Wef global cybersecurity outlook 2022," World Economic Forum, Tech. Rep., 2022. [Online]. Available: https://www3.weforum.org/docs/WEF_Global_Cybersecurity_Outlook_2022.pdf
11. P. Nespoli, D. Papamartzivanos, F. Gómez Mármol, and G. Kambourakis, "Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1361–1396, 2018.
12. Y. Stefinko, A. Piskozub, and R. Banakh, "Manual and automated penetration testing. benefits and drawbacks. modern tendency," in *2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*, 2016, pp. 488–491.
13. W. Xiong, E. Legrand, O. Åberg, and R. Lagerström, "Cyber security threat modeling based on the mitre enterprise att&ck matrix," *Software and Systems Modeling*, vol. 21, no. 1, pp. 157–177, Feb 2022.

14. T. Yadav and A. M. Rao, "Technical aspects of cyber kill chain," in *Security in Computing and Communications*, J. H. Abawajy, S. Mukherjea, S. M. Thampi, and A. Ruiz-Martínez, Eds. Cham: Springer International Publishing, 2015, pp. 438–452.
15. C. P. Castaño, "Hacktricks," https://book.hacktricks.xyz/welcome/readme, Accessed on April 26, 2023.
16. C. Richards, "Boyd's ooda loop," 2020.
17. G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Sunnyvale, CA, USA: Insecure, 2009.
18. D. Kennedy, J. O'Gorman, D. Kearns, and M. Aharoni, *Metasploit: The Penetration Tester's Guide*, 1st ed. USA: No Starch Press, 2011.
19. C. Polop, "Peass-ng," https://github.com/carlospolop/PEASS-ng, 2023.
20. M. Kurant, A. Markopoulou, and P. Thiran, "On the bias of bfs (breadth first search)," in *2010 22nd International Teletraffic Congress (lTC 22)*, 2010, pp. 1–8.
21. sshuttle, "Sshuttle," https://github.com/sshuttle/sshuttle, 2023.
22. R. Alford, D. Lawrence, and M. Kouremetis, "Caldera: A red-blue cyber operations automation platform," in *Proceedings of the 32nd International Conference on Automated Planning and Scheduling*, June 13-24 2022.