Department of Information Engineering and Computer Science

Bachelor's Degree in Computer Science

Final Dissertation

Comprehensive Analysis of Breach and Attack Simulation Tools

A Theoretical and Framework-Driven Assessment of Their Capabilities, Advantages, and Limitations

Supervisor
Prof. Silvio Ranise

Student
Matteo Bregola

Co-supervisors
Dott. Pietro De Matteis
Dott. Matteo Rizzi
Dr. Salvatore Manfredi

Academic year 2023/2024

Acknowledgements

I want to express my gratitude to Professor Silvio Ranise for giving me the opportunity to work in the cybersecurity department at FBK. The environment is thrilling, and people share valuable knowledge that has helped me on this project. Furthermore, I thank my co-supervisors Pietro, Matteo, and Salvatore, who provided constant support not only from a technical point of view but also from a human one. You can establish a welcoming atmosphere that makes people more confident and the results more effective. Finally, I want to leverage this occasion to thank my family for standing by me during this three-year journey. Mamma, Papà, and Anna, I achieved this accomplishment because you have always supported me regardless of the results. Knowing I always have a place where I feel at home gives me the strength and security to keep moving forward.

Dedicated to Maria and Deanna.

Contents

Abstract

Breach and Attack Simulation Tools have recently emerged as an effective approach to security testing aimed at assessing an organization's resilience by simulating cyberattacks. Despite several solutions being available, this technology lacks a standardized definition and a clear understanding of its functionalities, advantages, limitations and objectives. This thesis addresses these gaps by analyzing existing research and examining available *BAS Tools* to identify their core components and organize the knowledge. Additionally, a framework is proposed to outline potential features of a generic *Breach and Attack Simulation Tool*, providing a basis for evaluating and comparing different solutions. This framework offers a structured approach to analyze a *BAS Tool* and assess its efficacy, thereby assisting security experts in choosing the most appropriate tool for their needs by selecting among a set of core features.

1 Introduction

1.1 Motivation and Objective

The importance of cybersecurity is rapidly growing, transcending its traditional confines within the realm of computer science to assume a significant role in the organization of contemporary companies. Citing a recent report[1], it is discerned that the average financial repercussions on enterprises of data breaches have surged by 10% in the last year, culminating in a figure of 4.88 million USD in 2024. A forecast[2] suggests that the annual impact of cybercrime on the global economy is projected to reach 23.84 trillion dollars by 2027. As these attacks become more complex and dangerous [1], the demand for new cybersecurity solutions has grown more urgent, making the automation of security testing essential to address the rising complexity of systems and threats. *Breach and Attack Simulation Tools* (*BAS Tools*) have emerged as promising solutions to this critical need, offering automated testing processes that reduce manual effort. These tools aid organizations in fortifying their cyber defences by emulating adversarial tactics, enabling them to assess and mitigate potential security vulnerabilities effectively. The technology's value depends on combining its technical capabilities with the relevance of the knowledge about attacks and adversarial strategies underpinning it.
Gartner[3] reports that *BAS Tools* have passed during 2023, the "peak of inflated expectations". According to their interpretation, the technologies in this stage have received the maximum expectations in their life-cycle. However, these solutions are in a primary phase, in which the number of applications and the knowledge about them is lacking behind the hype. During an early study of *BAS Tools*, we encountered this "expected" scarcity of information relevant to the technology, especially from scientific research. To the best of the author's knowledge, there is no comprehensive survey of *BAS's* state of the art, and it is missing a clear definition of the technology in terms of capabilities, advantages, and limitations. Furthermore, the open source projects relative to *Breach and Attack Simulation* are few and seem to be far behind the commercial solutions. Moving from this knowledge gap, the main objective of this thesis is to provide an overview of *Breach and Attack Simulation Tools*, elucidating their goals, how they operate, the pros and cons, and comparing the technology with other solutions.

1.2 Structure of the Work

The thesis is structured in five chapters. First, we provide some background knowledge necessary to understand better some concepts used in this dissertation and to contextualize *BAS Tools* in the cybersecurity field. Then, we analyze the core aspects of the tools, studying the definitions, the advantages and limitations, and how a user interacts with this software; finally, we briefly compare *BAS Tools* with other solutions. Chapter 4 proposes the *BAS Tools* framework, exploring its objectives and design choices. In the following section, we delve into a case study of the framework and leverage this study to evaluate its functionalities and limitations. Ultimately, we review the whole work, drawing final considerations and exploring possible directions for further analysis.

[1] IBM, "Cost of a Data Breach Report 2024", https://www.ibm.com/reports/data-breach, last assessed on 11/08/2024.
[2] World Economic Forum, "2023 was a big year for cybercrime – here's how we can make our systems safer" https://www.weforum.org/agenda/2024/01/cybersecurity-cybercrime-system-safety/, last assessed on 11/08/2024.
[3] Gartner, "Hype Cycle for Security Operations, 2023", https://www.gartner.com/en/documents/4547399, last assessed on 22/08/2024.

2 Background

This chapter contains the background knowledge essential for this thesis. The concepts of *Techniques, Tactics, Procedures(TTPs)* and *Advanced Persistent Threats(APTs)* are clarified, followed by an excursus on cyber threat models and eventually a paragraph on security testing.

2.1 TTP

Analyzing and simulating cyber-attacks requires the usage of precise and shared terminology such as *TTP*. According to the National Institute of Standards and Technology (NIST)[1], *TTP* describes the "behavior of an actor" highlighting how the idiom allows for clearly defining the actions of a threat actor at a high level. The term has a military origin [32] and is an acronym for *tactics*, *techniques* and *procedures*. The NIST glossary explains how each word is used to categorize the descriptions of the actor's behaviour based on the amount of details given. Accordingly, a *tactic* is "the highest-level description of this behavior" whereas a *procedure* is a "highly detailed description in the context of a technique".

Maymi' et al. [32] propose a semantic definition of *TTP* accompanied by a logical representation using Directed Acyclic Graphs (DAGs). To provide their interpretation, they introduce two terms. The first is *task*, defined as a "clearly defined action or activity specifically assigned to an individual or organization". The second term is *primitive action*, described as a "single step in some decision-making or execution process" like a "single line of code or a push of a button". Following these notions, *Procedures* are characterized as an aggregation of *primitive actions* in which the order is fundamental, while *techniques* are "non-prescriptive ways or methods used to perform *tasks*" and *tactics* are instantiations of one or multiple *techniques* that interact with one or multiple resources (file, domains, credentials, etc.).

MITRE developed a threat model called ATT&CK [13, 44] to analyze cyber-attackers actions. At a high level, it places *tactics* which express "why of an ATT&CK technique or sub-technique" and embody the "tactical objective: the reason for performing an action". Each *tactic* has several *techniques* or *sub-techniques* that represents " 'how' an adversary achieves a tactical objective by performing an action". *Procedures* are not directly involved in the model, as the design paper of ATT&CK states that *procedures* "are the specific implementation adversaries have used for *techniques* or *sub-techniques*" [44]. To thoroughly understand the concept of *TTP*, we analyze an example from the framework:

Discovery is one of the fourteen *tactics* present in the current "Enterprise" edition. It encompasses the various techniques that aim to gain information about the system.

Account Discovery is a *technique* of the "Discovery" *tactic* that describes the objective of collecting metadata related to the accounts of the compromised infrastructure.

net user can be considered as a *procedure* for the "Account Discovery" *technique*, i.e., a PowerShell command that displays the user accounts on the computer.

2.2 APT

The term *APT*, an acronym for *Advanced Persistent Threat*, found its origin in the U.S. military, where it was employed to describe Chinese espionage groups [43]. Over the years, the expression has

[1] National Institute of Standards and Technology (NIST) Glossary, https://csrc.nist.rip/glossary/term/cybersecurity, last accessed 03/07/2024.

been widely adopted and is now commonly used within cybersecurity. NIST defines an *APT* as "an adversary with sophisticated levels of expertise and significant resources, allowing it through the use of multiple different attack vectors (e.g., cyber, physical, and deception), to generate opportunities to achieve its objectives [...]"[2]. Chan et al. [11] conducted a comprehensive study about APTs in which they adopted the previously mentioned definition, highlighting that APTs represent a new class of threats with four specific distinctions from the traditional ones for the following reasons:

Specific targets and clear objectives: the targets of these attackers are governments and high-level companies. Their objective is to gain *strategical*, *competitive* and *financial* assets. The APTs create dangers to the public by potentially damaging critical infrastructures, thereby causing physical harm and political problems, as reported by Selmanaj [43].

Highly organized and well-resourced attackers: these groups consist of several experienced hackers. They are well organized and often possess vast resources, not only from the economic point of view but also in terms of protection, as many of them are backed by national governments.

Long-term campaign with repeated attempts: attacks carried out by these groups may take years or be repeated several times over an extended period. One typical characteristic of APTs is the objective of maintaining access to the compromised system or creating a backdoor [43].

Stealthy and evasive attack techniques: the attacks aim to be as stealthy as possible, and many APTs consider undetectness an essential objective.

Attributing attacks to a specific APT, i.e. proving accountability, is a complex challenge that requires a shared effort between different entities in the cybersecurity community and implies taking into consideration several aspects like the techniques used or the skill level [43]. Despite these difficulties, it is possible to design an *attack model* common for all the *APTs* [11] or provide a further detailed model specific for a single *APT*, as shown by Korban et al. with APT3 [46]. Different cyber threat intelligence frameworks, reports, and forums collect information about APTs from various sources. The MITRE ATT&CK framework has a section called "Groups" in which it presents a list of 152 "teams" of attackers, many of whom are defined as APTs. For each group, MITRE offers different details like their origin, the associated groups, the aliases, the campaigns performed, the tactics, techniques, and software they have leveraged, and the CTI sources that have been analyzed to gather this knowledge. Analyzing a specific APT can provide valuable insights and ease the understanding of their capabilities and behaviour. For example, in March 2024, the U.S. Department of Justice (a subsection of the Office of Public Affairs) published an article about seven individuals associated with the Chinese government and identified them as members of the APT31 (also known as Zirconium [8, 17]) [35]. The U.S. state wants the people involved in this group as they have been charged with computer intrusion due to targeting people "closely associated with U.S. presidential campaigns and candidates" and "individuals in the international affairs community" [35, 8]. Delving deeper into their actions, we can discover that they used more than 20 *techniques* belonging to 10 different *tactics* and that they sent more than ten thousand malicious tracking emails [17, 35].

2.3 Cyber Threat Models

NIST defines a *threat* as "Any circumstance or event with the potential to adversely impact organizational operations[...], organizational assets, or individuals[...]" and as the "potential for a threat-source to successfully exploit a particular information system vulnerability" [3]. Threat modelling is the process of analyzing these threats in a logical way, considering both the offensive and defensive side, evaluating the environment, and the resources that are threatened and used [37].
As shown by Granata and Rak [27], the number of published papers on threat models is remarkable,

[2]National Institute of Standards and Technology (NIST) Glossary, https://csrc.nist.gov/glossary/term/apt. last accessed 01/07/2024.

[3]National Institute of Standards and Technology (NIST) Glossary, https://csrc. nist.gov/glossary/term/threat. last accessed 02/07/2024.

and there has been a positive trend in the last few years. Tatam et al. [30] performed a review of threat modelling, describing how they can be categorized by different features like the approach, the distinction between graphical and formal methods and manuals against automated. From the approach point of view, models can be divided into four classes: *risk-based*, *attacker*, *data-centric* and *system-centric*. As pointed out by Sada et al. [2], the models are difficult to compare because the "objectives and the scopes of the frameworks are different"; in fact, the "combination of threat models can leverage of each other's strengths to minimize weaknesses" [30].
Beyond these differences and besides the fact that each threat model can have dissimilar use cases or audiences, we found prevalent consensus in the academic cybersecurity community on the relevance of threat models in simulations. Xiong et al. [49] reported that threat models can be employed in conjunction with attack simulations to assess the system's security and to "provide probabilistic evaluations of security". Bakker [7] further emphasizes this idea of threat modelling as a precondition for building realistic emulations. He states that describing adversaries and their behaviours improves the simulation's structure, improving defenders' communication and understanding. Furthermore, Blake et al. [44] define the ATT&CK model as a "conceptual tool" that can be leveraged to "perform testing through red teaming or adversary emulation". These considerations highlight that *BAS Tools* provide a practical way to apply knowledge about cyberattacks in real-world scenarios, and the more accurate and organized this knowledge is, the more effective the tools will be. Therefore, developing an effective *BAS* solution requires a strong understanding of attackers, their strategies, and attack methods.
For these reasons, the following paragraphs will briefly introduce the threat models relevant to studying *Breach and Attack Simulations*.

2.3.1 Cyber Kill Chain (CKC)

The Cyber Kill Chain is a framework developed in 2011 by Lockheed Martin that describes at a high level the steps that an adversary should follow to reach their objectives [12, 23]. The framework identifies seven elements that characterize an attack:

Reconnaissance involves gathering information about the target. This step could be further subdivided into identification, selection and profiling [50].

Weaponization consists of creating the exploit by leveraging the information obtained in the previous phase [50].

Delivery implies delivering the weaponized element to the target.

Exploitation involves executing the delivered element on the victim.

Installation consists of installing the malicious software on the target.

Command and control involves establishing a command and control channel that permits remotely performing actions on the victim's assets.

Actions on objectives involves performing actions on the assets to fulfil the initial objectives.

The framework effectively models the possible behaviour of an attacker from the initial phase to the actions he performs on the compromised target. Nevertheless, as mentioned by Bakker [7] and Pauls [39], this framework lacks some details on actions accomplished by the attacker in the so-called *post-compromise* phase, that is after having reached control of the target system. For this reason, the Cyber Kill Chain model can be considered more adaptable to penetration testing than *BAS*.

2.3.2 MITRE ATT&CK

MITRE ATT&CK is a "curated knowledge base and model for cyber adversary behaviour, reflecting the various phases of an adversary's attack life-cycle and the platforms they are known to target" [44]. The definition emphasizes how the framework is not only a behavioural threat model but also aims to gather information about attackers.
From the modelling point of view, it has three components:

Tactics "denoting short-term, tactical adversary goals during an attack" [44].

Techniques "describing the means by which adversaries achieve tactical goals [44].

Sub-techniques "describing more specific means by which adversaries achieve tactical goals at a lower level than techniques" [44].

The relationship between these components can be visualized in the ATT&CK Matrix [14, 15], which helps understand how the abovementioned component can be used to analyze an attack vector. The current version covers three technology domains: enterprise, mobile, and ICS (industrial control systems). Each technology domain supports specific operating systems and software: for example, the enterprise matrix contains data about the following platforms and architectures: Windows, macOS, Linux, Azure AD, Office 365, Google Workspace, SaaS, IaaS, Network, and Containers. Figure 2.1 shows a part of the enterprise matrix, whereas the complete version comprises 14 tactics and 235 techniques.

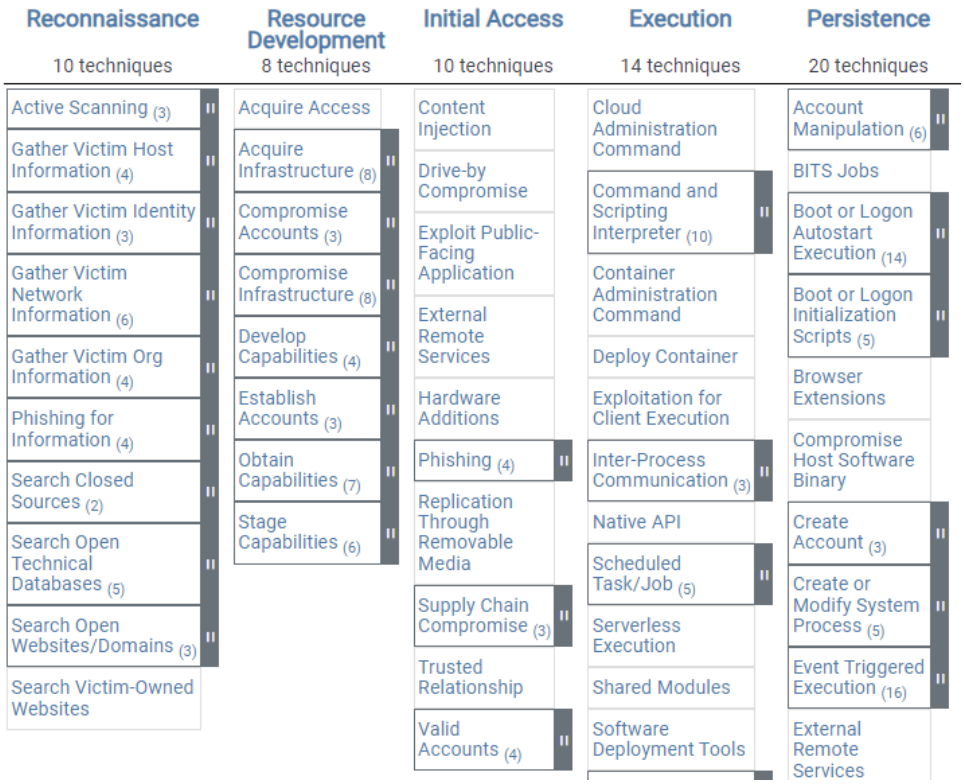

Figure 2.1: Partial Enterprise Matrix of MITRE ATT&CK.

Regarding the information provided by the framework, the ATT&CK website [14] documents real usage of the techniques, groups of attackers with associated data, mitigations related to specific technology domains, software used by attackers, and many others.
There are different reasons why ATT&CK is probably the best threat model to be used as a basis for developing a *Breach and Attack Simulation Tool*. Firstly, the model design is aligned with this objective since it was born "out of a need to systematically categorize adversary behaviour as part of conducting structured adversary emulation exercises" [44] and as a result, MITRE developed Caldera [16]. Furthermore, the framework is supported by a renowned corporation and is periodically updated by a dedicated team backed by a large community. Finally, different projects like Atomic Red Team [9] have been developed based on this framework and can be leveraged in new undertakings.

2.3.3 Unified Kill Chain (UKC)

Paul Pols developed the Unified Kill Chain in 2017 by extending CKC and MITRE ATT&CK with improvements proposed by other authors, applying the study of other models, APTs, and red team operations[4] [39]. The resulting unified model contains eighteen phases, which permit the modelling of

[4] A "Red Team" is a group of people authorized to simulate realistic cyberattacks against an organization to evaluate its security defences and provide insights for the defenders (Blue Team).

a broad range of attacks with high detail. The eighteen phases can be grouped as shown in Figure 1.2 to provide an abstract way to visualize an attack.

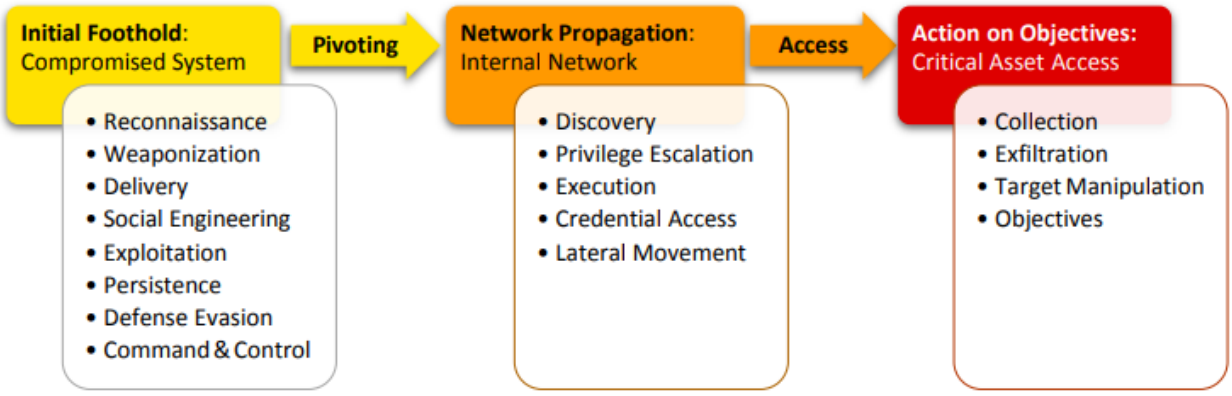

Figure 2.2: Unified Kill Chain's phases grouping.

As its creator pointed out, the fact that UKC is more exhaustive than other models could help red teams improve threat emulations. Nonetheless, to the author's knowledge, no *BAS Tool* is currently explicitly mapping their operations to the UKC.

2.4 Security Testing

Cybersecurity aims to defend systems and infrastructure from attackers, ensuring integrity, availability, authentication, confidentiality, and non-repudiation [36]. To uphold these principles, cybersecurity experts operate in a dual-mode manner: they apply security measures and evaluate the readiness of the infrastructure, trying to discover possible threats. The second element is formally known as "information security assessment". The NIST describes an *information security assessment* as the "process of determining how effectively an entity being assessed (e.g., host, system, network, procedure, person—known as the assessment object) meets specific security objectives" [38]. Moving from this definition, the NIST defines *security testing* as one of the three types of information assessment together with *examination* and *interviewing* [38]. The decision to separate the concept of *security testing* from *security examination* is not a shared approach in the community. For example, Felderer et al. [25] keep these two elements together and apply the former, implying both aspects. Beyond the different conceptual definitions, the practice of *security testing* is a standard operation in cybersecurity, and its objective of verifying and validating systems to grant the security of a system is not disputed. *Security testing* is often regarded as a "black art" [25] because, despite the existence of guides such as the "technical guide to information security testing and assessment" [38] and various standards, the task remains exceedingly complex. There is an abundance of methodologies of security testing that can be grouped based on the *objective*, the *scope*, the *accessibility*, the *phase of the software life-cycle* and many others [25]. Felderer et al. [25] propose four macro classifications of security testing techniques:

Model-based focuses on the requirements and the system's design, being particularly helpful in the planning phase of the software since it is aligned with the standard principle of "security by design".

Code-based testing and static analysis involves the static analysis of software (or infrastructure). The category includes techniques like vulnerability scanning for the code or the analysis of protocols.

Penetration testing and dynamic analysis focus on performing tests interacting with the system or running simulations on a digital twin in a cyber range.

Security regression testing testing phase performed during maintenance.

It is worth noting that this multitude of security testing techniques is coupled with a profusion of security testing tools that can be leveraged to accomplish the techniques previously mentioned. For example, Viegas et Kuyucu [48] provide a non-exhaustive list of 189 tools divided into 23 categories. These considerations permit us to understand better why security testing is a complex field and why deciding which methodology and tools to adopt can be daunting. To clarify how *BAS Tools* can be placed in this context and to summarize what has been mentioned in this paragraph, a conceptual visualization is proposed in Figure 2.3. The schema is based on the categories proposed by Felderer et al. [25], and the NIST definition of security assessment [38].

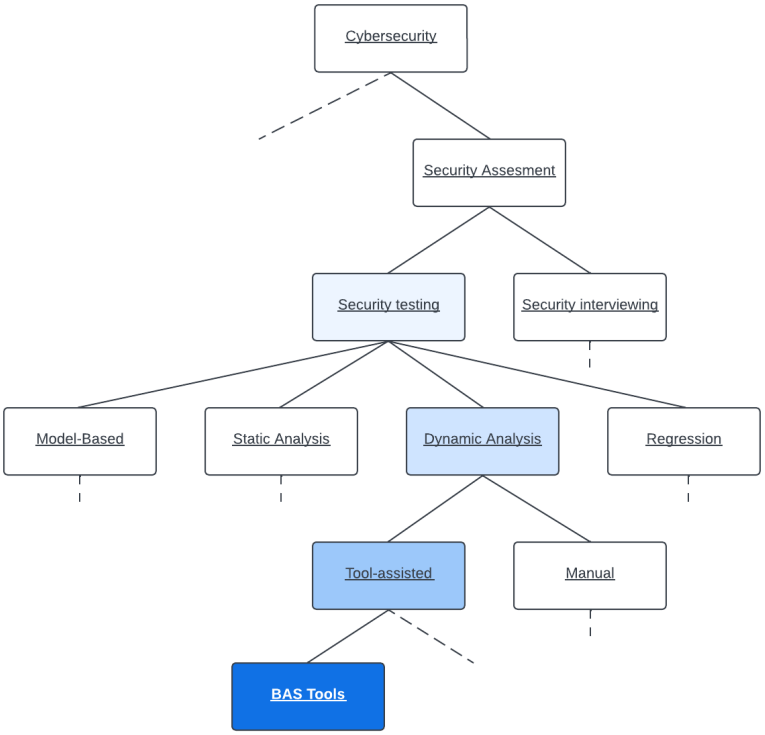

Figure 2.3: Abstract visualization of *Breach and Attack Simulation Tools* in the context of security assessment.

However, it is essential to note that this representation is arbitrary due to the absence of a universal definition of *security assessment* and many other categorizations are possible [43], including also non-cited elements like *risk assessment*.

3 BAS Tools Theory

3.1 Research Methodology

It was important during this research to have a schematic and reproducible methodology to gather information in order to have a complete and correct overview of the technology. The academic publications were sought using four search engines: "Google Scholar", "IEEE Xplore", "ACM Digital Library" and "Springer Link". For each of them, the research was done by restricting the year of release to the span 2004 to 2024 and using the following keywords[1]: " 'breach and attack simulation' ", "breach and attack simulation", " 'BAS tool' " and " 'BAS tools' ". The keywords and years were chosen due to a preliminary analysis of which terms and periods returned more satisfactory results. This methodology returned 98 papers, which were then analysed and categorised based on their relevance to this work. The information presented in this chapter results from the study of these articles, which is empowered by the study of the tools and other papers found while schematising the knowledge.

3.2 BAS Tool Definition

This paragraph examines the definition of *"Breach and Attack Simulation Tool"*. The first section presents characterizations from the academic literature retrieved with the mentioned methodology. The second section focuses on a subset of definitions from the vendors of *BAS Tools*. Finally, we draw considerations on the definitions listed and propose a comprehensive one.

3.2.1 Academic definitions

Kruck [31] describes *Breach and Attack Simulations*: "technologies that focus on repeatable, safe, and extensible cybersecurity exploitation to identify, quantify, and provide feedback on defences and defenders".

Horta et al. [28] state that *BAS* is an "advanced security testing method" that identifies "vulnerabilities in security environments by mimicking the likely attack patterns and techniques used by malicious actors".

Chen et al. [10] define *Breach and Attack Simulations* as an "advanced penetration method for automated evaluation of security situations" and then highlights the importance of integrating vulnerability correlation, multi-step attacks, and exploit chains in these tools.

Ryu et al. [40] analyse *BAS* by distinction from other simulators, noting how it is "capable of automating and implementing the attack scenario that the user has selected and implementing attacks to vulnerabilities, in practice, based, on the agent installed at the subject system".

3.2.2 Industry definitions

IBM [41] defines *Breach and Attack Simulation* as an "automated and continuous software-based approach to offensive security" which "complements more traditional security tools by simulating cyberattacks to test security controls and provide actionable insights". Furthermore, the article stresses that *Breach and Attack Simulations* "use the real-world attack tactics, techniques, and procedures (TTPs) employed by hackers to proactively identify and mitigate security vulnerabilities before they can be exploited by actual threat actors".

Picus Security [42] provides a definition related to *BAS Tools*: "A BAS tool is a cybersecurity solution designed to simulate real-world cyberattacks on an organization's infrastructure in a safe

[1]Keywords enclosed in single quotes (e.g., 'breach and attack simulation') were used to search for exact matches of those terms.

and controlled manner. By mimicking the tactics, techniques, and procedures (TTPs) used by potential attackers, it assesses the effectiveness of the organization's security measures, identifies vulnerabilities, and offers recommendations for improvement".

Cymulate [20] states that *BAS* is a "process that allows businesses to simulate cyber-attacks on their systems and networks to identify weaknesses in their security posture".

3.2.3 Considerations and definition proposal

Different papers emphasize how cyberattack simulation research is "fragmented and inconsistent" and lacks a "fully unified view" [22]. This element is particularly evident if we analyse the taxonomy: there are no shared definitions, and the use of language is ambiguous and inconsistent [22, 43, 31]. *Breach and Attack Simulation Tools* evidence this phenomenon; we searched the term in five cybersecurity glossaries[2] and none of them listed the definition of "Breach and Attack Simulation", "BAS" or "Breach and Attack Simulation Tool". The absence of a common characterization presents an opportunity for further research; however, we can attempt to delineate some underlying reasons.
The first justification is the relative novelty of the technology. MITRE CALDERA [16], which can be considered the open-source precursor of *BAS Tools*, was released in 2017 [45]. Correspondingly, many *BAS*'s sellers have started to commercialize their software between 2010 and 2020[3].
Another reason comes with the fact that scholarly research is lacking behind the introduction of new technologies in the market, as reported by Kruck [31]. Therefore, even if the idiom has nowadays been commonly adopted commercially, the absence of academic research makes it challenging to synthesize knowledge into a standard definition.
Finally, the presence of similar terms, often used interchangeably, poses an obstacle to delineating a definition. For example, *"Adversary Emulation"*, *"Adversary Simulation"*, and *"Automated Red Teaming"* are expressions adopted to describe analogous technologies.

Despite these considerations, the articles mentioned in the previous paragraph attempt to draw a general definition of *Breach and Attack Simulation*, whereas Picus[4] provides a definition specific to *BAS Tools*. These definitions present various key concepts about *Breach and Attack Simulation Tools*:

Automation: the core idea of this technology is to simplify the process of performing manual simulations, minimizing the effort required by the users. Software must imply some automation to be considered a *Breach and Attack Simulation Tool*; otherwise, it falls in the general red team application category.

Realistic simulation: a *Breach and Attack Simulation Tool* should be able to finalise accurate cyber-attack simulations that resemble real attacks accomplished by APTs and malicious actors. These simulations must be able to perform multiple steps and apply a progressive approach.

Defence-oriented: the tool aims to assess the security posture of the user's infrastructure, showcasing the weakness discovered during the simulation and providing mitigations. Furthermore, this reporting should respect the first principle of automation.

Continuous testing: a *Breach and Attack Simulation Tool* should be developed to enable continuous use and permit integration with the customer's workflow.

Reproducible operations: the simulations must follow a precise, sharable and reproducible schema. The results should have a standardized format.

Given these principles, we propose the following definition of a *Breach and Attack Simulation Tool*:

[2] A reference to the glossaries is reported in Appendix A.
[3] Esecurity Planet article, https://www.esecurityplanet.com/products/breach-and-attack-simulation-bas-vendors/, last accessed 06/07/2024.
[4] Picus Security Validation Platform: https://www.picussecurity.com/security-validation-platform.

a *"Breach and Attack Simulation Tool"* is a security testing software that enables simulating realistic attacks with an automation-oriented approach. The simulation must be standardized, reproducible, and allow for continuous testing. The tool aims to examine the system's security and produce results accordingly, outlining vulnerabilities and suggesting mitigations.

3.3 Advantages and Benefits of BAS Tools

The analysis of the definitions of *Breach and Attack Simulation* presented in the previous paragraph has brought to attention five key advantages of *BAS Tools*: *automation*, *realistic simulation*, *defence-oriented approach*, *continuous testing*, and *reproducible operations*. We will delve deeper into the benefits of the technology by emphasizing other boons and grouping them by category.

From an **economic** point of view, *BAS Tools* and similar technologies have been frequently cited as beneficial [31, 4, 3]. This software can enhance red teams' work and decrease the expertise required to perform simulations. These two features are accomplished using automation and offering "pre-defined" payloads and test suits. Before their introduction, red team engagements would spend a lot of time in configuring simulations and gathering information about common attacks and vulnerabilities. Reducing these hurdles makes operations easier and more effective, allowing for a smaller workforce or less specialized personnel, thereby reducing the overall costs of the red team [31, 4, 3]. The economic benefits don't involve only the cost associated with red teams; they encompass other aspects, such as reducing costs associated with security solutions. Companies commonly acquire different security solutions like *Endpoint Detection and Response*(EDR), *Anti-Virus*(AV), *Security Information and Event Management*(SIEM). The presence of multiple security software poses different difficulties, such as evaluating their performances, checking if they have been correctly configured, and analyzing whether they overlap to avoid using multiple solutions capable of performing the same actions. Many *BAS Tools* can assess the response of these defence systems by performing simulations and comparing the results of each solution [3]. This data can be analysed to determine which solution, among the adopted ones, has the highest return on investment (ROI) based on the user's requirements. Finally, the tool can improve the security posture, consequently reducing the impact of attacks and the associated costs. These considerations about the economic benefits of *BAS Tools* are based on the assumption that we are adopting a free solution, whereas, in the context of commercial solutions, we should also consider the price of the tool itself. The vendors claim high ROI [18] for technologies that include *BAS*; however, more independent studies should be carried out to provide a general assessment.
The second category of advantages is related to how *Breach and Attack simulation Tools* can be involved in **training** security teams [43, 21]. The first projects towards establishing the *BAS Tool* technology were commitments aimed at helping red teams. As the technology improved, the focus shifted to a purple team approach [43, 3] in which also the "defending" blue team is involved. Many *BAS Tools* have developed functionalities that streamline this process: they permit differentiated access and assign different capabilities based on the users' role. Furthermore, they simplify the communication between the two teams thanks to reports of attacks and results. The tool can, therefore, be used to simulate attacks to test the defenders' capabilities in case of crisis. This process increases the overall security posture, ensuring defenders can manage critical incidents [26]. Regarding the training advantages, it is also worth mentioning that analysing the network from an attacker's perspective can improve the defender's understanding of their system [33].
Another benefit of *BAS Tools* is the global improvement of the **work organization**. The ability to subdivide accesses based on the roles, schedule simulations, generate reports, and the possibility of integrating the technology with ticketing and communications platforms reduce the effort required to fulfil non-technical operations. Additionally, having identifiable and repeatable simulations permits evaluating the system's security over time and consequently the team's performance [26].
BAS Tools also represent the answer to the need for a shift in the **cybersecurity approach**. Traditional red team operations focus on performing penetration testing to evaluate and improve the network perimeter. *BAS* solutions focus on the more modern concept of assuming that a breach of

this perimeter will occur eventually [26]. *BAS Tools* extend the testing capabilities, allowing to investigate what an attacker could accomplish after acquiring access to the system. The potential of this technology lies mainly in this aspect; the data retrieved from these simulations represent empirical evidence that can be leveraged by blue teams [33].
Finally, some advantages are present only in some specific tools; for example, *BAS* can be adopted to test the system's compliance with security standards, as shown by Kruck [31].

3.4 Usage Methodology

The definition of *Breach and Attack Simulation Tool* presented in paragraph 3.2 and the list of its advantages in paragraph 3.3 delivered a high-level description of this technology and why it is used. In this section, we will analyse with more precision how a *BAS Tool* operates by exploring the simulation lifecycle from the initial planning phases to the final analysis of results.

The automated nature of *BAS tools* should not lead to the misconception that these solutions are ready-made and it is sufficient to execute simulations. Conversely, the initial stage of a *BAS Tool* engagement should involve organizing and planning the operations from a high-level perspective. Selmanaj [43] emphasises that proper pre-engagement planning, tailored for the specific organization structure and needs, is fundamental to exploit the full potential of adversary emulation. This consideration is also effective for *Breach and Attack Simulation*, and as evidence, many vendors leverage the support offered, especially during the initial setup, as a key selling point to buy their solution [19, 34]. The support does not exclusively involve configuring the solution from a technical point of view but also from a strategic one, analysing the simulations that are effective for the buyer's needs. The initial "pre-engagement" phase should cover different aspects of the simulation [43]:

Objectives: why should the simulation take place and what is the final goal. There is a substantial range of reasons behind a simulation, from testing readiness against a specific threat to personnel training.

Scope and Targets: which resources and assets of the organization will be involved in the simulation. There should be a clear indication of which elements of the IT infrastructure, what data, and who will be the target of the simulation. It is essential to carefully analyse these aspects to avoid confidentiality, integrity, or availability breaches.

Methodology: how the simulation will be performed and under which assumptions. This aspect also involves considering the approach and the knowledge of the testers. Some simulations may be performed with a black-box approach in which attackers have no internal information about the infrastructure, while other ones should adopt a white-box approach, offering the team to leverage all the internal sensible information.

Duration: when the simulation will start, the time limit, and possibly how often it should be repeated.

Rules of engagement: it completes the information offered by the scope by imposing explicit limits to the simulation regarding authorizations to perform specific actions on specific assets.

Communication plan: it decides who and how should be informed about the simulation.

Personnel involved: it delineates who will be involved in the simulation lifecycle, from the planning phase to the analysis of the results.

The last element is particularly relevant for *BAS Tools*. In the previous paragraphs, it was briefly mentioned that this technology is primarily used by red teams; however, the staff involved should be more heterogeneous. *BAS* solutions should be developed to offer a purple team functionality in which the simulation results are easily accessible to the defending blue team. This approach can also support the opposite sequence of actions: the blue team shares potential flaws with the red team in order to test their impact. Furthermore, the tool should facilitate communication with stakeholders

and managers who do not have technical knowledge. If the tool provides results accessible to non-experts, then the last phase of analysing results can engage more people.
Once the "pre-engagement" phase is accomplished, the tool's configuration phase will be the next step in the simulation's life cycle. This stage is highly solution-dependent, related to the tool's architecture and depends on the deployment methodology. *BAS Tools* can be different not only from a deployment point of view, ranging from on-premise solutions to Software as a Service (SaaS), but also for their target environment. Some tools can operate in production environments, while others must be adopted in an emulated infrastructure called "cyber-range". The choice of these two factors depends on the organization employing the tool. Several factors should be considered in this decision, such as security policies and the expertise of the tool's operators. Is worth mentioning that there is a substantial difference between tools developed for testing on real production environments and the ones that are thought for cyber-ranges. Running tests on production environments requires the simulation to respect strongly the scope and the rules of engagement to avoid unwanted harm to the organization. This approach usually implies having revertible simulations that may have the downside of being less effective and realistic. Among these different architectures, during our studies of *BAS Tools*, we have found out that the *agent-based* methodology is the most common (beyond the difference between production and simulated environment). This typology of architecture is presented at a high level in Figure 3.1. The *agent-based* architecture presents three main components:

Testers: personnel involved in the simulation interacting with the *BAS Tool*. The interaction between the tester and the tool depends on the specific functionalities of the tool, which usually offers a graphical user interface.

BAS Tool: is the centre of command of all the operations, typically installed directly on devices accessible to testers. The software can communicate with agents and vice-versa by leveraging a command and control channel. There are several ways in which this connection is established, for example, Caldera [16] requires running a payload on the agent system while other solutions require installing a software on the targets.

Agents: the various elements of the testers' infrastructure that will be the target of the simulation. The type of agents supported depends on the specific solution, and they can be endpoints, personal devices, servers, databases and many more.

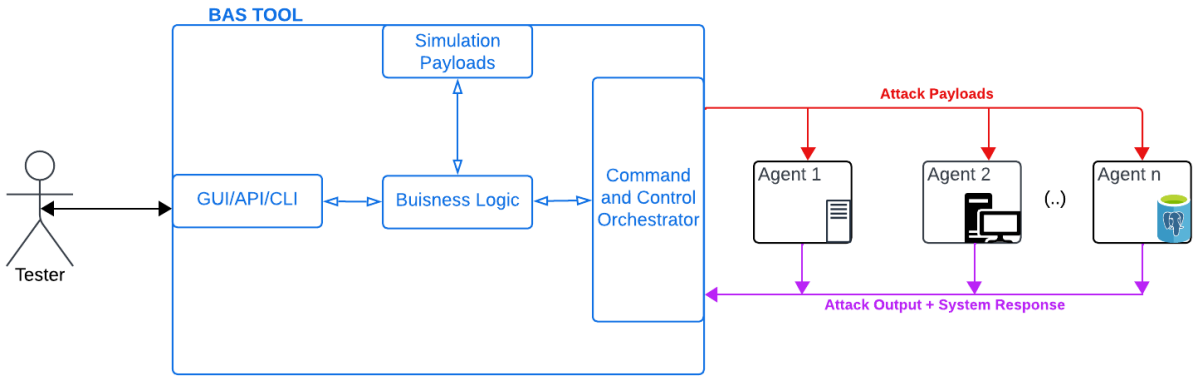

Figure 3.1: Abstract visualization of *Breach and Attack Simulation Tools* in the context of security assessment.

We can now leverage this architecture to analyse the generic operation workflow after the initial pre-engagement and setup phases. The first action is selecting the specific simulation that was decided upon in the planning phase. Some tools adopt a *white-box* approach in which the tester can read the entire simulation payload deployed on targets. Conversely, with a *black-box* approach, the user cannot

have access to the native code of the simulation. Once the simulation has been selected, the tester can configure different aspects of the simulation and provide additional information. The level and ease of this configuration depend on the tool; we are going to provide a non-exhaustive list of possible aspects that can be modified:

Logic of execution: how the simulation attack steps will be chained.

Obfuscation: if and how the messages between agent and tool will be encoded.

Output parsers: how the output of each step will be parsed and how this data will be leveraged in the next actions.

Jitter: how often should the agent and controller contact each other.

Timeout: maximum time of execution of each step.

Targets' information: additional information about the target that can be leveraged for a more effective attack (e.g., user privileges and credentials).

Specific parameters : parameters that should or could be specified for the simulation. For example, specific directories, files, or data to be exfiltrated or damaged.

When the simulation has been configured, the last step before executing it is choosing which pre-configured agents will be involved in the simulation. When all the setup is completed and the tester launches the operation, the tool starts the attack by sending payloads to the agents. The agents then return metadata to the tool; this data generally belongs to two possible categories, as shown in Figure 3.1: the output of the executed commands and the system response. The first category comprehends the actual "result" of the attack; for example, if the attack involves retrieving all the directories of the target, the output can be a list of discovered directories. The second category contains knowledge of how the system responded to the attack and can be reported in different ways and by different subjects. The system response can be analysed by sending the operating system's security logs or events generated by security solutions installed on the targets to the tool. In this case, the simulation may be structured in multiple steps; the tool sends payloads and retrieves the previously mentioned responses progressively. This approach is particularly effective because it permits the tool to analyse the knowledge gathered in a step (or by a specific agent) and re-use it in the following steps (or for other agents). During the simulation's execution, the tester may be able to see the status of the operations and partial results, stop the simulation or alter its pre-defined procedures.

Once the simulation has finished, the tool should organize the results and create a report of the attack performed and the relative system's response. During the study of the tools, we have noticed a wide variety of what is reported and how. In general, the tool should be able to report how effective the attack has been and whether the objectives of the simulation were reached. The tool should analyse the defences of the targets, possibly being able to distinguish the actions of each defence system. The results of each simulation should be standard, and the tool ought to store each result to analyse the security posture over time. Finally, aligning the tool's functionalities to the definition in 3.2 should provide mitigations to the vulnerabilities discovered during the testing.

3.5 Limitations

In this paragraph, we will analyse the limitations of a generic *BAS Tool* beyond the individual characteristics of specific solutions.

The first limitation can be found in the inherent restrictions of attack simulations compared to real attacks. Bakker [7] highlights that despite the effort put into building a realistic simulation, the final motivation of the red team will still be related to improving the defences, distinguishing the simulation from an authentic attacker. The article emphasises that simulations have limitations imposed by the testers' companies, such as time constrictions, resource constrictions, access constrictions, and, more

importantly, legal and ethical ones.

Another limitation lies in the "accessibility" of *BAS Tools*. Many authors underlined that these kinds of solutions are effective when the organization adopting them has a mature cybersecurity culture and practices [7, 43]. Selmanaj [43] provides a guideline of aspects to consider when evaluating a company's readiness for adversary emulation operations. These indications suggest evaluating the resources, analysing the current security practice and awareness, conducting an internal security audit, engaging stakeholders and elucidating the objectives. The author states that using these solutions without adequate evaluation and preparation may lead to ineffective results or, worse, damage to the organization. To showcase this statement, we take, for example, a startup with a restricted budget and no security software adopted. Buying a *BAS Tool* can be disruptive for two reasons. Firstly, the funds allocated to this solution can reduce the resources available for more critical needs during this phase of the organization, such as purchasing a Firewall software or an EDR. Secondly, suppose the personnel using the tool is not adequately trained. In that case, the tool's impact may be irrelevant or potentially damage the overall system's confidentiality, integrity, or availability.

This latter aspect of the tester's abilities permits highlighting other limitations of *BAS Tools*. We previously mentioned that one advantage of this technology was reducing the level of experience required by the users; however, this should not lead to considering these tools as a standalone plug-and-play addition. Different aspects of the simulation are not yet automated, which limits the tool's capabilities and creates the need for testers with good cybersecurity knowledge or who have been trained to use the software. For example, as mentioned in paragraph 3.4, choosing which simulations to perform against which target is a challenging task that can hardly be automated and still requires manual effort. Baiardi et al. [43] emphasize that another limitation of *BAS* solutions is their capabilities of analysing results, arguing that most tools "do not rank the vulnerabilities they discover according to their impact on robustness" and even if they try to provide a priority, the ranking is "not system dependent as it considers the intrinsic properties of a vulnerability and neglects the context of the vulnerability - i.e., the system where it appears". These two limitations have a common origin in the difficulty of automatizing environment awareness [6, 3]. This issue can be explained by analysing their relationship with ground security concepts. Both the previous limitations involved the notion of risk, in the former one we highlighted the need to choose simulations that pose a real risk for the organization that is using the tool while in the latter the need to rank discovered vulnerabilities based on risk. The risk is considered the product of the likelihood of an event times the impact of an event [5]. While the likelihood could be estimated through simulations, evaluating the impact involves considering a multitude of factors and depends on the different stakeholders, making its estimation a complete field of study on its own. This abundance of elements should then be converted and standardized in a machine-readable format to be used as input for the software. Therefore, we understand that the problem is duplex: it is difficult to make environment-aware decisions, and it is even more challenging to convert environmental aspects into input for software to make these decisions automatically.

Moreover, a partial shortcoming of *BAS Tools* is that they are based on existing data and discovered attacks. Although this approach assures that the attacks simulated are realistic, at the same time, they won't be able to find zero-day vulnerabilities and new threats [47, 29].

Finally, these technologies' capabilities are generally weak in the so-called "pre-compromise" phase [7]. This term describes the attacker's actions in the initial moments of an attack, where the malicious actors try to gather information about the victim and gain the initial foothold. One reason behind this deficiency is that a subset of these tools was designed with the assumption of breach and aim to test the defences from that moment forward[6] [5]. Bakker [7] suggest that another justification for this limitation comes from the challenges in this phase, both from a simulation point of view and a detection one. Furthermore, he hypothesises that this domain is lacking behind the others because it was

[5]NIST Glossary: https://csrc.nist.gov/glossary/term/risk

[6]Prelude Operator Documentation: https://docs.preludesecurity.com/v1/docs/basic

introduced later in the MITRE ATT&CK Framework. Among these considerations, the first one relative to the assumption of breach is arguably the most relevant. This idea of testing the organization's security response is one of the elements that permit distinguishing a *BAS Tool* from any penetration testing software. Penetration testing is a security testing methodology that focuses primarily on the "pre-compromise" domain and aims to circumvent the system's defences to access the organization's resources. This field of study is complex and highly specialized in the specific system that is the subject of study. Therefore, developing a *BAS Tools* with high "pre-compromise" capabilities would go beyond the general idea of *BAS Tool*, and it would compete with penetration testing platforms that experienced long development and research. Despite these considerations, a *BAS Tool* should develop abilities of this phase because they are essential to have an effective and realistic lateral movement during the simulation.

3.6 Terminology and Technology Comparison

One of the most difficult challenges encountered during this research was related to the terminology. As mentioned in the definition paragraph, this field of study lacks common terminology, and similar terms are often used interchangeably, making it a daunting task to understand and distinguish different technologies. To provide an idea of the magnitude of the challenge, we present a non-exhaustive list of terms that are related to *BAS Tools* and *Breach and Attack Simulation*:

- Automated Adversary Emulation
- Autonomous Adversary Emulation
- Adversary Simulation
- Automated Adversary Simulation
- Autonomous Adversary Simulation
- Automated Penetration Testing
- Automated Red Teaming
- Vulnerability Scanning

In Figure 3.2 we tried to show how these terms are somehow related by analysing the trend over the years of published research that contains a subset of these keywords. The results demonstrate that each term presents a similar increasing trend, indicating that, besides their differences, the research area partially overlaps.

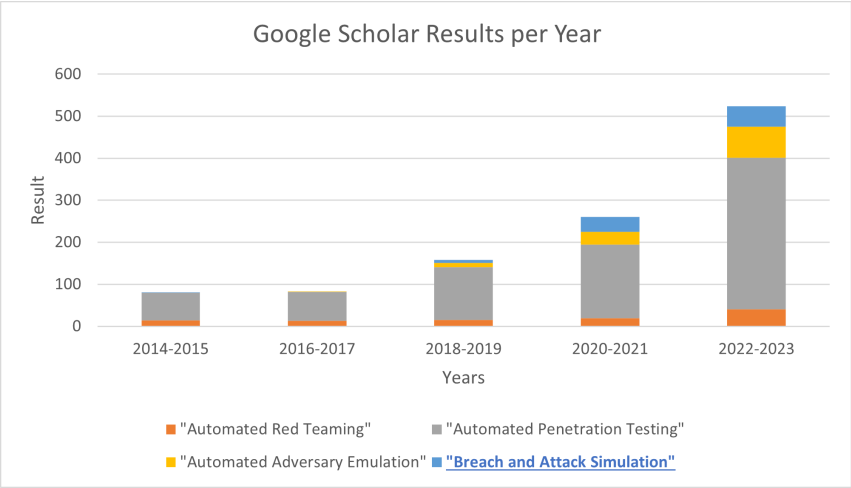

Figure 3.2: Research Result of terms related to *BAS* across the years.

Despite this complex situation, we will propose a differentiation of *BAS Tools* from the other terms. Before delving into this task, it is worth mentioning that these terms are generally used to describe the software associated with a specific methodology. For example, *Adversary Emulation* is

a specific approach and the software solutions that help perform it are called with different names such as *Automated Adversary Emulation*, *Autonomous Adversary Emulation*, *Adversary Emulation Platform*, etc. To gather all these variations, each of the following sections will use the methodology to convey all the associated ways to name the software.

3.6.1 Adversary Emulation and Adversary Simulation

Selmanaj [43] states that there is no agreed definition of *adversary emulation*, and the terms emulation and simulation are used interchangeably. MITRE [44] defines *adversary emulation (AE)* as "The process of assessing the security of a technology domain by applying cyber threat intelligence about specific adversaries and how they operate to emulate that threat". Besides the considerations that some authors made to distinguish emulation and simulation[7] [7] in this context, we are more interested in understanding the difference between *AE* solutions and *BAS Tools*. In general, an *adversary emulation software* is focused on simulating the actions of specific malicious threat actors and on the chaining of actions to generate an attack vector. *BAS Tools* can simulate real APT operations as *AE* but can also be exploited to test simpler abilities to check for specific vulnerabilities. Furthermore, the main difference between the two solutions is arguably their support for the blue team. While *AE* solutions commonly offer little to no support in analysing the results of an attack simulation, *BAS Tools* leverage this element as their key selling point. Accordingly, many commercial solutions provide an accurate study of the results and offer useful details to the blue team, such as the efficiency of the various software (EDR, AV, etc.). Finally, *BAS* products may provide suggestions and mitigations to improve the system's response against the corresponding simulation.

3.6.2 Penetration Testing

As explained in the limitation section (paragraph 3.5), *penetration testing* solutions primarily focus on the "pre-compromise" phase whereas, even if *BAS Tools* can support operations in this domain, their scope is broader and they analyse with more detail what can happen after the attacker has gained the initial foothold. For this reason, these two technologies can be considered complementary; a red team could move in parallel, testing both the external perimeter with penetration testing and the internal defences leveraging *BAS Tools*. Furthermore, *BAS Tools* are designed to be autonomous, enabling them to follow a continuous assessment approach, whereas penetration tests are typically conducted periodically. Finally, penetration testing aims to find and exploit vulnerabilities, while *BAS* focuses on validating the effectiveness of security measures and identifying gaps.

3.6.3 Red Teaming

A high number of cybersecurity solutions can be categorised as *red teaming software*. This typology of software contains functionality that can enhance and simplify the work of a red team. Furthermore, this category includes solutions that range from particular objectives, such as cracking passwords, to more general-purpose red teaming tools like Metasploit[8]. *BAS Tools* partially belong to this group, even if their capabilities go beyond the red teaming and can also be leveraged by the defence team.

3.6.4 Vulnerability Scanning

Vulnerability Scanning tools are a class of software used to scan to identify and report vulnerabilities within architecture, a network or application. These tools test systems by using a database of known vulnerabilities and performing specific queries to verify if the target configuration is vulnerable to some specific weaknesses. Many solutions empower this core idea with other functionalities, such as ranking the risk of the discovered vulnerabilities, evaluating the system's compliance with specific security standards, and providing mitigations to the detected weaknesses. Besides some exceptions, these tools can be categorized as static analysis security testing software, following the schema proposed in Figure 2.3. This permits highlighting the primary difference between this kind of solution and a *BAS* solution: the latter is based on a dynamic approach that tries to exploit the vulnerabilities that it discovers during the attack. Furthermore, *vulnerability scanners* have a broad analysis that generally returns independent results, whereas *BAS Tools* are more focused on discovering how to chain the vulnerabilities to get the maximum control over the victim.

[7] Redpoint Cyber: https://www.redpointcyber.com/emulation-vs-simulation/
[8] Metasploit: https://www.metasploit.com/

4 BAS Tools Analysis Framework

4.1 Motivation and Objectives

The study performed in the previous chapter delineated some issues related to *BAS Tools*'s literature. We emphasized that the technology lacks a unified view, comprehends solutions with different features, and employs diverse terminology. Furthermore, during the analysis of published research, we discovered that this subject has been little studied, and a schematization of knowledge is missing, as well as a complete guide to compare different solutions and analyze a specific one. To the best of the author's knowledge, no publication provides a detailed, schematic and repeatable procedure for studying a particular tool while simultaneously allowing for comparison with other tools. Some work focuses on comparing a specific characteristic among different solutions (e.g. Elgh [21]), while others provide general indications on comparison elements (e.g. Evangelakos [24]) or a brief description of tools. Moreover, some research try delineating some high-level elements to compare (e.g. Ferraz [3] and Bakker [7]); however, they are too general to be used to understand in detail the functionalities of a solution. The only known work that presents defined criteria for evaluating and comparing a tool is an unpublished paper by Zilberman et al. [51]. Although their research talks about "threat emulators", their work aligns with the abovementioned objectives and presents objective guidance on comparing different solutions. Despite its relevance and valuable contribution to this work, it presents some limitations. Firstly, their evaluation leverages MITRE ATT&CK as a reference to assess the number of attack techniques provided, thereby creating difficulties in analysing software that does not adopt it. Secondly, it does not consider some features present in some *BAS* solutions that are valuable and deserve attention. This shortcoming might result from the study's exclusive analysis of open-source solutions, ignoring potentially useful functionalities in commercial ones.

Given these considerations, throughout this chapter, we aim to delineate a comprehensive framework that includes distinct features and elements beneficial to diverse applications, namely:

- Understanding the core of *Breach and Attack Simulation Tools*, including their fundamental constituents and distinguishing characteristics.
- Analyzing the advantages and limitations of a specific solution.
- Supplying a basis for comparing and contrasting different solutions, facilitating informed decision-making.
- Providing an exhaustive list of features that can be leveraged to develop a new tool or enhance existing ones, thus encouraging innovation within the domain.

4.2 Research Methodology

The proposed framework results from a conjunct analysis of papers and tools. The documents examined have been searched with the methodology reported in section 3.1. The software selection for this analysis presented a significant challenge in maintaining objectivity and establishing a rigorous methodology. Given the many available options, the general criteria have been to analyze popular solutions with accessible documentation, demos or tutorials. Appendix B provides the tools analyzed, divided by their licensing model. Due to time and resource constraints, it was impossible to test these solutions, therefore the features presented in the framework are retrieved from the documentation of the software rather than their testing.

4.3 Framework Design

The proposed *Framework* consists of a list of 164 elements that are logically organized as shown in picture 4.1.These elements are initially categorized into two primary classifications: **Features** and **Comparison Elements**.

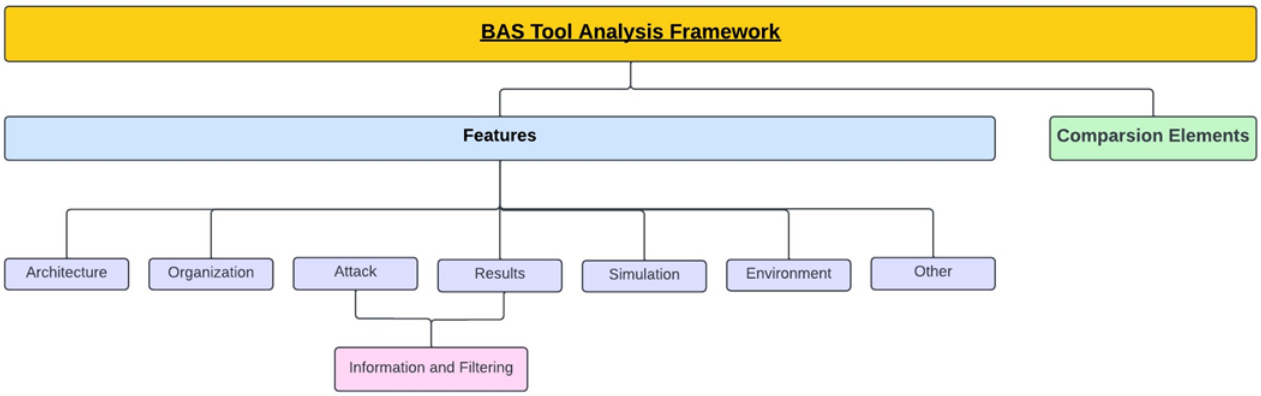

Figure 4.1: Framework Structure.

Features encapsulate the functionalities inherent within the tool, such as the graphical user interface. Elements falling under this category are systematically employed, delineating their presence or absence (e.g., presence of override functionality) or enumerating a list of attributes (e.g., supported operating systems). This section is delimited from qualitative analysis and instead emphasizes a quantitative examination, discerning the mere availability of features instead of their effectiveness. Moreover, features have been further categorized into seven groups for a more schematic overview and a faster comparison. The classes *Attack* and *Results* also have a common sub-section called *Information and Filtering*.

Comparison Elements are envisioned to facilitate quantitative and qualitative analyses when juxtaposing tools. Nevertheless, while certain elements within this category, such as price, are readily attainable, the majority pose challenges due to subjective variability, frequent fluctuations, or reliance on contextual factors. For instance, assessing the *Operator Expertise* presents inherent complexities, necessitating dedicated study for meaningful comparison. Despite these hurdles, these elements are valuable for making informed decisions about tool selection. Hence, they are included within the framework but need a separate section. The list of features that can be used to compare two software could be very detailed; this section focuses on analyzing relevant elements when comparing two different *BAS* solutions rather than a generic software.

The framework names the features in a "positive" and non-exclusive way, meaning that a tool could theoretically offer all the features[1]. Nevertheless, it is not intended to apply this study as a checklist but to analyze all the possible elements of *BAS Tools* and choose the subset aligned with the desired use case. Developing a tool with all the features would be challenging, but more importantly, it would be performance-inefficient, economically unsustainable and too complex to use. Another noteworthy consideration is that the features are delineated and enumerated from a client standpoint, highlighting the functionalities provided to the user. For this reason, when categorizing under the architecture section, the analysis of the pros and cons of various solutions pertains not to the benefits for the tool developers but to those for the tool purchaser.

Each one of the 164 elements of the *Framework* is characterized at a minimum by a name, a ref-

[1]Two features, respectively A9 *Deployment Platforms* and A10 *Target Platforms*, contains a list of elements instead of a "present" or "absent" value. The justification for this comes from the fact that there are too many operating systems and platforms, and creating a feature for each would be inefficient in terms of interface effectiveness.

erence, a description, and a list of benefits (see Figure 4.2). The list of benefits, called "advantages", explains why a feature could be helpful for the user, helping in the decision process.[2]. Furthermore, some features belonging to the *Architecture* category presents also drawbacks that can be used to understand better which architectural approach is better suited for the users' needs. The "disadvantages" section is not present in elements belonging to other categories because it would not add value to the framework. In many cases, the only downside in including a specific functionality in a tool would be having a worthless addition based on the reader's specific purposes. To illustrate this decision, we can take the *Automatic Report* feature as an example; it is clear that if a tool offers this service, it would be beneficial, but there are no apparent drawbacks of having it beyond having a feature that the client may not need. Finally, there could be a notes section in which other explanations are presented, as well as citations and other considerations

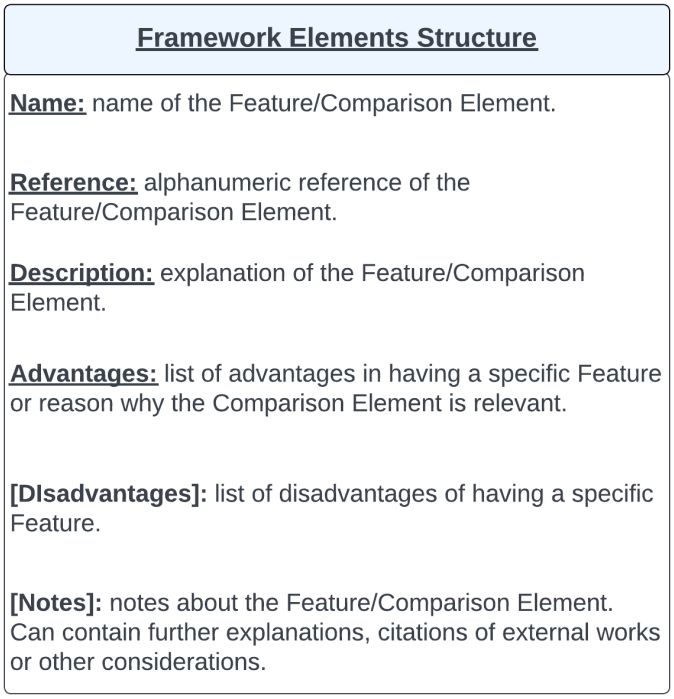

Figure 4.2: Framework Elements Structure.

4.4 Framework

In the following subsections, we analyze each category in more detail and specify which type of elements it contains. For readability, the description of each feature is presented as an external document on Github[3], divided by category.

4.4.1 Architecture

The *Architecture* category comprehends features describing the software technical components and the deployment methodology. This section is crucial for deciding whether a specific solution suits a particular use case. As previously mentioned, it is not generally recommended to exclusively verify how many features a tool offers but rather to study which ones align with the user's needs. This reasoning is particularly relevant to the features in this section. For instance, a tool that offers both software as a service and on-premise solutions in a single package can be unnecessarily costly if the buyer's system is not compatible with cloud environments.

[2]Since the features A9 and A10 comprehend a list of elements, there is no "advantages" section.
[3]https://github.com/risingfbk/BAS-Tools-Framework

4.4.2 Organization

The *Breach and Attack Simulation Tools* available on the market offer different simulation capabilities that can be distinguished not only by their objectives but also by how they are structured. Many solutions subdivide the general simulation concept into smaller elements that can be grouped or chained to perform a simulation. There are different possibilities for performing this subdivision; this framework provides a two-element subdivision compatible with the solutions adopted by the various tools. These two elements are named *Ability* and *Operation*. An *Ability* has a basic and straightforward objective, achievable by running a few commands; multiple abilities can be grouped into an *Operation* to reach a more complex goal [4]. A *Simulation* involves one or many operations. Therefore, the features of this section permit the reader to verify if the tool offers a logical structure similar to this, how much it is customizable, and the benefits of having multiple fine-grain elements rather than a whole "simulation object". It is essential to consider that it may not be possible to directly compare these features with the ones in *BAS* solutions due to a different simulation subdivision, and an adjustment may be needed.

To show the compatibility of the proposed solution with the tools, we can compare it with MITRE Caldera[5] and AttackIQ[6].
MITRE Caldera has three basic components: *Abilities*, *Adversaries* and *Operations*. Caldera defines an *Ability* as a "specific MITRE ATT&CK tactic/technique implementation which can be executed on running agents", an *Adversary* as "groups of abilities, representing the tactics, techniques, and procedures (TTPs) available to a threat actor" and *Operations* "run abilities on agent groups".
AttackIQ's *BAS* offers different *Assessments* that are composed of different *Tests*; each test can have multiple *Scenarios*.
Table 4.1 explains how each concept of the analyzed *BAS* solutions can be converted to our framework.

Caldera	Attack IQ	Framework
Ability	Scenario	Ability
Adversary	Test	Operation
Operation	Assessment	Simulation

Table 4.1: Tool to Framework terminology and logic conversion.

4.4.3 Attack

The features in this section aim to analyze the software's general attack capabilities. This list comprises thirteen features; five of them (C9 to C13) are related to the attack "configuration", meaning the selection of targets, activities, and their order. In particular, the features *Activity Filtering* and *Activities Combined Filtering* are the attributes used to verify if the tool permits the filtering of activities based on the elements described in the *Information and Filtering* subsection 4.4.5.

4.4.4 Results

This section outlines the features that describe the type of simulation results presented and how they are displayed. An effective tool offers a clear understanding of how the system responds to the simulation, which targets are more vulnerable and which activities represent the highest risk for the system. It is important to provide as much information as possible and permit the user to filter and visualize only the ones that he considers useful[7]. Another key aspect of *BAS* analyzed in this section is whether the tool offers users suggestions on how to fix vulnerabilities.

4.4.5 Information and Filtering

Information and Filtering is a subcategory of both *Attack* and *Results* categories. As described in section 3.2, the essence of *BAS* is to simulate the attacker's actions. To enhance the tool's capabilities, it is fundamental to provide the users with information about these actions, permitting operators to

[4]See the notes section of *Ability* and *Operation* features for a more detailed explanation.
[5]https://github.com/mitre/caldera
[6]https://www.attackiq.com/solutions/security-control-validation/
[7]The features related to filter capabilities are presented in the *Information and Filtering* subsection.

have a broader view of the simulation. In the *Organization* category 4.4.2, it has been presented the difference between *Action* and *Operation*; in this section, the term *Activity* will be used to comprehend both to simplify the descriptions.
The activity's information can be used principally in three different ways:

- To give more detail of the activity to the user,
- To filter activities before performing an attack simulations,
- To filter the simulation report to analyze the results in further detail.

Given these use cases, the framework permits, in this section, to check not only if the tool provides that specific information but also if it can be used to filter the activities or to sort the results. To perform this test, when it makes sense[8], the features are virtually arranged in groups of three:

- The first one, with a reference starting with "X", describes the specific information it is referring to.
- The second one, with reference starting with "Y", checks if it is possible to filter between all the activities offered by the tool based on the information described in the "X" feature and tests if this subset can be used to run a simulation. In particular, the *Activities Filtering* feature in the *Organization* section is used to verify if the tool offers this filtering capability.
- The third one, with reference starting with "Z", checks if it is possible to analyze the simulation results by filtering them based on the information described in "X" or to generate a report including only the activities that match that specific "X" feature. In particular, two features in the *Results* section, respectively, *Analytics* and *Result Filtering*, are used to verify if and how the tool offers this filtering capability.

The following example illustrates this reasoning:

- *Update* [X11]: The tool reports the last activity's update date.
- *Update Filtering* [Y11]: The tool permits filtering among all possible activities to only those updated within a certain time range.
- *Update Results* [Z11]: The tool permits filtering the simulation results based on the activities' last update to analyze the results only of that particular range.

Figure 4.3 shows the logical connection between the information and how it can be used to filter activities before launching a simulation or to sort out the results.

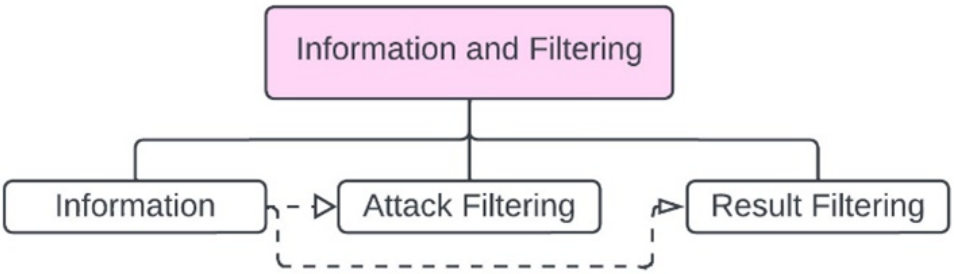

Figure 4.3: Logical relationships between features in *Information and Filtering* category.

[8]See Note 1 at the end of the section.

Note 1: Not all the "informative" (X) features have equivalent "filtering" (Y) and "results" (Z) features since they may be useless. If we take *Explanation* as an example of an "informative" feature, then the correspondent filtering ability could perform a text comparison that would probably lead to selecting activities that contain similar words rather than activities that are actually similar. A similar reasoning can be used to filter the results. To check rapidly if a feature "X" has the corresponding "Y" and "Z" features, it is sufficient to read the reference: related features have the same number after the letter.

Note 2: We have decided to separate the "filtering" and "results" features rather than combining them into a single category, allowing for a more fine-grained analysis. Moreover, suppose future work assigns an importance score to the features. In that case, filtering activities' usefulness differs from filtering results.

4.4.6 Simulation

The features in this section analyze the overall capabilities of managing simulations before, during, and after their execution.

4.4.7 Environment

This section aims to analyze whether and how the tool is integrated into the user's environment and adapted to its needs. A *BAS* solution is typically used by an organization's IT or cybersecurity team. Therefore, to streamline their work, the tool should be flexible not only from a technical standpoint, enabling interaction with other elements, but also from an organizational perspective, allowing use by different individuals with various roles.

4.4.8 Others

This section groups the features useful for analyzing the tool but not logically comprehended in the previous sections.

4.4.9 Comparison Elements

As explained in the Framework Design section 4.3, this category aims to enumerate the elements that can be used to compare different solutions. Unlike the previous features, these elements may be difficult to evaluate objectively, and developing a precise metric can be challenging in some cases. Finally, as noted in the introduction of this chapter, it is essential to remember that this section focuses specifically on features valuable for *BAS* comparison, omitting general comparison elements applicable to software in general.

5 Framework Application

5.1 Motivation and Objectives

In this chapter, we will accomplish a case study of our framework leveraging a tool called MITRE Caldera[1]. The primary motivation behind this analysis is to assess the framework's effectiveness in evaluating a *BAS Tool*, determining whether it can accurately identify its strengths and weaknesses, and ensuring that the proposed features adequately cover the tool's functionalities.
There is a threefold motivation behind choosing Caldera as a case study. Firstly, the tool is developed and supported by MITRE, one of the most relevant cybersecurity organizations, which started this project almost ten years ago [2]. Secondly, it is open source and offers valuable documentation[3]. Finally, it has become a reference for the field; it has more than five thousand stars on GitHub, and other tools (like OpenBAS[4]) use it as an addition. It is worth mentioning that the software is defined as an "Automated Adversary Emulation Platform"[5] rather than a *BAS Tool*. We do not consider this aspect a limitation of our testing because, as mentioned in sections 3.2.3 and 3.6, this field of study has ambiguous terminology and similar solutions often use different definitions. Additionally, a more detailed analysis of this platform can give an argument supporting or denying the difference between *AE* and *BAS* proposed in section 3.6.1.

5.2 Analysis Methodology

We have analysed MITRE Caldera, checking which framework's features it supports and excluding the *Comparison Elements* since their evaluation would have required a more complex study, as stated in paragraph 4.3. This examination was performed on release 5.0.0 by reading the corresponding documentation[6] and testing the software. Manual testing was performed by installing Caldera v5.0.0 from Github on a machine running Ubuntu v22.04.2 and using a Docker(v24.0.5) image with Ubuntu v22.04.2 to emulate one endpoint of the system.

5.3 Results

The complete case study's report can be found on Github[7]; for each feature, it documents whether the tool offers it or not (except for A9 and A10[8]). Furthermore, a feature subset includes a note section and a reference. The former supplements the feature's evaluation with details or elucidates how the tool offers functionality. The latter references the documentation about the specific feature or indicates how it can be exploited through the tool's GUI.
Table 5.1 shows the results of Caldera according to the framework, divided by category and represented as the percentage of features offered out of the total in each category.

The data indicates that the tool's performance differs greatly based on the category. It has an excellent result (90%) in the *Organization* category, although this outcome is partially biased since this category was developed taking as reference MITRE ATT&CK, and Caldera leverages it too. The tool performs well in the *Attack* and *Simulation* categories (69% and 71%), reflecting the software's

[1] https://github.com/mitre/caldera
[2] Caldera Blog on Medium: https://medium.com/@mitrecaldera/welcome-to-the-official-mitre-caldera-blog-page-f34c2cdfef09
[3] Caldera Documentation, https://caldera.readthedocs.io/en/latest
[4] Filigran OpenBas, https://filigran.io/solutions/open-bas/
[5] Caldera Website, https://caldera.mitre.org/
[6] Caldera Documentation, https://caldera.readthedocs.io/en/5.0.0/
[7] Framework Repository, https://github.com/risingfbk/BAS-Tools-Framework
[8] See note 2 of section 4.3

Category Name	Value
Architecture	54%
Organization	90%
Attack	69%
Results	28%
Information and Filtering	31%
Simulation	71%
Environment	40%
Other	41%
Overall	**43%**

Table 5.1: MITRE Caldera Results.

satisfactory capabilities in helping red team operations. More than half of the features presented in *Architecture* are available, whereas the categories *Other and Environment* reveal results slightly below the passing mark. Finally, the tool has a poor score on *Results* and *Information and Filtering* categories (respectively 28%, 31%). The unsatisfactory performance on the *Result* category was expected; this solution was born to help the red team's operations and is considered an *Adversary Emulation Platform*. For these reasons, its primary focus lies on the attack, while the support offered in analyzing the results and providing mitigations lags behind commercial *BAS* solutions. The sub-optimal outcome in the *Information and Filtering* category is because the tool's filtering is based on MITRE ATT&CK's, which permits filtering based on *tactic*, *technique* and *platform* but leaves out other valuable details like the APT correlation or the last update time. Furthermore, the reports are available in PDF, JSON, and CSV formats, but there are limited capabilities for analysing the results directly from the platform.
Overall, the tool offered 63 of the 145 analysed features, which can be considered a decedent result considering the intrinsic limitations of applying a Framework designed for *BAS* solutions to an *Adversary Emulation* software. The tool's performance was notably affected by lower evaluations in categories that distinguish *AE Tools* from *BAS Tools*, which significantly impacted the final results.

5.4 Considerations

The case study has shown that the proposed Framework effectively highlights a tool's strengths and weaknesses. This capability is achieved thanks to different categories, which help focus on specific aspects of the software rather than considering it as a monolithic element. Furthermore, the examination has shown that the Framework's features are sufficient to encompass the software's main characteristics while excluding the analysis of the more technical aspects. For example, the Framework permits the verification that Caldera leverages an *agent-based* approach but omits scrutiny on how this agent is implemented and which type of technology is used to establish the connection. Although including these kinds of details may be helpful, the downside would be increasing the Framework's dimensions, risking making it unusable. This compromise between the Framework's usability and level of detail can be considered the first limitation discovered during the case study. Another partial constraint can be found in the evaluation's objectivity. Sometimes, it has been difficult to state the feature as "present" or "absent" since intermediate situations require more "expressive" values to avoid a subjective evaluation. As an illustration, we can take the feature *Activities Filtering (C9)*, which should be considered "present" if the user can filter the abilities to include in the simulation. Caldera permits searching abilities based on a filter but does not automatically permit the selection of all the ones that respect the filter; however, it has been evaluated as "present" even if it only partially respects the feature's description. For this issue, some features (17 over the total of 145 analysed) have a graphic mark with a note explaining why the evaluation returned a partial outcome. A possible solution to reduce this matter is shifting from this evaluation's methodology to a feature-specific one in which each feature has its assessment values. This approach has been used by Zilberman et al. [51] with good results; however, it may be too complex to apply it with a Framework of this dimension. Their

solution is more efficient for comparing different solutions but loses accessibility. With our approach, a client can easily decide on a list of features they need and check what the tool offers them, while with their methodology, this operation would be less straightforward. The last improvement suggested by this case study is the weighting of the features to give more helpful scores. Caldera's results were highly influenced by the performance on *Results* and *Information and Filtering* categories. Still, the latter contains features that are generally less relevant than the ones in other categories, and additionally, the features in this category are not entirely independent[9]. Deciding on an importance scale and providing value to each feature may compensate for these issues, generating more correct results. However, it is worth mentioning that this would introduce a subjective metric since a feature has no intrinsic value; rather, it can be found in the user's needs. Furthermore, the score should not be used to compare two solutions directly; it may be an interesting key performance indicator, but this Framework was not developed to rank tools.
Finally, the case study has proven that the differentiation between *Adversary Emulation* and *Breach and Attack Simulation Tools*, as proposed in section 3.6.1, is effective.

[9]Remember the relationships between features "X", "Y", and "Z" presented in subsection 4.4.5.

6 Conclusion and Future Work

6.1 Summary and Conclusions

In this thesis, we tried to remedy the lack of a comprehensive analysis of *Breach and Attack Simulation Tools* in the academic literature. This objective was fulfilled by clearly defining the technology in chapter 3, which leveraged other publications and the addition of our contribution resulting from an in-depth study of the tools. This analysis comprehended a specification about the definition of *BAS Tool*, its advantages and limitations, an overview of their functionalities and usage, and a comparison with other technologies. This work has then raised the need to create a framework capable of highlighting the functionalities of a *BAS Tool*, analyzing its strengths and weaknesses and providing a basis for comparing solutions. Subsequently, the framework has been tested with a case study to assess its capabilities and indicate how it can be exploited. The conjunction of the theory chapter and the proposed framework constitutes a knowledge base for understanding the technology and the solutions available.

6.2 Future Work

This thesis aimed to contribute to gathering information about an ambiguously-defined technology with scarce publications. To follow this goal, we realized that it could be helpful to make this content more accessible. Therefore, we developed a website[1] containing some of the information presented in this work with enhanced features like a "finder" functionality that permits interacting with the framework more dynamically to choose the tool. This site poses a basis for some interesting future works; for example, the number of case studies could be increased to evaluate the framework's effectiveness in comparing solutions and to establish a public repository that can assist in selecting the tool best suited to individual needs. Other exciting directions for future works would be developing a weight system for the framework's features or moving toward a multi-value evaluation as proposed in section 5.4. Finally, a deeper analysis of some aspects of *BAS Tools* would be appealing, like analyzing their economic impact, examining artificial intelligence applications, or studying which simulations to prioritize.

[1]The Beta is currently available at: https://rising.fbk.eu/BASTools/.

Bibliography

[1] Ahmad Al-Hawamleh. Predictions of cybersecurity experts on future cyber-attacks and related cybersecurity measures. *International Journal of Advanced Computer Science and Applications*, 14:2023, 02 2023.

[2] Bader Al-Sada, Alireza Sadighian, and Gabriele Oligeri. MITRE ATTCK: State of the art and way forward. *ACM Comput. Surv.*, aug 2024.

[3] Ferraz Tomás Almeida. Breach and attack simulator. Master's thesis, Universidade de Coimbra, 2022.

[4] Andy Applebaum, Doug Miller, Blake Strom, Henry Foster, and Cody Thomas. Analysis of automated adversary emulation techniques. In *Proceedings of the Summer Simulation Multi-Conference*. Society for Computer Simulation International, 2017.

[5] Andy Applebaum, Doug Miller, Blake Strom, Chris Korban, and Ross Wolf. Intelligent, automated red team emulation. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, page 363–373. Association for Computing Machinery, 2016.

[6] Fabrizio Baiardi. Avoiding the weaknesses of a penetration test. *Computer Fraud & Security*, 2019(4):11-15, 2019.

[7] D.R. Bakker. Autonomous emulation of adversary procedures in the (pre-)compromise domain. Master's thesis, University of Twente, 2022.

[8] Tom Burt. New cyberattacks targeting U.S. elections. Microsoft Blog, `https://blogs.microsoft.com/on-theissues/2020/09/10/cyberattacks-us-elections-trump-biden/`. last accessed 01/07/2024.

[9] Red Canary. Atomic red team. `https://atomicredteam.io/`. last accessed 02/07/2024.

[10] Junhan Chen, Rufeng Liang, Man Zhang, Chengcong Zheng, Xun Huang, Hui Lu, Xiang Yu, and Zhihong Tian. Vulnerability correlation, multi-step attack and exploit chain in breach and attack simulation. In *2023 IEEE 12th International Conference on Cloud Networking (CloudNet)*, 2023.

[11] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In Bart De Decker and André Zúquete, editors, *Communications and Multimedia Security*, pages 63–72. Springer Berlin Heidelberg, 2014.

[12] Lockheed Martin Corporation. Cyber kill chain. `https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html`. last accessed 02/07/2024.

[13] The Mitre Corporation. Mitre att&ck. `https://attack.mitre.org/`. last accessed 01/07/2024.

[14] The Mitre Corporation. Mitre att&ck enterpise matrix. `https://attack.mitre.org/`. last accessed 02/07/2024.

[15] The Mitre Corporation. Mitre att&ck enterpise matrix navigator. `https://mitre-attack.github.io/attack-navigator/`. last accessed 02/07/2024.

[16] The Mitre Corporation. Mitre caldera. https://caldera.mitre.org/. last accessed 02/07/2024.

[17] The Mitre Corporation. Mitre zirconium. https://attack.mitre.org/groups/G0128/. last accessed 01/07/2024.

[18] XM Cyber. 2022 total economic impact study. https://info.xmcyber.com/total-economic-impact-study. last accessed 10/07/2024.

[19] XM Cyber. Technical support. https://xmcyber.com/support/. last accessed 14/07/2024.

[20] Cymulate. Breach and attack simulation (bas). https://cymulate.com/breach-and-attack-simulation/. last accessed 06/07/2024.

[21] Joakim Elgh. Comparison of adversary emulation tools for reproducing behavior in cyber attacks. Master's thesis, Linköping Universitya, 2022.

[22] Viktor Engström and Robert Lagerström. Two decades of cyberattack simulations: A systematic literature review. *Computers & Security*, 116, 2022.

[23] Rohan M. Amin Eric M. Hutchins, Michael J. Cloppert†. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Lockheed Martin Corporation*, 2017.

[24] Gus Evangelakos. Keeping critical assets safe when teleworking is the new norm. *Network Security*, 2020(6):11–14, 2020.

[25] Michael Felderer, Matthias Büchler, Martin Johns, Achim D. Brucker, Ruth Breu, and Alexander Pretschner. Chapter one - security testing: A survey. volume 101, pages 1–51. Elsevier, 2016.

[26] Daniel Goldberg. Living with decade-old vulnerabilities in datacentre software. *Network Security*, 2019, 2019.

[27] Rak M. Granata D. Systematic analysis of automated threat modelling techniques: Comparison of open-source tools. *Software Qual J*, 32:125–161, 2024.

[28] Antonio Jose Horta Neto, Raimir Filho, and Renato Marinho. A multi-criteria approach to improve the cyber security visibility through breach attack simulations. pages 330-343, 09 2022.

[29] Aws Jaber and Lothar Fritsch. Towards ai-powered cybersecurity attack modeling with simulation tools: Review of attack simulators. In Leonard Barolli, editor, *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 249–257, 2023.

[30] Matt Tatam Bharanidharan Shanmugam Sami Azam Krishnan Kannoorpatti. A review of threat modelling approaches for apt-style attacks. *Heliyon*, 7, 2021.

[31] Gregory P. Kruck. *Combating Non-Compliance: Leveraging Breach & Attack Simulation Techniques to Continuously Validate Information Assurance Controls*. PhD thesis, Marymount University, 2023.

[32] Fernando Maymí, Robert Bixler, Randolph Jones, and Scott Lathrop. Towards a definition of cyberspace tactics, techniques and procedures. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 4674–4679, 2017.

[33] Doug Miller, Ron Alford, Andy Applebaum, Henry Foster, Caleb Little, and Blake E. Strom. Automated adversary emulation: A case for planning and acting with unknowns. Mitre Corporation, 2018.

[34] NetSPI. Netspi breach and attack simulation (bas). https://www.netspi.com/breach-and-attack-simulation/. last accessed 14/07/2024.

[35] U.S. Office of Public Affairs. Seven hackers associated with chinese government charged with computer intrusions targeting perceived critics of china and u.s. businesses and politicians. https://www.justice.gov/opa/pr/seven-hackers-associated-chinese-government-charged-computer-intrusions-targeting-perceived. last accessed 01/07/2024.

[36] National Institute of Standards and Technology (NIST). Nist glossary: Cybersecurity. https://csrc.nist.rip/glossary/term/cybersecurity. last accessed 03/07/2024.

[37] National Institute of Standards and Technology (NIST). Nist glossary: Threat. https://csrc.nist.gov/glossary/term/threat. last accessed 02/07/2024.

[38] Karen Scarfone Murugiah Souppaya Amanda Cody Angela Orebaugh. Technical guide to information security testing and assessment. *NIST Special Publication 800-115*, 2017.

[39] Paul Pols. The unified kill chain. https://www.unifiedkillchain.com/assets/The-Unified-Kill-Chain-Thesis.pdf, 2017.

[40] Sungwook Ryu, Jinsu Kim, Namje Park, and Yongseok Seo. Preemptive prediction-based automated cyberattack framework modeling. *Symmetry*, 13(5), 2021.

[41] Josh Schneider. What are breach and attack simulations? IBM blog: https://www.ibm.com/blog/breach-attack-simulation/, 2024. last accessed 06/07/2024.

[42] Picus Security. Bas tools. https://www.picussecurity.com/resource/glossary/what-are-bas-tools, 2023. last accessed 06/07/2024.

[43] Drinor Selmanaj. *Adversary Emulation with MITRE ATT&CK*. O'Reilly Media, Inc., 2024.

[44] Blake E. Strom, Andy Applebaum, Doug P. Miller, Kathryn C. Nickels, Adam G. Pennington, and Cody B. Thomas. MITRE ATT&CK®: Design and philosophy. *IEEE Trans. Inf. Theory*, 2020.

[45] MITRE Caldera team. Welcome to the official mitre caldera™ blog page! https://medium.com/@mitrecaldera/welcome-to-the-official-mitre-caldera-blog-page-f34c2cdfef09, 2022. last accessed 06/07/2024.

[46] Christopher A. Korban Douglas P. Miller Adam Pennington Cody B. Thomas. Apt3 adversary emulatio. Mitre corporation whitepaper, 2017.

[47] Matthew J. Turner, Erik Hemberg, and Una-May O'Reilly. Analyzing multi-agent reinforcement learning and coevolution in cybersecurity. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 1290–1298. Association for Computing Machinery, 2022.

[48] Virgilio Viegas and Oben Kuyucu. *Security Testing and Attack Simulation Tools*, pages 263-290. Apress, 2022.

[49] Wenjun Xiong and Robert Lagerström. Threat modeling – a systematic literature review. *Computers & Security*, 84:53-69, 2019.

[50] Tarun Yadav and Arvind Mallari Rao. Technical aspects of cyber kill chain. In *Security in Computing and Communications*, pages 438-452. Springer International Publishing, 2015.

[51] Polina Zilberman, Rami Puzis, Sunders Bruskin, Shai Shwarz, and Yuval Elovici. Sok: A survey of open-source threat emulators. arXiv: https://arxiv.org/abs/2003.01518, 2020.

Appendix A Cybersecurity Glossaries

Organization/Company Name	Reference
National Institute of Standards and Technology (NIST)	https://csrc.nist.gov/glossary
SANS Institute	https://www.sans.org/security-resources/glossary-of-terms/
National Initiative for Cybersecurity Careers and Studies (NICCS)	https://niccs.cisa.gov/cybersecurity-career-resources/vocabulary
Information Systems Audit and Control Association (ISACA)	https://www.isaca.org/resources/glossary
Cisco Systems Inc.	https://www.cisco.com/c/en/us/td/docs/security/glossary/b_security_glossary_terms.html

Appendix B Software Analysed

B.1 Open Source Solutions

Tool Name	Reference
APTSimulator	https://github.com/NextronSystems/APTSimulator
Atomic Chain Reactor	https://github.com/redcanaryco/chain-reactor/wiki/Defining-custom-reactions
Atomic Test Harness	https://github.com/redcanaryco/AtomicTestHarnesses
ATTPwn	https://github.com/Telefonica/ATTPwn
BadBlood	https://github.com/davidprowe/BadBlood
Caldera	https://github.com/mitre/caldera
Cascade-server	https://github.com/mitre/cascade-server
Firedrill	https://github.com/FourCoreLabs/firedrill
Flightsim	https://github.com/alphasoc/flightsim
Infection Monkey	https://github.com/guardicore/monkey
Invoke Atomic	https://github.com/redcanaryco/invoke-atomicredteam
Leonidas	https://github.com/WithSecureLabs/leonidas
Metta	https://github.com/uber-common/metta
OpenBAS	https://github.com/OpenBAS-Platform/openbas
Pacu	https://github.com/RhinoSecurityLabs/pacu
PowerSploit	https://github.com/PowerShellMafia/PowerSploit
Prelude Operator	https://www.preludesecurity.com/products/operator
PurpleSharp	https://github.com/mvelazc0/PurpleSharp
Social Engineer Toolkit (SET)	https://github.com/trustedsec/social-engineer-toolkit
Stratus Red Team	https://stratus-red-team.cloud/
The Browser Exploitation Framework (beEF)	https://github.com/beefproject/beef
Vectr	https://github.com/SecurityRiskAdvisors/VECTR
Zed Attack Proxy (ZAP)	https://github.com/zaproxy/zaproxy

B.2 Commercial Solutions

Tool Name	Reference
AttackIQ	https://www.attackiq.com/
Cymulate	https://cymulate.com/breach-and-attack-simulation/
NetSPI	https://www.netspi.com/breach-and-attack-simulation/
Picus Security	https://www.picussecurity.com/
SafeBreach	https://www.safebreach.com/breach-and-attack-simulation-platform/
XM Cyber	https://xmcyber.com/