SoK: A Survey of Open Source Threat Emulators

Sunders Bruskin[1] and Polina Zilberman[1] and Rami Puzis[1] and Shay Shwarz[1]

***Abstract*— Threat emulators are tools or sets of scripts that emulate cyber-attacks or malicious behavior. Specifically, threat emulators can launch single procedure attacks or give one the ability to create multi-step attacks, while the resulting attacks may be known or unknown cyber-attacks. The motivations for using threat emulators are various: cutting costs of penetration testing activities by having smaller red teams, performing automated security audits in organizations, creating baseline tests for security tools in development, supplying penetration testers with another tool in their arsenal, etc.**

In this paper, we review various open-source threat emulators and perform qualitative and quantitative comparison between them. We focus on tactics and techniques from MITRE ATT&CK matrix, and check if they can be performed and tested with the reviewed emulators. We develop a comprehensive methodology for the evaluation and comparison of threat emulators with respect to general features such as prerequisites, attack manipulation, clean up and more. Finally, we present a discussion on the circumstances in which one threat emulator should be preferred over another.

This survey can help security teams, security developers and product deployment teams to examine their network environment or their product with the most suitable threat emulator. Using the provided guidelines, a team can choose the best threat emulator for their needs without checking and trying them all. To the best of our knowledge, no academic comparison was made between threat emulators.

I. Introduction

It is essential for IT security professionals to find frailties in security systems before cyber-criminals seek to exploit them. Red teams are groups of white-hat hackers that perform penetration testing by assuming an adversarial role. In addition to finding un-patched vulnerabilities one of the most important objectives of a red team is evaluating the organization's security readiness, active controls, and countermeasures by emulating a full attack cycle. Extensive what-if analysis is necessary to evaluate organization's response to an attack that penetrated its premises [32], [27], [4], [34], [33], [24].

Threat emulation platforms, such as CALDERA [9], Red-Team Automation (RTA) [21], or Advanced Persistent Threat Simulator (APTSimulator) [38], significantly speed up and simplify the what-if analysis in diverse scenarios. Threat emulators also make it easier for IT professionals who are not qualified red team practitioners to test organizations security controls and countermeasures. *Using a good threat emulator it is easy to challenge the organization's security controls in wide variety of realistic multi-step attacks in a controlled manner.* To the best of our knowledge, no rigorous comparison between threat emulators was published.

In this paper, we survey general purpose open source threat emulators suitable for post-compromise what-if analysis excluding tools targeted only at specific types of systems. For example, security assessment tools, such as Sqlmap [1] and W3af [2], web application attack and audit frameworks respectively, are not included in this survey because they too specific and are not general purpose threat emulators. Various fuzzing tools [8], [42], [17] as well as network and vulnerability scanners such as Nmap [23] or Nessus [35], are excluded from this survey because they are not used to emulate complex multi-step attack scenarios.

The threat emulators are reviewed and compared with respect to tactics and techniques from MITRE ATT&CK matrix and criteria such as prerequisites, attack manipulation, and clean up. The contributions of this article are four-fold:

1) We present well-defined criteria and detailed methodology for evaluation and comparison of threat emulators.
2) We present a thorough review the following threat emulators: APT Simulator, Red Team Automation, Uber's Metta, Caldera, Atomic red team, Infection monkey, PowerSploit, DumpsterFire, Metasploit, BT3 and Invoke-Adversary.
3) We present two taxonomies for categorizing threat emulators.
4) We provide actionable guidelines for choosing the best threat emulator given a specific environment and a set of security assessment tasks.

The rest of the paper is structured as follows. Section II details the background on attacking capabilities matrix. In Section III we define the ecosystem of threat emulation. Section IV present the criteria for comparison, with respect to which we suggest reviewing and comparing threat emulators. A comprehensive methodology for evaluating threat emulators is presented in Section V, followed by highlights of the reviews of eleven threat emulators. Next, in Section VI we introduce and discuss two taxonomies and guidelines for choosing threat emulators. Section VII summarizes the paper.

II. Background: MITRE ATT&CK

MITRE's ATT&CK is an open knowledge base of adversarial tactics and techniques which is continuously updated based on real-world observations [37]. MITRE's ATT&CK knowledge base is considered state-of-the-art when it comes to describing common behaviour of adversaries. Many companies, such as D3 Security and AlienVault, rely on this knowledge base in threat intelligence that they consume or provide.

[1]Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, P.O.B. 653 Beer-Sheva, Israel. {sundersb,shwarzs}@post.bgu.ac.il, {polinaz,puzis}@bgu.ac.il

Most threat emulators are designed with respect to the techniques listed in the MITRE ATT&CK Enterprise matrix[1]. Attacking capabilities of threat emulators include the implementation of different techniques that comprise various cyber-attacks. MITRE ATT&CK matrix can help systematically evaluating the versatility of attack scenarios that can be created by an emulator and consequently estimating the ability of the red team to discover security flaws during the assessment process.

First we introduce the MITRE terminology used in the rest of this paper. *Tactics* represent the adversary's technical goals, such as creating persistence, privilege escalation, defense evasion, and more. An attack's lifecycle, also referred to as the kill-chain [18], [30], [22], is typically a series of goals achieved during the course of the attack. Every tactic includes a variety of *techniques*, also known as attack patterns [6], that achieve similar goals. A technique may achieve multiple goals, thus being associated with a number of tactics. Techniques are abstract descriptions distilling the essence of many similar attack implementations. The specific implementations of the techniques are referred to as *procedures*. For example, in order to achieve persistence (a tactic) an adversary creates a scheduled task (a technique) using the following Windows command: `schtasks /create /sc /daily /tn <folder path>\<task name> /tr` (a procedure).

Next, we briefly explain the tactics listed in MITRE's ATT&CK matrix and their contribution to the success of an attack. These tactics will be used as attributes in the detailed threat emulators review in Section V.

Initial Access tactic contains all the techniques that enable malware to infiltrate a machine or a network environment. Penetration testing focuses on this tactic.

Execution tactic contains all the techniques that result in the execution of attacker controlled code on a victim's machine. The more *execution* techniques the attacker has in his toolbox, the more obstacles, such as PowerShell execution policy, he can overcome.

Persistence assures that even if the compromised machine restarts, the malware shall persist. The fact, that initial access is risky for the adversary and require significant effort, makes persistence crucial in the attack's life cycle.

Privilege escalation is required in order to invade important assets accessible with superuser permissions only. Superuser (or administrative) permissions also facilitates achievement of other goals such as defense evasion.

Defense evasion tactic contains techniques that enable the attack to stay under the radar of security defense tools.

Credential access is a goal pursued by most attacks. Having multiple user credentials facilitates privilege escalation, persistence, and defense evasion. For example, specific user credentials may be required to access a key system.

Discovery refers to techniques used by a malware to discover assets, network topology, security controls, vulnerabilities, etc.

[1]https://attack.mitre.org/techniques/enterprise/

Lateral movement techniques are used to reach the goal assets spreading out from the initial entry point. Emulating lateral movement is crucial threat emulators when evaluating network security systems.

Collection tactic contains techniques for collecting the target data assets. In addition to espionage and intelligence, collection also facilitates defense evasion and credential access tactics.

Exfiltration tactic includes techniques for transporting collected data out of the environment to the attacker.

Command and control tactic includes techniques that enable the attackers' interactions with their tools providing input on-the-go.

III. Threat Emulation Eco-System

Here we define the threat emulation ecosystem. Threat emulators are an integral part of the full red teaming process. Within this ecosystem we define the stakeholder identities, relevant software processes, and their relations. We map the MITRE Tactics (see Section II) to the software components of the red teaming process. The ecosystem elements are framed with respect to four threat emulation use cases: training, security tool assessment, organizational security assessment, and what-if analysis.

A. Stakeholders

Sponsor is the organization that employs a red team for security assessment.

Threat actor is the adversarial entity that targets the sponsor. Threat actor's motivations may vary depending on the sponsor's assets. His attacks compromise the sponsor's operational environment. Information about the threat actors, most relevant to the assessed organization, will enable the red team to adequately play the attacker's role [7], [10].

Red team is hired to challenge the security of an organization. Red teams mimic the behavior of threat actors [25], [28], [24]. Red teams may include penetration testers, social engineers, reverse engineers, intrusion detection specialists, and in rare cases also language specialists [3]. Sponsor and red team agree on the assessment target and scope in light of the critical organizational assets and possible attack vectors [7], [28]. The scope is also limited by ethical considerations and legislation [24].

Blue team defends the organization systems by deploying and using the security tools to detect and mitigate the activities of threat actors.

White team is responsible for the coordination, execution, and analysis of the red team blue team exercises [26], [14]. White teams are most important during training.

B. Software and tools

Operational environment is the hardware and software that supports the organizational operation. The security needs arise from the type of assets within the operational environment, threat actors, and regulation [44]. The operational environment also constrains which security assessment and training can be performed in the real operational environment

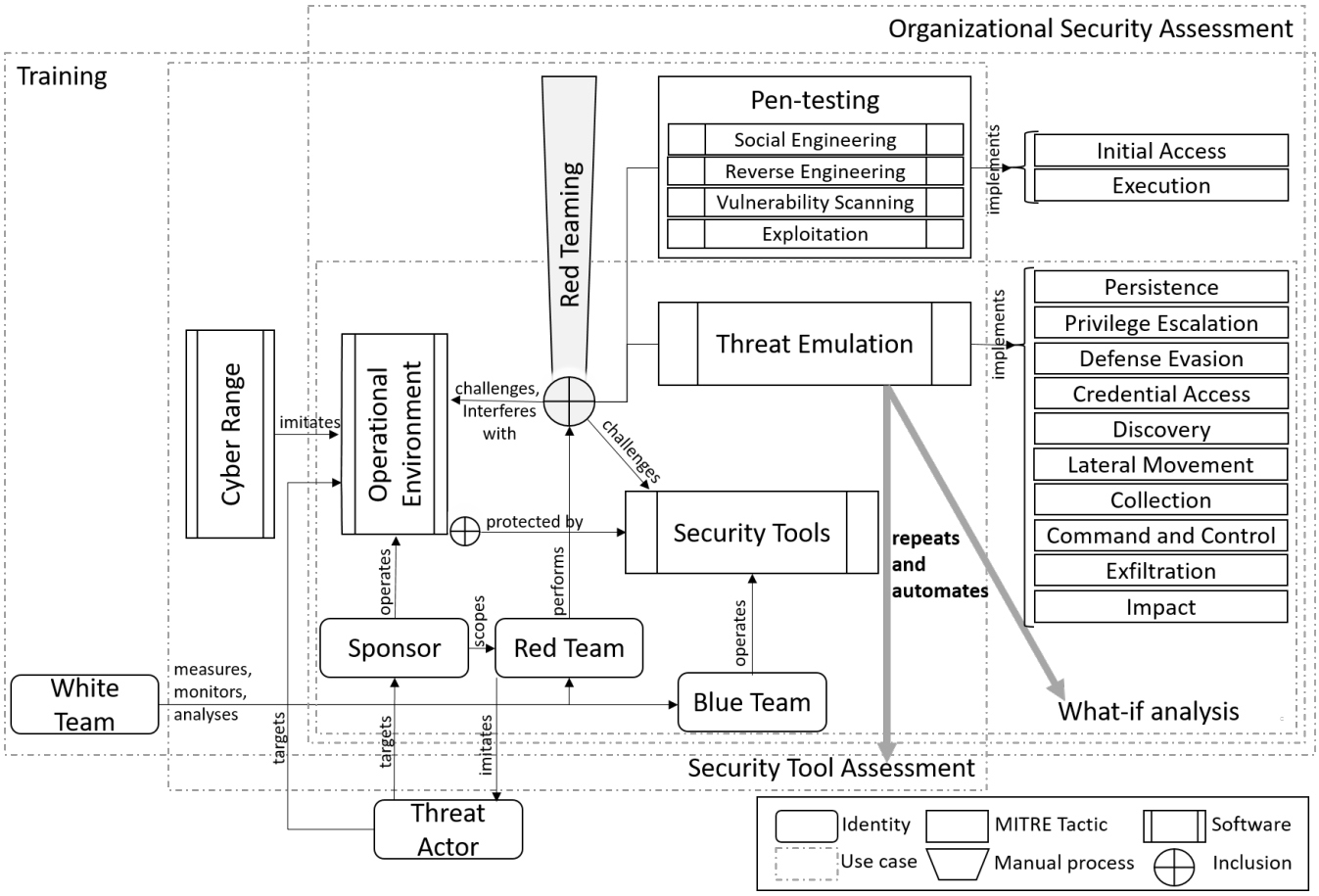

Fig. 1. The threat emulation ecosystem

and which must be performed in a non-operational environment such as a cyber range [25].

Cyber range is a term used for a virtual environment that is created for cybersecurity training. Some operational environments are too critical to allow the risk of malfunction during training or security assessment. In such ceases, a safe environment which imitates the real operational environment is required [25], [28]. A good example is critical infrastructures and especially supervisory control and data acquisition (SCADA) systems. A single glitch resulting from the red team activities in such a system could lead to disastrous malfunction. An additional reason to use a cyber range is the fact that threat emulation can cause the triggering of false alarms, which are hard to distinguish from real attacks [44].

Security tools represent the collective notion of intrusion detection, anti-virus, firewall, SIEM (security information and even management), EDR (end-point detection and response), and other systems that are deployed within the operational environment to protect the organizational assets. Security tools are operated by the blue team.

Vulnerability scanners, reverse engineering tools, exploits, fuzzers, etc. are all part of the tool-set of red teams. They are used by red teams in order to actively probe the operational environment and the security tools for weaknesses and vulnerabilities. Following Applebaum et al. [4] and Adkins [3] we include under penetration testing all tools used for identifying opportunities for unsolicited access or code execution including social engineering.

Today cyber security defense paradigms assume that the system is already compromised [11]. Consequently perimeter defense is insufficient and the defenses need to identify and mitigate malicious activities throughout the whole attack cycle. Security array assessment of the organization in post-compromise scenarios is critical. Repeated assessment using red teams is expensive due to the required expertise. Manual red teaming also lacks the ability to replicate the security assessment in fully repeatable manner.

Threat emulators, software tools that are the main focus of this survey, compensate the lack of sufficient expertise of the red team members and aid with automation of post-compromise red team activities [4]. Although, human white hat (ethical) hackers cannot be replaced by automatic tools in the mission of red teaming, however, using threat emulators can increase the efficiency of the red teams or reduce the expertise level required from the red teams [28]. Challenging the security array in many different scenarios rapidly and repeatedly is especially important during training and what-if analysis. When emulating attack techniques that may cause damage to the operational environment or personnel (e.g. persistence, credential access, lateral movement, collection,

impact) threat emulators usually operate by leaving forensic evidence that mimics execution of specific attack procedure. The emulators may execute real attack procedures without causing actual damage or compromising the organization.

Vulnerability scanners and threat emulators may interfere with the network and systems in place. Therefore, they should be used with caution within the operational environment. The extent to which penetration testing tools are employed (frequency, intensity, volume, etc.) is defined by the sponsor and the red team within the scope of red teaming.

Red team blue team exercises should be used in a dedicated environment, cyber-range, where there is no interference to the operational environment. The cyber range should imitate the operational environment to the extent possible in order to increase the suitability of the security tools and their operators (the blue team) to the organization.

C. Red teaming use cases

Training

The terms red team and blue team originate from military exercises where a blue team is in charge of protecting an asset, and the red team's role is to challenge the blue team's plan of defense iteratively [25]. The result should be a comprehensive understanding of the asset's security mechanisms/protocols (and their weaknesses) as well as improved response in terms of mitigation and efficiency. In some cases, a white team is in charge of overseeing the whole training exercise(s), coordinating between teams, and analyzing their performance. Often the training takes place in a controlled environment like a cyber range.

Threat emulators can be used to train the blue teams and test their capabilities in handling security events. The emulators can create predefined and well-designed scenarios that provide full coverage of essential detection and response cases. In absence of a qualified red team, a threat emulator without extensive security expertise can use threat emulator to train the blue team members in common scenarios.

Security Tool Assessment

The cybersecurity landscape is flooded with security products. Organizations' security needs can vary. The security requirements may depend on the type of assets an organization has, the regulation it is subject to, organization's size, and other characteristics of the operational environment.

Security tool assessment has two perspectives: assessing the contribution of a tool to the security preparedness of the organization, and testing the resilience of a tool (or new version of it), also known as subversive exploitation [28], [24]. Assessment of security products should never compromise an organization [26]. An organization can use threat emulators to perform comparisons and assessments of security tools. To asses security products, the assessor can use a threat emulator to launch attacks that mimic real scenarios without compromising the organization's security.

Having the ability to create and reproduce the same simulated attack helps comparing and contrasting security products of the same type. MITRE ATT&CK Evaluation[2] is an example of a service that assesses capabilities of different types of security tools. In particular they articulate the adversarial techniques that each evaluated tool is able to mitigate.

[2]https://attackevals.mitre.org/

Organizational Security Assessment

Security regulations and standards such as the EU General Data Protection Regulation (GDPR) [40], ISO/27001, or the NIST catalog of security and privacy controls [19] define a set of security controls as well as standard security assessment activities. For example, section CA-8 in the NIST Special Publication 800-53 on security controls defines the need for independent security assessment using red teams and red team exercises.

Due to the constant development of the cyber-threats landscape, organizations must routinely review, document and update all security-related aspects. The red team role is to identify weak points and security breaches through interaction with different planes of the organization: systems, users, applications, etc. [28], [24]. In-house or outsourced red teams assess the organization's security by simulating attacks that do not compromise the security but thoroughly test it [31]. Launching a wide range of emulated attacks allows the organization to better understand vulnerabilities and weaknesses it has; as a result, the organization can take measures to improve security.

What-If analysis

During a what-if analysis, the user tries to check two things. First, the system/organization's response and robustness to a specific change of parameters [5] in different granularities. The second thing which can be checked is the impact and response to failures which helps prioritizing and managing such failures.

By using threat emulators, the user can automate parts of the what-if analysis process. Threat emulators allow the reproduction of scenarios while allowing to change only specific parameters, thus allowing to isolate specific variables and key aspects.

IV. Criteria for Comparison

The primary objective of threat emulation is speeding up and simplifying the what-if analysis of organization's security controls and countermeasures in diverse scenarios. Choosing the best threat emulation framework depends on the capabilities of the red / blue teams as well as on various organization constraints. This subsection presents criteria that we suggest using for the comparison and evaluation of threat emulators.

A. Environment

Checking whether or not the emulator can operate in the target environment is the first basic criteria for the emulator's compatibility. This includes operating system compatibility, required changes in the security array, special privileges, etc.

1) Operating system compatibility: Usually organizations work with different operation systems (OS). *Ideally a threat emulator should support all operation systems of the endpoints.* We do not include in the OS compatibility criteria

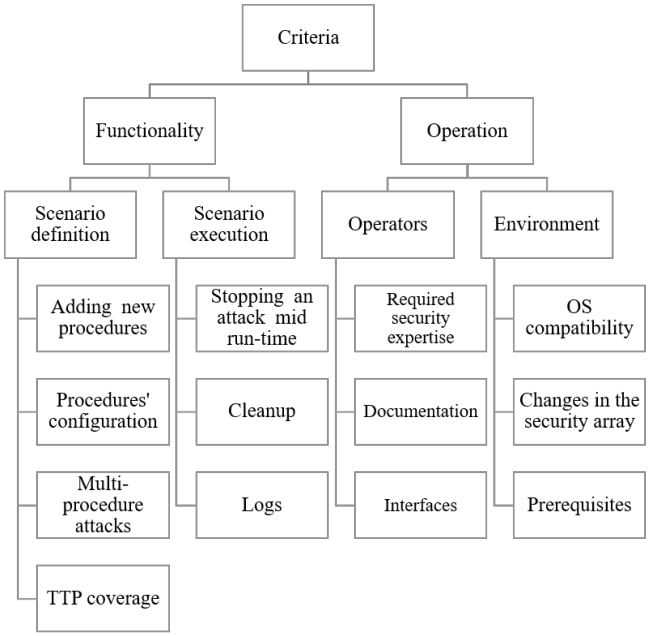

Fig. 2. Taxonomy of criteria for comparing threat emulators

requirements of the CNC component of the emulator because it is not part of the tested organizational environment.

The OS compatibility criterion may include one or more of the following OSes: Windows, Linux, MacOS, Android, iOS. Due to the ubiquitous nature of mobile devices it is very important to include them in tested attack scenarios. Thus, we include mobile platforms in this threat emulator evaluation criteria, although, none of the reviewed emulators support them.

2) Changes in the security array: Every organization has various security tools that provide some level of protection. anti-virus (AV) and firewall (FW) are parts of the organizational security array. Some emulators require disabling AV and FW to operate successfully. For example, a threat emulator that relies on Remote Administration Tool (RATs) may require disabling the real-time anti-virus protection in order to operate successfully. Such emulator is not suitable to evaluate the anti-virus protection.

A good threat emulator should operate without requiring changes in the organizational security array. In the ideal case, the evaluated security controls and countermeasures are able to detect the emulated attacks but not the emulator's agent.

This criterion value is the set of security controls that need to be disabled to execute the emulator:

- ∅ The emulator can successfully operate without disabling any security controls.
- AV anti-virus software should be disabled.
- FW Firewall should be disabled.

3) Prerequisites: This criterion will specify if there are any special prerequisites for a threat emulator to work. It may include third party software tools (such as mimikatz), hardware components (such as a dedicated CNC server), special privileges (if the emulator's agent requires superuser privileges), etc. *Similar to any redistribute package a good threat emulator should be self contained.*

This criterion can include one or more of the following values:

- ∅ The emulator is self contained and does not require special privileges.
- 3d Requires installation of third party tools on the endpoints.
- Priv Requires special privileges (e.g. superuser or running in a kernel mode).

We consider an emulator to be self contained if it includes everything that is required to operate on the endpoints. We exclude from this criterion any prerequisites for the dedicated servers because their impact on the deployment is minimal compared to required installations on the endpoints. In addition, since we focus on open source threat emulators we assume that all prerequisites are open source as well.

B. Scenario Definition

When evaluating security solutions it is often required to mimic complex and sophisticated real attacks. A threat emulator should be able to define all steps of the emulated attacks and produce realistic artifacts. It includes both the ability of the emulator to leave detectable traces as well as to cover the tracks emulating defense evasion techniques.

1) Adding new procedures: Cyber criminals constantly develop new exploits and perform unknown attacks. Hence, *the ability to add new procedures is crucial for a future-proof threat emulator.* A good emulator should allow adding new procedures for techniques it already implements as well as procedures that introduce new techniques.

This criterion is binary:

- Yes The emulator facilitates adding new custom procedures.
- No The emulator does not provide interfaces and infrastructure for adding custom procedures.

2) Procedures' configuration: Different organizations have different assets, network structure, and security controls. Consequently, it is important to be able to easily modify the operation of an emulated attack in order to fit the organization. For example, one should be able to configure the IP range, ports, and protocols of a procedure that implements network scanning.

On the one hand, if the emulator does not supports neither adding new procedures nor configuring the built-in procedures, its applicability is very limited. On the other hand, adding a new procedure instead of configuring an existing procedure is extremely ineffective. Moreover, the ability to configure procedures is required when emulating a large number attack variants in a row. *A good threat emulation framework should provide the facilities for configuring both built-in procedures and new custom procedures.* For example, an emulator that executes remote code on an endpoint should at least ensure that the parameters of the executed procedures are set before transferring the code to the endpoint.

We consider three possible values for the configurability of the threat emulator:

- Low The emulator contains a set of procedures with hard-coded parameters.
- Med. The emulator supports configuration files or repeated execution of the same attack scenario with different parameters.
- High The emulator provides facilities to easily configure new custom procedures.

3) Multi-procedure attacks: An attack can be referred to as a set of procedures launched along a specific timeline. In order to emulate a realistic attack, it is required to combine multiple procedures together. *Threat emulators should both provide multiple built-in attack scenarios and support the creation of new multi-procedure attacks.* Build-in attack scenarios may serve red teams for rapid testing of new security solutions, IT professionals who are not qualified red team practitioners, and novice security personnel for training. The ability to create custom attack scenarios recombining tactics, techniques, and procedures is very important for challenging the organizational security array and for performing diverse what-if analysis.

This criterion may include one or more of the following values:

- $\emptyset$ No support of multi-procedure attacks.
- Built-in Includes ready to run multi-procedure attacks.
- Custom Multi-procedure attacks can be added.

4) TTP coverage: Versatility of a threat emulator is derived from the assortment of attack techniques it can emulate. To map the capabilities of a threat emulator we refer to the set of tactics, techniques, and procedures in the MITRE ATT&CK knowledge base. The more tactics a threat emulator covers the more complete an emulated attacks can be. The more techniques a threat emulator implements the more comprehensive assessment can be done by using it. In an ideal case *a threat emulator should contain multiple procedures implementing each and every known adversarious technique.*

At least one procedure/technique is required to claim that a technique/tactic, respectively, is supported by an emulator. We define three levels of TTP Coverage based on the supported tactics and techniques as follows:

- Low the number of supported tactics is less than six or the number of supported techniques is less than 20.
- Med. the emulator supports more than six tactics and the number of techniques is between 20 and 40.
- High the number of supported tactics is eight or more and the number of supported techniques is more than 40.

The above thresholds are quite low taking into account the 12 tactics and 266 techniques currently listed by MITRE in the Enterprise ATT&CK matrix. Unfortunately, the overall TTP Coverage provided by current threat emulators is very low. The thresholds were set such that they partition the set of emulators into three non-trivial groups.

C. Scenario execution

1) Stopping an attack mid run-time: When examining an operational environment during the execution of a simulated attack an operator may want to stop the attack for various reasons. For example, a simulated attack might consume too much computational resources or cause other damage or crucial updates may be required to the endpoints. There may also raise a need to change the attack scenario after the evaluation process have started. For example, the red team may realize that the scenario is missing some crucial components for effective assessment. In this case the red team would like to stop and restart the evaluation. Consequently, *it is important to have the option to stop an emulated attack at any stage before it finishes running.*

Note, that changing the attack scenario while it is being executed results in a new attack scenario that is different from both the new one and the old one. Thus, we do not consider it a useful capability.

This criterion is binary:

- Yes The emulator supports stopping an attack scenario while it is running without far reaching consequences.
- No The emulator does not provide such functionality and the operator is required to manually stop all the attack components.

2) Cleanup: After an emulated attack scenario has finished or has been stopped in the middle *it is required to roll back the machines to their previous state, and cleanup all traces of the attack.* This functionality is especially important when emulating large numbers of attacks. Cleanup may include for example, changed registry values, files, etc.

During cleanup a threat emulator should remove artifacts that it has created. However, send and forget actions performed by the emulator, such as pinging a remote server, may be logged by monitoring system. As a result an alert may be created and stored within the SIEM. A threat emulator is not aware of the security, logging, or monitoring tools in the network. In case that the threat emulation is performed on a dedicated cyber-range it is possible to roll-back the machinery state and continue emulating further attacks.

However, in a operational network reverting the response of the security tools to the activity of the threat emulator is not a reasonable requirement. This will create a too strong coupling between the red-team and blue-team instrumentation. Nevertheless, the operational staff should be aware to possible consequences of the threat emulation including possible collateral damage and false alarms [44].

This criterion is ternary:

- No The emulator does not support cleanup.
- Proc Cleanup included at the level of individual procedures.
- Attk Cleanup included at the level of multi-procedure attacks.

Although, none of the reviewed emulators facilitate cleanup implementation in new custom procedures, it is important to encourage such implementation at the API level.

A good example of cleanup facilitated by a framework are the teardown functions in unit tests. In case of multi-procedure attacks both procedure level and attack level cleanup (teardown) implementation is required.

If the emulator supports stopping an attack mid run-time then we expect the cleanup functionality to be invoked when the attack is interrupted. In case that cleanup of an interrupted attack is not possible we consider it to be a bug rather than assessment criterion.

3) Logs: Threat emulator logs can help the operator to understand whether the attack was executed successfully, which stages of the attack were performed, whether or not the required assets were acquired etc. This criterion refers to the capability of a threat emulator to automatically produce and store log files. It may have the following values:

- No — There are no login capabilities implemented. If needed the operator can log the attack steps.
- Base — Log entries are created at the beginning and the end of the emulated attack execution.
- Adv. — A log entry is generated for every executed procedure.

Note that, if an emulator does not support multi-procedure attacks then, this criterion receives the value Base by default.

D. Operators

1) Required security expertise: One of the goals of using threat emulators is simplifying the security assessment process. Ease of usage and execution of attacks is an important factor when evaluating a threat emulator. Requiring expert knowledge and meticulous setup and configuration of a threat emulator is counterproductive. *A good threat emulator should allow an operator without security expertise to execute and configure all build-in attacks.* Of course, adding new procedures requires cyber security expertise. This criterion refers only to the expertise required to configure and execute attacks provided within the emulator's original package.

We define two levels of Required Security Expertise as follows:

- Low — Built-in attacks can be executed out of the box or precisely following step-by-step instructions provided with the emulator.
- High — Configuring and executing attacks provided with the emulator requires, cyber security knowledge that is not included in the provided instructions.

Note that this criterion is different from the next documentation criterion that considers the coverage and detailization of the provided documentation. Here we only consider the prior cyber security knowledge required to launch attacks.

2) Documentation: In addition to intuitive user interface it is important to inspect the emulator documentation. While some emulators are poorly documented collection of scripts, some are well documented software frameworks. Good documentation shortens the learning curve and facilitates full utilization of the emulator capabilities. *An operator that is not familiar with the emulator should succeed in creating and executing attack scenarios as well as in interpreting the results relying only on his cyber-security knowledge and the provided documentation.*

This criterion refers to the completeness of the documentation. It may have the following values:

- Full — The documentation is sufficient to setup all the emulator's components, execute built-in attack scenarios, create new ones, and interpret the execution results.
- Miss — Documentation of some of the critical functionality is missing.
- None — The documentation is insufficient to setup the emulator or execute build-in attack scenarios.

3) Interfaces: It is important to map the interfaces of an emulator in order to better match the capabilities of the operators and the organizational needs, such as the desired level of attack customization or the frequency of security assessment. Threat emulators may be operated through a simple command line interface (CLI) or a graphical user interface (GUI). Novice operators may find GUI more accessible and more intuitive. Command line tools are usually required for automating security assessment but have a steeper learning curve. Some emulators also support scripting (SRP) to produce custom attack scenarios. A few emulators, such as Atomic Red Team and Metasploit, provide application interface (API) for management of the attack scenarios. *Ideally an emulator will provide all the above interfaces.*

The emulator interfaces criterion may include one or more of the following values:

- GUI — Some of the emulator functionality is accessible through a GUI.
- CLI — Some of the emulator functionality is accessible through a CLI.
- API — Some of the emulator functionality is accessible through an API.
- SRP — The emulator supports scripting of attack scenarios or experiments.

When all functionality relevant to the scenario definition and execution is accessible through GUI, CLI, or API we mark the respective interface with asterisk (*).

V. Detailed Evaluation of Threat Emulators

In this section, we present a comprehensive evaluation methodology and review eleven open-source threat emulators based on the criteria discussed in Section IV.

A. Evaluation Methodology

In this subsection we describe the methodology for evaluating a threat emulator with respect to the criteria detailed in Section IV. The expert questionnaire for emulator assessment is detailed in Appendix VIII. The step-by-step evaluation process is depicted in Figure 3.

First we check and list the emulator components, such as the CNC server and agents; *prerequisites*, such as third-party tools, libraries, and required privileges; needed in order to launch the threat emulator. If third-party tools or special privileges are required to be setup on the endpoints on which

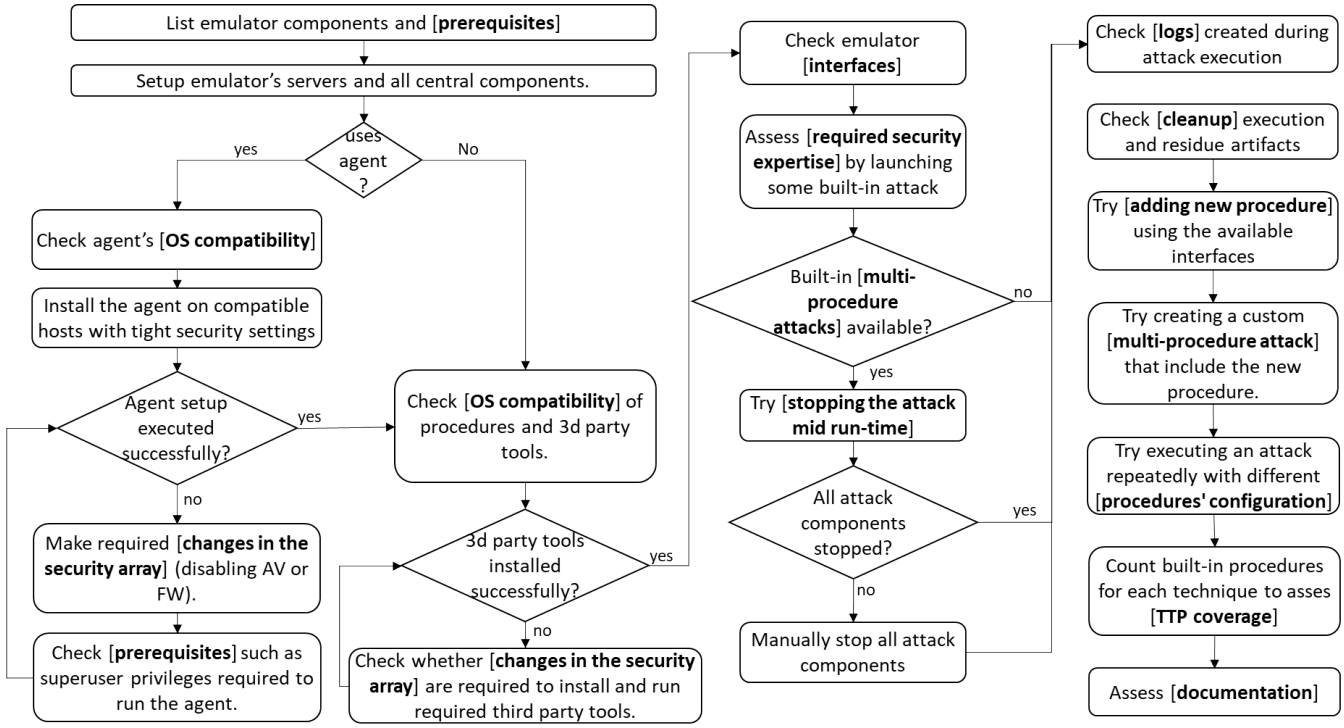

Fig. 3. Threat emulator evaluation process

the emulated adversarial activity will be executed then we check the values 3d or Priv, respectively in the *prerequisites* criterion. Otherwise, the value of this criterion is set to $\emptyset$.

Next, we record whether or not the emulator uses agents to emulate adversarial activity. The *OS Compatibility*, is determined by the emulator's agents if it uses them. In absence of agents the *OS compatibility* is determined by the built-in procedures and the required third party tools (e.g. mimikatz). CNC server, tools for analyzing the results, and other emulator's components that can run on dedicated machines which are not a part of the original operational environment are not considered for *OS compatibility*.

Next we check whether *changes in the security array* are required to successfully setup the agents and install third-party tools. For a threat emulator that works with agents we check if the agent can be installed without disabling the local firewall and anti-virus software. At this stage we also check whether the anti-virus alerts on the third party tools or any emulator files that should run on the endpoints.

After setting up the emulator we check the available *interfaces* using which the emulated attacks can be defined and executed. We check whether the emulator has GUI, API or CLI, and whether it supports scripting (SRP). For each *interface*, we check if all functionality relevant to scenario definition and execution is available via the *interface*. If so, we add asterisk (*) to the *interface*'s respective value.

To evaluate the threat emulator with respect to *required security expertise* we start with launching a built-in attack. If it is possible to launch the attack without delving into arguments or configuration while precisely following the documentation, this criterion will receive the value Low. Simple meaningful names of procedures contribute to Low *required security expertise*. However, if the configuration of the attack (including choosing the relevant procedures) requires knowledge in cyber security beyond the documented configuration steps this criterion will receive the value High.

We check whether the emulator includes built-in *multi-procedure attacks*. If it does, we mark the value Built-in for the criterion *multi-procedure attack*, execute it, and try to *stop the attack mid run-time*. We search for the stopping functionality in the available interfaces. For example, in case the threat emulator has GUI, we look for a "stop" button. If such functionality is provided, and when activating it all the attack components are being stopped, then this criterion receives the value Yes, otherwise, it receives No.

After the attack has stopped we examine the created *logs*, if such exist. If the created *log* describes each procedure that has been executed this criterion will receive the value Adv. However, if only the attack metadata, such as start time, end time, endpoint, etc., was logged, without information about execution of procedures, this criterion will receive the value Base. If none of the above exists this criterion will receive the value No.

After the attack has stopped we also examine the emulator's *cleanup* functionalities. If *cleanup* is available, we verify that it is executed correctly, i.e., that the attack traces/artifacts have been removed. For example, if a file was created then it should be deleted. If a registry key was modified it should be restored to the previous value, etc. As noted before, send and forget actions cannot be *cleaned up*.

In addition, we examined the timing of the *cleanup*. If the *cleanup* has been executed after each procedure separately, this criterion will receive the value Proc. If the *cleanup* has been executed after the execution of the entire attack scenario, this criterion will receive the value Attk.

Next we examine whether the emulator supports *adding new procedures* and creating custom *multi-procedure attacks*. First, we try *adding a new procedure* using the available interfaces. If the emulator is script based we add the new script to the relevant folder. We conclude that an emulator does not support *adding new procedures* if it requires changing the emulator's source code, for example, inserting a new procedure into an existing script. To conclude that the new procedure is added successfully we execute a simplest attack containing this procedure.

Afterwards, we try creating a new *multi-procedure attack* either using the available *interfaces* or by combining available scripts. If a new procedure was added in the previous step then the new multi-procedure attack should include it. If the new custom multi-procedure attack was created and executed successfully, we mark the value Custom for the criterion *multi-procedure attack*.

Having the new multi-procedure attack, or any built-in attack, we examine the *procedures' configuration* before execution. For each procedure that may require arguments, such as endpoint's IP for lateral movement or data exfiltration, a file name to search, etc. we check how can these arguments be specified. If the arguments are specified using a configuration file or using the emulator's *interfaces* before execution of the attack the criterion value is set to Med. Otherwise, if the arguments are hard-coded we change them in the code and set the criterion value to Low. Finally, we set the criterion to High, if when *adding a new procedure* the emulator enforced an API that supports future configuration changes.

To evaluate the *TTP coverage* of an emulator we are first required to list all the built-in procedures that the emulator provides. We map each procedure to the techniques that it implements. Next, each technique is mapped to the tactics that can be achieved by using it. The mapping is performed according to the names of the procedures, techniques, and tactics using mainly the MITRE ATT&CK Enterprise matrix[3]. Note, that the names of the emulator's built-in procedures might not match to their names in the MITRE ATT&CK website. In such cases, we map the procedures to techniques according to their functionality. We summarize the mapping results in a vector of 11 dimensions, one for each tactic, except for initial access tactic. The value in each dimension is the total number of techniques that the threat emulator supports in the respective tactic. Initial access is not included in current evaluation of threat emulators because it is mainly covered by penetration testing tools which are out of the scope of this survey.

Last but not least, we conclude with the assessment of the *documentation* quality. By *documentation* we refer to comments in a procedure's implementation, README files, text in the respective GIT page or website, tooltips and help options in the emulator's interface, manuals, etc. Throughout the emulator's evaluation process we were assisted by the available documentation. If every operation that we were required to do was well documented, this criterion receives the value Full. If one of the critical functions, required for attack definition or execution, was not well documented, this criterion receives the value Miss. If the documentation does not include helpful instructions for setting up the emulator and executing built-in attack scenarios we set the value of *documentation* criterion to None.

B. Reviews' Highlights

Each threat emulator was manually examined to determine: (1) which of the MITRE ATT&CK tactics and techniques are implemented, and (2) how each of the criteria discussed in Section IV are addressed. This subsection presents the highlights for each reviewed threat emulator. A table with the detailed review of the emulators is added as a supplementary material to the paper.

1) Red Team Automation (RTA): Red Team Automation[4] is a framework comprised of Python scripts aimed for blue teams to test their security mechanisms. It has no central components such as a CNC. Since RTA is written in Python it can be used only on Python supported OS and on endpoints that have Python installed on them. py2exe can be used to operate RTA on Windows OS without Python. To use all the built-in functionality, RTA requires additional third-party software. For example, in order to simulate lateral movement psexec or xcopy are required.

RTA includes minimal documentation with setup and execution instructions on its Git page[5] and in comments within its scripts. The scripts can be combined to create a multi-procedure attack. Additionally, it is possible to edit procedures or add new ones using Python. RTA requires Python knowledge to be used effectively.

There is no log file that documents the emulated attack execution and the changes made to the endpoint. But RTA has a cleanup process which is useful to revert the changes within each procedure.

RTA scripts implement a decent number of techniques from MITRE's ATT&CK matrix. However, the coverage of the tactics is not uniform. For example, persistence and defense evasion tactics are well covered with 10 and 20 techniques respectively, while lateral movement has only three techniques implemented in RTA. RTA can also generate network traffic that mimics communication with the attacker's CNC.

RTA provides both procedures that anti-viruses usually do not detect and common procedures that can be detected by anti-viruses. RTA does not require disabling security tools in order to launch most of the procedures.

Overall, RTA is suitable to test an entire network and is capable of simulating real-life threats, due to its versatility

[3]https://attack.mitre.org/techniques/enterprise/

[4]https://github.com/endgameinc/RTA

[5]https://github.com/endgameinc/RTA

of tactics and techniques. The fact that this tool is Python based makes it easier to add advanced techniques and more sophisticated attacks.

2) Metasploit: Metasploit[6] is one of the most used penetration testing toolkits. It focuses on initial access and execution tactics and contains a large library of vulnerabilities and exploits [16]. Metasploit can also be used as a threat emulator for security assessment thanks to the post-compromise tools it contains.

Metasploit attack modules may deliver a payload to the compromised endpoint in order to perform the post-compromise activities. Meterpreter is the Metasploit's most known payload. Meterpreter provides a good coverage of post-compromise tactics including discovery, collection, persistence, privilege escalation and more. It employs third-party tools such as psexec and BruteForce for lateral movement.

Metasploit does not have a library of built-in multi-procedure attacks. But extensive attack scenarios that mimic real APTs can be scripted. For example, AutoTTP[7] is a scripting package that enables generating and executing multi-procedure attack scenarios using the API of Metasploit and Empire[8]. AutoTTP organizes the attack procedures along an attack life cycle suggested by its developer Jym Cheong.

Metasploit server components are a part of Kali Linux OS, but can also be installed on Windows OS. Metasploit target endpoints may run Windows, Linux and MacOS. Most of the attacks performed by Metasploit are not detected by standard anti-virus tools.

An operator needs to have extensive cyber-security knowledge and some knowledge in Ruby in order to both launch built-in attacks and add new attack procedures. In order to ease up on the operator, a GUI version of Metasploit, called Armitage, has been developed[9].

3) Invoke-Adversary: Invoke-Adversary[10] is a PowerShell script intended for blue teams to test their security mechanisms. Since Invoke-Adversary is written in PowerShell it can be used only on PowerShell supported OS and on endpoints that have PowerShell installed and enabled on them.

Invoke-Adversary main menu is based on MITRE's ATT&CK matrix tactics list. The menu allows the operator to execute a single technique for each tactic. Invoke-Adversary implements six to ten techniques for each of the following tactics: persistence, discovery, credential access, defense evasion, command and control, and execution. However, Invoke-Adversary provides only one technique for the collection tactic. Lateral movement is not supported.

Procedures provided by Invoke-Adversary for the credential access tactic are usually not detected by an anti-virus. Other provided procedures are common and can be detected by anti-viruses.

Since Invoke-Adversary is a PowerShell script, it can be modified by an operator that has sufficient knowledge in PowerShell and in cyber-security. Such an operator can edit procedures, add new custom procedures, and compose multi-procedure attacks. Invoke-Adversary does not provide cleanup functionality. It outputs a log to the console at the end of attack execution.

Overall, Invoke-Adversary is suitable for testing a stand alone endpoint, but operators are required to have cyber-security expertise in order to execute attacks that combine more than one procedure.

4) Blue Team Training Toolkit (BT3): BT3[11] is a platform that generates network traffic that mimics malicious activity. It is primarily used as blue team training kit. BT3 offers free components and paid components. The free components include 14 network attack simulations and 10 hash collisions. The attack components cannot be combined into a multi-procedure attack scenario.

BT3 has to be installed on Linux server. However, the network simulation can target endpoints with any OS. Hash collision can be performed only on the Linux server.

BT3 does not require Linux knowledge. It provides install.sh script for easy setup. However, in order to execute a malicious network traffic simulation, the operator should recognize attack procedures and the required configurations by the procedures' names. Consequently, cyber-security knowledge is required to correctly operate the network simulations. BT3 Web site provides sufficient reference on all BT3 components. Some BT3 components require additional third-party tools such as LHOST, SSL, etc.

BT3 does not have a cleanup functionality. Files created from hash collision need to be manually removed by the operator.

BT3 only performs malicious network traffic simulation and known malware hash collision files.

Overall, BT3 is suitable to test an IDS or hash based anti-virus. The fact that BT3 does not have multi-procedure attacks and does not support various tactics and techniques makes it hard to use for threat emulation.

5) Advanced Persistent Threat (APT) Simulator: APT simulator[12] is a Windows batch script, which aims to be as simple as possible with little user interaction. APT simulator runs on Windows with no additional setup requirements and with required configuration. It does not support other OSs.

The procedures' documentation including execution instructions appear in comments inside the code. However, having sufficient expertise, it is possible to add new procedures and create new attacks by editing and combining batch scripts. Sophisticated procedures and techniques can be created by adding third-party tools to APT simulator's folder and writing an appropriate batch script that utilizes them.

Most of the procedures implemented in APT simulator were detected by an anti-virus. The procedures detected by

[6] https://www.metasploit.com/

[7] https://github.com/jymcheong/AutoTTP

[8] Empire is a PowerShell and Python post-compromise agent https://github.com/EmpireProject/Empire

[9] http://www.fastandeasyhacking.com/

[10] https://github.com/CyberMonitor/Invoke-Adversary

[11] https://www.bt3.no/

[12] https://github.com/NextronSystems/APTSimulator

an anti-virus are listed in the tool's Git page. Currently, APT Simulator implements techniques for persistence and discovery tactics (eight and seven techniques, respectively).

APT simulator lacks lateral movement capabilities. It has no CNC component and no cleanup or logging functionalities.

Overall, APT Simulator is suitable for endpoint device testing. It can be deployed and ran in minutes on Windows environment and it provides a basic testing for the security of an endpoint.

6) Caldera: Caldera[13] is a threat emulator developed by the MITRE organization. It is designed to test the security of a Windows system [4]. Caldera includes remote access agent, database, and server components. Caldera server can be installed on Windows server or Windows 10 platforms. Caldera's agents can run on Windows. The installation and configuration process of Caldera's components is longer than that of most of the threat emulators. In order to use Caldera's remote agent, it is required to disable the anti-virus software running on the endpoint.

Once installed, Caldera's graphical interface allows the operator to control all the agents and get visual feedback about an ongoing attack. All the attacks in Caldera are taken from MITRE ATT&CK matrix and information about the procedures can be found in the MITRE documentation. Attack procedures can be chosen from the Caldera's library and combined together to create custom multi-procedure attacks. Caldera gives the operator the option to import his own Remote Administrative Tool (RAT) to perform additional procedures or have a different CNC communication. Caldera's lateral movement capabilities allow choosing the first compromised machine ("patient zero") out of the endpoints with Caldera's agents. Other tactics that Caldera covers are persistence, privilege escalation and discovery with seven, six and eleven techniques respectively.In addition, Caldera provides a built-in cleanup functionality.

Overall, Caldera provides user friendly interface and functionality to compile new attacks from existing procedures. Caldera is most suitable for those who search for a threat emulator with a large variety of built-in features, lateral movement and CNC communication using RAT, and a large set of discovery procedures.

7) Infection Monkey: Infection Monkey[14] is a threat emulator used for testing and assessing the defenses against lateral movement and discovery tactics (mostly initial access). Infection Monkey has a CNC component and it requires RAT to be installed on the endpoints. Since anti-virus tools label the RAT used by Infection Monkey as a threat, it is required to disable anti-virus tools for Infection Monkey to function properly. Infection Monkey server can be installed on any OS with no additional knowledge required to launch an attack, while its RAT component can be installed on Windows and Linux (but not on MAC OS).

The emulator provides the ability to choose the first compromised machine out of the endpoints with the Infection Monkey RAT. It reports about machines that were infected in the process of lateral movement. Infection Monkey is designed with a purpose to test the defenses against lateral movement. But it lacks versatility of tactics and supports low number of procedures.

The techniques and exploits used by Infection Monkey come from real-world scenarios, making it close to a real attacker's behavior when it comes to lateral movement. Notably, most of the procedures, if executed separately without the RAT, would have remained undetected by anti-viruses.

The progress of each attack execution is logged in the interface and a detailed review of the results is provided. Infection Monkey provides a clean up option which is useful when examining live systems.

Overall, Infection Monkey is a good threat emulator for testing network defenses. Moreover, its easy installation process together with a graphical user interface, cleanup, logging, and OS versatility make it a suitable choice for novice operators that are looking for an easy way to test network environment.

8) PowerSploit: PowerSploit[15] is a collection of PowerShell based scripts aimed for various security assessment tasks. Since PowerSploit is based on PowerShell, it runs natively on any Windows OS with PowerShell installed. In addition, there are adds-on that can be installed on MacOS and Linux to enable using PowerShell.

New procedures can be implemented using PowerShell scripts. PowerSploit's scripts can be manually combined into a custom multi-procedure attack. PowerSploit lacks cleanup and logging functionalities.

PowerSploit supports a high number of sophisticated procedures. As a result it requires cyber-security expertise in addition to PowerShell knowledge. PowerSploit supports various techniques that cover persistence, privilege escalation, defense evasion, and discovery tactics (18, 9, 18 and 18 techniques respectively). Also supports credential access, collection, and lateral movement tactics with 7 ,7 and 6 techniques, respectively.

Overall, PowerSploit can be a quick and easy baseline assessment tool. It is well documented, and it contains a high amount of procedures, specifically a high number of procedures that try to avoid getting caught by anti-viruses.

9) DumpsterFire: DumpsterFire[16] is a platform for building attacks using techniques the majority of which are not mentioned in MITRE ATT&CK matrix. For example, creation of 5000 files, search for hacking tools in Google, and running of a video in loop on YouTube. DumpsterFire can be executed only on Linux systems. DumpsterFire does not have a cleanup option, and documentation can be found only their GitHub page and within the procedure scripts in form of comments. Logs of the execution are shown on the CLI.

A multi-procedure attack is built by choosing techniques

[13]https://github.com/mitre/caldera

[14]https://github.com/guardicore/monkey

[15]https://github.com/PowerShellMafia/PowerSploit

[16]https://github.com/TryCatchHCF/DumpsterFire

from a menu. DumpsterFire provides a set of built-in multi-procedure attacks packed with a well-built CLI. In addition, DumpsterFire implements a variety of discovery and credential access techniques (13 and 6 techniques, respectively). However, DumpsterFire lacks CNC and lateral movement techniques. Overall, DumpsterFire supports low number of techniques covering few tactics. However, the provided techniques and the simple interface make DumpsterFire useful for examining the basic security of Linux endpoints in an easy and accessible manner.

10) Uber's Metta: Metta[17] is a threat emulator developed by Uber organization. Its main purpose is assessing the endpoints' security. But it also includes a set of network security testing procedures. Uber's Metta can be executed on endpoints with Linux, Windows and MacOS. In order to use Metta, an organization needs to install Redis server, Python 2.7 and Vagrant, which makes the installation process a bit more complicated than other threat emulators.

Metta provides built-in attacks, each of which executes all the techniques that achieve a specific tactic. For example, an attack that consists of techniques all of which achieve collection of a user's data. Such an attack does not emulate the complete attack cycle, rather focusing on a wide assessment of specific defense targets.

Metta also provides the operator with the ability to create custom multi-procedure attacks. Metta supports CNC and lateral movement tactics. But lateral movement procedures are only implemented for Linux. Metta also provides various techniques for achieving discovery, credential access and defense evasion tactics (ten, seven, and seven techniques, respectively). Persistence, privilege escalation, collection and exfiltration tactics are supported by Metta with few techniques only. Most of the procedures implemented in Metta were not detected by anti-virus tools.

Metta has no logging or clean up functionalities.

Consequently, Metta is suitable for operators who need to test specific parts of endpoint security across many platforms.

11) Atomic Red Team: Atomic Red Team a threat emulation library of lightweight security tests that can be rapidly executed by security teams[18]. Attack scenarios can be executed using command line, PowerShell, and Shell. It is compatible with all major operating systems. Atomic Red Team also provides an API written in Ruby.

Atomic Red Team supports multiple techniques for achieving persistence, privilege escalation, defense evasion, credential access, and discovery tactics (20, 9, 24, 9 and 16 techniques, respectively). It also includes procedures for lateral movement and simulates CNC communication. All procedures are mapped to the MITRE ATT&CK matrix. It is possible to add new custom procedure scripts.

In addition, Atomic Red Team uses third-party tools, such as mimikatz[19] for performing credential dumping technique. Most of the procedures are undetected by anti-virus tools. Atomic Red Team lacks logging and cleanup functionalities. Documentation is provided only as comments in the procedures' scripts.

One of the main objectives of Atomic Red Team is enabling the operator to execute procedures quickly with none to minimal installation required, however, Atomic Red Team may require cyber-security knowledge in order to be used effectively.

[17] https://github.com/uber-common/metta

[18] https://atomicredteam.io/,https://github.com/redcanaryco/atomic-red-team

[19] https://github.com/gentilkiwi/mimikatz, https://www.varonis.com/blog/what-is-mimikatz/

VI. Taxonomies of Threat Emulators

In this section we present two taxonomies of the reviewed threat emulators. The Technical Compatibility taxonomy, presented in Figure 4, categorizes the reviewed threat emulators according to their execution type and their environment evaluation capabilities. The Attack's Complexity and Diversity taxonomy, presented in Figure 5, illustrates the trade-off between the knowledge required from the operator and integration difficultly vs. the versatility and attack complexity.

A. Threat Emulator's Technical Compatibility

The purpose of this taxonomy is to help an operator in choosing a suitable emulator for his needs based on the system's architecture and the test subject (a set of connected endpoints or a single endpoint).

a) Execution Type:

Script Based Execution. A threat emulation is launched via a script or a set of scripts. In script-based threat emulators each script is a procedure that executes a certain technique, thus a collection of scripts can represent an attack. Script based attacks are more "code" oriented; customizing attacks and launching multi-procedure attacks may require coding knowledge.

Interface Based Execution. A threat emulation with an interface is simpler to run, since most of the interaction is via the interface.

b) Lateral Movement Capability:

Not Available. No simulation of lateral movement can be done by the threat emulator, hence the emulator is suitable for endpoints only.

Available. A threat emulator can perform lateral movement, thus giving the ability to test network defenses and network isolation.

In addition, Figure 4 shows which operating systems are supported by each threat emulator, whether an emulator has traces cleanup capability, the capability of adding new procedures and techniques, if it is possible to choose the order in which the machines will be attacked, and if it has the possibility of stopping an attack during its execution.

B. Attack's Complexity and Versatility vs. Required expertise and Difficulty of Integration

The taxonomy presented in Figure 5 illustrates the trade-off between the versatility and complexity of attacks supported by a threat emulator and the knowledge and effort required to operate it.

On the one hand, we ranked threat emulators according to their ability to express attacks. Higher versatility, as

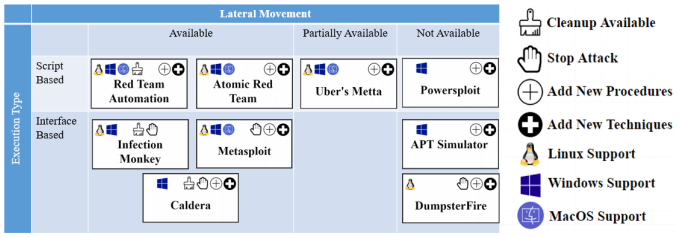

Fig. 4. Technical Compatibility Taxonomy

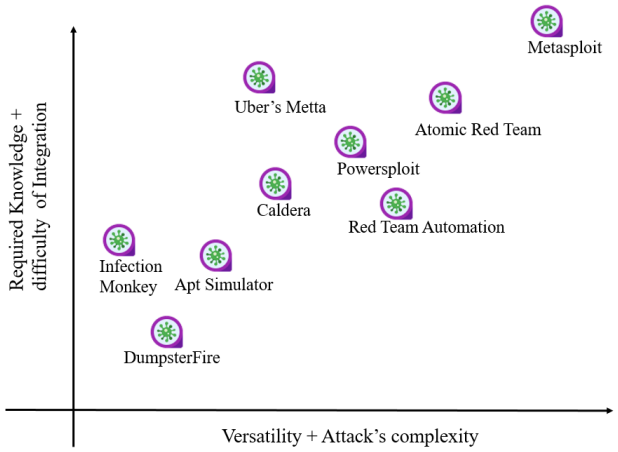

Fig. 5. Attack's Complexity and Diversity Taxonomy

mentioned in previous sections, means that more attack scenarios can be created, more tactics can be examined and better understanding of the security array limitations can be gained. Attack's complexity represents the quality and sophistication of attacks. The more complex an emulated attack is, the better the chances it has to resemble a real attack. Presumably, a machine or a set of connected machines that can withstand a high complexity emulated attack, will be able to defend against a real, similar attack.

On the other hand, an operator can be required to have significant knowledge and training to being able to successfully operate a certain threat emulator. Moreover, the more preliminary installations (e.g., database, agents, server, etc.) are required, the longer it takes to integrate the emulator and prepare it for operation.

As can be seen in Figure 5, the higher a threat emulator on the versatility and attack complexity scale, the higher it is on the required knowledge and difficulty of integration scale.

C. Guidelines for Selecting Threat Emulators

Based on the above taxonomies, we formulate guidelines for choosing the appropriate threat emulator given a specific environment and a set of security assessment tasks.

- When checking how exploitable a machine is or how easy it is to get from a certain entry point in a network to a specific machine (by using exploits and malicious activity), an operator should choose a threat emulator with lateral movement option (in order to traverse from one machine to another). Such emulators can be seen in Figure 4.
- When examining the effectiveness of measures for data protection (defending assets such as credentials, important files, configurations, etc.) an operator should look for a threat emulator that has a many (not necessarily sophisticated) techniques. The purpose is to check as many techniques as possible.
- If the purpose is to check if a software is better than baseline defenses, such as anti-virus and firewall, the operator should use a threat emulator with high or medium amount of procedures that cannot be detected by an anti-virus. That is to say, a threat emulator with high level of complexity should be used.
- When the purpose is to test if specific software is good at least as the default anti-virus of Windows, an operator should use a threat emulator with procedures that can be detected by anti-virus and check if the examined software also detects them. This test can be regarded as the baseline test for the examined software.
- If the purpose is to check whether the software can find advanced persistence threat attacks, an operator should use a threat emulator that can reflect the software's abilities against APT's. Therefore, the threat emulator should have the following characteristics: high or medium level of complexity, support for multi-procedure attacks, high amount (more than 30) of undetectable procedures, lateral movement, and an option to add more procedures and techniques.
- Experience and knowledge of the operator is an essential factor when deciding which emulator to use. Some emulators such as Uber's Metta may require coding knowledge to create, launch and manage attacks. In some circumstances a simple threat emulator is sufficient, while in others there is no choice but to choose a threat emulator that requires a qualified experts. For example, when testing tools or security systems that monitor operating system (e.g., changes to registry keys, file creation or DNS requests), simple to operate threat emulators, such as APTSimualtor or DumpsterFire, are sufficient. In contrast, when testing advanced IoCs (e.g., process injection, file encryption or credential dumps) threat emulators, such as PowerSploit and Red Team Automation, are required. Moreover, if the operator wishes to execute real CVEs and create real-life attacks, it is needed to use threat emulators that may require experts to operate correctly the attacks or even to extend or create the attacks.
- Integration difficulty is something that could affect the usefulness of a threat emulator in some situations. Time and effort spent on configuring and deploying a threat emulators can vary between emulators. If the goal of assessment is to perform baseline checks and launch a simple attack on a machine, it is advised to choose an emulator that does not require difficult integration. For instance,APT Simulator or Infection monkey

Figure 6 presents a high-level flow of selecting a threat

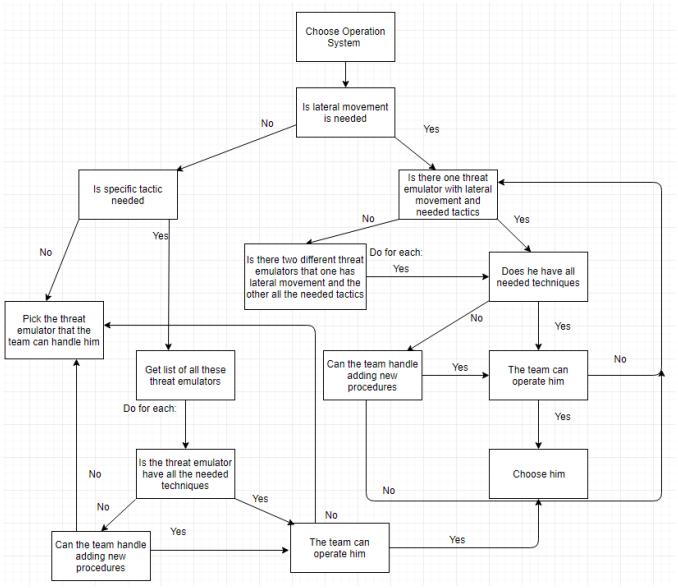

Fig. 6. Choosing the fittest threat emulator

emulator.

1) It is essential to determine what execution environment will be tested. Consequentially, one should look at the technical compatibility taxonomy (see Figure 4) and choose a group of threat emulators that work on the chosen platforms.
2) The next consideration is the necessity of lateral movement. The right column in the technical compatibility taxonomy (see Figure 4) shows the emulators with lateral movement capabilities.
   If lateral movement is required alongside other tactics, there are two options: 1) take two separate threat emulators, one for lateral movement tactics and one for the other tactics; 2) use a single threat emulator that contains all the tactics we need. The reason the first option may be preferred over the later is the fact that threat emulators with high number of tactics and procedures may require previous knowledge and not always simple to operate.
3) The coding knowledge of the operator is relevant when deciding which emulator to use. If an operator is familiar with scripting languages such as Python, batch or PowerShell he can use script based threat emulators to manage the procedures and create new attack scenarios. If the operator lacks coding knowledge, a tool with built-in attacks should be used (see the last row in Figure 4).
4) Threat emulators with high amount of diverse techniques may require cyber-security knowledge, due to the fact that high number of different techniques require high knowledge about their usage and purpose. When checking basic and standard techniques such as file, registry and user manipulation, the simplest threat emulator can be used to emulate these kind of techniques without any previous knowledge and without any effort from the operator. Attacks that contain a high amount of techniques may have sophisticated procedures and, thus, require the operator to understand how to use the procedures to implement the attacks.
5) An operators who wants to test high number of scenarios requires a tool with a high number of procedures and techniques. High number of procedures gives the operator the ability to test high amount of triggers on his system and evaluate their effect in various ways.
6) Once an operator tries to check how penetrable, i.e., vulnerable to initial access and execution tactics an organization is, the operator must consider using Metasploit, since most of the CVEs that Metasploit employs implement initial access and execution techniques. Metasploit gives the operator the ability to deploy malicious code after the initial access, thus giving the operator an option to continue the attack using third-party or self-written tools.

VII. Summary

Threat emulators are viable as a complementary tool for personnel training, security assessment of a product or the organizational environment, and what-if analysis. We reviewed eleven open-source threat emulators and presented a detailed and comprehensive evaluation methodology for qualitative and quantitative comparison between them. The review is based on well-defined criteria discussed in Section IV. We also presented two different taxonomies which can help an operator choose the right emulator for the right job in an intuitive manner: (1) a taxonomy that categorizes and labels threat emulators according to various technical criteria; and (2) a taxonomy that shows a trade-off between the expertise required from an operator and the attack complexity that can be achieved using a threat emulator. The review methodology with respect to Section IV, the detailed reviews of threat emulators, and the taxonomies allowed us to deduce guidelines for choosing the most appropriate threat emulator in a given situation.

Finally, Section V and Section VI can guide security personals when designing a new threat emulator or offensive security tools. To the best of our knowledge, no such survey was done on this subject.

REFERENCES

[1] sqlmap: Automatic sql injection and database takeover tool. http://sqlmap.org/. Accessed on: 2019-11-10.

[2] w3af: Web application attack and audit framework. http://w3af.org/. Accessed on: 2019-11-10.

[3] Gary Adkins. Red teaming the red team: Utilizing cyber espionage to combat terrorism. *Journal of Strategic Security*, 6(3):1–9, 2013.

[4] 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*, ACSAC '16, pages 363–373, New York, NY, USA, 2016. ACM.

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

[6] Sean Barnum. Common attack pattern enumeration and classification (capec) schema description. *Cigital Inc, http://capec.mitre.org/documents/documentation/CAPEC_Schema_Description_v1*, 3, 2008.

[7] M. Bishop. About penetration testing. *IEEE Security Privacy*, 5(6):84–87, Nov 2007.

[8] Marcel Böhme, Van-Thuan Pham, Manh-Dung Nguyen, and Abhik Roychoudhury. Directed greybox fuzzing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 2329–2344, New York, NY, USA, 2017. ACM.

[9] The MITRE Corporation. Caldera. www.mitre.org/research/technology-transfer/open-source-software/caldera, April 2019. Accessed on: 2019-11-10.

[10] Patrick Engebretson. *The basics of hacking and penetration testing: ethical hacking and penetration testing made easy*. Elsevier, 2013.

[11] Mike Gault. The cia secret to cybersecurity that no one seems to get, Dec 2015.

[12] Walter M Grayman, Avi Ostfeld, and Elad Salomons. Locating monitors in water distribution systems: Red team–blue team exercise. *Journal of water resources planning and management*, 132(4):300–304, 2006.

[13] Jürgen Großmann and Fredrik Seehusen. Combining security risk assessment and security testing based on standards. In Fredrik Seehusen, Michael Felderer, Jürgen Großmann, and Marc-Florian Wendland, editors, *Risk Assessment and Risk-Driven Testing*, pages 18–33, Cham, 2015. Springer International Publishing.

[14] D. S. Henshel, G. M. Deckard, B. Lufkin, N. Buchler, B. Hoffman, P. Rajivan, and S. Collman. Predicting proficiency in cyber defense team exercises. In *MILCOM 2016 - 2016 IEEE Military Communications Conference*, pages 776–781, Nov 2016.

[15] Pete Herzog. Open-source security testing methodology manual. *Institute for Security and Open Methodologies (ISECOM)*, 2003.

[16] Filip Holik, Josef Horalek, Ondrej Marik, Sona Neradova, and Stanislav Zitta. Effective penetration testing with metasploit framework and methodologies. In *2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 237–242. IEEE, 2014.

[17] Christian Holler, Kim Herzig, and Andreas Zeller. Fuzzing with code fragments. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 445–458, Bellevue, WA, 2012. USENIX.

[18] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[19] Joint Task Force Transformation Initiative. Security and privacy controls for federal information systems and organizations. *NIST Special Publication*, 800(53):8–13, 2013.

[20] Hamdi Kavak, Jose J Padilla, and Daniele Vernon-Bido. A characterization of cybersecurity simulation scenarios. In *Proceedings of the 19th Communications & Networking Symposium*, page 3. Society for Computer Simulation International, 2016.

[21] Devon Kerr. Introducing endgame red team automation. www.endgame.com/blog/technical-blog/introducing-endgame-red-team-automation, March 2018. Accessed on: 2019-11-10.

[22] Robert M. Lee, Michael J. Assante, and Tim Conway. Analysis of the cyber attack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 2016.

[23] Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, USA, 2009.

[24] Steve Mansfield-Devine. The best form of defence the benefits of red teaming. *Computer Fraud & Security*, 2018(10):8 – 12, 2018.

[25] Robin Mejia. Red team versus blue team: how to run an effective simulation. *CSO Online-Security and Risk*, 2008.

[26] J. Mirkovic, P. Reiher, C. Papadopoulos, A. Hussain, M. Shepard, M. Berg, and R. Jung. Testing a collaborative ddos defense in a red team/blue team exercise. *IEEE Transactions on Computers*, 57(8):1098–1112, Aug 2008.

[27] Michael Muehlberghuber, Frank K. Gürkaynak, Thomas Korak, Philipp Dunst, and Michael Hutter. Red team vs. blue team hardware trojan analysis: Detection of a hardware trojan on an actual asic. In *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, pages 1:1–1:8, New York, NY, USA, 2013. ACM.

[28] Jacob G. Oakley. *Professional Red Teaming: Conducting Successful Cybersecurity Engagements*. APress, 1st edition, 2019.

[29] Sandip Patel and Jigish Zaveri. A risk-assessment model for cyber attacks on information systems. *Journal of computers*, 5(3):352–359, 2010.

[30] Paul Pols. The unified kill chain: designing a unified kill chain for analyzing, comparing and defending against cyber attacks. Master's thesis, Cyber Security Academy, The Hague, December 2017.

[31] Shirley Radack. Guide to information security testing and assessment. Technical report, National Institute of Standards and Technology, 2008.

[32] J. Rajendran, V. Jyothi, and R. Karri. Blue team red team approach to hardware trust assessment. In *2011 IEEE 29th International Conference on Computer Design (ICCD)*, pages 285–288, Oct 2011.

[33] S. Rastegari, P. Hingston, Chiou-Peng Lam, and M. Brand. Testing a distributed denial of service defence mechanism using red teaming. In *2013 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pages 23–29, April 2013.

[34] Aunshul Rege and Joe Adams. The need for more sophisticated cyber-physical systems war gaming exercises. In *ECCWS 2019 18th European Conference on Cyber Warfare and Security*, page 403. Academic Conferences and publishing limited, 2019.

[35] Russ Rogers. *Nessus network auditing*. Elsevier, 2011.

[36] Yaroslav Stefinko, Andrian Piskozub, and Roman Banakh. Manual and automated penetration testing. benefits and drawbacks. modern tendency. In *2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*, pages 488–491. IEEE, 2016.

[37] Blake E Strom, Joseph A Battaglia, Michael S Kemmerer, William Kupersanin, Douglas P Miller, Craig Wampler, Sean M Whitley, and Ross D Wolf. Finding cyber threats with att&ck-based analytics. Technical report, Technical Report MTR170202, MITRE, 2017.

[38] Nextron Systems. Aptsimulator. https://github.com/NextronSystems/APTSimulator. Accessed on: 2019-11-10.

[39] Colin Tankard. What the gdpr means for businesses. *Network Security*, 2016(6):5–8, 2016.

[40] Paul Voigt and Axel Von dem Bussche. The EU general data protection regulation (GDPR). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 2017.

[41] A. Waksman, J. Rajendran, M. Suozzo, and S. Sethumadhavan. A red team/blue team assessment of functional analysis methods for malicious circuit identification. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–4, June 2014.

[42] T. Wang, T. Wei, G. Gu, and W. Zou. Taintscope: A checksum-aware directed fuzzing tool for automatic software vulnerability detection. In *2010 IEEE Symposium on Security and Privacy*, pages 497–512, May 2010.

[43] Steffen Weiss. Industrial approaches and standards for security assessment. In *Dependability metrics*, pages 166–175. Springer, 2008.

[44] JD Work. In wolf's clothing: Complications of threat emulation in contemporary cyber intelligence practice. In *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–8. IEEE, 2019.

[45] X. Zhang, K. Xiao, M. Tehranipoor, J. Rajendran, and R. Karri. A study on the effectiveness of trojan detection techniques using a red team blue team approach. In *2013 IEEE 31st VLSI Test Symposium (VTS)*, pages 1–3, April 2013.

VIII. APPENDIX: EMULATOR EVALUATION QUESTIONNAIRE

A. Environment

Questions this section relate to the interactions between the emulator and the environment where it is deployed. The following questions relate to the emulator's main components and dependencies.

	Yes	No
Q1: Does the emulator use agents?	☐	☐
Q2: Does the emulator executes scripts on the endpoints?	☐	☐
Q3: Does the emulator executes third party tools on the endpoints?	☐	☐

1) OS compatibility: Out of the following questions, answer only the first question relevant to the emulator. W, L, and M refer to the Windows, Linux and Mac operating systems respectively.

	W	L	M
Q4: [Q1=Yes] Are the emulator's agents compatible with the followings operating systems?	☐	☐	☐
Q5: [Q2=Yes] Does the emulator contain procedures compatible with the followings operating systems?	☐	☐	☐
Q6: [Q3=Yes] Does the emulator use third party tools compatible with the followings operating systems?	☐	☐	☐

2) Changes to security array: Out of the following questions, answer only the first question relevant to the emulator.

	Yes	No
Q7: Did the system's firewall interrupt with the emulator work flow?	☐	☐
Q8: [Q7=Yes] Did the system's firewall blocked the connection between the CNC and the RAT?	☐	☐
Q9: Did the system's real-time anti-virus interrupt with the emulator work flow?	☐	☐
Q10: [Q9=Yes] Did the system's real-time anti-virus interrupted with any RAT functionality?	☐	☐
Q11: [Q9=Yes][Q1=Yes] Did the system's real-time anti-virus interrupted with the agents functionality?	☐	☐
Q12: [Q9=Yes][Q2=Yes] Did the system's real-time anti-virus interrupted with the scripts functionality? (i.e delete powershell scripts)	☐	☐
Q13: [Q9=Yes][Q3=Yes] Did the system's real-time anti-virus interrupted with the third party tools functionality?	☐	☐

3) Prerequisites: Out of the following questions, answer only the first question relevant to the emulator.

	Yes	No
Q14: [Q1=Yes] Do the emulator's agents require special privileges to operate?	☐	☐
Q15: [Q1=Yes] Do the emulator's agents require special privileges from the systems to take advantage of their full functionality?	☐	☐
Q16: [Q2=Yes] Do the emulator's scripts require special privileges to operate?	☐	☐
Q17: [Q2=Yes] Do the emulator's scripts require special privileges from the systems to take advantage of their full functionality?	☐	☐
Q18: [Q3=Yes] Do the emulator's third party tools require special privileges to operate?	☐	☐
Q19: [Q3=Yes] Do the third party tools require special privileges from the systems to take advantage of their full functionality?	☐	☐

B. Operators

Questions in this section asses the ease of operating of the emulator.

1) Required security expertise: Consider a novice red team member, such as software engineer or computer science student, who is technology savvy but without cyber security expertise.

	Yes	No
Q20: Can the novice red team member execute built-in attacks by following instructions provided with the emulator?	☐	☐

2) Documentation:

	Yes	No
Q21: Does the emulator have any kind of documentation?	☐	☐
Q22: [Q21=Yes] Is the documentation sufficient to setup all the emulator's components?	☐	☐
Q23: [Q21=Yes] Is the documentation sufficient to launch build-in attacks?	☐	☐
Q24: [Q21=Yes][Q39=Yes] Is the documentation sufficient to create new custom procedures?	☐	☐
Q25: [Q21=Yes][Q46=Yes] Is the documentation sufficient to create new multi-procedure attack scenarios?	☐	☐
Q26: [Q21=Yes] Does the documentation describe how to interpret the attack's execution results?	☐	☐

3) Interface: In the following matrix fill in whether or not the functionality (row) is available through the UI (column).

	GUI	CLI	API
Q27: Executing attacks	☐	☐	☐
Q28: Configuring procedures	☐	☐	☐
Q29: Stopping attacks mid run-time	☐	☐	☐
Q30: [Q37=Yes] Accessing logs	☐	☐	☐
Q31: [Q39=Yes] Adding new custom procedures	☐	☐	☐
Q32: [Q46=Yes] Adding new multi-procedure attacks	☐	☐	☐

	Yes	No
Q33: Does the emulator support scripting of attack scenarios?	☐	☐

C. Scenario execution

Questions in this section asses to the emulator's functionality related to attack scenario execution.

1) Cleanup:

	Yes	No
Q34: Does the emulator have cleanup functionality?	☐	☐
Q35: [Q34=Yes] Must the cleanup occur after every individual attack procedure?	☐	☐
Q36: [Q34=Yes][Q46=Yes∨Q46=Yes] Can the cleanup be executed only at the end of a multi-procedure attack?	☐	☐

2) Logs:

	Yes	No
Q37: Does the emulator have logging capabilities?	☐	☐
Q38: [Q37=Yes] Is every executed procedure logged during an attack?	☐	☐

D. Scenario definition

Questions in this section asses the functionality related to defining and configuring attack scenarios.

1) Adding new procedures:

	Yes	No
Q39: Does the emulator support adding new custom procedures through either one of the interfaces?	☐	☐
Q40: [Q39=No] Are procedures implemented using scripts?	☐	☐
Q41: [Q39=No][Q40=Yes] Can a new script be added to the collection of procedures?	☐	☐

2) Procedures' configuration:

	Yes	No
Q42: Can the emulator procedures be configured through configuration files?	☐	☐
Q43: [Q42=No][Q28=GUI] Does the emulator support repeated execution of the same attack with different parameters?	☐	☐
Q44: [Q39=Yes] Can new custom procedures be configured using the same methods as built-in procedures (e.g. various interfaces or configuration files)?	☐	☐

3) Multi-procedure attacks:

	Yes	No
Q45: Does the emulator include built-in multi-procedure attacks?	☐	☐
Q46: Does the emulator support adding new custom multi-procedure attacks?	☐	☐

4) TTP coverage: In order to answer the following questions, fill-in the MITRE ATT&CK matrix for the emulator considering only built-in procedures.

	<6	6 to 8	>8
Q47: The number of tactics is	☐	☐	☐
	<20	20 to 40	>40
Q48: The number of techniques is	☐	☐	☐