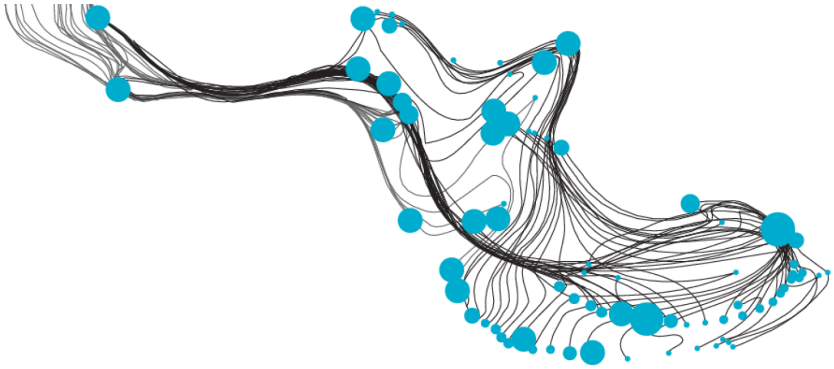

MSc Thesis Computer Science

Autonomous Emulation of Adversary Procedures in the (Pre-)Compromise Domain

D.R. Bakker
d.r.bakker-1@alumnus.utwente.nl

April 21, 2022

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)
Services and Cybersecurity Group (SCS)

Committee:
Prof. Dr. Andreas Peter
Dr. Mattijs Jonker
Tijme Gommers (Northwave)
Robert Diepeveen (Northwave)

UNIVERSITY OF TWENTE.

NORTHWAVE
Intelligent Security Operations

ABSTRACT

In a world with ever-evolving digital threat, offensive testing in the form of adversary emulation has become an important means of keeping organisations secure. Generally, this process is manually carried out by red teams, but carries several limitations - mainly those of time, cost and consistency - hindering effectiveness, accuracy and widespread adoption. Automation could enhance manual adversary emulation, with additional benefits to security control validation and security control development use cases. Automation efforts have been made in the form of Autonomous Adversary Emulation (AAE) and Breach and Attack Simulation (BAS) solutions, but they largely focus on post-compromise adversary behaviour. In this work, we investigate the potential for autonomous emulation of adversaries in their (pre-)compromise procedures. Through a threat intelligence based approach, we implement a platform for autonomous (pre-)compromise adversary emulation in the form of an extension to MITRE CALDERA, a state-of-the-art AAE framework. Using this extension as a vehicle for further analysis, we identify several domain-specific limitations and challenges that currently exist in AAE frameworks, to which we propose high-level solutions. Finally, we show its behaviour in a dynamic testing range and explore its performance in the context of several real-world applications - showing a measurable improvement over related automation efforts in the process.

CONTENTS

ACRONYMS

AAE Autonomous Adversary Emulation

AD Active Directory

API Application Programming Interface

BAS Breach and Attack Simulation

C2 Command and Control

CART Continuous Automated Red Teaming

CIDR Classless Inter-Domain Routing

CTI Cyber Threat Intelligence

CKC Cyber Kill Chain

FQDN Fully Qualified Domain Name

IAB Initial Access Broker

IOC Indicator of Compromise

NAT Network Address Translation

OSINT Open Source Intelligence

RAAS Ransomware-as-a-Service

RAT Remote Access Trojan

RCE Remote Code Execution

RDP Remote Desktop Protocol

SDO STIX Domain Object

SIEM Security Information and Event Management

SMTP Simple Mail Transfer Protocol

SSH Secure Shell

SSRF Server-Side Request Forgery

STIX Structured Threat Information Expression

TIBER Threat Intelligence-Based Ethical Red-Teaming

TTPS Tactics, Techniques and Procedures

UKC Unified Kill Chain

VPN Virtual Private Network

WINRM Windows Remote Management

1 INTRODUCTION

In an increasingly digital world, malicious threat actors and organisations find themselves playing an ever-evolving tug-of-war. In order for organisations to stay on top of this game, it is ideally continuously emulated by red and blue teams, which play the roles of attacker and defender respectively. Red teaming is an exercise in cybersecurity where ethical hackers attempt to breach defensive controls of an organisation, often with a very broad scope and with the goal of reaching an organisation's most valuable assets. In order to correctly determine whether an organisation is vulnerable to certain real-world threat actors, it is desirable to execute realistic attacks during red team engagements. By gathering relevant Cyber Threat Intelligence (CTI) and modelling realistic attack scenarios, adversaries can be accurately emulated by a red team.

Well-executed red team engagements prove to be very effective way of strengthening the security posture of organisations. However, these engagements are only fully effective when a mature blue team is in place and when they are conducted on a regular basis. In practice, not all organisations have the resources to meet these requirements. Making offensive security tests more accessible could help these organisations in bringing their security posture to a sufficient level.

Automation has always had a big role in the offensive security industry, but more complex tests like adversary emulation are still a highly manual process. A development that has been gaining traction is the concept of Autonomous Adversary Emulation (AAE). Between 2016 and 2018, researchers at MITRE wrote several papers on the use cases for autonomy in this domain and the fundamental challenges that are faced in building AAE solutions. They note that a high majority of automation efforts in the offensive security domain are focused on the technique level, while AAE solutions aim to extend this to the tactical level [3]. Given a highly sophisticated planning engine and a wide range of adversary profiles and techniques, this could conceivably result in autonomous emulation of large parts of the adversary lifecycle. These developments would make offensive security operations more accessible and continuous, further empowering blue teams to test and harden security controls and allowing them to assess gaps in their defenses at an earlier stage.

Based on their research, MITRE developed and published CALDERA[1] in 2019. The framework, although highly experimental and still having large

1 https://github.com/mitre/caldera

shortcomings, has shown its potential. In subsequent years, a similar concept found its way to the industry and started appearing on the market in the form of *breach and attack simulation* or *security validation* tools. Compared to research done at MITRE and the implementation of CALDERA, commercial implementations are mostly geared towards *automatically simulating* individual steps in the adversary lifecycle against controlled and known environments, rather than *autonomously emulating* the path of an adversary in unknown environments. While this turns out to be a large difference, lots of similarities exist between these categories of tools as well - sharing many practical use cases and challenges.

1.1 MOTIVATION

While existing efforts in implementing AAE already provide a rather strong foundation in emulating threat actors in their post-compromise Tactics, Techniques and Procedures, they fall short in emulating the (pre-)compromise[2] steps of the adversary lifecycle. Similarly, many offerings in the Breach and Attack Simulation (BAS) market are still lacking support in the (pre-)compromise domain [6].

Various reasons are central to why this domain is not receiving priority in the implementation of these solutions:

1. The well-adopted principle of assumed breach
2. The fact that the pre-compromise domain was only recently recognized in the MITRE ATT&CK framework [19]
3. Inherent challenges in detection and mitigation of threat in the pre-compromise domain
4. Relatively large amounts of uncertainty from the point of view of an attacker, compared to post-compromise operations
5. Fundamental challenges in autonomously running operations across multiple hosts, that haven't entirely been solved in the AAE domain yet

While the application of the principle of assumed breach - threat actors will eventually always find a way in - will turn out to be favorable in certain circumstances, there is still a lot of value in making it more difficult for attackers to breach the virtual perimeter. The activity in the (pre-)compromise domain is well-observed by means of listings of Initial Access Brokers (IABs), threat actors that specifically focus on the (pre-)compromise stages of the adversary lifecycle with the goal of selling initial access to the highest bidder. A report from Digital Shadows shows that 2020 has seen the largest number of online listings from IABs, and 2021 is on track to be surpassing that number [25]. They find that these threat actors will often target low-hanging

2 reconnaissance, resource development, gaining foothold and implanting C2

fruit - precisely the category of where automatic security testing can make a difference.

Additionally, exploring and partially solving AAE for the (pre-)compromise domain will likely prove to be a valuable testbed for multi-host operation, because of the inherent requirement to laterally move between hosts in completing the (pre-)compromise stage of the adversary lifecycle. Finally, being able to autonomously emulate adversarial activity in this domain will build understanding on the defensive side, providing valuable input for security decisions.

1.2 RESEARCH QUESTION

In this work, we will explore the room for automation of the emulation of adversaries in their (pre-)compromise Tactics, Techniques and Procedures and close the gap in the current range of Autonomous Adversary Emulation and Breach and Attack Simulation solutions. After exploring the modus operandi of threat actors and collecting a representative set of emulation scenarios, we construct a basis for autonomous pre-compromise adversary emulation in the form of an extension to a state-of-the-art autonomous adversary emulation framework. We define one main research question that is central to this work:

RQ How can threat actors be emulated in their (pre-)compromise tactics, techniques and procedures in an autonomous fashion?

To give further structure in answering this research question, we define the following milestones for the research phase:

MS 1 Understanding the modus operandi of threat actors during the (pre-)compromise phase of the adversary lifecycle (reconnaissance, resource development and initial foothold)

MS 2 Constructing a representative set of emulation scenarios covering the (pre-)compromise phase

MS 3 Integrating the previously defined set of emulation scenarios into a plugin for an adversary emulation framework

MS 4 Validating the plugin against a test environment and testing its application to real-world use cases

Additionally, we deem the following concerns of high importance while working towards answering the main research question:

- Optimizing towards strategies that yield opportunities for detection and mitigation, ensuring future potential towards autonomous purple teaming and immediate usefulness for blue teams

- Reducing manual operator input and intervention where possible in terms of (1) development of new emulation scenarios and (2) providing the tool with starting input and/or constraints, ensuring accessibility towards defenders

1.3 CONTRIBUTIONS

This work delivers three main contributions. Our first contribution is a review of existing work in the domain of automated adversary emulation. We present an exploration and comparison of several existing approaches. Second, we provide - to the best of our knowledge - the first fundamental effort at autonomously emulating threat actors in the initial stages of the adversary lifecycle, by extending a state-of-the-art AAE framework. This effort will provide a valuable basis for implementing additional adversary emulation scenarios that concern the (pre-)compromise domain. In this research however, we use this effort as a vehicle for further analysis, allowing us to identify several limitations and challenges - specific to emulating threat actors in the first stages of the adversary lifecycle - that exist in AAE frameworks. Our final main contribution, which should be of independent interest, comes in the form of high-level theoretical solutions to specific AAE framework-level gaps.

Along with the previous contributions, we make several sub-contributions. The first is a frequency analysis on technique usage in the (pre-)compromise domain, which is based on several sources and types of CTI. Through this analysis, we devise several scenarios that together reach notable coverage of the modus operandi of threat actors in the first stages of the adversary lifecycle. These scenarios form a common ground in characterizing and reasoning about the role and challenge of automation in the adversary emulation process. Finally, through the evaluation of our *proof-of-concept*, we show where our solution fits in the broader landscape of automation in offensive security and provide ideas on the objective evaluation of such solutions.

1.4 OUTLINE

CHAPTER 2 builds the context for this work - we explore relevant literature and engineering work on adversary modeling, threat intelligence and (autonomous) adversary emulation.

CHAPTER 3 provides a technique-level analysis of the modus operandi of threat actors in the (pre-)compromise domain based on several types of Cyber Threat Intelligence (CTI) sources. Additionally, it introduces a carefully balanced set of emulation scenarios that will be used for the implementation of a *proof-of-concept* for emulating adversaries in the (pre-)compromise domain.

CHAPTER 4 describes the implementation of this *proof-of-concept* - an extension to a state-of-the-art post-compromise Autonomous Adversary Emulation (AAE) framework. This extension has two main components: a platform that facilitates autonomous emulation of adversaries in the (pre-)compromise domain, and the previously introduced scenarios that are built on top of that platform. Both of these components will be dicussed comprehensively. Finally, this chapter proposes several theoretical solutions to gaps that have been identified at the framework level while implementing the *proof-of-concept*.

CHAPTER 5 provides an evaluation of this *proof-of-concept*. We show how the *proof-of-concept* performs in a dynamic testing range. Moreover, we evaluate potential applications, and where possible, evaluate performance compared to existing work.

CHAPTER 6 concludes this work. We provide a concrete answer to our research question, show any limitations that apply to this research and present several directions for further research.

2

BACKGROUND AND RELATED WORK

In this chapter, we introduce preliminary knowledge that forms the basis for the research presented in this thesis. In section 2.1, we explore the practices in the industry that keep enterprises digitally secure, focusing on manual adversary emulation. Section 2.2 introduces previous work related to adversary modeling and the different tactical phases that build up a cyber attack. In section 2.3, we explore the field of Cyber Threat Intelligence (CTI), with a focus on types, sources and dissemination formats. Finally, section 2.4 evaluates the domain of Autonomous Adversary Emulation (AAE), exploring several solutions and illustrating the typical architecture of an AAE framework.

2.1 RED TEAMING AND ADVERSARY EMULATION

Several means exist to keep organisations safe against digital threat. One of the proactive methods that can be employed is *adversary emulation* - closely mimicking real adversaries in their Tactics, Techniques and Procedures (TTPs) based on relevant threat intelligence, before real threat comes along. This process is usually carried out by *red teams*. In this section, we seek to explore the practice of manual adversary emulation, prior to exploring the concept of Autonomous Adversary Emulation (AAE).

2.1.1 *Simulation and emulation*

When reasoning about the replication of adversary behaviour, both 'adversary emulation' and 'adversary simulation' seem to be used interchangeably. Popular red teaming frameworks, like TIBER [11] and AASE [30], denote the exercise as 'simulation', while 'emulation' is commonly used by automated solutions. For lack of an official distinction between the two terms in the context of offensive security, we will follow the definition of both terms applied to this domain, and the anaylsis of NVISO Labs based on practical examples [14].

According to the Cambridge dictionary, the definition of emulation is "to copy something achieved by someone else and try to do it as well as they have"[1], while the definition of simulation is "to do or make something that looks real but is not real"[2]. Applying these definitions to offensive security would yield the following differentiation [14]:

1 https://dictionary.cambridge.org/dictionary/english/emulate
2 https://dictionary.cambridge.org/dictionary/english/simulate

EMULATION Staying close to the TTPs of a specific threat actor, executing the attack just like they would. Availability of accurate threat intelligence on a specific threat actor is an important prerequisite. Many practical constraints exist which make real emulation very hard or even impossible to carry out, and actual emulation is therefore not necessarily the status quo in offensive security.

SIMULATION When executing an attack simulation, aspects of real attacks are being used. This allows red teams to be more creative and use adversary TTPs as they see fit. From the defensive side, this might look like a real attack, while it is only a simulation.

2.1.2 *Process*

A strong foundation in formalising an approach towards adversary emulation has been laid out by financial regulators in several jurisdictions. This was stimulated by the G7 Cyber Expert Group, who, in an effort to increase consistency across penetration testing efforts, published the *G7 fundamental elements for threat-led penetration testing for the financial sector* [12]. From these fundamentals, the process was formalised by the European Union in the Threat Intelligence-Based Ethical Red-Teaming (TIBER) framework [11]. While designed for the defense of the core financial infrastructure of the European Union, the framework is sector-agnostic to large extent. Taking the most critical technical phases from the TIBER-EU framework yields the following process for a test:

1. The scope of the test is determined between the tested entity, the threat intelligence provider and the red team. This scope is generally very broad to make the test as realistic as possible.
2. A threat intelligence provider passively gathers intelligence on the tested entity, closely mimicking the reconnaissance steps that would be undertaken by threat actors, yielding an overview of the digital presence of the tested entity. Additionally, they report on the strategic understanding of the tested entity and its critical functions. Actors that form realistic threat towards the tested entity are gathered, including corresponding high-level attack scenarios.
3. The red team develops detailed and credible attack scenarios from the point of view of the adversary, based on the information received from the threat intelligence provider and their professional experience.
4. The red team executes the adversary emulation process, closely basing their actions on the previously developed attack scenarios while documenting their steps.
5. Together with the blue team of the tested entity, a *purple teaming* session is held. The goal of this session is to maximise the learning experience,

by comparing the actions executed by the red team to the findings of the blue team.

2.1.3 *Limitations*

Certain limitations exist in the accuracy of adversary emulation carried out by red teams. From literature, we identify four main differences that potentially influence the accuracy of an emulated attack. These differences emerge in either the time path of the attack or the usage of TTPs.

- The main motivation in red teaming engagements is improving blue team capabilities [15], while a real adversary will be mainly motivated towards reaching their objectives
- Red teaming engagements are generally strictly time-bound, while this is usually not the case for real targeted attacks [30]
- Real adversaries do not, or only selectively adhere to laws, ethics and/or social norms and can therefore go to greater lengths in achieving their objectives [30]
- Red teams are directly or indirectly under control of the organisation that is subject to a simulated attack, which is not the case during real targeted attacks [30]

2.2 ADVERSARY MODELING

Modeling the behaviour of threat actors is an essential part to both their emulation and the implementation of defensive controls. Describing adversarial TTPs using a common taxonomy forces a level of structure in both of these use cases and helps in communication and understanding between entities in the information security industry.

Adversary behaviour can be modeled at various levels of abstraction. We speak of a tactical level when considering high-level objectives of adversaries. Going down the ladder of abstraction, we reach the technique level, which describes how attackers reach their tactical objectives. On the procedural level, we look at the precise actions an adversary is taking to reach their technical goals. Procedures are the implementations of the techniques found on a higher level of abstraction.

2.2.1 *Tactical models*

CYBER KILL CHAIN Lending the concept of a kill chain from the military, Lockheed Martin introduced one of the first models for adversary behaviour in cyberspace, called the Cyber Kill Chain (CKC) [13]. The CKC consists of 7 high-level tactical phases, mostly describing behaviour until installing adversary-controlled software on the first host in the target network.

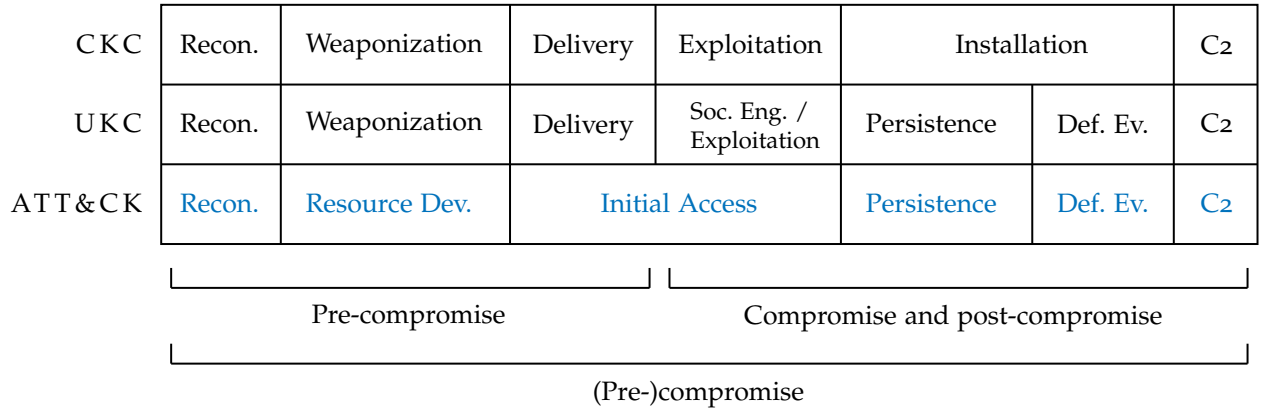

Figure 2.1: Comparing the tactical phases in the (pre-)compromise domain across kill chain models

The model is oftentimes considered outdated in the context of enterprise network attacks, because it does not consider adversarial objectives after breaching the network perimeter in great detail.

MITRE ATT&CK™ MITRE develops a knowledge base and model for adversarial behaviour called ATT&CK [27]. Among others, the framework contains an adversary model focusing on enterprise network attacks. ATT&CK describes adversary behaviour on both tactical and technical levels, and is therefore often visualised as a matrix with tactics as columns and techniques as rows. Tactics and techniques are labelled with unique identifiers, allowing the framework to be used as a reference for many different purposes in the field of cybersecurity. Up until ATT&CK version 8, preparatory techniques (reconnaissance and resource development) were covered in a separate domain called PRE-ATT&CK. Given that this is a very recent update, most literature still refers to PRE-ATT&CK when addressing the preparatory domain and many CTI providers do not cover preparatory techniques in their reports. In the most recent version of ATT&CK[3], preparatory techniques have been integrated into the Enterprise domain.

UNIFIED KILL CHAIN In 2017, Paul Pols introduced the Unified Kill Chain (UKC) [21]. The kill chain is constructed through a combination of the original Cyber Kill Chain as defined by Lockheed Martin, MITRE ATT&CK, and observations from APTs and Red Teams. The UKC is the most advanced kill chain model in terms of tactical complexity, containing 18 phases that describe the lifecycle of an attack.

2.2.2 *The (pre-)compromise domain*

Figure 2.1 shows the tactical phases that need - in general - be traversed by an adversary to gain an initial foothold and Command and Control (C2) in an

3 https://attack.mitre.org/resources/updates/updates-october-2020/

enterprise network environment. For the Lockheed Martin Cyber Kill Chain (CKC), this entails the full kill chain apart from its last phase. The framework is very much perimeter-focused and does not consider adversarial activity behind the perimeter in detail. The Unified Kill Chain (UKC) separates its phases into three high-level groups, of which the first one describes gaining initial foothold, containing the phases as portrayed in the figure. The MITRE ATT&CK framework is different - it is time-agnostic at a tactical level, but it is trivial to map its tactics to time-aware kill chain models like CKC and UKC. However, due to certain design decisions in the ATT&CK framework, it cannot be ruled out that techniques from tactics that are not shown in figure 2.1 will have to be used occasionally to reach the tactical goal of C2.

In the absence of a generally accepted name for this high-level phase of the adversarial kill chain, we will refer to the group of tactics in figure 2.1 as the *(pre-)compromise* domain. The point where an operation reaches the *post-compromise* phase generally depends on the attack path that is being followed, but following literature on Autonomous Adversary Emulation (AAE) implementations [5, 17], we will use *post-compromise* for all tactics that follow the installation of a C2 agent on the first host in the internal network.

Since its introduction, MITRE ATT&CK has become a widely popular adversary model in the industry, and it is the only framework in this comparison that also describes adversarial activity at the technique level. This is a useful feature when developing and reasoning about AAE solutions, and we will therefore continue using it as the adversary model in the remainder of this work. In section 2.2.2.1, we will shortly introduce each of the tactical phases in MITRE ATT&CK that are generally traversed by an adversary until gaining initial foothold and C2 on the first host in a target enterprise network.

2.2.2.1 *Tactical phases*

RECONNAISSANCE The reconnaissance tactic describes adversarial activity that involves actively or passively collecting information that can be used in later stages of the operation. This can include Open Source Intelligence (OSINT) on employees but also scanning for public-facing infrastructure.

RESOURCE DEVELOPMENT The adversary prepares the attack by setting up infrastructure, accounts and building capabilities.

INITIAL ACCESS The adversary uses techniques that gain them an initial foothold in the target network, ranging from spearphishing to exploiting vulnerabilities or using valid accounts.

PERSISTENCE After gaining access to a system within the target network, an adversary may opt to employ techniques that give them persistent access.

This allows them to continue the operation, even when their initial method to gain access stops working.

DEFENSE EVASION An adversary needs techniques to evade defensive mechanisms while compromising the target. These techniques include disabling endpoint protection software and obfuscation.

COMMAND AND CONTROL The adversary uses techniques to create a connection between their own infrastructure and compromised hosts in the target network. Communication is oftentimes done using covert methods to prevent detection.

2.3 CYBER THREAT INTELLIGENCE

After establishing necessary knowledge in the field of adversary modeling, the question arises as to how to fill these models with necessary information about threat actors. Cyber Threat Intelligence (CTI) describes information that can be used towards the prevention, detection and mitigation of cyber attacks. We are interested in types of CTI and how it is being disseminated in computer-readable formats, with the goal of being able to make informed decisions in expanding Autonomous Adversary Emulation (AAE) solutions, and the automation of ingestion of new CTI in AAE solutions.

2.3.1 *Types*

Chismon and Ruks [8] identify that CTI is a very broad domain and define four subtypes: technical, tactical, operational and strategic threat intelligence. Within this separation, tactical threat intelligence concerns information about the modus operandi of threat actors and would be most useful when emulating adversaries. The production of tactical threat intelligence requires a relatively high amount of manual research as opposed to other types of threat intelligence, causing it to be predominantly available from commercial providers.

2.3.2 *Dissemination*

Ramsdale et al. [22] analyse and compare several dissemination formats for CTI. In their assessment, they identify more than 20 criteria which they use to determine their capability. While their criteria largely focus on support for technical indicators, their analysis on the versatility and popularity of dissemination formats is useful for the use case of adversary emulation. They conclude that Structured Threat Information Expression (STIX) [18] has good apparent support in threat intelligence sharing platforms and communities but is not yet widely used and often poorly implemented.

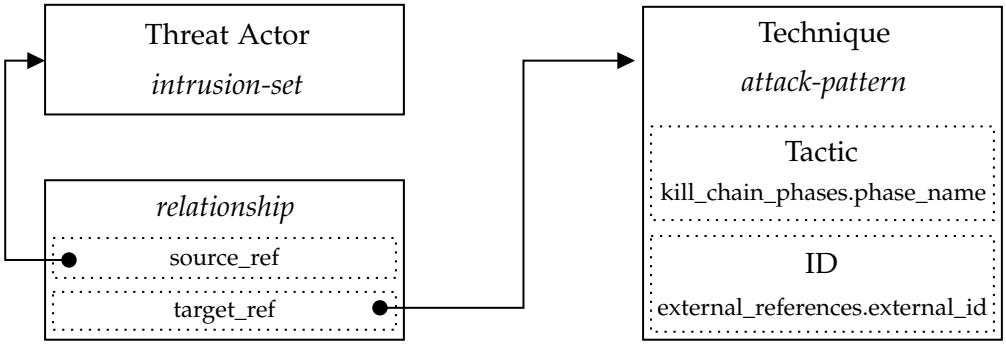

Figure 2.2: Using Structured Threat Information Expression (STIX) to express tactical Cyber Threat Intelligence (CTI) modeled in MITRE ATT&CK (STIX SDOs in cursive, fields of STIX SDOs in small font, MITRE ATT&CK concepts in normal font)

When enumerating offerings from CTI providers, some indeed support STIX, while others disseminate tactical CTI in the form of human-readable reports or custom computer-readable formats. MITRE maintains one of the only public tactical CTI resources and disseminates the information in both STIX and a custom JSON format[4].

2.3.2.1 *STIX*

In the past few years, Structured Threat Information Expression (STIX) [18] has been gaining popularity in the CTI domain. Tactical threat intelligence can be stored in STIX by using a selection of STIX Domain Objects (SDOs), which are the 'building blocks' of the format, connecting them through STIX relationships, and referencing to a tactical model for adversary behaviour like MITRE ATT&CK. Figure 2.2 shows how this can be achieved by comparing SDOs to tactical concepts that are found in the MITRE ATT&CK framework. STIX supports many more SDOs, making the format highly versatile and allowing it to express any type of CTI. However, its extensiveness could lead to ambiguity, causing users to implement the format in various ways while expressing identical information in STIX.

2.4 AUTONOMOUS ADVERSARY EMULATION

Automation has always had a big role in the offensive security industry, but more complex tests like adversary emulation require large amounts of manual intervention. A development that has been gaining traction is the concept of Autonomous Adversary Emulation (AAE). Between 2016 and 2018, researchers at MITRE wrote several papers on the use cases for autonomy in this domain and the fundamental challenges that are faced in building AAE solutions. They note that a high majority of automation efforts in the offensive security domain are focused on the technique level, while AAE solutions aim to extend this to the tactical level [3]. Given a highly sophisticated plan-

4 https://github.com/mitre/cti

ning engine and a wide range of adversary profiles and techniques, this could conceivably result in autonomous emulation of large parts of the adversary lifecycle. In this section, we further introduce the AAE domain. We explore existing implementations and set a basis for further reasoning about AAE by introducing a common architecture for an AAE framework.

2.4.1 *Implementations*

Autonomous Adversary Emulation is a rather new and broad concept, and the term is used to label a wide range of products and solutions, with greatly varying goals and capabilities. We discuss several open-source implementations that are open-source and either suitable for further research or in another way relevant to the remainder of this work. We define several criteria to assess their place in the AAE domain and shortly compare the solutions against these criteria.

Additionally, we discuss commercial implementations with similar functionality - Breach and Attack Simulation (BAS) solutions. While these fill a meaningful place in the market, making it important to track their development, they are not a suitable basis for research because they are generally closed-source and lack potential for extensibility.

2.4.1.1 *Non-commercial or open-source implementations*

MITRE CALDERA Applebaum et al. [17] propose an AAE framework and implement it in the form of MITRE CALDERA[5]. After its introduction in 2016, the framework has been regularly updated and is still an active research project at MITRE. The framework focuses on automating adversary emulation in the post-compromise domain, having limited ability in gaining initial access and requiring manual operation until the network perimeter has been breached. In the context of red teaming engagements, MITRE claims that CALDERA addresses the problems of "cost, time, and personnel, as the system can conduct an assessment without requiring any operator involvement" [5]. In practice, the framework is seeing most of its use as a basis for researching tactical-level autonomy for offensive security.

PRELUDE OPERATOR Released in 2021 [31], Prelude Operator[6] is an AAE product which packs many features that are implemented in CALDERA, while placing high importance in ease-of-use. In its current state, the tool is a great execution engine for adversarial TTPs with autonomous traits, but lacks in the more advanced planning capabilities that are seen in MITRE CALDERA. Analogous to CALDERA, the tool has no built-in capability to perform AAE in the (pre-)compromise domain. The tool is currently under very active development and will likely be an important instrument in bringing Autonomous Adversary Emulation to a wide audience and bringing

5 https://github.com/mitre/caldera
6 https://www.prelude.org/platform/overview

previous research into practice. While Prelude Operator is technically a commercial product, the tool is built to be very extensible and open to free non-commercial use, making it likely to become a relevant foundation for future research.

ATOMIC RED TEAM The Atomic Red Team library[7], an open-source initiative from Red Canary, contains a large amount of adversarial procedures (*atomics*) designed to test detection and security controls. Atomics are directly tied MITRE ATT&CK techniques and consist of execution steps for each platform they support, together some metadata about the procedure they emulate. Atomics are not designed to be composed - they mostly emulate an adversary in the 'noise' they generate, rather than actually gaining information or tactical progress.

A simple execution framework, called Invoke-AtomicRedTeam[8], has been built around the library. Using the tool, defenders can easily kick off atomics in the library to test their security controls. This execution framework can also form a basis for scripting attack paths consisting of multiple procedures. While the library and lacks any form of automation on a tactical level, it is unique in its completeness and simplicity - it packs procedures for a relatively large amount of the techniques in the post-compromise domain of the MITRE ATT&CK framework. It is therefore a rather popular tool amongst blue teams.

INFECTION MONKEY GuardiCore's Infection Monkey[9] is an open-source BAS solution whichs shows high similarity to most commercially available products. The solution features a small library of emulation steps mapped to the MITRE ATT&CK framework. Unlike other solutions presented in this section, Infection Monkey is mostly focused on propagating across hosts within a network, rather than extending access and gaining information at the host level. The solution does not have sophisticated planning capabilities and is largely independent from central C2. Instead, the agent works recursively and makes its own decisions. Therefore, most control over the tool is lost once it is deployed - it will start recursively propagating through a network until it has no actions left. Apart from the ability to set an attack depth, similar to a time-to-live in networking protocols, and providing an IP exclusion list, it is not possible to set up constraints for the emulation process.

Table 2.1 compares the four solutions presented in this section. We distinguish solutions that are designed to be *autonomous*, preventing human intervention as much as possible, and solutions that are *automated*, largely designed to be deployed in a known environment and requiring a relatively large amount of input. Other factors setting solutions apart are the level at

7 https://github.com/redcanaryco/atomic-red-team
8 https://github.com/redcanaryco/invoke-atomicredteam
9 https://www.guardicore.com/infectionmonkey/

which they automate, their extensibility, and their scope in the context of enterprise networks.

SOLUTION	TYPE	LEVEL	EXTENSIBILITY	SCOPE
MITRE CALDERA	Autonomous	Tactical	Full	Post-compromise
Prelude Operator	Autonomous	Tactical	TTPs, Plugins, Agents	Post-compromise
Atomic Red Team	Manual	Technique	TTPs	Post-compromise
Infection Monkey	Automated	Tactical	TTPs	Full

Table 2.1: Comparison of actively maintained and open-source adversary simulation/emulation solutions

2.4.1.2 *Commercial implementations*

While the concept of *autonomous* adversary emulation mostly lives in academia, several commerical solutions exist that have lots in common with AAE. Most of these products are marketed as Breach and Attack Simulation (BAS) solutions, while others are labeled as Continuous Automated Red Teaming (CART) or Security Validation. When enumerating these products, it is also common to find products that are creatively rebranded vulnerability assessment solutions. Especially the BAS domain is getting lots of attention in the enterprise security landscape - with Gartner identifying high benefit and ranking the domain at the peak of its hype cycle [20]. Essentially, Breach and Attack Simulation (BAS) can be seen as a practical implementation of Autonomous Adversary Emulation (AAE) concepts.

Solutions of note are those sold by AttackIQ[10], XMCyber[11], Picus[12], SafeBreach[13], FireEye[14] and Cymulate[15] [6, 20]. Because these products are closed source, generally expensive, and do not have clear public-facing documentation, it is difficult to assess their capability. Based on marketing resources, these solutions generally focus on simulating post-compromise TTPs and run on proprietary libraries of abilities which are attributed to the MITRE ATT&CK framework. Their focus generally lies in simulating adversaries in known, simulated environments, with the goal of continuously validating security controls. In most cases, they directly integrate with Security Information and Event Management (SIEM) and other detection platforms allowing for immediate intervention when security controls fail.

10 https://attackiq.com/blueprints/automated-security-validation/
11 https://www.xmcyber.com/breach-and-attack/
12 https://www.picussecurity.com/product/threat-emulation
13 https://safebreach.com
14 https://www.fireeye.com/mandiant/security-validation.html
15 https://cymulate.com/breach-and-attack-simulation

2.4.2 *Architecture*

As discussed in the previous section about AAE implementations in academia and the industry, their goals and use cases widely differ. We need a framework that is (1) autonomous, (2) targets autonomy on a tactical level and is (3) open-source and extensible when considering the goal of autonomously emulating adversaries in their (pre-)compromise TTPs. MITRE CALDERA and its evolution, Prelude Operator, largely satisfy these requirements. We will use their architecture as a basis throughout the remainder of this work, allowing for a common ground to reason about AAE implementations, and a basis for building a proof of concept.

Figure 2.3 shows a high-level representation of the most important concepts that form the CALDERA adversary emulation framework, and the means in which they interact. In this section, we will shortly introduce these concepts.

AGENTS Agents are the Remote Access Trojans (RATs) that run on remote hosts and communicate back to the AAE framework at set intervals. As is customary in the implementation of these RATs, a wide range of (covert) methods can be used to communicate with the C2 server, the AAE framework in this context.

FACTS All knowledge that is required to be stored across the execution of abilities is expressed in the form of facts. Facts have a standardised identifier, called a trait. An example of a trait would be `target.public.port` to express open network ports. A fact trait can be instantiated multiple times. Facts can be tagged with a score, allowing for expressing a level of certainty or priority over other facts of the same type. Additionally, facts can be related to each other with named, standardised links. This results in a graph-like knowledge model.

It is possible to manually source an operation with facts through the use of CALDERA `fact sources`. In the context of pre-compromise adversary emulation, this typically becomes a necessity because of the wide scope and large uncertainty that comes with this domain.

ADVERSARIES Considering the requirement of staying close to the TTPs of a real attacker, adversaries define a range of abilities that are available during an operation, based on tactical CTI.

PARSERS Raw output from the execution of abilities is ingested by parsers, which extract and store facts based on predefined logic. Parsers can be very specific to a certain ability (when extracting information from a specific tool), or broadly applicable (finding email addresses in a long string).

ABILITIES Abilities encode adversarial TTPs into the AAE framework. They consist of steps that are required for their execution, payloads that are needed over the course of their execution, and definitions of facts that are either required for the ability to run or potentially produced by the ability.

REQUIREMENTS Requirements impose constraints on the facts that can be used to satisfy an ability. They can either verify the value of a single fact, or impose constraints on the relation between two facts. The latter is a more common use case for requirements, given that fact values are generally constrained with rules.

RULES Through the notion of rules, boundaries can be placed on the values that are stored in facts. In contrast to requirements, rules work on the operation level and Several use cases exist for rules, one of them being the requirement to keep the framework from exploring and attacking hosts that are out of scope.

OBJECTIVES Objectives define rules, in the form of logic clauses on fact values, that need be satisfied to complete an operation. These would be the 'crown jewels' of a traditional red team. A smart planner, combined with a set of abilities that pose requirements and potentially produce certain facts, can solve towards reaching the objectives that are defined for an operation.

PLANNERS Based on all information that is available in the context of an operation, a planner makes decisions on which ability is executed at which agent. Planners can be very simple, executing all abilities that are tied to an adversary in sequence, but should ideally be 'smart', making their decisions based on the state of the operation and adversary-specific logic.

OPERATION Within an operation, the framework leverages the previously introduced notions (a group of agents, at least one adversary and a planner) to execute the emulation and gather results.

LINKS Links are generated by planners and form the combination of an agent, an ability, and a set of facts that satifies the requirements for that ability. For every iteration, the framework internally uses the generated set of links to give orders to its agents.

2.4.3 *Automated planning*

An important challenge in AAE is building the planners that intelligently and automatically compose sequences of abilities to achieve pre-determined goals, while staying close to the procedures of a real adversary. While most elements required in a model AAE solution require mostly engineering effort, automated planning can be considered an active research domain of its own.

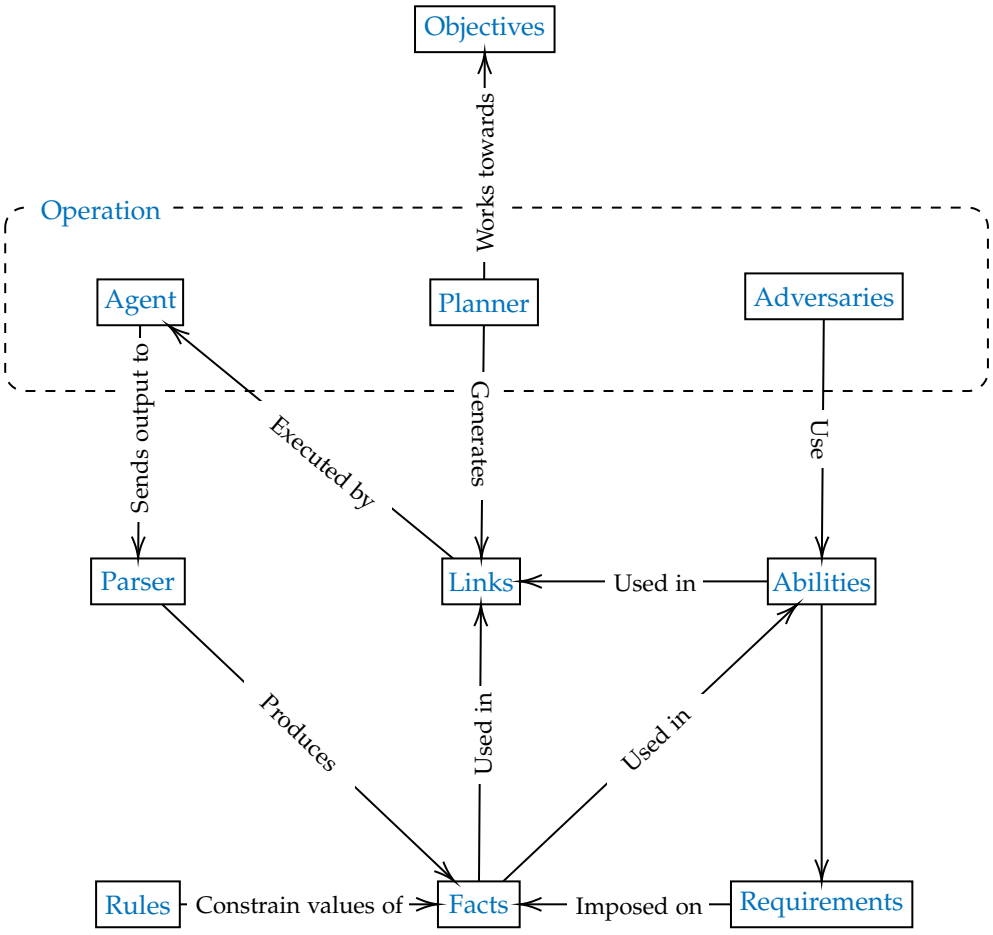

Figure 2.3: Global architecture of the MITRE CALDERA adversary emulation framework

2.4.3.1 *Automated planning in literature*

Most academic work in automated planning is very specific to certain technology domains or is constrained to vulnerability exploitation as a means of lateral movement. Miller et al. [27] point out that this is not a realistic scope, given that adversaries often use benign functionality in a malicious way to reach their goals. Over the years, researchers at MITRE have published several articles on automated planning and execution in the context of enterprise network environments. In [5] and [4], they construct and analyse several planning strategies for adversary emulation. In [27], they note that most of these strategies, while performing well in small-scale tests, do not scale to the large data model that CALDERA is using. They point out that attackers face unbounded uncertainty about the environment they are operating in and the outcomes of their actions, making it hard to fully plan an attack beforehand, and resort to using simpler conditional planners.

Bland et al. [7] use Petri nets to model several common attack patterns and implement a reinforcement learning algorithm to navigate these nets. In their results, they demonstrate potential for the use of formal methods to model cyber attacks and the use of reinforcement learning to improve the planning of these attacks. However, their method does not handle uncertainty - all potential states of an attack and the action space of an attacker need to be modeled beforehand.

The Microsoft 365 Defender Research Team [29] builds a platform called CyberBattleSim[16] to allow training and testing automated agents based on reinforcement learning in simulated network environments. Attacking agents have limited knowledge about the network and have exploration and exploitation capabilities, allowing to expand knowledge and foothold in the network. Defending agents have a predefined probability of detecting an ongoing attack and have the capability of resetting compromised nodes. By assigning rewards to the successful exploitation of nodes in the simulated network based on their intrinsic value, reinforcement learning algorithms can be guided towards an optimal strategy.

2.4.3.2 *Automated planning in AAE frameworks*

MITRE CALDERA provides some basic constructs that can be used to build planners. Additionally, the framework is packed with several basic planning algorithms, which are heavily influenced by previous research done at MITRE. The most basic planning algorithm in CALDERA sequentially executes the actions as defined in an adversary profile and stops once it does not meet the preconditions of the next action or when it reaches its objectives. Additionally, the framework is supplied with a batch planner that executes all available actions on all available agents where all preconditions for executing the action are met. As a basis for building more intelligent planners, the framework supports the notion of *buckets*, which are macro-level categories

16 https://github.com/microsoft/CyberBattleSim

of abilities that can be used to model state machines for use in a custom planning algorithm. Using this construct, planners can behave in a more intelligent way and follow a high-level tactical path towards their objectives. CALDERA supplies a predefined buckets planner which is highly inspired by the tactics in the MITRE ATT&CK framework. While popular in the research domain, planning strategies based on reinforcement learning have not yet been applied to MITRE CALDERA or other AAE frameworks.

3

MODUS OPERANDI

Recall the first milestone in this research:

MS 1 Understanding the modus operandi of threat actors during the (pre-)compromise stage of the adversary lifecycle

Threat intelligence is the cornerstone of adversary emulation - it is impossible to accurately emulate an adversary without understanding their modus operandi first. Evidently, this also applies to the implementation of Autonomous Adversary Emulation (AAE) solutions. The aim of this chapter will be twofold: gain adequate understanding of adversary modus operandi in the (pre-)compromise domain, and use this knowledge to intelligently narrow the scope of the remainder of this work. In this context, 'adequate' means having broad understanding of technique usage frequency in the (pre-)compromise domain, and good understanding of modus operandi during the initial access phase.

To certain extent, AAE frameworks need to be agnostic towards the adversaries they emulate. In this context, a common taxonomy like MITRE ATT&CK essentially forms an abstraction layer between the adversary and the implementation of their TTPs. For this reason, our initial analysis on adversary modus operandi in the (pre-)compromise domain will be focused on technique usage frequency. In section 3.1, we describe the steps taken and the output of this analysis. Given the output from the former exploration, we further concentrate our analysis on the initial access phase of the adversary lifecycle, by introducing additional CTI sources and manually ingesting relevant information from these sources. The initial access phase will play an important role in remaining steps of this research, and it is important to raise confidence in data on technique usage frequency for this phase.

However, while very useful in making choices in terms of scope for the remainder of this work, data on technique usage still lacks both broader context and adversary-specific detail. Adversaries do not carry out their operations using only one or a few techniques, or randomly combine techniques throughout their campaign. Tactical context plays an important role that needs to be understood to see how techniques interplay. Additionally, given that techniques are merely an abstraction of adversary procedures, more detail could be necessary to actually implement and emulate adversaries. To proceed with remaining steps of this research, we need to have a few specific and representative scenarios that can be implemented into a plugin for an AAE framework. We will collect these scenarios from public CTI sources and shortly introduce them in section 3.2.

3.1 TECHNIQUE USAGE

For an initial and broad exploration on technique usage by threat actors in the (pre-)compromise domain, we choose to take a quantitative approach in analysing technique usage frequency, as this allows for ingesting larger amounts of data while preserving objectivity. This exploration involves two parts: a broad analysis on technique usage frequency in the (pre-)compromise domain in section 3.1.1, and a more focused analysis on technique usage frequency during the initial access phase of the adversary lifecycle in section 3.1.2.

3.1.1 *Analysing STIX-formatted CTI*

For a broad analysis on technique usage frequency in the (pre-)compromise domain, we opt to source CTI in a structured format. Structured Threat Information Expression (STIX) has become a popular format for expressing various types of CTI and is also the format of choice for this analysis. Sourcing data in a structured format allows for consuming large amounts of CTI, while making the analysis agnostic to the provider of CTI and increasing reproducibility.

Commercial parties like Kaspersky[1] and Mandiant[2] disseminate high-quality tactical CTI in STIX, but their services are costly and inaccessible to the public. To the best of our knowledge, the only exception is MITRE's knowledge base on adversary behaviour[3], which is freely accessible and forms one of the pillars for the design of the ATT&CK framework. MITRE disseminates this data formatted in STIX in a public GitHub repository[4].

In order to collect these statistics, we wrote a Python script that parses STIX repositories with ATT&CK mappings into technique-level statistics on adversary behaviour. This script has been published on GitHub Gists[5]. The script collects attributions of technique usage to threat actors and computes statistics based on this data. Because the amount of attributions widely varies across tactics, we normalise the results by counting the total amount of attributions in each tactic and showing the relative share of attributions each technique has in its respective tactic.

Figure 3.1 shows the resulting statistics for the 5 most commonly attributed techniques for each tactic in the (pre-)compromise domain.

1 https://www.kaspersky.com/enterprise-security/apt-intelligence-reporting
2 https://www.mandiant.com/advantage/threat-intelligence
3 https://attack.mitre.org/groups/
4 https://github.com/mitre/cti
5 https://gist.github.com/DiedB/d0fc88a099ed1743b779d15819e1db31

Reconnaissance	Resource Development	Initial Access	Persistence	Defense Evasion	Command And Control
Gather Victim Identity Info. T1589 33%	Acquire Infrastructure T1583 32%	Phishing T1566 38%	Boot or Logon Autostart Exec. T1547 18%	Obfuscated Files or Info. T1027 16%	Ingress Tool Transfer T1105 23%
Phishing for Info. T1598 26%	Develop Capabilities T1587 19%	Valid Accounts T1078 23%	Valid Accounts T1078 16%	Masquerading T1036 11%	App. Layer Protocol T1071 19%
Active Scanning T1595 11%	Obtain Capabilities T1588 15%	Drive-by Compromise T1189 12%	Scheduled Task/Job T1053 15%	Valid Accounts T1078 9%	Web Service T1102 9%
Gather Victim Network Info. T1590 7%	Establish Accounts T1585 15%	External Remote Services T1133 10%	Hijack Exec. Flow T1574 9%	Indicator Removal on Host T1070 9%	Proxy T1090 9%
Gather Victim Host Info. T1592 7%	Compromise Infrastructure T1584 9%	Exploit Public-Facing App. T1190 8%	Create or Modify System Process T1543 7%	Signed Binary Proxy Exec. T1218 8%	Encrypted Channel T1573 7%

Figure 3.1: MITRE ATT&CK technique usage frequency per tactic in the (pre-)compromise domain relative to total technique usage in that tactic, sourced from the MITRE ATT&CK Groups STIX repository

3.1.1.1 *Limitations*

While this broad exploration on technique usage frequency based on MITRE's CTI repository provides useful guidance, it is important to note that several factors could potentially negatively influence its accuracy:

- Creation or modification dates in STIX `intrusion sets` do not match the date on which the corresponding CTI report was made, making it difficult to filter for technique usage over a certain period of time in an automated fashion. The presented data is therefore an aggregation of CTI that has been published over the past decade, losing information about specific trends that are time-sensitive.
- MITRE CTI data is not a direct representation of research on threat actors, but an aggregation of publicly available CTI reports over the past decade. While the sample is quite large, its completeness and lack of bias cannot be guaranteed.
- ATT&CK defines techniques that appear in multiple phases (e.g. `Valid Accounts`). Tactical context is lost when threat group activity is attributed to these techniques. In this representation, this could mean that these techniques are overrepresented.
- Compared to other tactics, the MITRE CTI repository does not (yet) contain as many attributions in the first two phases of the ATT&CK lifecycle, negatively influencing statistic significance of the resulting numbers.

- While the sample size of individual attributions is large enough for this analysis, it is not large enough to identify correlations between the usage of techniques in a statistically significant way.

In future work, these limitations could partially be addressed by introducing additional sources for STIX-based CTI in the analysis. Given that this broad analysis is mostly used as background information for later parts of this research, it meets our needs despite its limitations. Major choices only depend on technique usage frequency in the initial access phase, for which we will further augment the analysis in section 3.1.2.

3.1.2 *Analysing threat reports*

As mentioned in section 3.1.1, several major CTI providers disseminate their data in STIX, but this information is not publicly available. However, these providers regularly publish reports on trends in adversarial behaviour, including statistics on technique usage frequency. In this section, we collect and use these reports to be able to paint a more reliable picture of technique usage in the initial access phase.

3.1.2.1 *Data sources and requirements*

Unlike in the STIX analysis, where the structure of source data is predictable and the only concern is quality and completeness of presented data, human-readable threat reports widely vary in their aim and structure. While many threat report sources were identified, they are often of limited use or not usable at all in a technique-centered analysis. For the analysis that follows, we require reports that comply to the following criteria:

- The report needs to include information about our tactical scope of interest - the initial access phase.
- The report should be broadly scoped, containing general findings about a wide range of threat actors instead of focusing on specific threat actors.
- Findings in the report should be quantified as much as possible.
- Freshness of presented data is important - we are only interested in reports that cover the past two years.

From our enumeration, we find four reports that sufficiently fulfill these requirements: those published by Mandiant [24], Sophos [26], CrowdStrike [9] and IBM X-Force [23].

3.1.2.2 *Report structure and coverage*

Mandiant [24] reports quantitatively on technique usage based on the amount of observations in their investigations. Similarly, CrowdStrike [9] reports on their observance frequency of all ATT&CK techniques. Sophos [26] reports

on the top 5 techniques observed for each tactic in the MITRE ATT&CK framework, but does not supply any numbers apart from the ranking itself. The report from IBM X-Force [23] has a broad tactical and strategical scope, but does not directly map observations to MITRE ATT&CK. However, most of their findings are quantified and can be manually mapped to MITRE ATT&CK techniques.

Coverage of preparatory (pre-compromise) modus operandi in the reports is little to none. This is partially caused by ATT&CK only recently being updated to cover these tactics while the reports still use older ATT&CK versions. Additionally, adversary behaviour in these phases is harder to detect and could stay out of sight of CTI providers. Only the report from Mandiant [24] contains statistics on pre-compromise technique usage frequency, but coverage is lacking to an extent that the information has little to no relevance in this analysis.

The reports sufficiently cover post-compromise adversary behaviour. However, we do not thoroughly analyse this domain. Manual analysis and comparison is more difficult because the domain is rather convoluted, and the domain will not play a large role in later stages of this research.

3.1.2.3 *Results and conclusions*

Table 3.1 shows initial access techniques and their observation frequency according to the analysed threat reports, for those techniques that are mentioned in at least three sources. Where available, percentages are mentioned which concern the total usage frequency of the technique in investigated attacks. Because of the design of MITRE ATT&CK, where the same technique can occur in multiple tactics, we cannot accurately determine what share of the observation frequency can be attributed to usage of `Valid Accounts` during the initial access phase for the CrowdStrike and Mandiant reports.

It can be observed that the reports do not entirely agree on technique usage frequency for the initial access phase. This is largely expected, given that there will be slight differences in the way CTI providers will source their information, carry out investigations, and report their findings. However, a conclusion that can be made is that the four techniques mentioned in 3.1 are more commonly observed than the 5 remaining initial access techniques - by a significant margin.

3.1.3 *Conclusions*

From analysis of MITRE STIX data and publicly available threat reports, we draw the following conclusions concerning technique usage in the (pre-)compromise domain:

- While the MITRE ATT&CK framework covers a wide range of techniques, threat actors generally seem to employ few of them. Mandiant [24] sees a 63% coverage of the MITRE ATT&CK framework across all

	MITRE	MANDIANT [24]	SOPHOS [26]	CS [9]	IBM [23]
Valid Accounts (T1078)[a]	2	?	4	?	3 (18%)
External Remote Services (T1133)	4	3	1	-	-
Exploit Public-Facing Application (T1190)	5	1 (29%)	2	1	1 (35%)
Phishing (T1566)	1	2 (23%)	3	2	2 (33%)

a This technique is used in multiple tactics in the ATT&CK framework, making it difficult to assess its actual frequency in the context of initial access attacks

Table 3.1: Initial access techniques and their occurrence according to threat reports from CTI providers and the previous analysis on MITRE's CTI repository, for all techniques that are mentioned in at least three sources

their investigations, with only 23% of techniques being used in more than 5% of intrusions.

- Activity of threat actors in their preparatory phase (reconnaissance and resource development) is not well-documented, partially because detection of techniques in this domain is relatively difficult and partially because CTI providers are not yet using the latest version of ATT&CK. Where necessary in later stages of this research, pre-compromise activity will be inferred from initial access techniques and other tactical information.
- Four techniques prevail in the initial access phase: phishing, exploitation of public-facing applications, usage of valid accounts and external remote services (the latter two often being used together). From CTI data, it is not possible to see a statistically significant difference in their occurrence.
- The post-compromise domain is more complex in nature - it contains a significantly larger amount of techniques, and adversaries do not necessarily need to traverse every tactic to reach their objectives. Given the choices that have been made in scoping this project, we will refrain from analysing modus operandi in this domain in more detail.

3.2 SCENARIOS

Recall the second milestone of this research:

MS 2 Constructing a representative set of emulation scenarios covering the (pre-)compromise phase

In the continuation of this work, we will require a set of scenarios that will form a basis for implementing and evaluating AAE in the (pre-)compromise

domain. These scenarios will contain the tactical context and procedure-level detail that is still missing from the former analysis on technique usage frequency in the (pre-)compromise domain. The process of collecting these scenarios will be manual, based on a qualitative analysis on publicly available Cyber Threat Intelligence (CTI). To be sufficiently representative and useful in the remainder of this work, the collected set of scenarios will have to fulfill the following requirements:

- Relevance in the current threat landscape - it should be plausible that organisations are (still) vulnerable to these scenarios in the real world
- Coverage of the (pre-)compromise domain in terms of technique usage should be maximised (through the combination of all selected scenarios)
- Full technique-level coverage of the four initial access techniques as identified in section 3.1.3
- For evaluation purposes, simulation of a vulnerable system or environment should be attainable

Although harder to encode in requirements, and probably implicit from previous findings, it is also important that the final selection of scenarios challenges the existing capabilities of the state-of-the-art in AAE frameworks.

In the remainder of this section, we will elaborate on the chosen set of emulation scenarios and test them against these requirements. Given the requirement of fully covering the four previously identified initial access techniques, we will centralise the chosen scenarios around these techniques.

3.2.1 *Vulnerability Exploitation*

The US Cybersecurity and Infrastructure Security Agency publishes a yearly report on routinely exploited vulnerabilities [10]. While the latest report mainly focuses on 2020, it also concerns vulnerabilities that have been actively exploited in 2021. These include vulnerabilities in several Virtual Private Network (VPN) solutions, like Fortinet and Pulse Secure, and a chain of vulnerabilities named ProxyLogon and ProxyShell that were found in on-premise versions of Microsoft Exchange between May and August 2021.

The most recent vulnerability, named ProxyShell, consists of three separate CVE!s that - when combined - can be exploited to gain Remote Code Execution (RCE) on the Exchange Server [28]. The first two vulnerabilities allow an arbitrary attacker to execute commands on the Exchange Server. Full RCE can be achieved through the third vulnerability - which misuses functionality to export mailboxes to write a web shell to the Exchange Server.

The ProxyShell chain of vulnerabilities forms an interesting scenario for challenging AAE capability in the (pre-)compromise domain, due to its high

versatility. Combining the initial two vulnerabilities allows for enumerating e-mail accounts on the Exchange Server, which can then be used by a sophisticated adversary in password spraying attacks or phishing campaigns. Additionally, internal phishing scenarios can be carried out using access to the Exchange PowerShell interface. Finally, exploitation of the third vulnerability can be used as a testbed for leveraging RCE in AAE solutions, which is largely uncharted territory in the domain.

3.2.2 *External Remote Services and Valid Accounts*

For this scenario, we choose to combine the External Remote Services and Valid Accounts techniques for the reason that in the context of initial access, successful usage of the Valid Accounts technique (`T1078`) always depends on the availability of External Remote Services (`T1133`). This has the implication that this scenario has to cover both techniques in a meaningful and representative manner.

Consulting MITRE's knowledge base on adversary behaviour[6], commonly observed external remote services in attacks are VPN solutions and remote desktop solutions, like Citrix and Microsoft RDP. Less prevalent are remote management services, like Windows Remote Management (WinRM) and SSH. For this scenario, we reduce the scope to remote desktop solutions, given that these services will also play a big role in the lateral movement phase. Additionally, being able to use remote desktop solutions will most likely still be a requirement after gaining access to an external-facing VPN service. Finally, lateral movement through remote desktop solutions forms an interesting challenge to existing AAE capability - underlying protocols are generally designed for human-computer interaction and state-of-the-art AAE frameworks have not yet demonstrated any capability in using these services to move laterally.

In the context of initial access attacks, threat actors could obtain valid accounts by brute forcing external services, or a combination of external reconnaissance and password spraying. Phishing is also a common method of obtaining credentials, but will be covered in a separate scenario.

It is not surprising that the combination of the previously discussed techniques is also being observed in the wild. CrowdStrike highlights the modus operandi of the REvil Ransomware-as-a-Service (RaaS) group in their yearly Global Threat Report [9, p. 20], who use the combination of password spraying and external-facing Remote Desktop Protocol (RDP) services to gain initial access in their campaigns. In the next phase of this research, we will implement a scenario around their modus operandi as documented by CrowdStrike.

The password spraying technique forms a useful test case for handling (partial) credentials in the knowledge model of AAE solutions. Emulating an

6 https://attack.mitre.org/techniques/T1133/

adversary interacting with an RDP service poses a challenge because the service is not designed for automated interaction. However, given its extremely widespread use - both internally and outside-facing - this renders a very useful baseline scenario for (pre-)compromise AAE.

3.2.3 *Phishing*

Adversary behaviour within the phishing domain can, generally speaking, be categorized in two ways. One is the means of delivery - email is most commonly used [16], but phishing via other services is also being observed, given that detection and prevention efforts for email phishing have come a long way over the past few years. The other is the immediate goal of the phishing campaign - main objectives are either obtaining credentials by luring users to follow a link to an adversary-controlled web page, or persuading the user into execution of malicious software that is either delivered through email attachments or links to external file hosting services.

For this scenario, we choose to use email as a means of delivery - threat intelligence sources show that this delivery method is still used in the vast majority of phishing campaigns [16]. Because storage and usage of credentials are already demonstrated in the valid accounts scenario (section 3.2.2), we focus on the remaining common path - generating and attaching malicious payloads.

A recent Emotet phishing campaign incorporates these two characteristics [2, p. 16]. This group utilizes weaponized Microsoft Office documents to deliver malware that yields them C2 on their targets. Cofense reports this as one of the larger phishing campaigns in 2020 [2]. Dynamic generation of malicious payloads and delivering malware through multi-stage droppers will prove to be a useful test for AAE capability in resource development and management.

4

EMULATING THE ADVERSARY

Recall the third milestone in this research:

MS 3 Integrating the previously defined set of emulation scenarios into a plugin for an adversary emulation framework

Previously, we have identified CALDERA and Prelude Operator to be the state-of-the-art frameworks in the Autonomous Adversary Emulation (AAE) domain. Given the high degree of extensibility and adoption level of the former framework, we will use it as a basis for building our integration. Effectively, this integration will consist of two main components - a platform that facilitates autonomous emulation of adversaries in the (pre-)compromise domain, and the previously introduced scenarios that are built on top of that platform.

This chapter is split into three parts. Section 4.1 dicusses the platform for our custom CALDERA integration - focusing on the engineering work and architectural choices. In section 4.2, we take the previously selected scenarios that have - until now - been discussed at a technique level, and discuss the specific implementation of these techniques that will eventually be used in the set of abilities that is bundled with our CALDERA integration. Finally, in section 4.3, we discuss several domain-specific limitations we identified in the CALDERA framework while implementing this integration. While the plugin platform works around these shortcomings, we believe they should be mitigated on the framework level and we propose several high-level solutions to these shortcomings.

4.1 PLATFORM

This section discusses the basis of our custom-built CALDERA integration and describes how it leverages and builds on the functionality provided by the CALDERA framework. Several fundamental design choices will be discussed in detail - the custom planner, its knowledge model, modes of operation, agent groups, and how it handles remote code execution. The source code for the plugin is available on GitHub[1].

4.1.1 *Assumptions*

We make several assumptions in the implementation of this proof-of-concept integration. Most of these assumptions are being made to scope out technical

1 https://github.com/DiedB/caldera-precomp

challenges that are not immediately relevant to the goals of this research project.

- The CALDERA Autonomous Adversary Emulation (AAE) framework is fully capable in post-compromise operation - with the addition of our proof-of-concept integration, it should be able to traverse the entire adversary lifecycle.
- The CALDERA server is running on a host that is fully exposed to at least the initial target host - no Network Address Translation (NAT) traversal is needed to establish communication between the CALDERA server and the initial target server.
- All hosts that CALDERA uses as vantage point for initial access attacks have all necessary prerequisites in terms of software to support execution of the abilities that are sent to them by the CALDERA framework.
- All hosts that CALDERA uses as vantage point for initial access attacks are identical from a networking perspective - externally, they share the same IP address, internally, they are in the same subnet.

4.1.2 *Agent groups*

CALDERA supports the notion of agent groups, which are essentially labels applied to an agent that can be used to administer the C2 agents that are provisioned by the framework. In the implementation of our proof-of-concept, we will uphold three distinctive agent groups, each with its own tactical purpose.

INITIAL Initial agents are used to run all abilities in the first three stages of the adversary lifecycle. In the context of initial access attacks, these agents are outside the organisation network. Given that the hosts these agents are running on are fully under our control, they can provide all software necessary to support CALDERA abilities. By spawning `initial` agents on hosts with various architectures and distributions, a broader basis for offensive tooling can be offered to the AAE framework.

RCE Agents in this group are used as a wrapper around abilities that yield RCE ability on a target host. The exact purpose of these agents will be further explained in section 4.1.4.

TARGET Upon gaining C2 on a target machine, the CALDERA plugin will spawn an agent in the `target` group. In our implementation, the planner will not generate any links for agents in this group, but the group can be used in transitioning towards post-compromise operation using other CALDERA plugins.

In the remainder of this chapter we will regularly refer back to these agent groups, especially when discussing automated planning.

4.1.3 *Adversaries and modes of operation*

The platform implements two types of `adversaries`. These types are in close relation with the previously introduced modes of operation we identify for automated security testing solutions, and can be chosen depending on the use case at hand.

UNRESTRICTED In unrestricted mode (the `precomp-unrestricted` adversary), CALDERA works autonomously within the (pre-)compromise domain, finding any path to initial compromise that can be found based on the capabilities that are implemented in our integration. Constraints imposed through the use of fact sources will still apply. This mode of operation follows the philosophy behind CALDERA and other AAE solutions, and it can be seen as a red team performing adversary simulation.

SCENARIO EMULATION In emulation mode (the `precomp-scenario-{name}` adversaries), CALDERA stays within the boundaries of specific scenarios. In all other aspects, CALDERA still operates autonomously. This mode of operation lies more closely to that of Breach and Attack Simulation (BAS) solutions in the sense that it is mostly following a one-dimensional scenario. This mode of operation lies closer to adversary emulation, given that the underlying scenario implements a single adversary and stays close to their TTPs.

4.1.4 *Handling remote code execution*

When designing abilities that are part of the tactical phases of inital access or lateral movement, one often encounters procedures that allow for code execution on the target host. Code execution could be reached through services that are designed for this purpose (like Secure Shell (SSH) or Windows Remote Management (WinRM)), or through the exploitation of certain vulnerabilities. In these situations, actually executing code on the remote host requires two separate actions: one that is run locally, to establish the pipe (a persistent connection or session) through which commands can be executed, and one that is run remotely.

Given CALDERA's current capability, without using workarounds that are not generally applicable or user-friendly, it is not possible to separate these two actions - both will have to be combined into one `ability`. Doing this goes directly against the philosophy of the framework - to strive for atomicity in designing abilities, while letting the framework make decisions in how to use them. Given the paradigms of autonomy and emulation, we deem it essential to be able to separate these two actions in our proof-of-concept.

To this end, we implemented a workaround in the form of a custom agent. This agent, which is a modified fork of CALDERA's Sandcat[2] agent, is open-source and can be found on GitHub[3]. By initialising this agent with several required arguments, it can modify commands that it receives from the CALDERA framework such that they are executed on a remote host. We will shortly introduce the lifecycle of an RCE agent, which consists of three phases - initialisation, operation and cleanup.

4.1.4.1 *Initialisation*

This agent is spawned on a host that is local to the operation through a standard CALDERA ability. This ability should be configured in a way that all preconditions for remote code execution are satisfied, e.g. having valid credentials or having confirmed that a vulnerability is working. In the ability, the following arguments should be passed to the execution of the RCE agent:

RCE COMMAND The agent needs a command that it can use to - in turn - execute commands remotely. This command should be filled with all necessary information which the framework has learned throughout the operation, such as a hostname and credentials. Additionally, it should contain an injection point for commands that should be executed on the remote system.

EXECUTOR The framework needs to know what executor to use for abilities that are sent to the RCE agent. An executor can be a certain command shell (e.g. `bash`, `PowerShell`) or direct process execution. In the architecture of the framework, this is something that is communicated to the framework by the agent - therefore, it is necessary to pass this information to the RCE agent on initialisation. The RCE agent could be equipped with the ability to detect the remote executor on its own, but we decided on a minimal implementation since the agent is merely a workaround as part of a proof-of-concept. In the majority of situations, the executor type is a priori knowledge - a specific vulnerability will generally exist on a certain operating system, and a specific remote administration solution always yields a certain command shell.

4.1.4.2 *Operation*

In operation, the RCE agent largely works like any other agent connected to the framework. Incoming commands are injected into the previously introduced RCE command, resulting in remote execution of incoming commands. Output from remote commands, if any, is sent back to the framework without any postprocessing.

2 https://github.com/mitre/sandcat
3 https://github.com/DiedB/caldera-precomp-gocat-rce

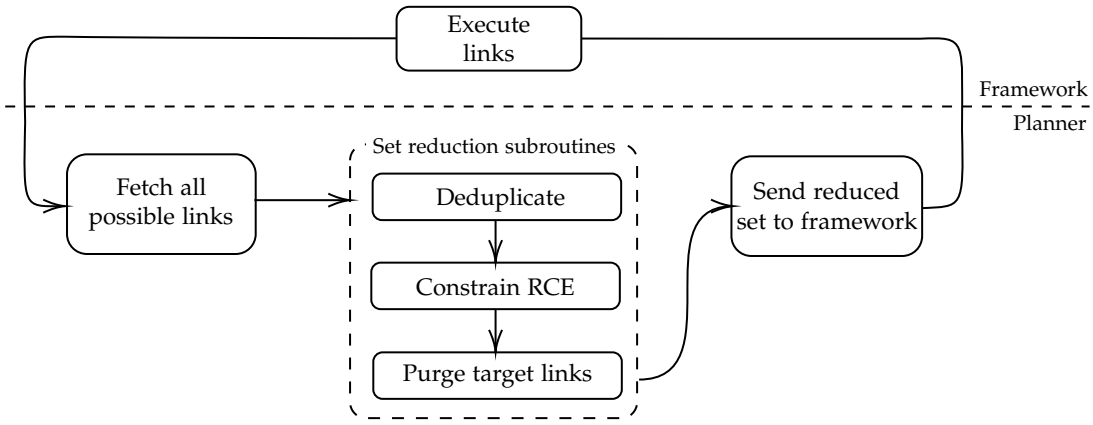

Figure 4.1: Schematic overview of the interaction between the CALDERA framework and the custom planner implemented in our CALDERA plugin

Tactically, an RCE agent is handled differently by the custom planner that is bundled with the proof-of-concept plugin. This distinction is further explained in section 4.1.5.

4.1.4.3 *Cleanup*

Because CALDERA does not provide an (internal) API to destroy agents, we pass a reasonable timeout to each RCE agent - ensuring that it cleans itself up after sufficient time has passed to reach all tactical objectives through the agent.

4.1.5 *Planning*

In order to deal with several limitations in CALDERA's built-in planners, which are specific to (pre-)compromise Autonomous Adversary Emulation (AAE), we developed a custom planner that is part of the plugin. While it is not the intention to tread into the automated planning domain with this research, some form of control on top of CALDERA's default planner is necessary to support emulation of the scenarios we defined and (pre-)compromise operation in general.

A global schematic overview of the planner and its interaction with the framework can be found in figure 4.1. We start by inquiring the CALDERA framework for all possible links - combinations of abilities, facts and agents - that it can generate based on the current state of the operation. The resulting set of links is passed through three subroutines with the intention of reducing it to only keep links that are relevant to the operation. These subroutines will be explained in section 4.1.5.1. Finally, the reduced set is sent back to the framework for execution. This process repeats until either an objective is satisfied or the reduced set of links is empty.

4.1.5.1 *Set reduction subroutines*

DEDUPLICATE For all agents in the `initial` group, we apply a form of deduplication - we do not want to run the same combination of an `ability` and `facts` on multiple `initial` agents, given that there exists no tactical difference between these agents. Only one unique combination remains in the set of generated links.

CONSTRAIN RCE When an `RCE` agent is available, we are only interested in running `abilities` that are part of the Command and Control (C2) tactic - upgrading the RCE capability we have to full Command and Control (C2). To this end, any links that are generated with an RCE agent and an ability that is not part of the C2 tactic, are removed from the set.

PURGE TARGET LINKS Given that the main objective of the plugin is reaching Command and Control (C2) on an initial host, which has been accomplished for agents in the `target` group, purge all links containing `target` agents from the set.

4.2 SCENARIO IMPLEMENTATION

In the previous section, we introduced a platform that facilitates autonomous emulation of adversaries in the (pre-)compromise domain. To be able to prove that this platform is effective, we will leverage it to implement the three scenarios that were introduced in section 3.2.

Essentially, each scenario needs to be decomposed into a set of AAE abilities. Because we want to support both autonomous simulation and scenario emulation, these abilities need be *atomic* and connected through a knowledge model. In this context, we will consider an ability to be *atomic* when it is reduced to the smallest possible action an adversary can take such that they either gain knowledge that is useful to their operation or an extension of their foothold. The knowledge model is introduced in section 4.2.1, and the abilities are introduced in section 4.2.2.

4.2.1 *Knowledge model*

Our *proof-of-concept* uses a typed knowledge model to give structure to operations. This knowledge model can be seen as an undirected, disconnected graph. While the graph is disconnected, we strive to connect graph nodes where possible and relevant, because this allows for better understanding of how facts were learned throughout an operation. In order to implement the scenarios we defined, we distinguish nine fact traits. The traits and the means in which they are related are shown in appendix A.1. We will shortly introduce the knowledge needed to start an operation in the remainder of this section.

In order to impose some initial guidance and constraints on the execution of an operation, it is possible to provide initial values for certain facts. By design, our integration expects the following fact traits to be present at the start of an operation.

IP RANGE The `target.range` trait can be filled with an IP range in Classless Inter-Domain Routing (CIDR) notation. This range will be honored in the reconnaissance process.

DOMAIN The `target.domain` trait can be filled with a Fully Qualified Domain Name (FQDN). This domain name can be used in the reconnaissance process to execute several reconnaissance steps to fill the knowledge model with additional information.

4.2.2 *Abilities*

A structured overview of all abilities that are part of our *proof-of-concept* can be found in appendix A.1. We will shortly discuss choices that were made in decomposing each scenario into abilities. Additionally, we build several abilities that serve multiple scenarios.

4.2.2.1 *Emulation accuracy*

While designing these abilities, we regularly make compromises in terms of accuracy in emulating the techniques of a threat actor. Some of these compromises are normal in manual adversary emulation as well and have minimal impact to emulation accuracy (e.g. not matching the IP address of a threat actor). Others could have a larger impact:

- Adversaries will sometimes use custom-built tooling and payloads, and CTI only observes and reports the result of execution of these tools. In the post-compromise domain, these artifacts can sometimes be collected because adversaries transfer their tools to the infrastructure they are attacking. Because this does - in general - not apply to (pre-)compromise operation, we have to make a best effort in emulating the procedures implemented by their custom tooling based on IoCs and log samples.
- In several instances, adversaries will perform manual actions during their operation. Automating these actions can have an impact on the accuracy of the emulation of their procedures. We are, for instance, limited to specific procedures when interacting with RDP-enabled hosts.
- We are balancing between the implementation of a mode for autonomous simulation and scenario-specific emulation. For abilities that are not immediately specific to a certain scenario, we choose a general implementation that matches an adversary at the technique-level but might not be the case at the procedure level.

- High-quality CTI, with the detail that is neccessary to emulate threat actors in a sufficient way, is not always available to us.

While, for the reasons above, compromises to emulation accuracy have to be made in some instances, we have no reasons to believe that this would have an impact on assessing whether the platform we designed is fit for emulation of specific adversaries in use cases where higher accuracy is important.

4.2.2.2 *Generic abilities*

NETWORK ENUMERATION Given that the plugin starts with a CIDR range of IP addresses, we need a method to identify active hosts and the services they expose before continuing the operation. We use a basic `Nmap` scan for this in ability `A1`.

SPAWNING C2 AGENTS Ability `A5` downloads and spawns a C2 agent on target hosts. In our implementation, this is the default agent that is used by the CALDERA framework. We do not mimick the procedures of a specific adversary because we consider adaptation to third-party C2 out of the scope of this `proof-of-concept`. We further address this limitation in section 4.3.3.

4.2.2.3 *Scenario-specific abilities*

We decompose each scenario defined in the previous chapter into several atomic abilities. For each of these scenarios, we will shortly discuss the choices that have been made in this process.

VULNERABILITY EXPLOITATION After scanning for the presence of the vulnerability using a basic and common check through `Nmap` (`A2`), we use a popular proof-of-concept implementation of a ProxyShell exploit[4] to which we make some slight alterations. We split the exploit in three atomic steps. The first step extracts email addresses from the Exchange server and stores them in the knowledge model of the operation (`A3`). The second completes the exploitation path and drops a webshell on the vulnerable server (`A4`). We modify the exploit to match the IoCs found in [1]. Finally, we extract the part that uses a planted webshell to execute commands on the remote server. Given that the exploitation process completes succesfully, an RCE agent is spawned that leverages this third step (`A10`).

EXTERNAL REMOTE SERVICES AND VALID ACCOUNTS After finding hosts that expose an RDP service and a potential username from the reconnaissance phase, we execute a password spray using `Hydra`[5], a popular tool for executing brute force attacks against several services. Given that a credential pair is found, an adversary would leverage it to start a session with the RDP service.

4 https://github.com/dmaasland/proxyshell-poc
5 https://github.com/vanhauser-thc/thc-hydra

Generally, this would be a manual task - the RDP protocol is not designed for automated interaction. Few tools exist to do this in an automated way - we use a modified version of `SharpRDP`[6] which is able to automate keyboard interaction and uses this to spawn processes. Because this tool did not initially support uppercase letters, we modified it to be able to send `Base64`-encoded PowerShell commands. We submitted a pull request to fix this limitation, and are using a patched version of `SharpRDP` in our scenario. This patched version is used by an RCE agent which we spawn when a credential pair is found (`A11`).

PHISHING We make two assumptions in the implementation of this scenario. We omit some parts of the resource development phase - setting up the prerequisites for a phishing campaign in terms of domain registration and reputable email infrastructure is a process that is relatively complex to automate and is not immediately relevant within the scope of this research. Additionally, we abstract the human element away and assume interaction on the receiving side that benefits the immediate objectives of the emulated threat actor. E-mail addresses are gathered using `Spiderfoot` based on the public domain that is present in the knowledge model (`A6`). To compile a malicious Microsoft Office document, we use `macro_pack`[7] and configure it to download a PowerShell dropper (`A5`) on macro execution (`A8`). Again, while this is a common implementation of the phishing technique, it is challenging to use payloads from real adversary campaigns. While these payloads are widely available, it is infeasible to modify these into a form where their execution benefits our operation. The compiled document is sent through a preconfigured Simple Mail Transfer Protocol (SMTP) server (`A9`), using an EML payload as template for the email. We use a sample from the TrickBot group, but the scenario could work with any phishing email sample.

4.3 EXTENDING CALDERA CAPABILITY

Throughout the implementation phase, we identified several shortcomings in the architecture of the CALDERA AAE framework. While we have applied workarounds to mitigate these shortcomings for the implementation of our plugin, we see potential for making several fundamental changes to the framework with the goal of making it more capable in emulating adversaries in the first stages of the attack lifecycle. In this section, we will propose these modifications - on an abstract level - for the main shortcomings we identified.

4.3.1 *Ability composition*

In section 4.1.4, we introduced the need for handling Remote Code Execution (RCE) in a flexible manner and implemented a workaround to satisfy

6 https://github.com/0xthirteen/SharpRDP
7 https://github.com/sevagas/macro_pack

our requirements in terms of autonomy and emulation accuracy. We do still make concessions in terms of accuracy in this workaround, however, with regard to persistence of connections through which commands are executed remotely. Additionally, application of the workaround in the process of developing new scenarios is not very straightforward.

When striving towards the most accurate emulation, there should be a possibility to keep the aforementioned connection alive. However, there will not be a persistent connection or session in all scenarios that yield the ability to execute code remotely - web shells generally receive a new connection for each command that is sent to them.

We will propose a general solution, which will work universally. This solution might not always yield the most accurate result from point of view of a defender, because the pipe is still being recreated for each command that is run remotely.

By extending the specification for abilities, one can indicate to CALDERA that an ability allows for substituting in another ability, the latter being executed on a remote host. The following fields should at minumum be available in composable abilities:

- A precondition which needs to be fulfilled before the ability can be used for substitution (verifying whether RCE is possible)
- A substitution marker for injecting commands (from other abilities) in the ability
- The type of executor that needs to be used for substituted abilities (`bash`, `PowerShell`, ...)

An extended solution, that allows for persisting pipes as well, is more complex from an engineering point of view. This will likely warrant the addition of an API for remote executors, implementing a subset of the functionality of the existing API for agents. Remote executors would be small programs that initialize a remote connection and listen for commands that can be sent over that connection.

While these modifications to CALDERA's internals are non-trivial, it is an absolute prerequisite to improve accuracy in emulation of adversaries through the initial access and lateral movement phases.

4.3.2 *Pre- and postconditions*

When designing `abilities`, one often encounters situations where certain constraints need to be imposed on the facts that are being used in the ability. By design, a simple constraint exists: the current state of the operation needs to be able to satisfy the set of facts used in an ability.

This does not always suffice - one might want to impose constraints on the relation between facts. Credentials are a prime example - combining

random usernames and passwords gathered within an operation, will probably not yield the desired result. In the CALDERA framework, this can be achieved by using `requirements`, which are separate Python modules that take a `fact` or a relation between `facts` as input, and based on some logic, return whether the requirement is satisfied.

In several situations, it is possible to write generic requirements that can be used by several abilities. We argue, however, that in practice many situations occur where requirements and facts need to be built specific to an ability. At that point, the model of requirements doesn't scale, and the process of designing abilities becomes rather tedious. In the process of developing our integration, we identify two distinguishable use cases where the requirement model falls short:

FACT VALUES Situations occur where the specific value of a fact is a condition for an ability to run. For example, this occurs when the knowledge graph contains hosts and the ports they expose, and abilities require a specific port being open. For each of these abilities, a custom requirement is needed that specifically checks for the value required by the ability,

ABILITY CHAINS While the AAE framework should generally be responsible for picking abilities to execute and the order in which to execute them, there are situations where implementing predefined chains of abilities is a necessity. For example, one can think of a scenario which chains the exploitation of multiple vulnerabilities - there is not much tactical meaning in executing individual abilities that exploit a single vulnerability each, but splitting them helps in debugging attack chains and evaluating detection performance. Using CALDERA's current capabilities, each of these abilities would need custom facts and requirements to ensure the chain is run in consecutive order - making the process of implementing abilities tedious and polluting CALDERA's knowledge model.

The solution we propose is twofold. Firstly, a custom query language that allows for making queries against CALDERA's knowledge model, including the set of links that has been executed in that operation. Secondly, an extension to the ability specification to include optional pre- and post-conditions written in this query language, where the former determines whether the ability is allowed to run and the latter determines success of the ability (reverting the alterations that were made to the knowledge graph in the execution of that ability). The implementation of these improvements would greatly reduce the complexity of abilities and the plugins around them, which facilitates an easier implementation process and easier sharing.

4.3.3 *Adapting to third-party C2*

Adversaries use a wide range of C2 frameworks in their operations. Popular choices are Cobalt Strike[8] and Covenant[9], but a multitude of frameworks exist. To accurately emulate an adversary, it is necessary to replicate the usage of their tooling in emulation scenarios. This is rather hard to do autonomously in the Command and Control (C2) category, as it will require the AAE framework to communicate with third-party C2 beacons.

CALDERA, in its current state, implements its own C2 framework. The framework is rather flexible in its design by supporting custom contact methods for agents, that can also be supplied through plugins. This functionality already goes a long way in terms of emulation accuracy, as it allows for replicating many properties of the communication used between the C2 beacon and the framework. In some cases, this functionality could even be leveraged to completely replicate the endpoints a third-party C2 agent uses to check in to the framework, effectively bypassing the third-party C2 framework.

However, for advanced C2 solutions, this approach will not work or not be able to scale. In these situations, direct integration with the third-party C2 framework could be a solution. This would require an extension to CALDERA that allows for starting persistent 'bridges' as part of an operation, which operate the third-party framework in an automated manner while sending and receiving all necessary information from the CALDERA framework.

8 https://www.cobaltstrike.com
9 https://github.com/cobbr/Covenant

5

EVALUATION

Recall the final milestone in this research:

MS 4 Validating the plugin against a test environment and testing its application to real-world use cases

This chapter consists of three parts. Section 5.1 introduces the environment in which we can perform this evaluation. In section 5.2, we evaluate the behaviour of the previously introduced CALDERA integration against this environment. Finally, in section 5.3, we identify several real-world use cases for our work, and evaluate its contribution to these use cases.

5.1 ENVIRONMENT

We construct a testing range to validate our previously introduced CALDERA integration and to compare it to similar solutions. A global overview of this environment is shown in figure 5.1. We display all hosts that are present in this range categorized by their purpose. Where relevant, the figure also displays networking architecture. In the following sections, we will introduce all relevant context and parameters used to construct the testing range.

5.1.1 *Networking*

We require granular control over networking in the testing range, for several reasons. Firstly, the range contains multiple vulnerable hosts - unwanted external interaction should be prevented by all means. Additionally, it is desirable to place extra safeguards around the AAE operations we are executing. While our plugin is designed to honor several configurable constraints, the application of safeguards on the network level will ensure operations do not break out of their boundaries.

To meet these requirements, we attach all hosts that are under our control to a virtual network[1]. Within this virtual network, we configure two separate IP ranges. One is being assigned to initial agents - the hosts CALDERA uses as a vantage point for emulated attacks. The other range is reserved for vulnerable hosts.

Where necessary, we use Azure VPN Gateways[2] to link networks together. These gateways form protected tunnels that provide certain external

1 https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview

2 https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpngateways

resources with access to (a segment of) the virtual network. These gateways are indicated with circle-ended connectors in figure 5.1.

5.1.2 *Offensive infrastructure*

5.1.2.1 *AAE*

The CALDERA framework is running locally on a Linux-based host. Modifications that were made to CALDERA's default configuration and the configuration used for our plugin are described in appendix B. The CALDERA deployment has two `initial` agents at its disposal - running on hosts with the latest versions of Windows 10 and Debian respectively. All software prerequisites for (pre-)compromise operation are available on these hosts.

5.1.2.2 *BAS*

For specific aspects in our validation strategy, we make a comparison against other solutions that are designed to validate security controls - Breach and Attack Simulation (BAS) solutions. To this end, we deploy the BAS platform offered by AttackIQ[3], as it is the only platform that is accessible to us.

Because BAS solutions do not have the capability to autonomously reach C2 on target hosts, agents need to be pre-installed on targets to simulate adversary behaviour after the virtual perimeter has been breached. Where necessary, we deploy and configure these agents on target hosts.

5.1.3 *Defensive infrastructure*

We deploy a defensive stack into the testing range which allows us to collect metrics from a defender's perspective. We choose to use Elastic Security as a SIEM, given that it is free to use for non-commercial purposes and easy to set up and integrate with endpoints. Moreover, it comes with a strong baseline of detection rules[4] that is openly and actively maintained based on fresh threat intelligence sources. Our deployment contains 623 rules which have all been enabled.

All three target endpoints are equipped with Elastic Agent[5]. This agent runs in the background and collects logs from several sources on the local machine before sending everything back to the Elastic Security stack. By default, these agents also act as an endpoint security solution - actively preventing malicious activity. We modify their configuration such that they only observe.

3 https://attackiq.com/breach-and-attack-simulation/
4 https://github.com/elastic/detection-rules
5 https://www.elastic.co/downloads/elastic-agent

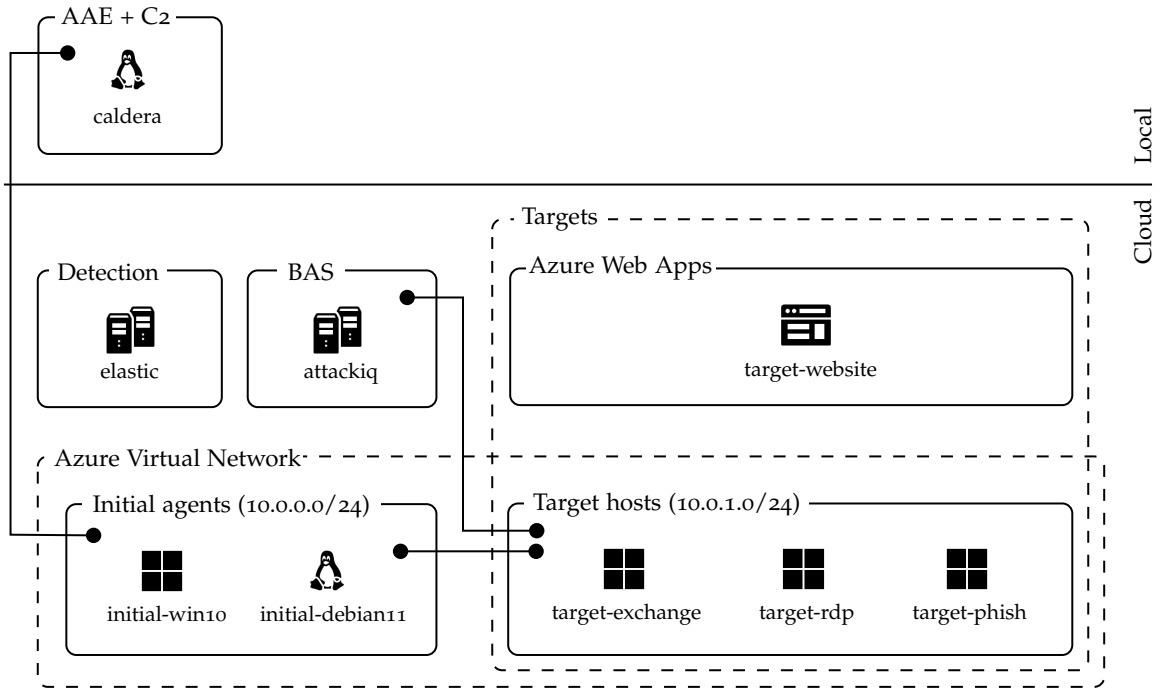

Figure 5.1: Topology of the testing range that will be utilized for anaylsis and validation

5.1.4 *Targets*

5.1.4.1 *Virtual servers*

TARGET-PHISH This host runs on a default Azure deployment of Windows 10, with the latest version of Microsoft Office installed.

TARGET-RDP This host runs on a default Azure deployment of Windows 10 with RDP enabled. The password of one of the RDP users is available on common password lists.

TARGET-EXCHANGE This host runs on Windows Server 2019 Datacenter. Microsoft Exchange Server 2019 CU9 is installed and configured - the last release before the ProxyShell vulnerability was patched.

All defensive controls that are in place by default are disabled on these hosts.

5.1.4.2 *Serverless infrastructure*

TARGET-WEBSITE This resource serves a static website. It is used as a target in the reconnaissance phase and meant to model a real-world company website. We modify the website template to include some names and email addresses that can be used in several scenarios.

5.2 EXECUTION

5.2.1 *Methodology*

To validate the basic functioning of our CALDERA integration, we will execute it in our testing range for each adversary that has been implemented. We configure the CALDERA framework and our integration as described in appendix B. The raw operation logs, as exported from CALDERA after an operation is complete (i.e. the planner is unable to generate valid links within the operation), are available on GitHub[6]. Based on the output of operations, we demonstrate that we are able to perform autonomous adversary emulation in the (pre-)compromise domain for the scenarios we defined, and we show the behaviour of our custom integration under various circumstances.

OPERATION CHARTS Operation logs will be illustrated using Gantt charts, since these convey the time path of various events that occur within an operation. Each column in a Gantt chart equals one second passing, and events are rounded to the nearest second. The top row of these charts shows agents that are part of the operation. More specifically, agents are shown based on the time they first call into CALDERA's internal C2 framework. In the remaining rows, we show links executed for each iteration the planner makes, connected to the agent that is executing the link using arrows.

For the execution of each link, we distinguish two phases. The first, depicted as a shaded block, denotes time taken between delegation of a link (the framework queuing the link for execution) and the specific agent retrieving said link for execution. Following our configuration, agents report to the framework on a two-second interval, resulting in an upper bound of two seconds for this phase. It should be noted that this phase does not always show up in the resulting chart, because it can be rounded down to zero seconds. The second phase, depicted as a white block, is the actual execution of the link by an agent. Links show the ability they contain, which is denoted by an ID that can be found in the ability reference in appendix A. Ability composition, where applied by the framework, is denoted with '←', where the left side is the ability that supports composition, and the right side is the ability that is being substituted.

5.2.2 *Results*

VULNERABILITY EXPLOITATION Figure 5.2 shows an execution log of the `precomp-scenario-vulnexp` adversary. Under this configuration, execution is linear - each ability produces knowledge or a tactical advantage that can be leveraged by exactly one new ability. After 109 seconds, an agent is spawned on the target and no new links can be generated.

6 https://gist.github.com/DiedB/52dcf0cadb0f192adea6ea0e01af7d7f

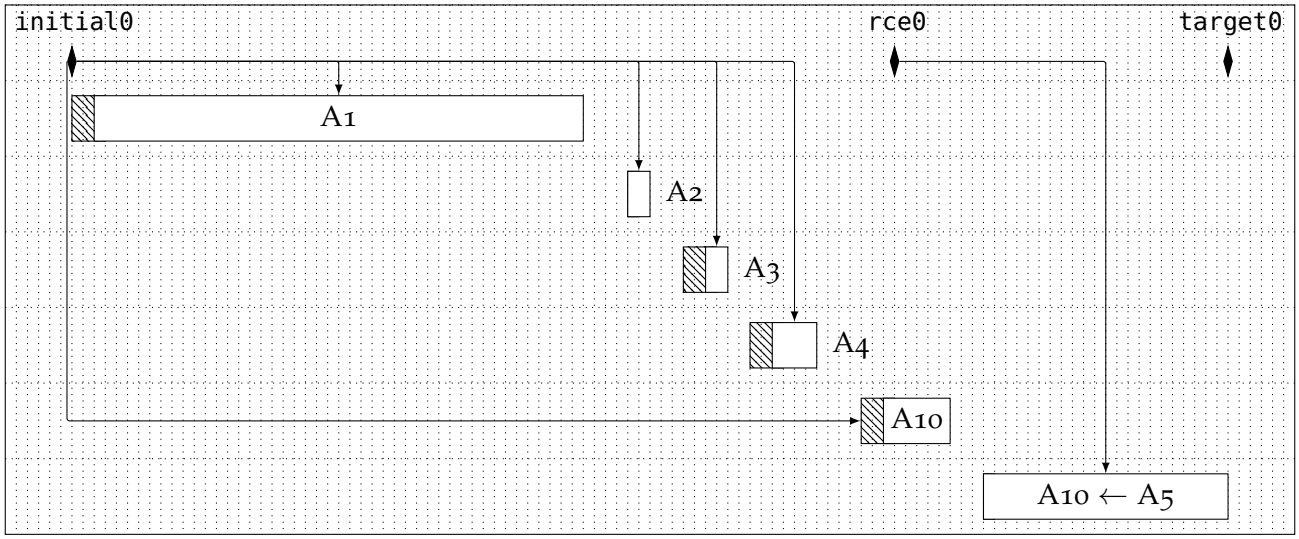

Figure 5.2: Execution of a CALDERA operation running under the `precomp-scenario-vulnexp` adversary. Further elaboration on reading these operation charts can be found in section 5.2.1.

EXTERNAL REMOTE SERVICES AND VALID ACCOUNTS Figure 5.3 visualizes an operation running under the `precomp-scenario-spray` adversary. Reconnaissance abilities (`A1` and `A6`) are executed in parallel as their knowledge requirements are fulfilled by available fact sources. Subsequently, a password spray is executed, gaining knowledge in the form of valid RDP credentials. Ability composition is applied on `A11` to reach the goal of spawning a `target` agent on the remote host.

PHISHING Figure 5.4 shows an operation running under the `precomp-scenario-phish` adversary. Reconnaissance (`A6`) and payload generation (`A8`) occur in parallel, after which the payload is staged and sent. Given that execution of the payload on the target system is a manual action, the time between the execution of ability `A9` and the first callback of agent `target0` does not hold any information in terms of plugin performance.

5.3 APPLICATION

As part of our validation strategy, we evaluate the application of our custom CALDERA integration to real-world use cases. We identify three main use cases:

SECURITY VALIDATION This is the use case that BAS platforms are designed to cover. Security can be tested by taking scenarios that simulate real threat and automatically executing these scenarios on a regular basis against an environment with security controls installed. Our CALDERA integration covers this use case in scenario emulation mode.

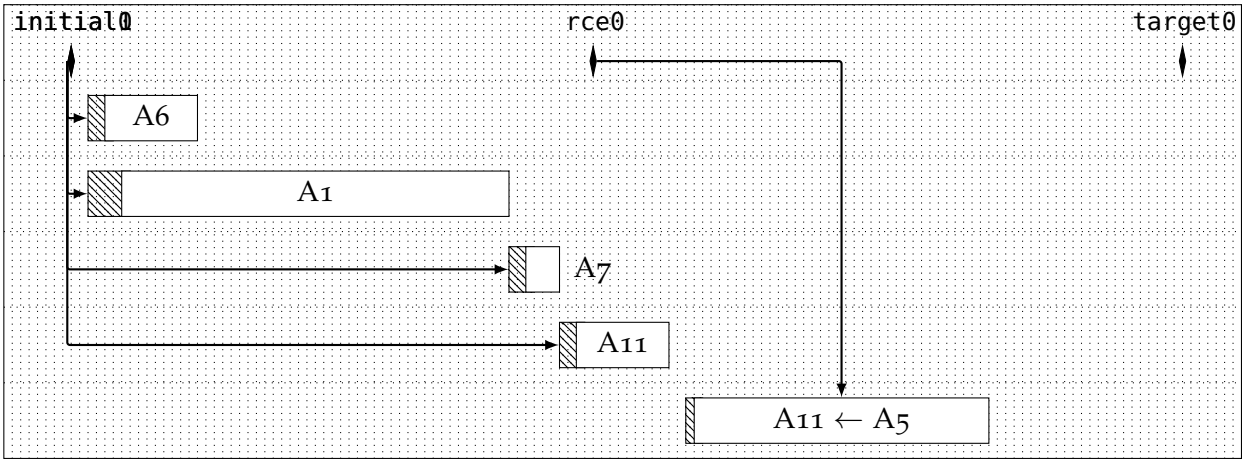

Figure 5.3: Execution of a CALDERA operation running under the `precomp-scenario-spray` adversary. Further elaboration on reading these operation charts can be found in section 5.2.1.

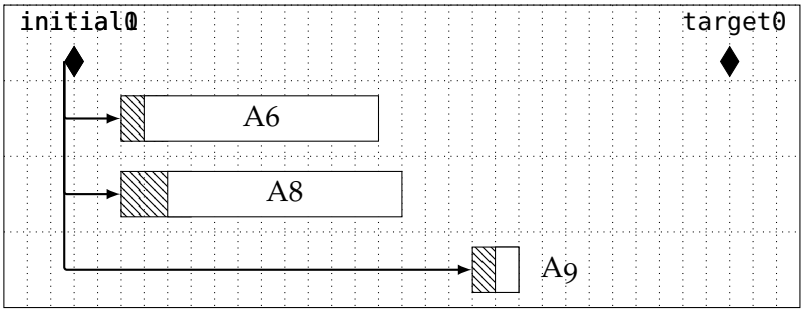

Figure 5.4: Execution of a CALDERA operation running under the `precomp-scenario-phish` adversary. Further elaboration on reading these operation charts can be found in section 5.2.1.

SECURITY CONTROL DEVELOPMENT Developing new security controls is done based on threat information, preferably as realistic and close to the defended environment as possible. Blue teams could download AAE scenarios from a public library, or receive them from a red team in a purple teaming session, and use them as a basis for 'tuning' their defensive stack. In scenario emulation mode, the framework would generate realistic indicators in the environment they are defending. This forms an effective basis for developing new security controls.

AUTONOMOUS RED TEAMING As introduced before, AAE frameworks could form an extension to a red team. The red team is more creative and sophisticated than AAE solutions, but the latter are consistent, cheap and can be executed on a regular basis. Our CALDERA integration covers this use case in autonomous mode.

In section 5.3.1, we will set up an experiment to compare detection performance between a security validation solution and our plugin in scenario emulation mode. Evaluation of the other use cases is more complicated - while we have strong reasons to assume our work provides a valuable contribution, proving its effectiveness requires a combination of expert evaluation and further extension in terms of scenarios - both of which are infeasible within the time we set out for performing this research. We will regard the evaluation of our work in these use cases as future work.

5.3.1 *Security validation*

Hypothetically, our AAE integration should - if implemented correctly - be able to trigger the same security controls as a BAS platform that is executing similar scenarios. Additionally, given the claim that our AAE integration is able to more accurately emulate real adversary procedures, we are interested in seeing whether the integration is actually able to trigger more security controls.

5.3.1.1 *Methodology*

For the experiment, we were able to access one BAS platform - the one built by AttackIQ[7]. Access was gained through their training resources, which give full access to the BAS platform for training purposes.

Given that the scenarios selected for this research are quite common in the real world, we find (very) similar scenarios in the library provided by AttackIQ. To construct a valid comparison, we carefully map our scenarios to those implemented in the AttackIQ BAS platform. This process is outlined in appendix A.3.

We then execute each of the scenarios in both AAE and BAS against the testing range, and collect relevant metrics from the detection stack.

7 https://attackiq.com/breach-and-attack-simulation/

5.3.1.2 *Results*

VULNERABILITY EXPLOITATION In the BAS platform, this scenario consists of three stages - port scanning, exploiting the vulnerability and executing a C2 agent on the remote host. The AAE implementation executes the same process through more atomic steps. For the comparison, we will combine these to align with the steps taken in the BAS scenario. Table 5.1 shows the detection rules that were triggered while running the scenario with both solutions.

The first stage in this scenario (mapping 1 in table 5.1) is a port scan which does not trigger any alerts in either execution. While this step would technically be detectable, alerting on it would likely yield many false positives. Therefore, this result is in line with expectations.

In the second stage (mapping 2 in table 5.1), differences appear in terms of detection between AAE and BAS executions. The difference for the second step can be explained because the BAS scenario does not actually use the entire ProxyShell vulnerability - it stops once a (malicious) email has been planted in one of the mailboxes. Normally, an attacker would then export the mailbox to plant a webshell on the filesystem of the target host - which does trigger several alerts in our detection stack. While this is a limitation with the specific BAS platform we test against, it does not necessarily have to exist in other BAS solutions.

The final stage (mapping 5 in table 5.1) is more interesting. Our plugin actually uses the knowledge and tactical advantage it has gained in previous steps - it drops and executes a C2 agent through the webshell that has been planted, triggering two additional detection rules in the process. A BAS platform would generally not be able to do this - rather, it uses its agent on the target host to simulate process execution. This also triggers an alert but does so in both executions.

EXTERNAL REMOTE SERVICES AND VALID ACCOUNTS The first stage in this scenario (mapping 1 in table 5.1) is a port scan - again, this does not trigger any alerts in our detection stack for either execution.

The second stage (mapping 3 in table 5.1) - a password spray on an external-facing RDP service - is being detected. The detection stack triggers on an RDP session being created from the internet during both executions. We would expect that a multitude of login attempts would also trigger an alert, but it does not.

Finally, the last stage (mapping 6 in table 5.1) uses the credentials that were found to drop a C2 agent on the remote host. While the BAS platform directly uses its agent on the remote host to execute a process, our AAE plugin actually employs the RDP session to drop the agent. This triggers a second alert in our detection stack.

PHISHING This scenario is not too interesting for comparison, as most activity is out of scope of our detection stack. The only stage that is tested - mapping 4 in table 5.1 - yields the same results between execution through AAE and BAS. Lots of alerts are being triggered, which is not not unexpected. Using malicious macros in Microsoft Office documents to install malware is a well-documented method of gaining initial access and strong efforts have been made on the defensive side.

5.3.1.3 *Conclusion*

This experiment shows that our integration is effective in emulating the scenarios we defined - it is able to surpass the BAS solution in terms of detection controls triggered. Additionally, there was no control our integration was not able to trigger where the BAS counterpart could. We show that this is not necessarily only because these specific BAS scenarios are lacking, but also due to architectural limitations that will likely exist in all BAS platforms. While AAE frameworks do not have all functionality that BAS platforms provide for validating security controls, AAE frameworks could certainly play a strong role in the execution aspect of this use case.

MAPPING	DETECTION AAE	DETECTION BAS
1	-	-
2	• Exporting Exchange Mailbox via PowerShell • Microsoft Exchange Server UM Writing Suspicious Files	-
3	• RDP (Remote Desktop Protocol) from the Internet	• RDP (Remote Desktop Protocol) from the Internet
4	• Process Execution from an Unusual Directory • Suspicious PowerShell Engine ImageLoad • Suspicious WMI Image Load from MS Office • Malicious Behavior Detection Alert: WMI Image Load via Microsoft Office • Malicious Behavior Detection Alert: RunDLL32 with Unusual Arguments • Malware Detection Alert • Malicious Behavior Detection Alert: Unusual PowerShell Engine ImageLoad	• Process Execution from an Unusual Directory • Suspicious PowerShell Engine ImageLoad • Suspicious WMI Image Load from MS Office • Malicious Behavior Detection Alert: WMI Image Load via Microsoft Office • Malicious Behavior Detection Alert: RunDLL32 with Unusual Arguments • Malware Detection Alert • Malicious Behavior Detection Alert: Unusual PowerShell Engine ImageLoad
5	• Process Execution from an Unusual Directory • Microsoft Exchange Worker Spawning Suspicious Processes • Webshell Detection: Script Process Child of Common Web Processes	• Process Execution from an Unusual Directory
6	• RDP (Remote Desktop Protocol) from the Internet • Process Execution from an Unusual Directory	• Process Execution from an Unusual Directory

Table 5.1: Detection rules triggered for each of the mappings as identified in appendix A.3, compared between AAE and BAS execution

6

CONCLUSION

This thesis explores the potential for further introducing automation in the process of adversary emulation, specifically through the following research question:

RQ How can threat actors be emulated in their (pre-)compromise tactics, techniques and procedures in an autonomous fashion?

Through extension of previous work on autonomous post-compromise adversary emulation by MITRE, we were able to present a *proof-of-concept* on autonomous (pre-)compromise adversary emulation for threat scenarios involving phishing, brute-forcing credentials against external-facing services and vulnerability exploitation. These threat scenarios are established based on statistical analysis on (pre-)compromise modus operandi of threat actors in chapter 3 and their combination shows sufficient coverage of the (pre-)compromise domain. While showing that autonomous emulation is possible for the specific scenarios we defined, and showing improvement over related automation efforts, we also look at the issue of doing autonomous (pre-)compromise adversary emulation in a broader sense by generalising domain-specific issues into a platform that facilitates building novel scenarios. Practical boundaries exist to the process of adversary emulation, and we identify several additional boundaries that emerge when automating the process, namely automation of procedures that are highly manual in nature and plugging into third-party C2 frameworks.

6.1 RESEARCH LIMITATIONS

The first limitation we encounter in this research is that we do not have Cyber Threat Intelligence (CTI) available at the level of detail and quality that would be desired. This limitation manifests itself in the analysis on technique usage frequency, but also in emulation accuracy of the scenarios we implement. Tactical Cyber Threat Intelligence (CTI) at the level of quality and detail that is needed to do adversary emulation does exist, but is only offered by commercial providers and comes at a high price.

Secondly, this work only implements a few emulation scenarios. Implementing scenarios is time-consuming work, especially because adversary emulation frameworks are still in their infancy and because each scenario needs target infrastructure to test against. While the scenarios in this work are carefully chosen and reach quite a good coverage of the (pre-)compromise domain in terms of technique usage, implementing additional scenarios and

increasing technique coverage would likely yield extra results in the form of emulation platform requirements and automated planning requirements. However, we do expect that implementing additional scenarios will quickly reach a point of diminishing returns.

Another limitation can be seen in the validation of this work, which was fully performed in a lab environment. While we made an effort to ensure that this environment closely reflects the real world, the research would benefit from additional expert evaluation by red and blue teams.

Finally, our main research question is broadly scoped, even broader than we could have imagined beforehand. Managing the scope of the research was a recurring issue during the research phase. While most choices that lead to a concrete limitation are already described in the foregoing enumeration, we would like to emphasise that there are many complex aspects to autonomously emulating adversaries, and we have only touched ground with our contribution.

6.2 FUTURE WORK

We identified that a large amount of relevant and promising scientific effort exists in the domain of automated planning for offensive security operations. This work is, nearly without exception, built and validated against modeled environments. Additionally, we identified that the factor of uncertainty - not having a priori knowledge of the environment you are attacking - is not always taken into account in related work. Simultaneously, AAE frameworks are strong in execution but do not implement the state-of-the-art in automated planning. Given that a baseline for automated emulation of adversarial procedures across the entire killchain now exists in the form of CALDERA with its existing capabilities and our extension, an interesting angle for future research would be to connect theory and practice. This would likely yield interesting insight in gaps that exist on either side of the coin.

Over the course of this research, we noticed that it is hard to assess and validate work on adversary emulation. Manual campaigns in the field of cybersecurity are often executed on a best effort basis, and most existing work on automating the process lies closer to simulation than emulation. Further improvements to the process of adversary emulation could be guided by an objective method of assessing the maturity of a campaign.

Finally, further research could be done into the applicability, usability and flexibility of various AAE solutions. Most existing work, including our research, does not surpass the *proof-of-concept* stage. While the ideas that have been implemented are promising for several practical use cases, they need to be applied to study their effects and limitations.

A

PLUGIN REFERENCE

A.1 ABILITIES

In this thesis, we regularly refer to the abilities implemented into our *proof-of-concept*. The table on the following page gives a more detailed overview of these abilities and several properties they carry. The leftmost column contains an identifier for each ability that is being used as a reference within this work.

ID	PLUGIN ID [a]	TECHNIQUE	D [b]	C [c]	DESCRIPTION
A1	`00837c5f-b757-4503-840b-c9fd-ec8ac2b7`	T1595.001	✓		Use Nmap to scan for public-facing services in an IP range
A2	`72d02ae7-0972-46c4-be19-3998-8c8fb13e`	T1595.002	✓		Use Nmap to scan for vulnerabilities in public-facing services
A3	`849a38ab-6fd2-4c2f-817a-a07c-9d60fbf4`	T1589.002 T1190 T1588.005	✓		ProxyShell email enumeration - use SSRF vulnerability to query AD for email addresses
A4	`f3369229-12b5-42b8-b26b-8726-2802d139`	T1190 T1505.003 T1588.005	✓		Drop webshell through chain of ProxyShell vulnerabilities
A5	`38b639c7-4812-4933-9356-c187-7d1eec03`	T1059.001 T1105 T1071.001 T1588.001 T1608.001	✓		Spawn C2 agent (on target host)
A6	`1b21c07f-3367-45eb-8231-6474-3bb292c1`	T1589.002			Find email addresses on a public website
A7	`46ca03a2-ffdd-4b6b-878c-84f7-0d1df7d4`	T1110.003	✓		Execute a password spray using Hydra
A8	`ebd7ebd0-2a8e-4f83-adc0-4e13-cbd47082`	T1588.001 T1608.001			Generate malicious document containing Sandcat agent
A9	`7b8a9df6-ba88-47a5-8de1-b1d3-82d4eff9`	T1566.001 T1204.002	✓		Send malicious payload via email - execution is assumed on receipt
A10	`554cc237-02c1-423f-ba52-e612-bd1b4d1c`	T1505.003	✓	✓	Execute a command through an ASP.NET webshell
A11	`bee06102-c9a1-42d9-8600-e910-f310bdbb`	T1078 T1059.003	✓	✓	Execute a command through an RDP session

a The ID for the ability as used in our CALDERA plugin

b Detectable - whether execution of the ability can (at least partially) be observed by the detection stack deployed in our testing range

c Composable - indicates whether an ability is designed to be composed with other abilities - this property is used in RCE agents

A.2 KNOWLEDGE MODEL

Our *proof-of-concept* uses a structured knowledge model which is used by abilities and the planner to keep track of the state of an operation and to make decisions. Table A.1 shows all fact traits that we use, including the abilities that use or produce facts of this trait and an example value. The knowledge model is a graph, and traits can therefore be related through (named) edges. These relationships are being shown in figure A.2.

Internal fact traits that are used as part of a workaround to mitigate missing functionality in the CALDERA framework are omitted here. We also omit traits that are being used to keep track of lateral movement and traits that express CALDERA's internal configuration, although these are both being used by our abilities.

TRAIT	SOURCE [a]	USING [b]	PROD. [c]	EXAMPLE
`target.range`	✓	A1	-	`10.0.0.0/24`
`target.domain`	✓	A6	-	`x.com`
`target.employee.email`		A9	A3, A6	`j.doe@x.com`
`target.employee.username`		A11	A7	`admin`
`target.employee.password`		A11	A7	`Summer2020`
`target.ip`		A2, A3, A4, A7, A11	A1	`10.0.0.1`
`target.port`		A2, A7, A11	A1	`3389`
`target.vulnerability`		A3, A4	A2	`proxyshell`
`target.webshell.url`		A10	A4	`x.com/shell.aspx`

a Whether this fact is expected to be present at plugin initialisation - if not, this fact will be 'learned' during the operation
b Abilities that use facts with this trait
c Abilities that produce facts with this trait

Table A.1: Fact traits in use by our proof-of-concept CALDERA integration, including the abilities that are using and producing these traits and example values

SOURCE TRAIT	EDGE	TARGET TRAIT
`target.employee.username`	`has_password`	`target.employee.password`
`target.remote.ip`	`has_open_port`	`target.remote.port`
`target.remote.ip`	`vulnerable_to`	`target.vulnerability`

Table A.2: Potential edges between facts in our proof-of-concept CALDERA integration

A.3 BAS SCENARIO MAPPING

To evaluate our implementation against BAS, we need to map our scenarios to those implemented in a BAS platform. For our experiment, we make this comparison against the BAS platform offered by AttackIQ. This mapping, including any relevant remarks on the mapping process, is provided in the table below. Because execution steps do not always have the same scope between the two solutions, we align our comparison with the largest step in either solution. Ability composition, where applied on the side of our plugin, is indicated with '←'. Any abilities that are out of scope of our detection setup are omitted in this mapping, as they are irrelevant in the experiment.

MAPPING	ABILITY ID [a]	ATTACKIQ ID [b]	REMARKS
1	A1	`0aecbfba-8818-489c-b816-4ff1-f73a1f24`	-
2	A2 A3 A4	`ddc774a4-5d02-416d-96d5-f9c6-805107c9`	The BAS scenario implements three plugin abilities in one - it verifies whether the system is vulnerable, enumerates email addresses from the Exchange Server and drops a malicious attachment into one of the mailboxes. It does not follow the last step of the exploit chain however - exporting the mailbox to plant the webshell on the filesystem.
3	A7	`25aed0af-be94-4eb0-9171-add0-f414060c`	The BAS password spraying scenario is configured to use the same username and password lists as used in the AAE scenario
4	A9	`04ed47b9-145c-46f6-9434-f9f5-af27a2d2`	The BAS scenario simulates process execution on a host - we pass the generated phishing document as payload.
5	A10 ← A5	`04ed47b9-145c-46f6-9434-f9f5-af27a2d2`	The BAS scenario simulates process execution on a host - we pass the same C2 agent as used in ability `A5` as payload.
6	A11 ← A5	`04ed47b9-145c-46f6-9434-f9f5-af27a2d2`	The BAS scenario simulates process execution on a host - we pass the same C2 agent as used in ability `A5` as payload.

a The ID for the ability as defined in the table in Appendix C.1
b The internal scenario ID as used in the AttackIQ BAS platform

B

CONFIGURATIONS

B.1 CALDERA

B.1.1 *Framework*

The framework is running with version `4.0.0-alpha.2`[1]. The following modifications are made to its default configuration:

```
# HTTP contact point for agent beacons
app.contact.http: http://172.16.0.2:8889
plugins:
  - sandcat       # Agent framework
  - precomp       # Our own plugin
  - stockpile     # Expected by the CALDERA framework
  - debrief       # For debugging completed operations
# Prevent interfering with Prelude Operator
port: 8889
```

B.1.2 *Agents*

The following modifications are made to CALDERA's agent configuration:

```
# Do not run any bootstrap abilities on agent check-in
bootstrap_abilities: []
deployments:
  # Sandcat agent - no other agents used
  - 2f34977d-9558-4c12-abad-349716777c6b
# Let agents beacon in on two-second intervals
# Do not implement jitter for testing consistency
sleep_max: 2
sleep_min: 2
```

1 https://github.com/mitre/caldera/releases/tag/4.0.0-alpha.2

B.2 PLUGIN

In the testing and validation process, the fact source for our plugin is configured as follows:

```
name: precomp-source
facts:
  # The target CIDR range for the operation
  - trait: "target.range"
    value: "10.0.1.0/28"
  # Domain name of target
  - trait: "target.domain"
    value: "website.precomp-test.local"
  # SMTP configuration
  - trait: "internal.phishing.smtp.server"
    value: "172.16.0.2:1025"
  - trait: "internal.phishing.from"
    value: "test@precomp-test.local"
rules: []
relationships: []
```

BIBLIOGRAPHY

[1] Microsoft Threat Intelligence Center (MSTIC). *HAFNIUM targeting Exchange Servers with o-day exploits*. 2021. URL: https://www.microsoft.com/security/blog/2021/03/02/hafnium-targeting-exchange-servers (visited on Nov. 29, 2021).

[2] *Annual State of Phishing Report*. Tech. rep. Cofense, 2021. URL: https://cofense.com/wp-content/uploads/2021/02/cofense-annual-report-2021.pdf.

[3] 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*. 2017.

[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*. 2017, pp. 1–12.

[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*. 2016, pp. 363–373.

[6] Anton Chuvakin Augusto Barros. *Utilizing Breach and Attack Simulation Tools to Test and Improve Security*. Tech. rep. Gartner, 2018.

[7] John A. Bland, Mikel D. Petty, Tymaine S. Whitaker, Katia P. Maxwell, and Walter Alan Cantrell. 'Machine Learning Cyberattack and Defense Strategies.' In: *Computers & Security* 92 (2020), p. 101738. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2020.101738. URL: https://www.sciencedirect.com/science/article/pii/S0167404818309799.

[8] David Chismon and Martyn Ruks. 'Threat intelligence: Collecting, analysing, evaluating.' In: *MWR InfoSecurity Ltd* (2015). URL: https://www.foo.be/docs/informations-sharing/Threat-Intelligence-Whitepaper.pdf.

[9] CrowdStrike. *2020 Global Threat Report*. URL: https://go.crowdstrike.com/rs/281-OBQ-266/images/Report2020CrowdStrikeGlobalThreatReport.pdf (visited on Sept. 24, 2021).

[10] Cybersecurity and Infrastructure Security Agency. *Joint Cybersecurity Advisory - Top Routinely Exploited Vulnerabilities*. 2021. URL: https://www.cisa.gov/uscert/sites/default/files/publications/AA21-

209A_Joint_CSA%20Top%20Routinely%20Exploited%20Vulnerabilities.pdf (visited on Nov. 28, 2021).

[11] European Central Bank. *TIBER-EU Framework*. URL: https://www.ecb.europa.eu/pub/pdf/other/ecb.tiber_eu_framework.en.pdf (visited on Mar. 22, 2021).

[12] G7. *G7 Fundamentals for Threat-led Penetration Testing*. URL: https://www.bundesbank.de/resource/blob/764690/792725ab3e779617a2fe28a03c303940/mL/2018-10-24-g-7-fundamental-elements-for-threat-led-penetration-testing-data.pdf (visited on Apr. 2, 2021).

[13] Eric M Hutchins, Michael J Cloppert, Rohan M Amin, et al. 'Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains.' In: *Leading Issues in Information Warfare & Security Research* 1.1 (2011), p. 80.

[14] Jonas Bauters. *Thoughts on Red Team Nomenclature*. URL: https://blog.nviso.eu/2020/01/23/thoughts-on-red-team-nomenclature/ (visited on Jan. 3, 2022).

[15] Ivan Kovačević and Stjepan Groš. 'Red Teams-Pentesters, APTs, or Neither.' In: *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE, pp. 1242–1249.

[16] Philippe Langlois. *Verizon 2020 Data Breach Investigations Report*. Tech. rep. Verizon, 2020. URL: https://www.verizon.com/business/en-gb/resources/reports/dbir/2020/.

[17] Doug Miller, Ron Alford, Andy Applebaum, Henry Foster, Caleb Little, and Blake Strom. *Automated adversary emulation: A case for planning and acting with unknowns*. Tech. rep. MITRE CORP MCLEAN VA MCLEAN, 2018.

[18] Cyber Threat Intelligence Technical Committee OASIS Open. *Introduction to STIX*. URL: https://oasis-open.github.io/cti-documentation/stix/intro (visited on Mar. 2, 2021).

[19] Adam Pennington and Jen Burns. *Bringing PRE into Enterprise*. URL: https://medium.com/mitre-attack/the-retirement-of-pre-attack-4b73ffecd3d3 (visited on Aug. 12, 2021).

[20] Shilpi Handa Pete Shoard. *Hype Cycle for Security Operations, 2021*. Tech. rep. Gartner, 2021.

[21] Paul Pols and Jan van den Berg. 'The unified kill chain.' In: *Cyber Security Academy* (2017).

[22] Andrew Ramsdale, Stavros Shiaeles, and Nicholas Kolokotronis. 'A Comparative Analysis of Cyber-Threat Intelligence Sources, Formats and Languages.' In: *Electronics* 9.5 (2020). ISSN: 2079-9292. DOI: 10.3390/electronics9050824. URL: https://www.mdpi.com/2079-9292/9/5/824.

[23] IBM Security. *X-Force Threat Intelligence Index 2021*. URL: https://www.ibm.com/downloads/cas/M1X3B7QG (visited on Sept. 24, 2021).

[24] FireEye Mandiant Services. *M-Trends 2021*. URL: https://www.arrow.com/ecs-media/16352/fireeye-rpt-mtrends-2021.pdf (visited on Sept. 24, 2021).

[25] Digital Shadows. *Initial Access Brokers: An Excess of Access*. URL: https://www.digitalshadows.com/blog-and-research/initial-access-brokers-listings-increasing-in-2021/ (visited on Aug. 12, 2021).

[26] Sophos. *The Active Adversary Playbook 2021*. URL: https://news.sophos.com/en-us/2021/05/18/the-active-adversary-playbook-2021/ (visited on Sept. 24, 2021).

[27] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. 'Mitre att&ck: Design and philosophy.' In: *Technical report* (2018).

[28] DEVCORE Research Team. *FROM PWN2OWN 2021: A NEW ATTACK SURFACE ON MICROSOFT EXCHANGE - PROXYSHELL!* DEVCORE. 2021. URL: https://www.zerodayinitiative.com/blog/2021/8/17/from-pwn2own-2021-a-new-attack-surface-on-microsoft-exchange-proxyshell (visited on Nov. 30, 2021).

[29] Microsoft 365 Defender Research Team. *Gamifying machine learning for stronger security and AI models*. URL: https://www.microsoft.com/security/blog/2021/04/08/gamifying-machine-learning-for-stronger-security-and-ai-models/ (visited on Apr. 16, 2021).

[30] The Association of Banks in Singapore. *Red team: Adversarial Attack Simulation Exercices - guidelines for the Financial Industry in Singapore*. URL: https://abs.org.sg/docs/library/abs-red-team-adversarial-attack-simulation-exercises-guidelines-v1-06766a69f299c69658b7dff00006ed795.pdf (visited on Apr. 3, 2021).

[31] Spencer Thompson and David Hunt. *Announcing Prelude*. URL: https://feed.prelude.org/p/announcing-prelude (visited on Aug. 29, 2021).