Knowledge-based Decision Making for Simulating Cyber Attack Behaviors

Stephen Frank Moskal
sfm5015@rit.edu
Knowledge-based Decision Making for Simulating Cyber Attack Behaviors

by

Stephen Frank Moskal

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science
in Computer Engineering

Supervised by

Professor Dr. Shanchieh Jay Yang
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
July 2016

Approved by:

Dr. Shanchieh Jay Yang, Professor  
_Thesis Advisor, Department of Computer Engineering_

Dr. Michael Kuhl, Professor  
_Committee Member, Department of Industrial and Systems Engineering_

Dr. Andres Kwasinski, Associate Professor  
_Committee Member, Department of Computer Engineering_
Title:

Knowledge-based Decision Making for Simulating Cyber Attack Behaviors

I, Stephen Frank Moskal, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

________________________________________
Stephen Frank Moskal

Date
Dedication

To my family. My parents Michael and Jane Moskal who have given me this opportunity and supported me throughout these years. My brother Michael Moskal II for the brotherly competition to motivate me to do just that much more. I could not do it without you and I love you guys with all of my heart.
Acknowledgments

First and foremost, the biggest thanks and highest praise goes out to Dr. Jay Yang who reached out to me when I was a second year undergrad student and miraculously stuck with me throughout this long but incredible experience. I cannot express how grateful I am to have someone who believed in me for this long and continue to do so into the next chapter of my career, this could not be done without you and here’s to the next couple of years. I would also like to thank Dr. Michael Kuhl for sticking with me, putting up with me, and introducing me to a side of engineering that I would not get otherwise. Then to Dr. Andres Kwasinski for stepping up to the plate and being apart of my committee so late in the game and I greatly appreciate it. Finally a big thanks to the original CASCADES team, Ben Wheeler and Derek Kreider for the ongoing support to make this project something that we never imagined to be years ago.
Abstract

Knowledge-based Decision Making for Simulating Cyber Attack Behaviors

Stephen Frank Moskal

Supervising Professor: Dr. Shanchieh Jay Yang

Computer networks are becoming more complex as the reliance on these networks increases in this era of exponential technological growth. This makes the potential gains for criminal activity on these networks extremely serious and can not only devastate organizations or enterprises but also the general population. As complexity of the network increases so does the difficulty to protect the networks as more potential vulnerabilities are introduced. Despite best efforts, traditional defenses like Intrusion Detection Systems and penetration tests are rendered ineffective to even amateur cyber adversaries. Networks now need to be analyzed at all times to preemptively detect weaknesses which harbored a new research field called Cyber Threat Analytics. However, current techniques for cyber threat analytics typically perform static analysis on the network and system vulnerabilities but few address the most variable and most critical piece of the puzzle – the attacker themselves.

This work focuses on defining a baseline framework for modeling a wide variety of cyber attack behaviors which can be used in conjunction with a cyber attack simulator to analyze the effects of individual or multiple attackers on a network. To model a cyber attacker’s behaviors with reasonable accuracy and flexibility, the model must be based on aspects of an attacker that are used in real scenarios. Real cyber attackers base their decisions on what they know and learn about the network, vulnerabilities, and targets. This attacker behavior model introduces the aspect of knowledge-based decision making to cyber attack behavior modeling with the goal of providing user configurable options. This behavior model employs Cyber Attack Kill Chain® along with an ensemble of the attacker
capabilities, opportunities, intent, and preferences. The proposed knowledge-based decision making model is implemented to enable the simulation of a variety of network attack behaviors and their effects. This thesis will show a number of simulated attack scenarios to demonstrate the capabilities and limitations of the proposed model.
# Contents

Dedication ................................................................. iii

Acknowledgments ......................................................... iv

Abstract .......................................................................... v

1 Introduction ............................................................... 1
  1.1 Importance of Cyber Security ........................................ 1
  1.2 Difficulty in Cyber Threat Analytics ............................... 2
  1.3 Variability of a Cyber Attack ......................................... 3
  1.4 Problem Statement .................................................... 4
  1.5 Motivation for Cyber Attack Behavior Modeling ................. 5

2 Related Work ............................................................. 7
  2.1 Attack Graph Generation and Analysis ............................. 7
  2.2 Behavioral Models ..................................................... 9
    2.2.1 Game Theory ..................................................... 10
    2.2.2 Agent Based Behavior Modeling ............................... 11
    2.2.3 Cyber Attack Ontologies ........................................ 12
  2.3 Cyber Attack Simulators and Analysis Tools ....................... 14
  2.4 Summary .................................................................. 15

3 Methodology .............................................................. 17
  3.1 Making of a Cyber Attacker ......................................... 17
  3.2 Cyber Attack Scenario Frameworks ................................. 20
  3.3 Modeling the Attacker’s Knowledge ............................... 22
    3.3.1 Skills, Rules, and Knowledge .................................. 23
    3.3.2 Network Knowledge ............................................. 24
    3.3.3 Opportunity: Using the Knowledge To Choose A Kill Chain Stage ................................................................. 25
    3.3.4 Fuzzy Logic Parameters ......................................... 27
    3.3.5 Using Parameters To Choose Kill Chain Stage ................. 31
3.4 Modeling Attacker Skills and Preferences
3.4.1 Capabilities: The Attacker’s Skill Set
3.4.2 Preferences: What the Attacker Chooses To Do Next
3.5 Combining Capabilities, Opportunity, Intent, and Preference Models

4 Design Implementation
4.1 General Process for Determining Attack Actions
4.2 Attacker Knowledge Modeling
4.2.1 Knowledge Base
4.2.2 Machine Knowledge
4.2.3 Attack Records
4.2.4 Updating the Knowledge Base
4.3 Initialization of the Attack Action Sets
4.4 Intent and Opportunities
4.4.1 Cyber Attack Kill Chain
4.4.2 Choosing Kill Chain Stage Using Knowledge: Implementation
4.5 Attacker Capabilities
4.6 Attacker Preferences
4.6.1 Variably Weighted Preferences Values
4.6.2 Preference Parameters To Determine Weights
4.7 Choosing an Attack Action
4.7.1 No Actions To Select

5 Experiments and Results Analysis
5.1 CASCADES Setup
5.1.1 Virtual Terrain Setup
5.1.2 Attacker’s Intent
5.1.3 Attacker Behavior Configurations
5.2 Analysis
5.2.1 Base Network Configuration
5.2.2 Misconfigured Network

6 Conclusion and Future Work
6.1 Conclusion
6.2 Future Work
6.2.1 Knowledge Upgrades
6.2.2 Fuzzy Logic
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Kill chain stages as defined by various groups and companies</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>The initial list of the fuzzy logic parameters available to the user from the simulation</td>
<td>29</td>
</tr>
<tr>
<td>3.3</td>
<td>The base set of attacker preferences that can be modified by the user</td>
<td>39</td>
</tr>
<tr>
<td>4.1</td>
<td>List of variables that represents one instance of the attacker’s machine knowledge</td>
<td>49</td>
</tr>
<tr>
<td>4.2</td>
<td>The list of variables being tracked by the simulator as statistics of the attacker’s actions</td>
<td>50</td>
</tr>
<tr>
<td>4.3</td>
<td>Weighting functions available to the user for some of the preference functions</td>
<td>62</td>
</tr>
<tr>
<td>4.4</td>
<td>The user configurable variables that are used to control the attacker’s preferences</td>
<td>64</td>
</tr>
<tr>
<td>5.1</td>
<td>List of the machines on the virtual terrain and their access permissions</td>
<td>69</td>
</tr>
<tr>
<td>5.2</td>
<td>The modified VT permissions with the misconfigurations in bold</td>
<td>70</td>
</tr>
<tr>
<td>5.3</td>
<td>ABM configuration summaries for each attacker</td>
<td>73</td>
</tr>
<tr>
<td>5.4</td>
<td>ABM preference summary for each of the attackers</td>
<td>74</td>
</tr>
<tr>
<td>5.5</td>
<td>Average number of steps taken by each of the attackers</td>
<td>74</td>
</tr>
<tr>
<td>5.6</td>
<td>Failed attack rate for each type of attackers on the base network</td>
<td>75</td>
</tr>
<tr>
<td>5.7</td>
<td>Average number of steps taken by each of the attackers on the misconfigured network</td>
<td>80</td>
</tr>
<tr>
<td>5.8</td>
<td>Failed attack rate for each type of attackers on the misconfigured network</td>
<td>80</td>
</tr>
<tr>
<td>5.9</td>
<td>Average percent change of attack steps between the base network configuration and the misconfigured network</td>
<td>80</td>
</tr>
<tr>
<td>5.10</td>
<td>Average percent change of attacks against the service “enterprise_linux” between the two networks</td>
<td>84</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Example of the cyber kill chain defined by MITRE. .......................... 14
3.1 Example setup of the linguistic variable “total failed actions” describing what is acceptable and unacceptable ............................ 26
3.2 The flow of the kill chain stage selection using fuzzy logic showing the inputs of the knowledge and intent. ................................. 27
3.3 Example fuzzy rule set for each of the kill chain stages in the MVKC .......................... 31
3.4 The distribution of membership for each of the kill chain stages in the MVKC .......................... 32
3.5 The membership function of the kill chain stages when the attacker has no knowledge on the network, choosing recon .......................... 32
3.6 The membership function of the kill chain stages after the attacker has recent performed recon scans, choosing breach .......................... 33
3.7 The membership function of the kill chain stages after finding a machine with intent, choosing exfiltration ................................. 34
3.8 The membership function of the kill chain stages after finding a machine with intent, choosing exfiltration ................................. 34
4.1 ABM basic black box design where the core logic only makes a single call to the ABM. .......................... 42
4.2 Internal ABM states to determine the final attack action from the attacker’s intent, opportunities, capabilities, and preferences. .......................... 43
4.3 The general process for the determine attack action method for the attacker behavior model called by the simulation core logic. .......................... 45
4.4 UML diagram describing the attacker’s knowledge design. .......................... 46
4.5 UML diagram describing the Knowledge Fields enumerated type. .......................... 50
4.6 UML diagram showing the three different stages ................................. 54
4.7 An example of a FCL file for JFuzzyLogic for the ABM. .......................... 57
4.8 An example of the state definition in the fuzzy logic representing independent states. .......................... 58
4.9 UML diagram describing the attacker’s capabilities design. .......................... 60
5.1 Architecture overview of the CASCADES simulator. .......................... 67
5.2 Visual representation showing the connections and machines in the virtual terrain being used. .................................................................................................................. 68
5.3 Comparison of the number of attack steps each attacker took over 1000 separate simulations. ................................................................. 75
5.4 Comparison of the occurrences of the recon and breach stages for each attacker type on the base configured network. .............................. 77
5.5 The percentage of times a service was exploited by each of the attackers on the base network. (Services pertaining to intent are bolded). 79
5.6 Comparison of the number of attack steps each attacker took over 1000 separate simulations on the misconfigured network ........................ 81
5.7 Comparison of the occurrences of the recon and breach stages for each attacker type on the misconfigured network. ................................. 82
5.8 The percentage of times a service was exploited by each of the attackers on the misconfigured network (Services pertaining to intent are bolded). *Service on misconfigured machine ................................. 83
Chapter 1

Introduction

1.1 Importance of Cyber Security

As the world becomes more and more technologically advanced and reliant on computers to keep track of their lives or run their businesses, the importance of keeping these systems secure becomes critical in every day life. However, in most cases, people do not realize how important cyber security is until their credit card numbers are stolen or their customers data is taken. This ignorance only has been realized widespread in late-2013 when the large corporation Target was infected with malware compromising the identities of over 70 million people worldwide [38]. FireEye, the company who was monitoring Target’s network at the time, reports that the median number of days between when the attack is successful to when the attack is discovered is 146 days [8]. Even though Target had many defenses implemented and detected the malware relatively early, enterprise level networks still can be extremely difficult and expensive to secure due to the configuration of the network like the services installed and access control.

To combat cyber attacks, defense tools and tactics like firewalls, intrusion detection systems (IDS), and penetration testing are used to help mitigate these threats. However most tools like firewalls and IDS’s only protect from attacks that are happening live and if they fail, there is nothing they can do further. Penetration testing, although can be very effective, is only performed at the time of the test and it effectiveness relies on the skills
of the tester. The issues with these techniques is that if the attacker has compromised the network in some way, it’s usually too late. This is why recently the research community has focused performing analytics on cyber attacks and threats to measure the effects and impact of certain attacks before the attack can actually happen. Cyber-threat analytics fills the gap between detecting cyber attacks and defending against cyber attacks predicting what could happen in various scenarios.

1.2 Difficulty in Cyber Threat Analytics

Prediction in any field of work is usually expected to come with some amount of uncertainty and inaccuracies, with meteorology being the extreme case since that is almost never 100 percent correct. As expected cyber attacks are no exception to this proposition however unlike weather prediction which has literally hundreds of years of previous weather data, there is little available data of detailed cyber attack information to base predictions off of. The issue is because of the sensitivity and complexity of this data most of the victims are hesitant to release this information. Sensitivity in the sense that releasing detailed information about an attack can expose even more vulnerabilities and information against them which could also negatively impact them from a business standpoint. These attacks are also complex requiring detailed logs and step-by-step analysis of vulnerabilities and techniques used during the attack which needs the actual attacker’s input to retrieve that data. Like meteorology where robustness is added to models of the weather to become as accurate as possible without physically modeling every variable possible, same concept can be applied to cyber attack modeling to reduce the number of variables to make up for the high uncertainly models. By modeling the networks and cyber attacks to various degrees it is possible make up for the lack of data to synthesize the effects of these attacks.
1.3 Variability of a Cyber Attack

To understand the difficulty of the prediction and modeling of a cyber attack, understanding the variability and complexity of an attack is key. First, the network and its configuration plays a major role in an attack. An average network may be hundreds to thousands of machines with various permissions, programs, and protection. Each of these dictate the sort of attacks that could happen on the network and is critical modeling an attack with some amount of realism. Permissions dictate whom can communicate, whom can be accessed from the Internet, etc. Access control is key to stopping unauthorized users accessing someone who they should not. Programs or services installed on a machine in a network defines the types of vulnerabilities that are available to an attacker. Management of the services can reduce the types of attacks that can happen on the machine. Then lastly the cyber attack defenses implemented on a network, a firewall for example, can impede attackers and possibly detecting attackers during the attack acting as a deterrent. It is obvious that to model the effects of an attack on the network that you must model a network but there is variability in the sense of the granularity the network is modeled. One may choose to model the network at a packet-level or use a real network, each has their pros and cons.

At a high level however, networks are relatively static and only introduces variability in the terms of what is possible in an attack. However the most variable aspect of a cyber attack is the attacker themselves. Modeling only the network could reveal what is possible to be done from an vulnerability perspective, but not whether or not it is actually possible to be done by someone. Since every human is different, the process at which an attacker will attack a network will vary between each attacker. This is interesting from a research perspective since by modeling the attacker one can not only see if their network is vulnerable but how it is vulnerable. Given the same network one attacker may approach the network differently than another, take different paths, and use different vulnerabilities which may effect the network very differently. One attacker maybe successful in ex-filtrating a critical asset in the network and leave little traces over the course of many days, while another
may run in and take what they can find quickly setting off many alarms. Understanding the differences between attackers and their behaviors can be used to analyze the effects of the attacks and then can be used for early detection and prediction later.

Interesting research has been conducted in the past 10 years to address the variability in cyber attacks and cyber attack behaviors to better analyze the impact of these attackers. On one side research has been done in using the vulnerabilities in the network to expose possible and realistic attack paths [18, 36, 52, 12]. However the downside to this approach is that these techniques only takes in account the vulnerabilities on the network exposing no real differences between types of attackers. Other works have addressed this issue by modeling the capabilities and opportunities of adversaries [46] or applying the methodology of Game Theory [50] to model the attacker and defense. None of these methods model the attacker based of information that the attacker gains throughout the attack. The information the attacker gains throughout the attack plays a critical role into what decisions the attacker makes. This concept can be seen in the agent based modeling techniques in the adversary simulation tool NeSSI2 [15] and the attacker behavior model in the Multi-stage Attack Scenario Simulation (MASS) [31]. These however fail to provide a structure in which the attacker learns specific details about the targets and be able to dynamically change goals and strategies through out the attack. This sort of knowledge based design to model an attacker allows for a flexible description of cyber attackers that provides the capability to model proactive and reactive behaviors.

1.4 Problem Statement

Previous works have identified two major factors that are dependent on each other for accurate prediction and modeling of cyber threats: the network configuration and the attacker. However this dependency between the threat and the target plays a role in both how successful an attacker is to achieving their goal and how secure the network is. Theoretically, if the network is perfectly secure then regardless of the attacker skill the attacker
should not be successful in achieving their goal. Likewise, a world-class attacker may find paths and vulnerabilities that professionals and analysts may miss due to the complexity of the network configurations and possibilities. This is the ongoing problem that penetration testers and threat analytic tools have tried to solve, expose possible vulnerabilities in a network. However, the other issue is how to analyze this dependency between the attacker and network and measure effects the attacker has on a network but also how network’s configuration effects the behaviors and intentions of the attacker. Current works that do address both the attacker and the network configurations face issues with: generating realistic attack scenarios, are inflexible with the types of attacks modeled, or are extremely difficult to configure to model certain scenarios. This is again due to the complexity of the attacker’s decision processes themselves and with the number of variables to consider when modeling a network. This work strives to find a balance between the model complexity of the cyber attacker along with the description of the network to measure the effects each of the models have on one another by examining the attacker’s actions taken when attacking a network.

1.5 Motivation for Cyber Attack Behavior Modeling

The motivation for this work is continuing on previous work by Moskal et al. [30, 31] where cyber attacks where modeled and simulated to analyze the possible attacks that could happen on the network. The main focus of this work is to model the cyber attacker’s behavior in a manner that allows for flexible description of many different types of attackers while maintaining reasonable realism in the types of attacks that can be performed. By modeling an attacker more like the way an attacker would actually think, this allows us to understand the differences different attackers have on the same network or how a single attacker can effect different types of network. This flexibility may help to alleviate the skills and time needed to perform this type of analysis on a network. The key goal is to develop a framework to model the decision process of an attacker based on both deterministic factors like the network and knowledge but also probabilistic factors to allow for randomness in
simulation. While the goal is not to be able to comprehensively model every type of attacker behavior but to identify what needs to be modeled to describe an attacker, much like what was identified for the network above.

By modeling cyber attackers more akin to how an actual cyber attacker would make decisions based on skills, rules, and knowledge, it is possible synthesize attacker behavior data that would be difficult to achieve otherwise. Combining both rule based and knowledge based attack action generation provides robust and diverse attack track generations while still providing realistic results because rules and knowledge are in constant check with each other. Meaning that the rules can not be applied if the knowledge is not developed enough and the flexibility of the knowledge can’t be used if the rules are too restrictive. Applying this design to simulations allows for a deeper understanding of what is the impact of many different types of attackers by analyzing the types of attacks were performed and being able to what the attacker needed to know to perform the attacks. Then finally this feeds back to the possible end user who is trying to protect their networks from attacks that penetration testers did not think of or other tools that do not have the defenses to defend against. This gives a deeper understanding of how the vulnerabilities are being used and how it can impact the network before the attack can happen which then something can be done about it.
Chapter 2

Related Work

Cyber-threat analytics is a relatively young field and is diverse in the types of approaches taken to perform predictive analysis of cyber attackers. These approaches consist of vulnerability assessments and mitigation, analytical approaches such as using attack graphs and game theory, and cyber attack modeling and simulation. Each have their own benefits and downfalls, where no approach is necessarily better than another due to the difficulty of predicting human behavior in the first place. In the later part of this chapter, we will shear the trade off between model accuracy, description flexibility, and ease of use. In the current work, one or two of three of these aspects are satisfied while significantly impacting the other aspects. Current works use mathematical models such as attack graph generation or game theory or other more theoretical approaches such as agent based modeling, attack ontologies, or simulations to aid in the analysis of adversaries.

2.1 Attack Graph Generation and Analysis

The purpose of network penetration test is to expose potential weaknesses in a network that is available to a potential attacker. With knowing the vulnerabilities of a network, the tester/attacker can then exploit them to further penetrate the network to learn more information. A penetration tester will use this information to expose more weaknesses until they have exhausted all their possible options. This is developing what is known as an attack graph which is a collection of all the possible paths an attacker could follow in a
network. This process was traditionally done by hand either by the attacker or the analysis team and can be a tedious process. In 2002, Sheyner et al.[44] formalized the process to automatically generate an exhaustive set of possible attack graphs given a network. Attack graphs are generated using by describing the network and the attacker’s knowledge of that network and then describing the set of states that describe the actual attacks that can happen.

Given a large network the number of possible attack paths could be extremely large. The follow up work by Jha et al.[19] proposed two methods to find which attack graphs are most critical and which defense techniques are most effective. The automatic attack graph generation still requires every different type of possible attack to be modeled. In their paper, Jha et al. only 4 possible types of attacks. In the case of Sheyner et al.[44] a network of two hosts with a IDS and firewall was modeled. This produced an attack graph of 5948 nodes with 68364 edges which is extremely large for very few types of attacks and an unrealistically small network. This method of analysis does not have the flexibility, scalability, or the ease of use needed to successfully assess the networks weaknesses.

Qui et al.[36] in 2004 used attack graphs to generate patterns of IDS alerts to aid in the prediction of future and ongoing attacks. Using these attack graphs and cyber attack domain knowledge, the likelihood of attack goals being performed can be evaluated to predict future attacks. Immediately, this is an issue because this method requires every attack graph to be converted into a casual network and a cyber security expert must analyze it for its likelihood. This has two problems, first attacks that do not strictly follow the attack plan cannot be modeled and the likelihood is based solely on the expertise of the expert. In 2010, Xie et al.[52] addressed this uncertainty of variation of attacks, successfulness, and accuracy of sensor alert data by combining attack graphs with Bayesian networks. This introduced bringing in real world vulnerability databases like the National Vulnerability Database (NVD) and the Common Vulnerability Scoring System (CVSS) to use real world data to give some basis for the likelihood without needing expert knowledge for every feature.
In 2014, Kotenko et al.[24] evaluates real-time attack graph generation for predicting the likelihood of the attacker’s next steps based off of various security events. From the security event, a base skill level of the attacker can be determined which then can be used with CVSS to determine the possibility of the next steps based on the position of the attacker on the network. A common issue between the aforementioned works is developing the base attack graph that describes the scenario and goals of the attacker. With the use of Common Attack Pattern Enumeration and Classification (CAPEC) from MITRE, Kotenko et al.[25] was able to generate attack graphs based off the real-world scenarios and use those scenarios to provide more realistic predictions and other attack graphs based off their previous works.

2.2 Behavioral Models

In the previous section, the works presented analyzes the network security based off the possible attacks that could be performed on the network given one or more scenarios. In these cases, the scenarios are defined and different attackers could share the same goal but whether or not they are successful is a different problem. Understanding the attacker’s impact on the network is important because realistically not every vulnerability can be closed and some may prioritize which vulnerabilities to address due to time. If there is a exploit that can be performed by any person and have a detrimental impact to the network (ex. Heartbleed from 2014) should have a higher priority than an exploit that only the top 1% of attackers could perform on a non critical machine. The earlier attack graph works from Jha et al.[19] and Qui et al.[36] only exposed various paths an attacker could take and not whether or not an attacker would. Where the more recent attack graph works such as Xie et al.[52] and Kotenko et al.[25] used publicly available cyber attack scenario data to create attack paths that were identified as realistic but still not accounting for attacker skills or behaviors. Current techniques, as shown in this section, apply techniques like game theory or agent based modeling to capture the behaviors and the decision processes
an attacker makes throughout an attack.

### 2.2.1 Game Theory

Applying game theory to the cyber domain seems to be a fitting application because cyber security naturally has two players which are most commonly referred to as the red team vs the blue team. The red team are the attackers who are trying to penetrate the network to achieve some goal. The blue team are the defenders, which could be the network administrator, the penetration testers, or the intrusion detection system. Game theory works well in this case because the red and blue team have contradictory goals and they are competing against each other. The red team is trying to break the defenses of the blue team and the blue team is trying to stop the red team at any phase. Roy et al. [39] in 2010 performed comprehensive research of various cyber security related game theory works and compared their usefulness. They came to three conclusions: 1) Current stochastic game models only consider perfect information and assume the defender can detect every attack. 2) State transition probabilities are fixed and do not consider changing knowledge in the model and 3) Models assume that all actions are synchronous (every action has a reaction) which is not realistic since the defense is not always aware of the offense and vice versa.

Since then, Wang et al. [50] improved on the attacker versus defense relationship by introducing the Nash equilibrium and having a non-cooperative offensive vs. defensive model. This provided significant gains in network security yet also, proved to be computationally intensive even with an unreasonable assumption that the skills of the attacker’s are all equal. In 2016 machine learning was applied with game theory by Chung et al. [5] to aid in reducing false positives. As mentioned in [39], IDS’s are never 100 percent correct. It was also assumed that the defense does not have any knowledge about the attacker’s skill or intent. Q-Learning was used as a model-free reinforcement learning technique to learn the optimal policy that will maximize the effectiveness of the technique being used. This method was found to respond well against focused attackers and even random attackers,
where the data for the attacker’s were developed from National Center of Supercomputing Applications (NCSA) where only a few attacker behaviors were present.

2.2.2 Agent Based Behavior Modeling

Agent based modeling is a technique that measures the effect of one or many autonomous agents on a particular system. Agent based modeling is used to re-create and predict the behaviors of very complex algorithms by using a very simple rule based system. Agent based modeling is popular in applications that exhibit a swarm like behavior like epidemic spreads [2] or the interaction of crowds of people in an area [34]. For example, the people in the crowd are agents and they are monitoring their speed, the position of other people, and trying to get to their goal which is a position in the system. Each agent is simply trying to minimize the time it takes to arrive at their goal, however other agents may impede each other. Bottlenecks can be then found in the system by examining the areas in which the most agents slow down [27]. Recently, this same methodology and technique was applied to the cyber security domain taking various approaches to defining the agents.

In 2005, Kotenko [22] applied the Agent Based Model to describe the interactions of botnets in a Distributed Denial of Service (DDoS) attack. Kotenko modeled the steps and actions to successfully perform a DDoS attack including the different initialized agents needed to perform the simulation. Two different types of agents were used in this work, the adversary attacking system which are the automated botnets attacking the system and then the security defense system which is the intrusion detection systems (IDS) and firewalls. For each type of agent, a state machine was created to describe the functionality and behaviors of the agent. For the attacker agent, the state machine comprised of the steps and outcomes of seven different methods to perform a DDoS. On the defense side, the agents were trained to perform certain scans on different levels of the network model (system, network, and global) and report their findings whether or not malicious activity was found. Then in 2011, Kotenko [23] applied this methodology to a simulator using
NS2 and the OMNeT++ INET Framework. Where normal users, attackers, and defenders were modeled together and the effects of a botnet was examined on a network. It was shown that the collaborative team based attacks were much more successful at performing a comprehensive DDoS attack on a network, as one would predict.

Then in 2011, Grunewald et al. [15] released an Agent based adversary detection and simulation tool called NeSSi2. The methodology was to model the entire network and attackers as agents in a packet based simulation to avoid complex mathematical models. The provided a simulation that could handle many different types of scenarios that could not be easy to perform in a real world environment. Each machine, router, and firewall are created as agents in an attempt to detect and identify DoS attacks in the network. NeSSi2 supports two types of attacks, Worm-spread scenarios which is similar to an epidemic spread problem and botnet based attack like seen before. Each attacker is also modeled as an agent, picking one of these attacks and performing it on the network. NeSSi2 simulates basic web traffic like HTTP requests and other applications in the hopes that the attacker’s behaviors can be identified in the simulation.

### 2.2.3 Cyber Attack Ontologies

Cyber attack ontologies describes the relationships between each of the factors of a cyber attack and their dependencies on each other. For example, it can describe all of the details in a vulnerability for it to be possible, like services and OS’s, user permissions, down to the requirements of the attacker like specific domain knowledge. The ontology can be described in a low level like specific vulnerabilities or the attacker’s individual behaviors to very high level to just the attacker’s high level intentions. In 2002, Gorodetski et al. [14] applied the methodology to the cyber domain and described the requirements of the top level ontology of a cyber attacker in three steps. First, all attacks are intention centered meaning that there must be a reason for the attack. Second, attacks correspond to the response of the adversary domain. This means that each attack depends on the output from
attacking the target network. Lastly, an attack intention can be represented as a collection of sub-intentions. This is describing the multistage attacks where there may be multiple steps before the attacker can achieve their overall goal.

Shepard et al.[43] in 2005 used their knowledge-based artificial intelligence inference system called CycSecure to develop a knowledge base about an network to perform security analysis. By running a set of Cyc daemons on the host machines on a network the Cyc system can develop the base map of the network and then ontologies developed from security experts and natural language processing from various sets of data (not disclosed) can be used to analyze the network. Cyc provides support for over 12,000 services, 683 vulnerability types, 354 types of software faults at the time. However these ontologies still need to be reviewed by a security expert and the authors admit that each of the ontologies require 10-20 minutes of review each. In 2010 Conesa et al.[9] analyzed the usability of the more versatile and general version of CycSecure called ResearchCyc and found it to be unorganized, difficult to use, and inflexible with out expert knowledge makes it difficult to use for research purposes. There has been no response to this paper and no further progress on CycSecure as been made since.

The Structured Threat Information Expression (STIX) from MITRE is a schema co-developed with the Department of Homeland Security to better describe cyber threat information. The STIX schema captures high-level concepts such as observables/indicators, incidents, TTP (Tactics, Techniques, and Procedures), targets, courses of action, and threat actors. This framework also defines what is called the “Cyber Kill Chain” which describes the phases or steps an attacker commonly takes throughout the overall campaign. An example of the different stages of the cyber kill chain can be seen in Figure 2.1.

The different “sides” of the attacks are important because it depending on what side the attacker is on, different defensive measures need to be taken. The left side of the kill chain represents the preparation phase of the attack where preventative defensive techniques can be used to impede the progress of the attacker. Likewise if the attacker is on the right side
of the attack, the attacker has compromised the target in some way and the defensive strategy changes to a damage control focus instead of preventative techniques. STIX focuses on individual attack campaigns and organizes it to represent the attack and the kill chain position by the tools the threat actor used, techniques/behaviors they exhibit, the interaction between threat actors, and the observables.

2.3 Cyber Attack Simulators and Analysis Tools

Simulation is an appealing approach to cyber-threat analytics because it provides repeatable results can is relevant to the specific application of the user. Simulations are usually robust and can be tuned to provide different results under certain circumstances. However as with most simulations, the simulation is only as good as the model being simulated. In manufacturing processes where simulations are widely used for measuring process efficiency, specific parameters and variables are defined to describe the possible actions in the simulation like part failure rates for example. In the case of cyber attacks the complexity is much higher because the variability of the network, complexity of the vulnerabilities, and the attacker themselves has to be taken into account.

Mentioned previously NeSSi2 from Grunewal et al.[15] developed a cyber attack simulator in which a network was modeled and simulated the effects of distributed denial of service (DDoS) attacks and work propagation. NeSSi2 performs well in applications where there is propagation of some threat not only in cyber attack applications but also testing the security of smart grids[4] in one case. Where NeSSi is strong in automated attacks like
DDoS, it does not model various other types of cyber attacks that require compromising multiple machines as it is focused on modeling the defense for DDoS attacks mostly.

In 2014, Moskal et al. [31] developed the Multistage Attack Scenario Simulator (MASS) which is the predecessor to the Cyber Attack Scenario and Network Defense Simulator (CASCADES) used in this work. MASS introduced the idea of modeling cyber attacks by modeling the network and its services which determines the vulnerabilities on the network along with describing a cyber attack scenario with the use of the Scenario Guidance Template (SGT). MASS also factored in the attacker’s behavior by modeling the skills and preferences of the attacker. This produced reasonable results for a given attack scenario producing multistage attacks using real vulnerability data and modeling intrusion detection systems. Where it fell short was that it did not actually model the attackers personality nor did it model the actual actions that the attacker could perform. Instead the behaviors were abstracted down to some numeric parameters like stealthiness or persistence which where difficult to describe and configure. The simulator itself was unpredictable and extremely difficult to configure but it laid out the framework and base for the next generator simulator CASCADES.

2.4 Summary

Modeling cyber attacks and attackers is a difficult problem because of all the possibilities. Attack graphs are shown to be useful but provide scalability issues that is dependent on network description and flexibility issues because of the massive amount of possible attack scenarios available. Simulation is an interesting concept and still a growing field but is hard to prove its validity because of the lack of cyber attack data available. In simulation there is a trade off between realism of attack scenarios because the simulation can be custom tailored to a single attack type and being able to simulate a wide variety of scenarios with potentially less accuracy. This work presents a method to generate attack plans and scenarios and balancing out the trade off equally between model accuracy, flexibility, and ease of
use.
Chapter 3

Methodology

Current techniques of modeling cyber attacker behaviors either lack the robustness to provide realistic attack scenarios or fail to reflect the reactive and dynamic nature of cyber attackers, as noted previously. To achieve realistic cyber attacker behaviors it is necessary to understand what is a cyber attack at a high level. Then to address the dynamics of an attack it is important to understand the decision process an attacker makes throughout an attack. What makes one cyber attacker different to another? This can be answered by looking at what makes one person different from another person, completely leaving out the cyber aspect out of it. Identical twins for example are biologically the same/similar but it is their experiences, their skills, and what they know that makes them different to their identical counterpart. Describing a person’s full skill set or experiences is impractical as the complexity of the model will be unreasonable. The main goal of this work is to identify key components of a cyber attacker and assess the realism and flexibility of modeling cyber attack behaviors in this form.

3.1 Making of a Cyber Attacker

The purpose of this work is not to model every possible behavior a cyber attacker may have but to develop a framework to investigate information needed to sufficiently model a variety of attacks. The reasons why a person would target and attack a network can be related to the general motivations of any crime and not just the cyber domain. U.S. criminal
law defines three aspects of a crime which can be applied to any crime: means, motive, and opportunity. Means is the ability the adversary could commit the crime, motive is the reason the adversary committed the crime, and opportunity is if the adversary had the chance to commit the crime. This can be applied to cyber attacks as the means whether the attacker has the skills to perform the attack, the motive is the goals and intent the attacker has for performing an attack, and opportunity is the vulnerabilities on the network. Holsopple et al. [16] touched upon this by using the capabilities and opportunities of ongoing attacks to predict future attack scenarios. The definition of a cyber threat as given by Gasper from U.S. ARL [13] is the combination as capabilities, opportunity, and intent (COI) lining up with the law definition. These three aspects of a cyber threat are critical in describing the skills, goals, and attack surfaces the attacker has on a network.

The issue of describing cyber attackers with only COI is that it does not describe the personality of the attacker. For example two attackers may have similar skill sets, the same intent, and are exposed to the same attack surface. What is left out is how the attacker uses skills and opportunity. One attacker may choose to be extremely aggressive and attack as fast as possible, getting in and out regardless of the alarms that were set off. Where another may choose to be very methodical in choosing their next attack and staying “under the radar” as much as possible. These differences need to be taken into consideration as these have very different impacts on the victim network. Comparing and contrasting these two examples are interesting as they have the same COI but the approaches and outcomes are very different. The first attacker may have been detected and some threat mitigation could have been performed, but the second one may have not been detected and it is critical to understand how they can be detected. This work proposes that not only capabilities, opportunity, and intent are required to model a cyber attacker but the preferences of the attacker must be considered to allow for diverse models.

The attacker behavior is defined as the decision process to develop a sequence of attack actions that represents certain characteristics of an attacker. The attacker will use their
capabilities, opportunity, intent, and preferences (COI+P) to choose an attack action which is only part of their overall behavior. However the order in which the COI+P is used has a drastically different meaning to the overall decision process. We propose to order these as intent, opportunity, capabilities, and preferences. Intent is first because the attacker will have some reason/goal for the attack. This goal can be as high level as just affecting as many machines as possible or to extract some information from a particular server. Without intent being first, there is no meaning to why the attacker is attacking the network. Then opportunity to perform the intent is second due to the attacker knowing what they want to do and the opportunity is if they know enough to perform the attacker’s intent. The notion of the attacker knowing information about the target is a topic that will be addressed later. If the attacker does not have the opportunity for the intent, then the attacker must reevaluate their next steps. Next, is the capabilities of the attacker which determines if the attacker has the skills to perform the intent actions if they have the opportunity. It is unlikely that an attacker will address their skill set before their intent and opportunities. Then lastly, is what does the attacker prefer to perform. With their COI determined, there may be many different options available to the attacker and this can only be realized after the other three have been determined.

This is the logical organization of each of these models, however it is recognized that this is not necessarily the optimal order for all cases. As seen later in the design, the order will not play a large role in determining an action. It does change the literal meaning of the process. For example if the capabilities is taken in account first then it would seem that the attacker is developing their attack around their skill set which may be the case for some, but most attacks are developed around some goal. This COI+P framework describes at a high level of how an attack action chosen but still does not define a structure of how real attacks progress. COI+P provides no logical structure of the types of attacks an attacker performs in real situations. To provide this, the structure of an attack scenario must be examined.
3.2 Cyber Attack Scenario Frameworks

There has been large debate on what steps an attacker makes to compromise a network. The military defense contractor Lockheed Martin developed the Cyber Attack Kill Chain\cite{1} in 2013 which defined a set of steps an attacker would take to complete a specific goal. These steps include reconnaissance, exploitation, installation of malware, and extraction to name a few. This gives a solid baseline of the steps it takes to take control over a network but does not encompass the types of scenarios for all of the attacks. It certainly does not necessarily apply to all of the types of attacks or attackers, some may not choose to clean their traces where others will certainly cover their tracks. Since then other companies have developed their own definitions of a kill chain that applies to their particular field. As seen in Table 3.1 is different versions of kill chains defined by other companies.

Although these kill chains range from typical cyber attacks to specific types of cyber attacks, there are some patterns that are present in most of them. It can be seen that most of the defined kill chains share three states: reconnaissance, breach, and exfiltration. Recon is critical to an attack because it is where the attacker discovers information about the network and possible target(s). Breach is the exploitation phase, each attack consists of exploiting a machine whether it is a denial of service or control of the machine. Finally the last stage exfiltration is where the attacker will perform their goal like extracting data or taking down a machine. We define the kill chain stages recon, breach, and exfiltration as the Minimum Viable Kill Chain (MVKC). The MVKC is the minimum steps that the attacker must take to provide realistic results for modeling. It is to note that these steps do not need to be taken in order necessarily but is defined as the typical order of an attack. For example in a multistage attack a attacker may not be able to perform their goal after they have breached the first machine, more recon and breaching may need to be performed.

As shown, the MVKC is the minimum number of kill chain stages required to model a realistic cyber attack but more stages may be required to describe more technical attack types or more advanced attacker behaviors. To account for this, a flexible framework should
Table 3.1: Kill chain stages as defined by various groups and companies

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Recon</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Phishing</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Breach</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Raise Privileges</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install Malware</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Maintenance</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Remove Traces</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Exfiltration</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
be implemented to account for different stages and the differences between them. Each of these different stages all correspond to a different part of the attack or a different set of actions. This poses an interesting issue because since there is no actual order for these stages, when is it appropriate to “perform” one of these stages? When the attacker performs recon over a breach is up to the attacker but what information is the decision based off of? The answer is the same as any decision process in any field which is knowledge, what the attacker knows.

3.3 Modeling the Attacker’s Knowledge

Knowledge plays a large role in the overall evolution of a cyber attack. What an attacker knows about a network determines their next steps and how they will progress to achieve their goal. Defining what key elements of an attack or network is important to model as knowledge is necessary to then develop a model to describe how an attacker uses this knowledge. It is important to note that in this context, knowledge refers to the information known about a particular network and not the skills and experiences of the attacker themselves. This is an important distinction as the overall decision making process takes in account the knowledge gained from the network with the skills of the attacker. Modeling the knowledge an attacker develops throughout an attack provides interesting benefits as it provides realism in the sense that an attacker can only make decisions on information about the network that is known about. Very rarely will an attacker perform a breach or exfiltration type attack without some prior knowledge. Even in the case of a random script kiddie, there is still a discovery phase looking for targets in which the attacker is learning about the field and searching for targets. As these random attacks fail, the “attacker” develops more and more information about the target(s). This applies also to advanced meticulous attackers that will learn about their network, understanding what is out there and choosing their target carefully based on the types of services on the targets and skill sets they have.
3.3.1 Skills, Rules, and Knowledge

Before describing the details of what needs to be represented as attacker’s knowledge, first the description of a knowledge-based design should be defined. Using the knowledge as the driving force of the decision-making process is a double-edged sword. On one side, using knowledge provides the capability of realistic attack scenarios while being able to understand what the attacker needed to know to perform that attack. While the other side describing this sort of behavior may become complex from a user stand point, as one cannot expect a user to describe all of the rules possible for the attacker. Applying case-based approaches to the cyber domain like from Kolodneer [20] is not appropriate due to cyber attackers not always following a ridged set of rules, as most cyber attacks are situational. Where the skills, rules, and knowledge based design from Rasmussen [37] provides a much more flexible design for how a cyber attacker makes decisions.

Rasmussen [37] defines three different types of decision making: skills, rules, and knowledge. Skills are the impulsive decisions that are based off of experience like catching a ball, you don’t think about it. The rule level is the “if... then” type of decisions similar to the case based approaches. Where the knowledge level is where there is no direct solution to the problem and the person has to think of a solution. An example of this is a physics problem where the solution to the problem is not known however the person goes through problem solving techniques to extract the solution out of what is known. This model is a good fit in this case because it combines the deterministic problem solving in the case where the attacker has a clear direction for the next attack attack action and also the more probabilistic methods in the case that the next step is not well defined because the goal is not possible at a given time for example. This work uses this design as a base to not strictly rely on rules to define the attacker behaviors which will be addressed in the design and configuration of the behavior model.
3.3.2 Network Knowledge

Cyber attackers when infiltrating a network choose their actions based on some sort of information that they have learned previously or during the attack. This applies to all attackers from script kiddies to the most elite attackers. Script kiddies must have some sort of target even if it is a shotgun approach where the attacker randomly targets machines to discover if they are vulnerable, some knowledge is gained from this technique as it is used to plan their next action depending on the result. Similarly elite hackers may have a target in mind and will perform reconnaissance techniques to learn more information about their target to carefully plan their next action. These statements above are vague as the definition of information/knowledge gain in the context of a cyber attacker has not been defined. Defining the set of data that an attacker can learn and most importantly use to choose their actions plays a major role in the validity of a knowledge based model.

To define what is the critical information an attacker needs to perform an attack, it is necessary to evaluate two popular tools that are used by hackers and penetration testers. NMap[32], a network scanning tool, and Metasploit [28], a penetration testing tool, are two pieces of software that are widely used in the field of computing security. NMap is the standard program among hackers of all skill levels and understanding it is a critical step for hackers to acquire more information about their target. NMap provides the capability of discovering hosts, operating system fingerprinting, port scanning, and service scanning. NMap also provides scripts for more advanced service detection, user privileges, and exploitation, however we are more interested in the base functionality as this is information that every remote attack requires. Thus at a minimum, the attackers knowledge base must contain IP addresses, open ports, operating system, and services running on the targets. This is reinforced by evaluating the basic needs of Metasploit to perform an exploit. At the very least to successfully compromise a machine, Metasploit requires some target IP(s), the target port, and a service to exploit with the correct OS installed. Metasploit also requires knowledge of the vulnerabilities on the target whether it is from skill or another tool, the
attacker must know that the vulnerability exists on the network.

At a base, the knowledge needed to successfully perform an attack is IP addresses, open ports, services, and vulnerabilities. The only major part that is left out is the actual topology of the network itself. Both NMap (short for Network Map) and Metasploit is aware of the permissions of the machines on the network when used correctly. It is critical to the overall attack scenario that the attacker understands the network as a whole and who can communicate with who. This is because certain machines can only be accessed by certain other machines deeper down in the network due to firewalls and permissions. By modeling the attacker’s awareness of permissions, it will allow the capability for attackers to base decisions based of the source machine and target machines later on as a preference. This allows the fine tuning of the attacker’s behavior in the case there is a specific preference to use one source over another.

3.3.3 Opportunity: Using the Knowledge To Choose A Kill Chain Stage

One of the key design requirements of this knowledge-based behavior model is flexibility. Instead of describing few behaviors in great detail like done in previous works, the goal is to describe a wide variety of behaviors based off of real world parameters the reflect how attackers react when they learn new information. Due to the nature of the data that the simulated attacker is making decisions on (i.e., network sizes, services, service types, machines, etc.), probabilistic models like Game Theory for example may be difficult to define many behaviors easily. Where a purely rule based system, while easier to define, struggles to handle situations where there is no direct rule corresponding to the current situation. A good model will provide the ease of a rule based system with the uncertainty and diversity of a probabilistic model. Introducing Fuzzy Logic into this model is an appropriate choice because it handles the case where there is uncertainty in what the attacker should do next but still driven by a set of rules.

Cyber attackers themselves are not purely probabilistic nor are they purely rule based.
There is situations where the attacker will be unsure on their next steps because they may not have enough information about their target to attack. This is where the fuzzy logic plays a role since it takes into account that certain situations are not as simple as a coin flip or a “if... then” rule. In this case, a set of input parameters from the simulation will be the inputs to the fuzzy logic controller which then they will be used to drive the rules to choose a kill chain stage. The user then can use these input parameters to configure a set of thresholds or distributions that are used as the linguistic variables. The user then can also describe the set of rules using the linguistic variables which will drive the membership for each of the kill chain stages. This membership can be seen in Figure 3.1 where the variable “total failed actions” is given as input to the fuzzy logic controller and is evaluated to either acceptable or unacceptable based on value.

It should be noted that this model does not preserve any order of the kill chain stages explicitly. In the descriptions of the kill chains from other organizations, each of them had their own order in which the stages should be completed to correctly describe a complete kill chain. This automatic ordering of the kill chain stages is left out in this model because the criteria for entering a stage is purely situational and based off the opportunity to perform these stages. If the user chooses to enforce the kill chain stage order then it is required to add in a rule based off the completion of a previous stage explicitly instead of it being
automatically enforced. The use cases for defining flexible ordering of states could be for covering up traces after attacks or some other action that relies on a previous state.

### 3.3.4 Fuzzy Logic Parameters

The parameters that are used for the fuzzy logic as the inputs to determine the kill chain stage must be defined carefully as the selection of the kill chain stage is critical to the overall functionality of the ABM. These parameters must be flexible enough to control many different types of attackers, scaleable to any size network, and most importantly based off values real attackers make decisions based upon. Reflecting back on the MVKC of Recon, Breach, and Exfiltration, a base set of variables can be created that can differentiate the different stages the attacker can perform based off the knowledge. How these variables are explicitly used will not be covered in this section because the usage is up to the user to define the thresholds of the membership.

Addressing the requirements of the fuzzy parameters the first is flexibility. Each attacker has their own agenda on what is their reasons to perform one of the kill chain states. For example, one attacker may find that service scanning one machine out of 3 known is enough to switch to the breach stage. Where another attacker may want to explore all the possible actions before moving on. This also can be applied to when the attacker stops their attack. Some attackers may not want to continue much farther after failing a few attempts. Whereas an script kiddie for example may try every possibility before giving up. This allows for a wide variety of behaviors that can be controlled by manipulating just a few parameters. The general process of evaluating the kill chain membership can be seen in Figure 3.2

![Figure 3.2: The flow of the kill chain stage selection using fuzzy logic showing the inputs of the knowledge and intent.](image-url)
The next requirement in the parameters is that they must scale regardless of the network size. This is because the behavior should be invariant to the size of the network. An attacker from a small network should be able to be applied to a large network and the only thing that should change is the number of steps taken. By no means does this mean because an attacker is successful on a small network that they would be successful in an enterprise network. What this means is that the ABM should not have to be changed if the network has changed. For analysis purposes, one of the main goals of this work is to be able to test a wide variety of behaviors over the same or different networks. An example of invariance to scale is if we choose the variables for our rule set as the number of known machines and number of service scans. The issue with these variables is that it is only an ever increasing value as the attack progresses as the attacker is constantly learning more and more information. Whereas if the ratio of these two parameters are taken in consideration, the attacker can progress through the network learning as much information as they please before moving on regardless of the network size.

Lastly to address the realism aspect of the these parameters. This is and will always be a debated topic on what is and isn’t realistic. As mentioned before, the goal of this work is not to make every attack as realistic as possible but to simulate them to a reasonable level. Thus parameters must be defined in a manner where a real attacker will make these “state transitions” upon. This work proposes the idea of taking into account newly acquired knowledge. For example if an attacker finds out recently the services of an potential target machine, then the attacker may be inclined to target it to compromise. However this has an interesting complication because what is considered “new” knowledge? What is considered new for one attacker maybe very different than that of another attacker. This work proposes a range in which the attacker finds information new for a fixed time period. As the simulation progresses, information will become less “fresh” as more information is uncovered and this is defined as the knowledge decay. With these requirements defined, the base set of parameters are defined in Table 3.2.
Table 3.2: The initial list of the fuzzy logic parameters available to the user from the simulation

<table>
<thead>
<tr>
<th>Variables</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known Machines</td>
<td>Number of machines known by the attacker</td>
<td>0-*</td>
</tr>
<tr>
<td>Service Scanned</td>
<td>Number of machines scanned by the attacker</td>
<td>0-*</td>
</tr>
<tr>
<td>Scanned-Pinged Ratio</td>
<td>Service Scanned Machines / Known Machines</td>
<td>0-1</td>
</tr>
<tr>
<td>Newly Discovered</td>
<td>Number of machines 'recently' discovered</td>
<td>0-*</td>
</tr>
<tr>
<td>Newly Scanned</td>
<td>Number of machines 'recently' service scanned</td>
<td>0-*</td>
</tr>
<tr>
<td>Newly Compromised</td>
<td>Number of machines 'recently' compromised</td>
<td>0-*</td>
</tr>
<tr>
<td>Newly Scanned-Pinged Ratio</td>
<td>NewlyScannedMachines / NewlyDiscoveredMachines</td>
<td>0-1</td>
</tr>
<tr>
<td>Machines With Intent</td>
<td>Number of machines with intent services known on them</td>
<td>0-*</td>
</tr>
<tr>
<td>Success Rate</td>
<td>Over all types of attacks, successfulActions/allActions</td>
<td>0-1</td>
</tr>
<tr>
<td>Total Successful Actions</td>
<td>Over all types of attacks, the count of successful actions</td>
<td>0-*</td>
</tr>
<tr>
<td>Total Failed Actions</td>
<td>The total count of failed actions</td>
<td>0-*</td>
</tr>
<tr>
<td>No Attack Count</td>
<td>Count of where the attacker failed to choose an attack</td>
<td>0-*</td>
</tr>
</tbody>
</table>

Each of these parameters are a representation of the information contained within the knowledge base or the records about the attack itself. Known machines represents the count of machines that the attacker has performed a discovery reconnaissance type action upon and likewise service scanned is the count of those machines that a service discovery reconnaissance type action has been performed upon. It is assumed that a machine must be known before it can be service scanned, meaning that the service scanned machines will always be less than or equal to the number of machines known. The ratio of these two variables can be taken representing how thorough the attacker is to learning the maximum possible information about the network available. A ratio of zero with a non-zero value for known machines means the attacker has not service scanned any of the available machines whereas a ratio of one denotes that the attacker has explored and service scanned every possible machine and potentially knows all the possible information they can at that given point of time.

To represent information based off of the time at which the information was learned, a set of parameters called “newly discovered/scanned/compromised” is defined to describe the time at which the machine was accessed. The attacker can decide what is considered newly learned information based on the number of attack steps from the current step.
that machines were discovered, scanned or compromised. This can be used to control the thresholds on when the attacker will switch to recon to breach if machines were recently scanned for example. Like with known machines and scanned machines, the ratio of the newly discovered and scanned machines can be used to control how thorough an attacker is to scan these individual machines. To address the differences between what each attacker considers new knowledge, a decay function shown in (3.1) where $s$ is the current attack step, $range$ is the number of steps the attacker back the attacker considers, and $decay$ is the weight value for the knowledge.

$$y_{weighted} = \sum_{i=s-range}^{s} decay^{s-i} \times numActions_i$$

(3.1)

This decay function provides the capability to favor the knowledge that was most recently gained and use it to determine the kill chain stage. The variable $decay$ is used to modify how much to favor knowledge from step to step and $range$ is used as a limit to how far back in the attack scenario to apply this decay function for the knowledge. If $range$ is set to zero, then all of the knowledge will be considered. A separate decay function is defined for discovered, scanned, and compromised machines allowing the possibility of independently controlling each of the knowledge variables.

To aid the attacker in performing their intent, the machines with intent variable is defined for the fuzzy logic. This counts all of the machines that the attacker has knowledge on that the current intent can be performed upon. Then lastly, certain parameters that are not directly based off the knowledge are available to better control the running model itself. These are statistics like the attacker’s success/failure rate and the amount of times the attacker could not choose an action (no attack action). This is primarily designed to control the special stop kill chain stage which is used to terminate the current attack scenario. In the case where the attacker fails too many times or the scenario diverges, the attack scenario can be ended safely and predictably. This is the second instance where the model can be ended other than from the intent being completed.
3.3.5 Using Parameters To Choose Kill Chain Stage

This model relies heavily on the variability of the knowledge at any given point and the knowledge is condensed down into the set of parameters seen previously in Figure 3.2. These parameters are configured as a set of linguistic variables in the fuzzy logic controller driving a set of rules defining the behaviors of the attacker in terms of a kill chain stage. Fuzzy logic is well suited for applications where there is not a clear boolean decision process for a situation and instead evaluating the membership of a function based on partial truths. In this case fuzzy logic is used to address the uncertainties the attacker may have on the rules defined. This represents the differences between attackers who may have a clear direction with well defined rules for certain situations and attackers who may be unclear on their next steps.

An example fuzzy rule set is defined in Figure 3.3 where a single rule is defined for each of the kill chain stages in the MVKC. The detailed definition of the values and thresholds for the linguistic variables for each of the parameters (as shown in Figure 3.1) is not shown for simplicity of the example. Each of the linguistic variables has two possible states in this example: high or low. The membership in these two states is defined by the attacker’s behavior which is configured by the user of the model.

IF Scanned-Pinged Ratio is low OR Newly Compromised Machines is high THEN Kill Chain Stage is RECON

IF Newly Scanned-Pinged Ratio is high AND Newly Compromised Machines is low THEN Kill Chain Stage is BREACH

IF Machines With Intent is high THEN Kill Chain Stage is EXFILTRATION

Figure 3.3: Example fuzzy rule set for each of the kill chain stages in the MVKC

For each of these kill chain stages, the membership functions for each of the kill chain stages must also be defined. In this example each of the kill chain stages are defined as independent shown in Figure 3.4 where the x-axis represents the distribution for the stages and the y-axis is the possible membership for each stage. The maximum membership for the kill chain stages in the MVKC, recon, breach, and exfiltration, are at x={5,15,25}
respectively.

Figure 3.4: The distribution of membership for each of the kill chain stages in the MVKC

As the attacker develops their knowledge base, the membership for each of these rules will change as the values for the linguistic variables change. In the case where the attacker knows nothing, where the rule defined denotes that the stage is recon if the ratio of scanned machines to pinged machines is low, which is the case with no knowledge, the stage is recon. Each of the other rules will not be activated because no machines are scanned and no machines are identified to contain the intent. This is shown in Figure 3.5 at x=5, where recon is defined to be located, is at the maximum membership and all other states are zero.

Figure 3.5: The membership function of the kill chain stages when the attacker has no knowledge on the network, choosing recon
Once the attacker has performed some recon, the values for pinged and scanned machines will increase. Likewise because the attacker just performed these actions, this information gain will be denoted as new information. This increases the values for the newly scanned to pinged ratio parameter and the attacker has not compromised a machine recently keeps that value low. This will produce full membership to the breach stage as seen in Figure 3.6 at x=15 where again all other values are zero.

Figure 3.6: The membership function of the kill chain stages after the attacker has recent performed recon scans, choosing breach

Once the attacker has chosen and performed breach, assume the attacker then performs recon from the recently compromised target (which the rule for recon defines) and the attacker finds their intent. In this case because of the “or” relationship and the decay function of the newly compromised machine definition, there will still be some membership for recon. However the membership of the linguistic variable for machines with intent is set to give full membership anytime that there is a single machine with the intent. This will make the rule for exfiltration to overpower the rule for recon as seen in Figure 3.7.

The higher membership of exfiltration will lead to exfiltration to be chosen not because the number is purely higher but because the kill chain stages with a higher x value is given priority over stages at lower x values (right most maximum). This is exaggerated in Figure 3.8 where both the breach and exfiltration stages have full membership but the exfiltration stage is chosen over the breach due to the right most maximum priority.
Figure 3.7: The membership function of the kill chain stages after finding a machine with intent, choosing exfiltration

Figure 3.8: The membership function of the kill chain stages after finding a machine with intent, choosing exfiltration
Fuzzy logic provides the platform to model the uncertainty that comes with certain decision processes and is used here for the cases where many variables contribute to the selection of a kill chain stage. The selection of the kill chain stages can be configured to be well defined rules or fuzzy as shown above, adding to the flexibility of the model.

3.4 Modeling Attacker Skills and Preferences

The intent and opportunities parts of the ABM describe at a high level what the attacker is going to do. Intent is the very high level goals of the attacker and where the opportunity is with the intent what can be done. The initial intent starts with a large set of possible things the attacker can do and the opportunity filters down that set by what is possible based off of the knowledge of the attacker. The capabilities will further filter this down based off of the skill of the attacker. After the COI is determined the result is a set of possible actions based off the attacker intentions, knowledge, and skills. From this set the attacker must choose their next action, which is where the preference of the attacker comes into play. As mentioned previously, the attacker may have a similar COI but because of their personality or other stimuli they may prefer to choose other targets or actions over another.

3.4.1 Capabilities: The Attacker’s Skill Set

The attacker’s skill can be difficult to define due to it being difficult to universally describe what makes one attacker better than another. This is because what makes one attacker better than another is purely situational. Two attackers with the same intentions may have very different results in the same network because one may be better trained in web applications where the other is more focused on database attacks for example. Debatably the overall attacker skill based on the ability to adapt their skills using the intent with their knowledge to work out an alternative solution. In this work, skill is defined by the attacker’s knowledge of certain exploits to perform as the previous steps already factored in the intentions and
knowledge into the decision making process. Defining attacker’s capabilities in terms of
the possible actions the attacker can perform provides the benefits of: easy to define, clear
meaning, and more inline real life situations (an attacker can only perform an attack they
know about). At the expense of flexibility (the attacker can not perform something that they
“kind of” know) and knowing whether or not the attacker actually can perform the attack
(we assume that if it is define, the attacker can perform it successfully).

By making these assumptions and defining the attacker’s skill in this way, this is now
defined as the capabilities of the attacker as opposed to the attacker’s skill. Previous works
would define skill as a numeric number [31] based off of metric describing the difficulty
of the attack. This sort of definition was scrapped due to no reliable data on what makes a
vulnerability more difficult to another. A numeric value describing the skill level is difficult
to understand the real world difference between two skill level values. Whereas by simpli-
fying the model and assuming that because the attacker has knowledge of the vulnerability,
the real world description of the attacker’s capabilities is much more meaningful and can
better describe the actions that they attacker’s would actually perform. This model is a
static description of the actions that the attacker can perform and does not account for any
sort of research an attacker would perform if they learned something about the network or
target in the middle of the simulation.

In this description of the ABM, the capabilities are factored in last due to it directly
relying on the outcome of the intent and opportunities. Note the usage of the word “last”
even though the preferences have not been factored in. This is because the preferences do
not effect the actual values of the possible actions that can be performed, only weighs them
according to the attacker’s preference to certain values. Reasoning for the order will be
discussed in the next section but the result will be the same but with a slightly different
meaning. The result of favoring in the capabilities is now the attacker is able to choose the
next action based off each of the factors and is the minimal set of actions at this point.
3.4.2 Preferences: What the Attacker Chooses To Do Next

The final piece of the ABM, the attacker’s preference, is what each attacker from each other and gives the variability in the simulation. Multiple attackers may have the same COI but the actions that they choose and the targets that they choose can be very different between one another. For example students that go through the same schooling with the same intentions of getting a job out of school will may choose to apply to different types of jobs and companies even though they very will could apply to the same. This is the same here, the attackers may have the same intent and know the same sort of information but the final details of the attack are up to the individual. The preferences plays a key role in the ABM because as mentioned it is where most of the variability of the simulation comes into play. All of the possible actions that could happen based off the COI has been determined and the attacker could theoretically choose any of them at this point. With a few key elements being weighed in to the possible actions from the COI, it is possible to develop a set of actions with the probability of performing an action is scaled according to the attacker’s preference within these elements.

First the variables on what variables can the attacker have a preference on must be defined. By looking that the parameters needed to perform an attack it is possible to develop a list of few variables in which an attacker can have a preference on. The most obvious parameter that the attacker has a preference on is the target machine. An attacker may choose one target over another because of particular services or vulnerabilities for example. The progression of the attacker and realism of the simulation relies on the preferences being defined well. Thus the target preferences also needs to include factors like targeting a previously compromised target or targeting a machine where the attacker failed. This allows the capability to tune the chances of an attacker targeting a machine they have already targeted in different circumstances. Attackers may have their reasons to target machine they previously targeted and this allows these machines to be favored or hindered. By taking these into account, the user of the simulation can tune the ABM to allow certain types of
behaviors in target decision if they choose.

With choosing the target the attacker must choose a machine to perform the attack from, this is generally referred as the source of the attack. Source preferences is not nearly as critical as the target preferences, it does however add into the realism of the simulation. If the source selection was completely unrestricted, the behavior that is seen is an attacker that is jumping around using different sources which is a legitimate behavior but not desired for every type of attack. Some attackers will compromise a target and then use that target as a source, trying to penetrate the network as deep as possible. Thus there is a preference parameter to prefer that newly compromised machine as the attacker’s next source. Likewise, if an attacker finds out that the source machine is a dead end (nothing connected to it) then there is a parameter to prefer that machine as a possible source less then others if desired.

The last piece of the preferences is the attacker’s preference on which actions they choose. The preference on actions can be based on the types of services they prefer, types of reconnaissance they want to perform, or even actions that are less likely to be detected. This is where the attacker’s “signature” can be modeled by the preference on certain actions. Every attacker will have a different set of preferred actions which makes each attacker unique from one another which could attribute how an attacker progresses through the network. The action preference also includes favoring or hindering the chance of repeating the same actions if performed previously or retrying the action after it previously failing. These factors are included in this model because the attacker must respond to situations that happened in previous cases like a failed attack or similar situation than before. Each of these provides ample flexibility to describe attacker’s signatures and patterns based off of their individual preferences and personality. The base set of preferences can be seen in Table 3.3.
Table 3.3: The base set of attacker preferences that can be modified by the user

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Preferences</td>
<td>Scales source probability on recently compromised targets</td>
</tr>
<tr>
<td>Persistence</td>
<td>How persistent an attacker is after failing an attack (weighs target and action)</td>
</tr>
<tr>
<td>Compromised Target</td>
<td>Likelihood to target a machine that has already been compromised.</td>
</tr>
<tr>
<td>Dead End</td>
<td>Likelihood to use a dead end machine as a source.</td>
</tr>
<tr>
<td>Preferred Actions</td>
<td>Scale certain actions by a preference weight</td>
</tr>
</tbody>
</table>

3.5 Combining Capabilities, Opportunity, Intent, and Preference Models

Each of these models in the ABM brings a critical aspect of the individual attacker. With this proposed structure of description of a cyber attacker by defining the attackers COI+P, the US criminal law definition of an adversary is satisfied with the COI along with describing the attacker’s personality with the +P. This gives a basis of what needs to be defined to model a range of different cyber attackers. When combining these models to eventually choose a single attack action the order in which these models are taken in consideration is required to be evaluated. This means that the order in which the attacker’s capabilities, opportunity, intent, and preferences are factored into the model has different logical meaning although should provide the same results in the end. In this situation, the preference of the attacker will be considered last all of the time because only the COI actually modifies the action sets.

Out of the six possibilities of the order of COI, the first of the models to be favored in determines the overall meaning. For example if the capabilities of the attacker are addressed first, this means that the attacker basing their attack based strictly their skills. The attacker will filter out their possible actions based on if they think they can attack it in the first place. This makes sense as an attacker will only choose their targets based on their skills they have but not particularly at the beginning before they figure out if the network can perform it. Where as if the opportunity is weighed in first in this model, the meaning is unclear because the attacker does not have a clear goal to have an opportunity on. The opportunity model
after all is the opportunity for the intent thus it is hard to place opportunity first since it depends on the intent. Placing the intent first logically makes the most sense because an attacker will always have some reason to begin an attack and then they can decide if they have the opportunity to perform that intent and whether or not they have the skills to perform that intent. This work chooses the order of intent, opportunities, capabilities, and preferences due to logically the most clear however as shown later the design is not effected by the order and will have the same results.
Chapter 4

Design Implementation

The purpose of this work is to model the attacker’s behavior and thought processes in a manner that better reflects real world attacks. Previous works to describe attacker’s behaviors uses purely probability based models or attack graph modeling that do not accurately portray the styles and actions an attacker takes to penetrate and infiltrate a real network. Instead of probabilistic models this design takes a information and knowledge based approach to deterministically model the attackers behaviors. Basing the attack action decisions based on what the attacker knows about the network and its vulnerabilities along with tune-able simulation parameters allows for a realistic simulation of various cyber attackers in a network.

This version of the ABM takes a modular approach in the way it determines the next action. In MASS [31], the ABM while it was a separate module from the rest of the simulator, much of the logic to determine the next action was burdened onto the core logic of the simulator making it very difficult to make changes. This design unloads the processing off of the core logic of the simulator where the core logic will make a single call to the ABM and it will return a single action as seen in Fig. 4.1. This is to allow many different types of behavior models to be implemented later on, where the ABM can be seen as a black box where the core logic does not depend on how the result of the ABM was achieved but the contents of the result. In this design, the ABM will consist of many different internal modules which can be viewed as a state machine that will incrementally develop the solution in
various steps.

Figure 4.1: ABM basic black box design where the core logic only makes a single call to the ABM.

First to model the attacker behaviors, the thought process of an attacker must be broken down into various steps or states to determine the next action the attacker is going to take. To recap, four states are proposed to represent the attacker choosing an attack are: intent, opportunities, capabilities, and preferences. Intent is what the attacker intends to do at any given time. This can be seen as the attacker’s goal or in the case of the simulation intent is the attack scenario that the user wants to simulate. The second state is opportunities which is the attack set the attacker has based on the network knowledge and the intent they have for the network. This based on what they know about the network and what they want to accomplish is the total opportunity the attacker has on their network regardless of their skill sets. However while the attacker may have an opportunity to attack a network, they may or may not be capable of take advantage of those opportunities which is the third state, capabilities. The attacker’s capabilities is seen as the attacker’s skill set or tool chest. Each attacker is different and the type of exploits and vulnerabilities that they are capable of using varies. Each of these three states are filters that will reduce the possible attack set down to a set that represents the attacks the attacker is relevant to the goal of the attacker, possible on the network that they are aware of, and the attacker is capable of. Then the attacker must choose from this set which is the forth and final state, the attacker’s preference. The attacker will choose the attack based on the types of attacks they prefer, services they know, or other parameters to choose the attack.
The goal of the ABM is to simply choose an attack action based off the intent, virtual terrain, and knowledge of the attacker. Initially a set of possible actions governed by the set up of the virtual terrain is compiled as the base set of actions. This is the set of all the possible actions that could be performed given full knowledge of the network and before any of the attacker’s knowledge is taken in account. As each step of the COI+P gets taken in account, this set of actions will decrease in size to a point where all of the attacker’s behaviors have been taken in account and an action can be chosen. This process can be seen in Figure 4.2 depicting both the COI+P process flow and the decreasing action set throughout the process.

These four states are the basic building blocks of the attacker’s thought process. However each of these stages focused on making decisions based on the knowledge the attacker has on a network. Modeling the knowledge of the attacker is the fundamental basis for this work and is what all the decisions regarding the attacker behavior is based off of. This design focuses on the evolution of the attacker’s knowledge throughout the attack and how each attacker chooses to use this knowledge to their advantage. By accurately modeling the key aspects of what the attacker bases their decisions off of when attacking a network, the types of behaviors that can be simulated can be simple or extremely complex.

### 4.1 General Process for Determining Attack Actions

Expanding on Figure 4.2, the actual process flow of the ABM can be created as seen in Figure 4.3. The most significant contribution is the depiction of the instantiation of the
attack action sets and the entry and exit points of the ABM. Each of the stages of the COI+P is represented as a subprocess in this diagram. Each of the subprocesses will be individually addressed in a separate section in this chapter. Each of the subprocesses will

It should be noted that by the time determine attack action is called the intent is already realized and it is up to the ABM to determine the opportunity to achieve the intent. The process flow at this stage of the ABM is straight forward, linear, and has few branches. This is in part because of the design choices to break every piece of the design in their own separate subprocesses to keep the complexity of the top level ABM down. This design only has two ending conditions, which one returns a chosen attack action by the attacker if all the steps of the COI+P was taken and the other is in the case where the simulation has been reported as completed by the attacker. This allows for a predictable flow of the program which makes it easier to add new features without changing the rest of the code and simpler to understand the overall concept.

4.2 Attacker Knowledge Modeling

The knowledge the attacker has plays a critical role in all the decision processes of this work and is the main contribution. This section will focus on the design of the attacker’s knowledge and how it is developed. The knowledge is modeled a data structure containing information about the network the attacker learns and in this design what is called the “Attacker Knowledge Base” (AKB) which contains key information about the network and is an evolving data structure as the attacker progresses throughout their attack. The key design strategy with the AKB is to make it as expandable as possible as this design provides a base for the knowledge as the ABM in the future advances the AKB can be easily expanded. By implementing a single update function allows the possibility to expand on different parts of the knowledge base like a plug-in, making it easy to implement new features. The update methodology will be addressed in more detail later in this section.

The attacker knowledge is broken up into three parts as seen in Fig. 4.4. The overall
Figure 4.3: The general process for the determine attack action method for the attacker behavior model called by the simulation core logic.
encompassing data structure is called Knowledge Base which contains many machine specific data structures called Machine Knowledge. Lastly is an enumerated type containing the different fields that are contained in Machine Knowledge that can be updated. Each of these different classes will be elaborated on further on later on in this section.

**4.2.1 Knowledge Base**

The Knowledge Base is a container of other network specific data structures that make up the entire Attacker Knowledge. There is one Knowledge Base object for each behavior model in which each of the other modules inside of the ABM have access to. The Knowledge Base contains variables keeping track of the last source and target pair for quick reference to keep track of where the attacker is in the network. This structure also contains look-up tables for the potential sources and known machines to be used to quickly reference IP Addresses of the machines that the attacker is aware of or have compromised. These look-up tables store the hash of the IP addresses in a TreeSet for quick sorting and searching though the list. Then the core of the knowledge base is the HashMap containing the mapping between IP address hashes and their Machine Knowledge object. This is the main data structure that represents the machine level knowledge the attacker has on the machine.

![Figure 4.4: UML diagram describing the attacker’s knowledge design.](image-url)
The knowledge base contains no methods that do any major calculations or processing and only has methods that interface with the underlying data structures. To avoid complicated data accessing all updating of the data structures are done through the update function and all machine knowledge accesses are done though the knowledge base. The machine knowledge updating process is encapsulated inside of the knowledge base to allow for a expandable update function.

4.2.2 Machine Knowledge

The machine knowledge object is the basis of each decision that is made in the ABM. The individual attributes that makes up the knowledge about specific machines provides a foundation for the ABM to make decisions off of and is imperative to the accuracy and flexibility of the model. The machine knowledge is represented as facts that the attacker has observed one way or another, through previous knowledge or knowledge gained from previous actions. While the information that they observed may or may not be correct due to defenses or changes in configuration, this does not change the attacker’s decisions because the attacker has not observed this change at the time.

The machine knowledge first starts off with the obvious parameter which is the observed IP Address of the machine. The wording here using “observed” is chosen very specifically to reflect that the IP address the attacker may observe could be different at any given time. This is because the network may have a moving target defense (MTD) system implemented in the network where the IP addresses, ports, and firewall permissions may change. The attacker may not know that one of these systems are in place on the network and is only what they know at the time the information was discovered. It was mentioned before that the information in the knowledge base, although facts, are only facts at the time the observation was made and it is not guaranteed that it is correct. This is critical since all information is based on what the attacker uncovered, attacks may become unsuccessful over time as their knowledge becomes outdated. This methodology also applies to the observed state of the
machine. As the network progresses in time, access vectors may no longer be available due to some defense strategy implemented thus the observed state of the machine is no longer consistent to the actual state of that machine on the network.

To keep track of where each machine can be accessed from, a list of possible sources is maintained. When choosing an attack, the attacker will need to choose which machine to attack but also what machine to attack from. Sources can either be just the internet, in which it will be an external IP address, or a previously compromised machine. This pairing is known as a source-target pair, in which will be passed to the core by the ABM as a part of the overall attack action result. Likewise, a list containing the known open ports on the machine is maintained. With each source-target pair, there is an associated port that is being used to access the machine. As with the other parameters, these are observed at the time the attacker has discovered the information and is subject to change at anytime. As a source may or may not be accessible after a certain time or a port has opened or closed without the attacker knowing.

The modeling of vulnerabilities in this design is based on MITRE’s Common Vulnerabilities and Exposures (CVE) which a single CVE represents a vulnerability on a particular service. The network is described as a collection of machines with services and each of these services has their own set of CVE’s. Knowing the type of services and how they are accessed, including the operating system, potentially gives the attacker options on the types of vulnerabilities that can be used on the machine. The machine knowledge object keeps track of the list of services known on a machine along with a mapping to service to open ports. The list of just services is used as a look up table to quickly search, sort, and union/intersect with other lists. The mapping structure is for after the source/target pair is chosen and the vulnerability to be used to then pick the port used for that vulnerability. Lastly, the vulnerabilities that was used on the machine is also kept track of. This is to represent persistence threats and a record of attacks that were previously used against that target. Generally if the target was vulnerable to an attack once, it can be used against them
again unless the target is updated. All of the available variables that represent the attacker’s knowledge for one machine can be seen in Table 4.1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observed IP</td>
<td>The observed IP of the machine</td>
<td>int</td>
</tr>
<tr>
<td>Observed State</td>
<td>State the attacker observes the machine as</td>
<td>NodeState</td>
</tr>
<tr>
<td>Accessible By</td>
<td>Sources that can access this machine</td>
<td>Set&lt;Int&gt;</td>
</tr>
<tr>
<td>Open Ports</td>
<td>The open ports on this machine</td>
<td>List&lt;Long&gt;</td>
</tr>
<tr>
<td>Known Services</td>
<td>The services known by the attacker</td>
<td>Set&lt;Int&gt;</td>
</tr>
<tr>
<td>Service To Port</td>
<td>Mapping between service and open port</td>
<td>Map&lt;Int,Set&lt;Long&gt;&gt;</td>
</tr>
<tr>
<td>Known Vulnerabilities</td>
<td>Known vulnerabilities on each service.</td>
<td>Map&lt;Int,Set&lt;Int&gt;&gt;</td>
</tr>
<tr>
<td>isDeadEnd</td>
<td>If this machine can be used as a source</td>
<td>boolean</td>
</tr>
<tr>
<td>Attempted Exploits</td>
<td>The attempted exploits by the attacker</td>
<td>Set&lt;Int&gt;</td>
</tr>
</tbody>
</table>

### 4.2.3 Attack Records

Along with the network knowledge, the attacker also can use information about their previous attacks to make decisions. This is known as the attack records in the ABM. These are general statistics about the different attacks that the attacker has performed in a much more general sense. The network knowledge was more about information that the attacker learns about the network where the records is stats about the attacks themselves, like success rate of the attacker or the result of the last action for example. Statistics like the total failed compromising actions, or total successful compromising action can be useful for controlling when the attacker completes their attack scenario or to tell the simulation to end. The records is also useful for general statistic tracking as results because the records also keeps track of when certain machines were discovered, scanned, and compromised. The list of variables being tracked in the records can be seen in Table 4.2.
Table 4.2: The list of variables being tracked by the simulator as statistics of the attacker’s actions.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Step</td>
<td>Current step of the attacker</td>
<td>int</td>
</tr>
<tr>
<td>Last Target</td>
<td>Last target of the attacker</td>
<td>int</td>
</tr>
<tr>
<td>Last Action</td>
<td>Last action used by the attacker</td>
<td>int</td>
</tr>
<tr>
<td>Last Result</td>
<td>Result type of the last action</td>
<td>ActionType</td>
</tr>
<tr>
<td>Total Actions Performed</td>
<td>Number of actions performed</td>
<td>int</td>
</tr>
<tr>
<td>Successful Actions</td>
<td>Total successful actions</td>
<td>int</td>
</tr>
<tr>
<td>Recon Actions</td>
<td>Number of recon actions performed</td>
<td>int</td>
</tr>
<tr>
<td>Successful Recon Actions</td>
<td>Total successful recon type actions</td>
<td>int</td>
</tr>
<tr>
<td>Compromising Actions</td>
<td>Number of compromising actions performed</td>
<td>int</td>
</tr>
<tr>
<td>Successful Comp. Act.</td>
<td>Total successful compromising actions</td>
<td>int</td>
</tr>
<tr>
<td>Step When Discovered</td>
<td>Step at which each machine was discovered</td>
<td>Map&lt;int,List&lt;int&gt;&gt;</td>
</tr>
<tr>
<td>Step When Scanned</td>
<td>Step at which each machine was scanned</td>
<td>Map&lt;int,List&lt;int&gt;&gt;</td>
</tr>
<tr>
<td>Step When Compromised</td>
<td>Step at which each machine was compromised</td>
<td>Map&lt;int,List&lt;int&gt;&gt;</td>
</tr>
<tr>
<td>No Attack Count</td>
<td>Count of NO_ATTACK attack types</td>
<td>int</td>
</tr>
</tbody>
</table>

4.2.4 Updating the Knowledge Base

With the stream of information incoming from the attack, an efficient and scaleable knowledge base update function is needed. By centralizing the update function within the knowledge base, this minimizes the work needed to add in new fields if and when the ABM Knowledge module is changed. However centralization is not enough to enforce easy scalability. By using an enumerated type that labels the update data, the update function can determine how to manipulate the data to update the knowledge and enforce strict type casting. The knowledge fields enumerated type can be seen in Fig. 4.5 showing the base set of modify-able fields

Figure 4.5: UML diagram describing the Knowledge Fields enumerated type.
In the update function within the knowledge base, the input to the function is a HashMap where the key is a KnowledgeFields enum and a value of object. As mentioned before the purpose of the knowledge field enum is to dictate what to cast the value object into. This is to allow the capability to perform bulk updates to the knowledge base since the hash map can take many different key value pairs. This update function also will report whether or not the update changed any values within the knowledge base. This is to inform the ABM that no new knowledge was uncovered in the last action. This could be useful in a case that a scan was performed on the same subnet from two different sources yielding no information gain.

### 4.3 Initialization of the Attack Action Sets

The flowchart of the ABM shown in Figure 4.3, the first subprocess that is addressed is the generation of the attack action sets. The attack action set is a data structure that contains not only the actions that can be performed by the attacker but also the possible source and targets for those actions. This creates a mapping between sources and targets and also a mapping between target and action. This data structure forms the bases for each of the COI+P functions in the ABM as each of these will take in the action set and filter it down by their criteria. The initialization of the attack action set requires the use of the virtual terrain and is the only time an ABM function interacts with another context model directly. This is due to the attack action set is initialized to the perfect knowledge of the network first and then the COI+P models will modify it and filter it.

The initialize function requires some sort of awareness of the knowledge of the attacker purely for efficiency purposes. For example, if the action sets indeed did start with full knowledge of the network in a large network, the action set would be extremely large because it would have to have effectively copy the virtual terrain each instance of determine attack action. To combat this issue, the action sets are initialized to only include the machines that the attacker is aware of. If the attacker is not aware of any machines then the
action set will only include the reconnaissance actions since they are the only actions that are possible to perform at this stage. A known limitation of this model is that it does not allow the capability of trying attacks on IP’s that do not exist, as some may try on random subnets. Once the possible targets are initialized, the sources can be any machine the current “targets” have access to which is accessible from the virtual terrain. Then lastly the actions for each of the possible targets are initialized to all the possible vulnerabilities on that potential target given the services on the virtual terrain. This then can be passed though the rest of the ABM to eventually pick a source-target-action grouping to make up a single action.

4.4 Intent and Opportunities

There are two different meanings for intent in this context. First is the intent of the attacker which is the goal of the attacker at that stage of the attack and the second type of intent is the goal of user of this model. This framework is designed for simulation and in the end is going to be used to produce simulated attack scenarios, thus the user of the simulator must be kept in mind. In MASS, the previous notion of intent was given by the Scenario Guidance Template (SGT) where the simulation was controlled by a state machine depending on the actions performed. This was further expanded on with the Attack Guidance Template (AGT) which supplied a set of attack actions to the ABM determined by simulation goals to influence the attacker’s actions.

This notion of guidance is preserved in this work. Where the intent is what the goal of the simulation is and the opportunities is how the attacker, based on their knowledge, can perform that intent. However this is slightly different that what is mentioned before where the AGT and ABM are two separate entities. The AGT represented strictly the goal of the simulation and the ABM in MASS was a separate entity uninfluenced by the AGT requirements. Whereas in this work it is more seen as the intent is the goal of the simulation and of the attacker, it is how the individual attackers reach the opportunity to perform the
The intent is given as input by another function (in this case the intent module) and the ABM uses that information to filter down the attack set by their knowledge. How the attack set is filtered down is based on not only the intent but the knowledge, the Cyber Attack Kill Chain methodology is used here as the basis of attack stages.

4.4.1 Cyber Attack Kill Chain

The cyber attack kill chain is used by many different organizations to describe the various stages and order that an attacker takes to infiltrate and attack a network. While some have 5 stages and others have 10+, there is a common set of stages regardless of the type of attack (Denial of Service, Phishing, Mobile, etc.). These stages were identified as the MVKC (Minimum Viable Kill Chain): Reconnaissance, Breach, and Exfiltration. The recon stage is the data collection stage, learning about the network and building the attacker’s knowledge base. The breach stage is when the attacker does not have their objective immediately “in-sight” and attacks a machine to progress into the network. Lastly exfiltration is performing some attack as part of the intent to either take down a machine, steal information, etc. These three stages are identified as the three minimum stages to perform an attack however in this design a forth special stage is implemented known as the Stop stage. This is to signify to the simulator that the attacker has completed their attack because they achieved their goal or they are giving up for some reason. The details of the stages are given in a later section.

Following the same style as the rest of the design, although three stages are identified as a minimum here, the stages are modular and extra stages can be added. As seen in Figure 4.6, the individual stages inherit from the abstract parent class KillChainStage and must implement the filterByIntent() method. Each of the kill chain stages are following a singleton pattern, making only one of each stages to be created at a time. This is due to when the kill chain stage is chosen, the method returns a kill chain stage object and only
the filter method is used and does not store information.

![UML diagram showing the three different stages]

Figure 4.6: UML diagram showing the three different stages

The ABM does not need to know specifically which stage it has chosen, it only needs to call the filter function to reduce down the attack set. To add in another stage only the filter method needs to implemented and the method that chooses the kill chain stage needs to be changed. Before choosing an stage, a brief understanding of differences between each stage is needed.

**Reconnaissance KC Stage**

The recon stage focuses mostly on gaining knowledge using the recon techniques described above, social engineering, or phishing (if implemented). The recon stage filter set does not contain any attacks that would be considered malicious and purely knowledge gain based. The purpose of the recon stage is for the attacker to learn more about the network and how the knowledge gained can be used against the network. If the attacker does not have enough information about the network to perform their intent, then the attacker must perform reconnaissance on the network. Not enough information could be not knowing IP addresses, the machine’s services, or the attacker is searching for key information about the network before they choose to attack the network. If this stage is chosen, the filter will remove any attack actions that are not of recon type from the resultant attack set.
**Breach KC Stage**

The breach stage is defined as the set of attacks the attacker may perform that does have malicious intent, but does not pertain to the specific goal of the attacker/simulation. In this case, the attacker has determined that there is enough knowledge about the network that they can move to the next stage. The overall goal of the breach stage is to compromise a machine with the intent of using it as a possible source to eventually reveal the knowledge to achieve their final goal. This is a critical stage for multi-stage attacks as most high-profile targets will be more than one layer deep within the protection of the network. This is also useful for advanced attacks that require prior authentication to the target machine before using another vulnerability to complete their goal later in the exfiltration stage. The filter in this stage is less aggressive and will filter out any attack actions that are not of recon type and not pertaining to the final goal of the attacker.

**Exfiltration KC Stage**

The exfiltration stage is extremely similar to the breach stage however instead of choosing any attack, only the attack actions pertaining to the goal of the attack is chosen. The reason for the difference between the breach and exfiltration stage is to specifically favor the actions that directly pertain to the goal. The separate intent module is monitoring whether or not this state is selected and the outcome of the selected action. If the action is successful, the intent module will update and return the next intent if it is available. In the cases where the intent module does not return a new intent, the attacker has then completed their intent successfully and the simulation is completed for that attacker.

**Stop KC Stage**

This kill chain stage is a special stage used primarily to signify the point at which the attacker should stop the attack track. If at any point in the simulation the stop state is chosen by the kill chain selection logic, the logic immediately will branch away from the
normal logic (capabilities and preferences) and trigger the core logic to end the attack track. The stop stage can be triggered by both simulation parameters and if the intent module has signified that the attack scenario is completed. This allows the both the simulator to control the when the attacker has accomplished the goal of the attack scenario and allows the behaviors of the attackers to determine the end of the attack based on what is learned through out the simulations. The filter in this stage returns nothing as it is not being used for any calculations.

4.4.2 Choosing Kill Chain Stage Using Knowledge: Implementation

Choosing the kill chain stage is a primary function of this work, as it represents the types of attacks based on the intent that the attacker has and the knowledge gained throughout the simulation. How these stages are chosen is critical to accurate simulations of attacker behaviors. Also to create many different types of behaviors, the design and configuration must be flexible to not restrict the types of behaviors that can be modeled. To recap, the kill chain stage is chosen using fuzzy logic where a user can set thresholds and rules on a given set of variables from the simulator to describe various different types of attackers. The fuzzy logic in this ABM uses the JFuzzyLogic [6] which is the standard package for implementing fuzzy logic control in Java. This takes in a “FCL” file which is a configuration file that contains all of the variable definitions, linguistic variables, state distributions, and rule functions. An example of the a FCL file for the ABM can be seen in Figure 4.7.

Using JFuzzyLogic simplifies the kill chain selection process significantly because the simulation variables described in the previous chapter simply needs to be inputted in to the fuzzy controller and call the evaluate function. Calling the evaluate function takes all the user input for the attacker’s behavior and chooses a state. Some key design choices and assumptions do need to be taken into account however to ensure that there is predictable results from the fuzzy logic controller. The first assumption is that every kill chain stage is independent from one another, meaning that kill chain stages do not have any actions in
common. This is the case for the MVKC as each of the states filter out each other when chosen, however if the stages have actions in common then it must be reflected in the fuzzy logic. An example on the fuzzy logic with independent states are shown in Figure 4.8.

Note the minimal overlap between the four distributions that represent the states, this is to say that there is nothing in common between these states. If there was a case where there were many possible actions that were shared between two stages, then the distributions for those stages should have overlap reflecting the amount of actions shared between them. This ambiguity allows for the behaviors and the rules to drive the decision between the two states which could provide interesting results when comparing attackers in the same situation. The next assumption is that the states shown in Figure 4.8 are ordered from left to right in least to most critical states. While the ABM does not directly preserve order in the kill chain, to aid in the realism some priority is given to more important states. This is known the JFuzzyLogic as “Right Most Maximum” where the state with the highest
membership to the farthest right on the state list will have priority. The example usage here is that the minute that the stop stage overpowers the rest of the stages, it should absolutely stop the attack because clearly something has happened that requires the attacker to stop. This is just an example as the priority of the stop stage can be lowered to favor other stages until it does not make nothing else can be done.

Reflecting upon the parameters described for the kill chain logic in the previous chapter, the definition of most of the variables are self explanatory and straight forward. Mentioned before, there is a notion “new knowledge” and what is described as new knowledge. This functionality of new knowledge gives the user to prioritize newly obtained knowledge to determine the next kill chain stage. First intuition is to describe a knowledge range which is how far back (in attack steps/actions) does the attacker consider the information as new. This could pose an issue because saying the “information from now and three steps ago is new” is very rigid because it gives very little flexibility in the fuzzy logic control in the thresholding because of the discrete values. For example if an attacker found 15 machines on step 1 and compromised one machine on step 2 with the knowledge range being three steps, even if the intentions of the user configuration was to compromise one machine out of the 15 and then use that as the next source the chances of that happening is very low because the 15 machines will overpower the next rule. This is an extreme case for one
situation that may actually be desired but the issue is that newer information may want to
be favored over older information.

The proposed solution to this issue is to weigh older information less than newer infor-
mation if desired. This functionality is defined as the knowledge decay function. Contained
in the attack records is the collection of “Step When” functions which returns the count of
actions given a range and the current step. For the example above, this function would re-
turn 15 for newly scanned machines and 1 for newly compromised for a knowledge range
of 2. The decay function will take some decay variable (0-1) set by the user and raise it to
the current step minus the step at which the action took place multiplied by the number of
actions as seen in (4.1).

\[ y_{\text{weighted}} = \text{decay}^{s_{\text{current}}-s_i} \ast \text{actionCount} \] (4.1)

4.5 Attacker Capabilities

The capabilities stage of the ABM describes the skill set of the attacker. While the intent
and opportunities stage described at a high level what the attacker would like to accom-
plish and the opportunities they have knowing what they know to accomplish that intent,
the capabilities is the skills the attacker has to take advantage of the opportunities. For
example, at any given moment a machine has vulnerabilities and there is an opportunity to
compromise the machine with the intent of stealing information. If this person attacking
the machine is a skilled and experienced hacker, they may be capable to successfully use
this opportunity with their skill sets to compromise the target. On the other hand if the at-
tacker is a script kiddie and automates their attacks, they may not be capable of performing
the specific vulnerability needed to take advantage of their opportunity.

The attacker capabilities describes the difference between the skill sets of the attacker
because after all if an attacker does not have the skill to perform an attack, it does not matter
whether they know about the vulnerability or not. Likewise an attacker that is not aware
of a vulnerability’s existence generally will not be capable of performing that attack either, unless they learn about the vulnerability during the attack in which is not modeled. The capabilities list is a static list describing either a whitelist of the skills they do have or a blacklist of the specific skills they don’t have. The whitelist is more difficult to define but more accurate to describing the skills of the attacker. Whereas the blacklist makes it easier to leave the skill set of the attacker more open ended for greater variety in attack sets. Fig. 4.9 shows the basic structure of the attacker capabilities and the functions with it.

<table>
<thead>
<tr>
<th>AttackCapabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>- capabilitiesSet : TreeSet&lt;Integer&gt;</td>
</tr>
<tr>
<td>- capabilitiesQuery : ArrayList&lt;Query&gt;</td>
</tr>
<tr>
<td>- isWhiteList : boolean</td>
</tr>
<tr>
<td>+ filterByCapabilities( TreeSet&lt;Integer&gt; ) : TreeSet&lt;Integer&gt;</td>
</tr>
<tr>
<td>+ generateCapabilitiesSet() : void</td>
</tr>
</tbody>
</table>

Figure 4.9: UML diagram describing the attacker’s capabilities design.

The capabilities object stores two representations of the attack set. First is the array list containing the actual query to the vulnerability database which is what is inputted as a part of the ABM input files. This will be used once to pull the action sets from the database by the method generateCapabilitiesSet() and will insert it into the capabilities set. The important fact to note is that if the set is declared as a whitelist or a blacklist, the query to the database will be the same. The blacklist will not pull every attack action that is not apart of the list because that is potentially a large list. If the set is a blacklist when performing the filter, the input set will have the blacklist’s entries removed. Whereas in the whitelist, the input set will be intersected with the whitelist. This will keep the sizes of the sets smaller and make it easier to perform calculations upon.

### 4.6 Attacker Preferences

Once all of the filtration of the attack action sets are complete, the preference model can then determine the probability to choose a source, target, and action independently. Along
with the attack action sets a mirrored version of the action sets is created but instead of the actual source, target, and action data the probabilities of choosing these parameters is created. Where the index for a specific source given a target in the attack action sets will directly correspond to its probability value in the mirrored probability sets. For each target there is a set of possible sources and each of those sources have a probability of choosing that source given a target. Likewise for each target there is a set of actions in which each action has a probability value of choosing the action given a target. This does create duplicate source and action values for a given target however this allows the capability for specific sources or actions to be weighed differently for different targets. Where as if there was only one list of source probabilities for all targets it would be difficult to keep track of what source belongs to what target and there may be a case where a source is eliminated for one target and not the other.

The probabilities of the attack action set are initially set uniformly as every target is equally likely to be chosen and every source and action given a target is equally likely to be chosen. For this preference model will directly modify the probability sets based of the user configured values for each of the preference parameters shown in the previous chapter. To model the preference of the source, target, or action a weighting scheme is employed to amplify certain values over an other. For example, if the attacker is defined to have a preference on mySQL actions the user can configure the preferences parameter to make those actions 10x more likely to happen than other actions. This action set can then be normalized and the multiplayer would still be preserved. The probability values will be modified by multiple different functions but because the only operation that is done to the probabilities is multiplication, order does not matter in this case.

4.6.1 Variably Weighted Preferences Values

The previous example only showed a static value as the weight to a probability where there are cases where the weights should change depending on some circumstances. Similarly
to the “new” knowledge concept of the kill chain logic, the attacker may have a preference on the source or target depending on the time at which it was uncovered. While in the kill chain the weight was only an inverse power function, the preference allows for four different functions based off of time to scale the preferences by. These four functions can be seen in Table 4.3 where $\alpha$ is the user set function modifier and $t$ is the difference between current attack step and the step at which the information was collected.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential</td>
<td>$\frac{1}{e^{-\alpha t}}$</td>
</tr>
<tr>
<td>Power</td>
<td>$\frac{1}{t-\alpha}$</td>
</tr>
<tr>
<td>Linear</td>
<td>$\frac{1}{t + \alpha}$</td>
</tr>
<tr>
<td>Constant</td>
<td>$\frac{1}{\alpha}$</td>
</tr>
</tbody>
</table>

The purpose of these is to allow the capability of certain functions to be able to weigh the probability of sources or targets depending on time. For example, in the case where the attacker has a preference of aggressively favoring targets that are the newest then they would choose a high $\alpha$ with the exponential function active. This makes it possible to favor the newest information and older information will be exponentially less favored. Where on the other side of the spectrum, the constant value will only scale the applicable values by the constant $\alpha$ and not take time into account. The $\alpha$ values are intended to be greater than or equal to 1 to give the highest priority to newer information as opposed to higher priority to older information, however this is not a restriction and a user could set the value to less than 1 if that is desired.
4.6.2 Preference Parameters To Determine Weights

This variable weighting method is only used for the source preferences, target preferences, and dead end preference. This is because this information is based off of time, where the action preferences are not. The source probabilities takes in account all the machines that are known to be compromised and determines how likely an attacker is going to use that machine as their next source. Whereas the target preferences is how likely an attacker is going to target a machine that has already been compromised. This is to prevent the attacker repeatedly targeting machines that they have already compromised which would gain them no new information in most cases, however in the case of a moving target defense that may not be the case. This is also the case for the dead end machines where a scan was performed and nothing was seen. In the case of a MTD and the attacker is aware of that, the attacker may want to revisit that machine at some point but for the most part the attacker would not favor using a dead end machine as a source again.

Persistence and preferred actions are the situations where there is no configurable functions available to the user. Instead for the persistence the attacker configures the mean and the standard deviation of the number of attempts the attacker will make to try a failed attack again. A random number is generated based off the normal distribution set and if the number of steps is less than the random number, the attack will be repeated. This means that if the attacker is going to be persistent, all the target probabilities that was not the last target will be zeroed out. This is also applied for the action also, the attacker can be persistent on the target and/or the action. An attacker may choose to try another vulnerability on the same target that they just failed on in which the last vulnerability performed is zeroed out. Preferred actions preference is the simplest of the preference modifiers because the user assigns a static weight to actions and it is applied to the action sets. This can be used to favor or deter the attacker from performing certain actions or preferring a specific type of reconnaissance. The compiled list of modifiable fields of the preference model can be seen in Table 4.4.
Table 4.4: The user configurable variables that are used to control the attacker’s preferences.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Configurable Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Preferences</td>
<td>$\alpha$, Weighting Function</td>
</tr>
<tr>
<td>Persistence</td>
<td>$\alpha, \sigma$</td>
</tr>
<tr>
<td>Compromised Target</td>
<td>$\alpha$, Weighting Function</td>
</tr>
<tr>
<td>Dead End</td>
<td>$\alpha$, Weighting Function</td>
</tr>
<tr>
<td>Preferred Actions</td>
<td>Map $&lt;$Action, Weight$&gt;$</td>
</tr>
</tbody>
</table>

4.7 Choosing an Attack Action

Once the probabilities have been fully modified by the preferences, the action selection process can be started. First the attack action sets must be reduced down into a single array. To start, all source, target, and action sets are normalized after the preferences are complete. For each target there is a corresponding target to source set/probabilities and a target to action set/probabilities. For a given target if there is 3 possible sources and 5 possible actions, this will result in 15 individual source-target-action grouping with 15 corresponding probabilities for one target. To combine the individual probabilities of the sources, targets, and actions, the values are simply multiplied together to yield one value. This is possible because each of the source and action probabilities only rely on a single target and are normalized with in and each of the target probabilities are normalized with each other. This will yield two lists one for the source-target-action values and another for the probabilities each of size total sources multiplied by the size of the total number of targets and the total number of actions left after the filtration.

Given the single probabilities set, a cumulative set of probabilities is generated based of this set. At this point the set is normalized and cumulative thus to pick an action a single random number is generated between 0 and 1. A binary search to find what index in the cumulative set does this random number fall in which directly corresponds to the chosen action in the source-target-action group set. This source-target-action grouping is then organized into the attack action data structure which will be used by the simulation core to process the attacker’s selection. This contains IP addresses, CVE data, and if needed
any parameters needed to simulate the attack like in reconnaissance actions. This action is returned to the core and the ABM process is completed.

4.7.1 No Actions To Select

Throughout the filtration process there may be cases where there is no attacks to choose from. What this generally means is that based off the COI, there was no actions that the attacker could perform. This can be due to the attacker not having the capabilities to perform any attacks based off the service knowledge they have or they have exhausted all of their options. This is known as a “NO_ATTACK” type in the simulation. This is a case that must be handled by the fuzzy logic and the attacker must react to this situation either by stopping the simulation or attempting to learn more information. Regardless it is worth addressing that there will possibly be cases where the attacker simply cannot progress due to their knowledge of the network.
Chapter 5

Experiments and Results Analysis

Co-developed with this ABM, a simulator called the Cyber Attack Scenario and Network Defense Simulator (CASCADES) will be used to analyze the output of the ABM. CASCADES’s only objective in this work is to take an attack action from the ABM and compile the effects of this action on the network. None of attack action selection logic is located anywhere within the core CASCADES logic and the statistics generated in this chapter are derived from the attacker’s perspective only. Four different types of attackers will be addressed in this work: an expert, an amateur, an exhaustive, and random attackers. For each of these types of attackers, their performance will be evaluated on the success rate, number of actions performed, the types of attacks performed, and the types of services exploited by the attacker for two different types of networks. Then lastly, a single attack track will be analyzed to show the results of a single attack scenario in a granular fashion.

5.1 CASCADES Setup

CASCADES is versatile cyber attack simulation platform with may features that are not used in this work. The ABM is a completely separate function to the CASCADES logic and the only functions used by the core logic is the determine next attack action and the update functions. The ABM resides in what is called the Attack Track in CASCADES which represents a single attacker with a single intent. The overall architecture for CASCADES can be seen in Figure 5.1.
The parts of CASCADES that will be used is the ABM, Intent, Virtual Terrain, and the Vulnerability Database. A moving target defense (MTD) will not be used as it has not been adapted from MASS to CASCADES. All intrusion detection systems and sensor noise simulations are deemed to be irrelevant to this work as only the ground truth actions are to be taken in account. For each experiment, 1000 simulations will be run with the same ABM and intent to generate an average behavior metric.

### 5.1.1 Virtual Terrain Setup

The configuration of the network, or in this case the virtual terrain, is critical to the process of testing the ABM. If the network is too small then there may be little variance in the results due to the low amount of options. The network being too large is not as much as a concern as this will amplify some of the differences between the attackers. However testing the patience and overall persistence of the attacker is not being tested where instead a few key design choices to test certain aspects of the attacker’s behavior. The virtual terrain that is in use in this work is seen in Figure 5.2 which describes 15 individual machines each with their own special characteristics.

Although this network is relatively small (in terms of number of machines) compared to enterprise networks and even a small business network, it has key design aspects that gives
it realistic and diverse results. First is the modeling of the separation of a guest network and the internal network. It is typical for a network to have one subnet that has internet connectivity but not access to the rest of the internal network like a guest network would have. This is to test the case where the attacker gains access to this machine and there is nothing else to do because the permissions are very limited. This network also is configured with firewalls in mind. A typical network will have a tiered permission set up where only specific machines can access other machines due to permissions. If a machine does not the correct permissions then the machine cannot access or even acknowledge that a machine exists. This allows for the ability to simulate multistage attacks where an attacker must gain access to other machines before progressing. The list of machines and permissions between each of these machines can be see in Table 5.1. The ID’s in this table directly correspond to the labels on the machine icons in Figure 5.2.

Note the names of each of the machines in this description of the VT. Each machine and its location in the network has some sort of relevance to represent the types of machines on a typical network. On level 1, which are all of the machines that are accessible from the external network, are all machines that typically will be exposed like a mail server or a web server. Whereas level 2 contains the machines that have more permissions to the network...
like developer machines, the domain server, and a print server. Then on the bottom end are all the internal services and critical machines like the storage systems, backup servers, and virtual machine servers. This gives a more realistic setup for a network that although not as large as a typical network but has all of the key characteristics of a real network.

Likewise the permissions between machine or the firewall was configured in a certain way to test how the attacker responds given certain situations. Already mentioned, the dedicated guest network with many vulnerabilities is set up on purpose to test the attacker’s response when there is nothing to learn from that machine. That machine is also chosen to have many vulnerabilities to possibly make it more appealing to some attackers than others. The concept of an administrative machine is also shown here which has access to every machine on the network which may provide interesting results as if that machine is compromised, the entire network can be accessed. Then lastly the backup sever can only be accessed by the level 3 machines and the administrator which effectively creates a
fourth level. With all these design elements taken in consideration, this provides the base opportunities for the attacker to perform their attack scenario.

A perfectly configured and managed network should contain the least amount of entry points and vulnerabilities, this however is not usually the case. It can be difficult to maximize the resilience of a network to a cyber attacker and misconfigurations or certain aspects of the network can be overlooked. To examine the effects of a misconfigured network, an intentional backdoor into the protected section network is inserted into the network. As an example, a developer wants to access the development GitLab server from outside of the network. Instead of using a VPN or some other method of accessing it safely, instead the developer exposed the server to the external network by changing the firewall permissions. This allows this machine to be directly accessible from the internet, bypassing the other layers of security into the network. These modifications can be seen in bold in Table 5.2.

Table 5.2: The modified VT permissions with the misconfigurations in bold.

<table>
<thead>
<tr>
<th>ID</th>
<th>Machine Info</th>
<th>Permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>External Router</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>Guest Wireless Router</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>Level 2 Router</td>
<td>X X X</td>
</tr>
<tr>
<td>4</td>
<td><strong>Level 3 Router</strong></td>
<td>X X X</td>
</tr>
<tr>
<td>11</td>
<td>Guest Windows 8</td>
<td>X</td>
</tr>
<tr>
<td>12</td>
<td>External Web Server</td>
<td>X X</td>
</tr>
<tr>
<td>13</td>
<td>External DNS Server</td>
<td>X X</td>
</tr>
<tr>
<td>14</td>
<td>External Mail Server</td>
<td>X X</td>
</tr>
<tr>
<td>15</td>
<td>External Web App Server</td>
<td>X X</td>
</tr>
<tr>
<td>100</td>
<td>Developer Machine (Unix)</td>
<td>X X X</td>
</tr>
<tr>
<td>101</td>
<td>Developer Machine (Mac)</td>
<td>X X X</td>
</tr>
<tr>
<td>102</td>
<td>Administrator (Windows)</td>
<td>X X X</td>
</tr>
<tr>
<td>103</td>
<td>OES Print Server</td>
<td>X X</td>
</tr>
<tr>
<td>104</td>
<td>OpenLDAP Server</td>
<td>X X</td>
</tr>
<tr>
<td>1000</td>
<td>VMWare ESXi</td>
<td>X X</td>
</tr>
<tr>
<td>1001</td>
<td>NAS</td>
<td>X X</td>
</tr>
<tr>
<td>1002</td>
<td>Backup Server</td>
<td>X</td>
</tr>
<tr>
<td><strong>1003</strong></td>
<td>Development GitLab</td>
<td>X X X</td>
</tr>
<tr>
<td>1004</td>
<td>MySQL Customer Info</td>
<td>X X</td>
</tr>
</tbody>
</table>
5.1.2 Attacker’s Intent

In CASCADES, the intent of the attacker is a separate model defined outside of the attacker behavior model. Based on the output of the attack action being performed, the intent module will return the attacker’s next intent or goal. In the ABM, the intent is only taken into account if and only if the attacker has some knowledge to perform the intent (the exfiltration kill chain stage). Because of this, between all of the experiments the intent will remain unchanged between runs however chosen to be sufficiently difficult enough to challenge each of the attackers. In this case, the intent of the attackers will be focused on the backup server in the third level of the network. This is done because the backup server has the most restrictive of access permissions as only the level 3 machines and the administrative have access to the machine.

5.1.3 Attacker Behavior Configurations

This thesis addresses four different types of attackers: the expert, the amateur, the exhaustive, and the random attacker. Both the expert and the amateur designed to measure the differences between capabilities of attackers while the comprehensive and random attackers are for measuring the general resilience to a brute force attack. Each of these attackers have interesting characteristics that makes them different from one another, showing a wide variety of behaviors that can be modeled with this work.

Amateur vs. Expert Behavior

The amateur behavior in this context is someone who has a limited skill set and does not have a general direction throughout the attack. Where as the expert attacker comes preloaded with some information about the types of services on the network already along with a much larger skill set than the attacker. The amateur attacker performs thorough reconnaissance scans which to a novice attacker provides the most information back but is also the most revealing and loudest from a IDS or defense perspective. Then once there
are known machines and services by the amateur attacker, this is when the breach stage will be chosen. The amateur attacker will be limited on the machines they can choose after this stage because their capabilities are limited to a few select services. Contrasting to the expert who is methodical about the amount of reconnaissance and breaches they perform. The expert attacker is chosen to be balanced on the amount of recon versus the amount of attacking they perform, only gaining enough information without drawing attention to themselves. A summary of the two different behaviors can be seen in Table 5.3.

The preferences between the two attackers also vary greatly. The preferences on the source machine are similar but the expert is more aggressive in using the last compromised machine as a source. This is the difference between using the exponential functions vs. the inverse or linear functions in the setup of the preference values. The exponential function of the expert attacker favors recently compromised targets much more aggressively than the linear function of the amateur. However, if the expert identifies the compromised target as a dead end (no new information can be learned from using it) then it will never choose that as a source again. Whereas the amateur attacker does not really recognize that it is a dead end and may try using it as a source again. As for the target machine preferences, the amateur behavior chooses anything that it knows about but slightly favoring newly discovered machines. The expert is much more aggressive in the target selection and favors targets that are uncompromised and recently discovered. Lastly, the preference on actions is what greatly differs these two attackers from one another. The amateur attacker has no preference on the types of actions but the expert attacker has preferences based on preexisting knowledge of the network. The expert is setup to acknowledge that this is a development company and certain common services on the protected section of the network is given preference like SQL, python, java, etc. This gives the attacker an advantage because they have an idea of what they are looking for. These differences in preferences can be seen in Table 5.4.
Table 5.3: ABM configuration summaries for each attacker.

<table>
<thead>
<tr>
<th></th>
<th>Amateur</th>
<th>Expert</th>
<th>Comprehensive</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capabilities</td>
<td>Limited</td>
<td>All</td>
<td>All</td>
<td>All</td>
</tr>
<tr>
<td>Recon Stage</td>
<td>Exhaustive</td>
<td>Balanced</td>
<td>Exhaustive</td>
<td>Random</td>
</tr>
<tr>
<td>Breach Stage</td>
<td>Scanned Any Machine</td>
<td>Balanced</td>
<td>Found Machines</td>
<td>Random</td>
</tr>
<tr>
<td>Exfil. Stage</td>
<td>Intent</td>
<td>Intent</td>
<td>90% Machines Compromised</td>
<td>Random</td>
</tr>
<tr>
<td>Stop Stage</td>
<td>5 Failures</td>
<td>5 Failures</td>
<td>Exhausted Known Options</td>
<td>5 Failures</td>
</tr>
</tbody>
</table>

**Comprehensive and Random Behaviors**

The comprehensive and random behaviors are much simpler nature because they are not necessarily representative of real behaviors. The comprehensive is the brute force, DoS-like behavior with the intentions of doing as much damage as possible. Where the random behavior is the base line of what could happen to a network if an attacker tried anything with no direction. The key characteristic of the comprehensive attacker is that it will exhaustively attack the network until it has 90% of the machines compromised one way or another. This is done by controlling the kill chain stages by determining how many machines are uncompromised compared to how many are known. Where the random will try anything at any time regardless of if it was done before. The summary of these behaviors can be seen in Table 5.3.

The preferences of these two behaviors are the the same story. In this case for the Random attacker, there are no preferences at all on anything. For each of the preference parameters the constant function modifier is used with a value of one, not affecting the probabilities of any actions at all. The comprehensive attacker will only target machines that they have not compromised yet and favor sources more recently found. The only preference on actions that the comprehensive attacker will have is on thorough recon scans because they are concerned about getting the most information and are unconcerned with how covert they are being. These differences can be seen in Table 5.4.
Table 5.4: ABM preference summary for each of the attackers.

<table>
<thead>
<tr>
<th>Source</th>
<th>Amateur</th>
<th>Expert</th>
<th>Comprehensive</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>Recently Compromised</td>
<td>Last Compromised</td>
<td>Recently Compromised</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Anything Known</td>
<td>Uncompromised only</td>
<td>Uncompromised only</td>
<td>None</td>
</tr>
<tr>
<td>Dead End</td>
<td>Low</td>
<td>Zero</td>
<td>Zero</td>
<td>None</td>
</tr>
<tr>
<td>Actions</td>
<td>None</td>
<td>Tailored</td>
<td>Thorough Scans</td>
<td>None</td>
</tr>
</tbody>
</table>

5.2 Analysis

In this section, the four different behavior models will be analyzed based off the number of steps taken, the success rate, the types of actions used, and the services exploited. Each of the behavior models were tested against two similar networks but one with a critical misconfigurations. Each of the simulations are independent from one another and the results of one attacker on a network do not effect the others. For each test, the attacker was simulated 1000 times and the statistics were averaged together to get an average response from the simulator. All tests were performed on a 2013 Apple MacBook Pro with an Intel i5 2.4GHz processor with 8GB of RAM. However timing results are not important in this work as there is no comparable benchmark or reference.

5.2.1 Base Network Configuration

Comparing the number of attack steps each attacker performs throughout all the simulations will illustrate not only the efficiency (how quickly the attacker achieves their goal) of the attacker but also the variation between each of the attacker’s. Even though the attackers in the same set of tests have the same behavior model configurations, the probabilistic nature of the preferences and attack selection logic allows for variation between each attacker. This variation can be seen in Figure 5.3 and the average attack steps by each of the attackers can be seen in Table 5.5.

Table 5.5: Average number of steps taken by each of the attackers

<table>
<thead>
<tr>
<th>Avg. Steps</th>
<th>Amateur</th>
<th>Expert</th>
<th>Comprehensive</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>27.5</td>
<td>27.6</td>
<td>54.9</td>
<td>47.4</td>
</tr>
</tbody>
</table>

74
Immediately it is evident that in terms of the attack step, the amateur and expert attackers are similar. The expert attacker has a slight edge over the amateur in terms of maximum number of steps but the average steps are identical. The minimum number of steps for the amateur and expert is 10 and 8 respectively, where the average steps was 27 steps. If all the correct steps are taken by the attacker, the best attack is significantly lower than the average. This concludes that the network being tested is too small and simple enough for even an average amateur attacker. When the attackers are successful the two attackers perform similarly. Examining failure rate does begin to expose the differences between these two attackers, as shown in Table 5.6

<table>
<thead>
<tr>
<th>% Failures</th>
<th>Amateur</th>
<th>Expert</th>
<th>Comprehensive</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>17.8</td>
<td>1.8</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The amateur attacker failed 17.8 percent of the time given this network configuration while the expert only failed 1.8 percent of the time. This disparity comes down to both the skill set and the patience of the attacker. In the case of the amateur, the behavior was configured to scan the network and then attack something that they are capable of. If the attacker at some point targeted the guest network and compromised the machine within, the
amateur attacker got confused and did not know how to proceed since that is a dead end. This also occurred with the expert attacker but only if the guest network was targeted first with a basic ping scan. This results in showing how the expert is more flexible in certain situations based on the knowledge of the network, if the expert had any other knowledge on the network other than that machine it does not fall into this trap.

The comprehensive and random attackers shows the network’s overall resilience to an attacker along with what would be considered the worst possible attackers. The comprehensive attacker is an attacker who is trying to do as much damage as possible where the random attacker will try anything until they are successful. Due to this, the number of attack steps are much greater than the conventional amateur and expert attackers. With the comprehensive behavior it is expected that the attack steps are higher because one of its requirements is to compromise every machine possible. The maximum number of steps is high because it searches for every machine possible and attacking machines that it has already attacked. Whereas the random attacker tries any action on any target, thus it is expected that the total steps are high. The random attacker being successful means that there are attack paths to the goal even though some are not entirely realistic. This is a brute force way of generating attack graphs to a goal because it will try anything to get to the goal. Due to the both the comprehensive and random attackers not having restrictive preferences on stopping the attack, both were successful 100 percent of the time.

The number of attack steps given this network is not a great representative of the behavior, just the performance of the attacker. Examining the different types of attacks the attacker performs gives a better indication of how the attacker uses each of these steps. Seen in Figure 5.4, the number of occurrences of the recon and breach kill chain stages are shown for each of the attackers.

The difference between the amateur and expert attacker becomes clearer when looking at the kill chain stages occurrences. The amateur attacker on average uses many less recon actions than the expert attacker. The amateur attacker favored using thorough scans because
it yields the most information and the attacker is unaware that is easily detected. Thus the amateur learns significant enough knowledge to immediately start attacking machines. The behavior of the amateur was to attack multiple machines if the attacker recently discovered and scanned them, which is shown by the significantly higher number of breach stages. The expert in contrast is near one-to-one recon to breach actions as an expert attacker is much more methodical in their approach. The behavior of the expert is to quietly step through the network and learn as much information as possible. The expert also favored using the ping scans and the connect scans for recon as they are the most covert actions. The one-to-one mapping between recon and breach is a desirable because it shows the attacker is trying to extract as much information out of the network as they progress.

Examining the comprehensive attacker, there is a large disparity between number of recon stages versus the breach stages. The difference is justified because the behavior of the attacker is to perform as much damage as possible by compromising every machine possible, explaining the significantly more breach stages than recon. The minimum value of breach has significance because it is the total number of machines on the network minus one. The reason for the minus one is because the intent machine counts as an exfiltration
stage and not a breach stage. The random attacker performs significantly more recon than breach. This in the end comes down to the rules chosen for the fuzzy logic behavior model. There is no direct way to randomly choose the conditions for the stages, instead greatly overlapping variables where chosen to drive the rules. The overlapping variables and rule set seems to favor recon more so than the breach stage.

The varying number of attack steps, failure rate, and the types of actions for each shows the flexibility of the model describing at a high level the differences between attackers. It can not determine the differences of skill and preferences of the attacker. This can be achieved by examining the types of services the different types of attackers used throughout their attacks. Figure 5.5 shows the percentage of times each service was exploited for each attacker. It should be noted again that the amateur has restricted capabilities and the expert, comprehensive, and random have unlimited capabilities. The expert attacker does however have preferences over certain services.

This analysis of the attacked services provides two purposes, one to obviously examine the individual behaviors of the attacker but also it exposes critical potential weaknesses in the system. If there is a particular service where many of the attacks are focused on, this raises red flags from an analyst perspective because there is a large security hole that needs to be addressed. In Figure 5.5 the service “ubuntu_linux” in particular stands out because all but the expert attacker have a significant number of attacks on that service. Digging deeper into why that is, two of the five externally facing machines are running a version of Ubuntu, the DNS server and web server. Thus it makes sense that this is the most active service because it is on the front line of machines to get attacked and is also the most vulnerable in terms of the number of vulnerabilities it has.

It can be seen that the number types of services used by the amateur and the others are drastically different. This comes down to the difference in capabilities between the attackers. The amateur was only given select vulnerabilities to common services like Ubuntu, SQL, and Java. Where the expert attacker has access to a wide variety of vulnerabilities
<table>
<thead>
<tr>
<th>Service</th>
<th>Amateur</th>
<th>Expert</th>
<th>Comprehensive</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>.net_framework</td>
<td>0.00%</td>
<td>0.25%</td>
<td>0.27%</td>
<td>0.17%</td>
</tr>
<tr>
<td>acrobat_reader</td>
<td>0.00%</td>
<td>1.26%</td>
<td>3.58%</td>
<td>2.77%</td>
</tr>
<tr>
<td>bind</td>
<td>0.00%</td>
<td>1.26%</td>
<td>1.26%</td>
<td>1.24%</td>
</tr>
<tr>
<td>chrome</td>
<td>0.00%</td>
<td>2.99%</td>
<td>7.60%</td>
<td>9.00%</td>
</tr>
<tr>
<td>debian_linux</td>
<td>12.71%</td>
<td>8.26%</td>
<td>3.61%</td>
<td>2.20%</td>
</tr>
<tr>
<td>enterprise_linux</td>
<td>13.76%</td>
<td>2.92%</td>
<td>9.60%</td>
<td>7.58%</td>
</tr>
<tr>
<td>esxi</td>
<td>0.00%</td>
<td>3.39%</td>
<td>2.63%</td>
<td>1.96%</td>
</tr>
<tr>
<td>exchange_server</td>
<td>0.00%</td>
<td>0.03%</td>
<td>3.57%</td>
<td>4.10%</td>
</tr>
<tr>
<td>firefox</td>
<td>0.03%</td>
<td>6.35%</td>
<td>2.42%</td>
<td>1.72%</td>
</tr>
<tr>
<td>freebsd</td>
<td>0.00%</td>
<td>0.03%</td>
<td>7.27%</td>
<td>6.06%</td>
</tr>
<tr>
<td>ftp_server</td>
<td>0.00%</td>
<td>0.08%</td>
<td>0.33%</td>
<td>0.22%</td>
</tr>
<tr>
<td>gitlab</td>
<td>0.00%</td>
<td>0.01%</td>
<td>0.05%</td>
<td>0.06%</td>
</tr>
<tr>
<td>http_server</td>
<td>0.16%</td>
<td>0.03%</td>
<td>0.80%</td>
<td>0.85%</td>
</tr>
<tr>
<td>java</td>
<td>2.81%</td>
<td>6.60%</td>
<td>0.36%</td>
<td>0.28%</td>
</tr>
<tr>
<td>java_1.6</td>
<td>0.21%</td>
<td>3.63%</td>
<td>0.03%</td>
<td>3.03%</td>
</tr>
<tr>
<td>javafx</td>
<td>21.53%</td>
<td>0.13%</td>
<td>3.36%</td>
<td>2.77%</td>
</tr>
<tr>
<td>mac_os_x</td>
<td>0.00%</td>
<td>0.07%</td>
<td>4.13%</td>
<td>0.11%</td>
</tr>
<tr>
<td>munin</td>
<td>0.00%</td>
<td>2.57%</td>
<td>0.13%</td>
<td>0.20%</td>
</tr>
<tr>
<td>mysql</td>
<td>0.00%</td>
<td>0.01%</td>
<td>0.37%</td>
<td>4.12%</td>
</tr>
<tr>
<td>networker</td>
<td>0.00%</td>
<td>6.62%</td>
<td>0.82%</td>
<td>0.15%</td>
</tr>
<tr>
<td>office</td>
<td>0.00%</td>
<td>0.87%</td>
<td>0.11%</td>
<td>6.45%</td>
</tr>
<tr>
<td>openenterprise_server</td>
<td>0.00%</td>
<td>1.92%</td>
<td>8.22%</td>
<td>0.89%</td>
</tr>
<tr>
<td>openldap</td>
<td>0.00%</td>
<td>0.90%</td>
<td>1.10%</td>
<td>1.11%</td>
</tr>
<tr>
<td>python</td>
<td>6.84%</td>
<td>0.44%</td>
<td>1.17%</td>
<td>1.16%</td>
</tr>
<tr>
<td>rsa_dataprotection_manager</td>
<td>0.00%</td>
<td>7.01%</td>
<td>0.25%</td>
<td>0.63%</td>
</tr>
<tr>
<td>rsa_data_protection_server</td>
<td>0.00%</td>
<td>0.54%</td>
<td>0.14%</td>
<td>8.41%</td>
</tr>
<tr>
<td>ruby_on_rails</td>
<td>0.00%</td>
<td>1.63%</td>
<td>6.89%</td>
<td>0.39%</td>
</tr>
<tr>
<td>safari</td>
<td>0.00%</td>
<td>2.43%</td>
<td>0.54%</td>
<td>1.31%</td>
</tr>
<tr>
<td>samba</td>
<td>4.11%</td>
<td>0.01%</td>
<td>2.10%</td>
<td>2.98%</td>
</tr>
<tr>
<td>sql_server</td>
<td>5.24%</td>
<td>15.03%</td>
<td>0.66%</td>
<td>0.59%</td>
</tr>
<tr>
<td>ssh2</td>
<td>0.00%</td>
<td>3.63%</td>
<td>0.84%</td>
<td>16.21%</td>
</tr>
<tr>
<td>ubuntu_linux</td>
<td>32.60%</td>
<td>7.55%</td>
<td>14.80%</td>
<td>0.02%</td>
</tr>
<tr>
<td>vsphere_client</td>
<td>0.00%</td>
<td>4.75%</td>
<td>0.05%</td>
<td>2.68%</td>
</tr>
<tr>
<td>windows_7</td>
<td>0.00%</td>
<td>4.45%</td>
<td>3.94%</td>
<td>1.77%</td>
</tr>
<tr>
<td>windows_8</td>
<td>0.00%</td>
<td>1.31%</td>
<td>1.34%</td>
<td>5.90%</td>
</tr>
<tr>
<td>windows_server_2008</td>
<td>0.00%</td>
<td>1.03%</td>
<td>4.79%</td>
<td>0.87%</td>
</tr>
<tr>
<td>workstation</td>
<td>0.00%</td>
<td>0.01%</td>
<td>0.80%</td>
<td>0.04%</td>
</tr>
<tr>
<td>xcode</td>
<td>0.00%</td>
<td>0.01%</td>
<td>0.02%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

Total Actions: 7515 7558 20878 10820

Figure 5.5: The percentage of times a service was exploited by each of the attackers on the base network. (Services pertaining to intent are bolded)
comparatively. The comprehensive and random attackers have similar distribution of attacks because their capabilities were not limited in anyway nor did they have preference on any actions. The expert’s chosen services are different because there was a heavy preference on SQL, Java, Debian, RSA type services. This allows the capability to test various different skill sets to understand how the attacker can penetrate the network.

5.2.2 Misconfigured Network

One of the goals of this work is to model the attacker and the network in way that will aid in understanding the relationship between the two. To exploit this relationship, the network was intentionally misconfigured to allow for a supposed to be heavily protected machine exposed to the internet. The same exact tests from the base network will be conducted on this less-than ideal network configuration to understand how the attacker can attack the network. It is important to understand how a single change to the network could possibly affect the network positively or negatively and how the attacker is affected by this change. For this experiment, the number of attack steps are summarized in Figure 5.6, the average number of steps are shown in Table 5.7, the failure rate in Table 5.8, and the average change of steps between the base and the misconfigured network is shown in Table 5.9.

Table 5.7: Average number of steps taken by each of the attackers on the misconfigured network

<table>
<thead>
<tr>
<th></th>
<th>Amateur</th>
<th>Expert</th>
<th>Comprehensive</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Steps</td>
<td>28.8</td>
<td>20.4</td>
<td>36.5</td>
<td>33.5</td>
</tr>
</tbody>
</table>

Table 5.8: Failed attack rate for each type of attackers on the misconfigured network

<table>
<thead>
<tr>
<th></th>
<th>Amateur</th>
<th>Expert</th>
<th>Comprehensive</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Failures</td>
<td>15.5</td>
<td>1.2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.9: Average percent change of attack steps between the base network configuration and the misconfigured network

<table>
<thead>
<tr>
<th></th>
<th>Amateur</th>
<th>Expert</th>
<th>Comprehensive</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. % Change</td>
<td>4.35</td>
<td>-26.3</td>
<td>-33.4</td>
<td>-29.1</td>
</tr>
</tbody>
</table>
Immediately it can be noticed that the average number of steps to achieve the goal for the expert, comprehensive, and random attackers decreased by 26.3%, 33.4%, and 29.1% respectively. Where the amateur attacker remained relatively unchanged and actually increasing by 4.35%. This is explained by the amateur attacker simply not having the capabilities to take advantage of the security hole in the network. Which is an interesting aspect to examine because not all attackers are made equal and regardless of how big of a problem this misconfiguration seems, not every attacker can take advantage of it. Although in this situation the seriousness of the hole can directly be observed by the change of the average steps for the comprehensive and random attackers. These attackers are unguided and randomly compromising machines and are approaching the average number of steps for the amateur attacker. It is expected that the expert attacker to benefit greatly from this misconfiguration but the random performing near the amateur attacker should be alarming. Looking at the minimum steps of the attackers, it can be observed that the expert and the random both completed their goal in 4 steps. Meaning that even when developing random attack paths some of them were equal to the best of the expert attacker.
The general trend from the base network for the kill chain stages applies here in the misconfigured network seen in Figure 5.7. The amateur remains unchanged from previously, the expert still performs recon and breach equally, comprehensive favors breach much more than recon, and random has significantly more recon actions. The major change is the general shift to the left for expert, comprehensive, and random where amateur stays in the same place.

Figure 5.7: Comparison of the occurrences of the recon and breach stages for each attacker type on the misconfigured network.

Mentioned previously, by looking at the services that were exploited it is possible to determine where some weak spots in the network are. Figure 5.8 shows the percentage of times the attacker exploits a particular service. As defined by the network, the machine that was misconfigured was the GitLab Development server which had two services running, RedHat Enterprise (“enterprise_linux” on figure) and GitLab. The amateur attacker did not have the capabilities to exploit this particular machine however does still have significant attacks on another machine also running some other version of RedHat. To show the difference between the base network and misconfigured, the percent change in the attacks on RedHat are shown in Table 5.10.

As expected, the expert attacker benefits the most from this misconfiguration boasting
<table>
<thead>
<tr>
<th>Service</th>
<th>Amateur</th>
<th>Expert</th>
<th>Comprehensive</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>.net_framework</td>
<td>0.00%</td>
<td>0.13%</td>
<td>0.27%</td>
<td>0.18%</td>
</tr>
<tr>
<td>acrobat_reader</td>
<td>0.00%</td>
<td>1.81%</td>
<td>3.39%</td>
<td>2.40%</td>
</tr>
<tr>
<td>bind</td>
<td>0.00%</td>
<td>0.91%</td>
<td>1.20%</td>
<td>1.12%</td>
</tr>
<tr>
<td>chrome</td>
<td>0.00%</td>
<td>6.31%</td>
<td>7.78%</td>
<td>8.31%</td>
</tr>
<tr>
<td>debian_linux</td>
<td>11.78%</td>
<td>7.46%</td>
<td>3.19%</td>
<td>1.74%</td>
</tr>
<tr>
<td>enterprise_linux*</td>
<td>13.73%</td>
<td>14.18%</td>
<td>12.83%</td>
<td>11.53%</td>
</tr>
<tr>
<td>esi</td>
<td>0.00%</td>
<td>1.67%</td>
<td>2.54%</td>
<td>1.71%</td>
</tr>
<tr>
<td>exchange_server</td>
<td>0.00%</td>
<td>3.50%</td>
<td>3.47%</td>
<td>3.66%</td>
</tr>
<tr>
<td>firefox</td>
<td>0.04%</td>
<td>1.48%</td>
<td>2.28%</td>
<td>1.89%</td>
</tr>
<tr>
<td>freebsd</td>
<td>0.00%</td>
<td>4.98%</td>
<td>6.96%</td>
<td>4.96%</td>
</tr>
<tr>
<td>ftp_server</td>
<td>0.00%</td>
<td>0.17%</td>
<td>0.25%</td>
<td>0.26%</td>
</tr>
<tr>
<td>gitlab*</td>
<td>0.00%</td>
<td>0.04%</td>
<td>0.17%</td>
<td>0.13%</td>
</tr>
<tr>
<td>http_server</td>
<td>0.13%</td>
<td>0.41%</td>
<td>0.70%</td>
<td>0.72%</td>
</tr>
<tr>
<td>java</td>
<td>2.55%</td>
<td>0.20%</td>
<td>0.30%</td>
<td>0.26%</td>
</tr>
<tr>
<td>java_1.6</td>
<td>0.24%</td>
<td>0.02%</td>
<td>0.02%</td>
<td>0.00%</td>
</tr>
<tr>
<td>javafx</td>
<td>21.63%</td>
<td>3.04%</td>
<td>3.47%</td>
<td>2.51%</td>
</tr>
<tr>
<td>mac_os_x</td>
<td>0.00%</td>
<td>2.78%</td>
<td>3.86%</td>
<td>2.81%</td>
</tr>
<tr>
<td>munin</td>
<td>0.00%</td>
<td>0.15%</td>
<td>0.14%</td>
<td>0.08%</td>
</tr>
<tr>
<td>mysql</td>
<td>0.00%</td>
<td>0.04%</td>
<td>0.35%</td>
<td>0.31%</td>
</tr>
<tr>
<td>networker</td>
<td><strong>0.00%</strong></td>
<td><strong>3.78%</strong></td>
<td><strong>1.31%</strong></td>
<td><strong>5.11%</strong></td>
</tr>
<tr>
<td>office</td>
<td>0.00%</td>
<td>0.09%</td>
<td>0.13%</td>
<td>0.08%</td>
</tr>
<tr>
<td>open_enterprise_server</td>
<td>0.00%</td>
<td>5.77%</td>
<td>7.36%</td>
<td>5.80%</td>
</tr>
<tr>
<td>openldap</td>
<td>0.00%</td>
<td>0.65%</td>
<td>1.04%</td>
<td>1.02%</td>
</tr>
<tr>
<td>python</td>
<td>7.28%</td>
<td>1.00%</td>
<td>0.97%</td>
<td>1.10%</td>
</tr>
<tr>
<td>rsa_data_protection_manager</td>
<td>0.00%</td>
<td>1.20%</td>
<td>0.40%</td>
<td>1.66%</td>
</tr>
<tr>
<td>rsa_data_protection_server</td>
<td>0.00%</td>
<td>0.68%</td>
<td>0.23%</td>
<td>0.77%</td>
</tr>
<tr>
<td>ruby_on_rails</td>
<td>0.00%</td>
<td>3.81%</td>
<td>6.67%</td>
<td>7.57%</td>
</tr>
<tr>
<td>safari</td>
<td>0.00%</td>
<td>0.39%</td>
<td>0.50%</td>
<td>0.49%</td>
</tr>
<tr>
<td>samba</td>
<td>4.18%</td>
<td>1.37%</td>
<td>1.88%</td>
<td>0.92%</td>
</tr>
<tr>
<td>sql_server</td>
<td><strong>5.46%</strong></td>
<td><strong>9.25%</strong></td>
<td><strong>1.09%</strong></td>
<td><strong>4.65%</strong></td>
</tr>
<tr>
<td>ssh2</td>
<td>0.00%</td>
<td>0.28%</td>
<td>0.62%</td>
<td>0.36%</td>
</tr>
<tr>
<td>ubuntu_linux</td>
<td><strong>32.98%</strong></td>
<td><strong>13.66%</strong></td>
<td><strong>14.71%</strong></td>
<td><strong>16.51%</strong></td>
</tr>
<tr>
<td>vsphere_client</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.02%</td>
<td>0.00%</td>
</tr>
<tr>
<td>windows_7</td>
<td>0.00%</td>
<td>2.59%</td>
<td>3.28%</td>
<td>2.35%</td>
</tr>
<tr>
<td>windows_8</td>
<td>0.00%</td>
<td>1.20%</td>
<td>1.26%</td>
<td>1.41%</td>
</tr>
<tr>
<td>windows_server_2008</td>
<td>0.00%</td>
<td>4.44%</td>
<td>4.70%</td>
<td>5.27%</td>
</tr>
<tr>
<td>workstation</td>
<td>0.00%</td>
<td>0.56%</td>
<td>0.59%</td>
<td>0.36%</td>
</tr>
<tr>
<td>xcode</td>
<td>0.00%</td>
<td>0.02%</td>
<td>0.06%</td>
<td>0.03%</td>
</tr>
</tbody>
</table>

**Total Actions**: 7540 5403 13880 7824

Figure 5.8: The percentage of times a service was exploited by each of the attackers on the misconfigured network (Services pertaining to intent are bolded). *Service on misconfigured machine
Table 5.10: Average percent change of attacks against the service “enterprise_linux” between the two networks

<table>
<thead>
<tr>
<th>Avg. % Change</th>
<th>Amateur</th>
<th>Expert</th>
<th>Comprehensive</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>-.235</td>
<td>384</td>
<td>33.7</td>
<td>52.1</td>
<td></td>
</tr>
</tbody>
</table>

a 384% increase in the number of attacks on RedHat compared to the base case. Then because the expert attacker could capitalize on this vulnerability, they were able to achieve their goal swiftly since they accessed the deepest layer of the network. The change in attacks on RedHat for the comprehensive and random attackers backs up the decrease in number of attack steps from previously. The comprehensive attacker only needed to compromise and perform a scan from the GitLab machine to access the entire network to attack. The random attacker benefited greatly from this hole because the fewer the steps to the goal the more likely it is to randomly choose it faster. Where as the Amateur saw no change at all because of this vulnerability because it couldn’t take advantage of it due to capabilities.

These results are unique compared to previous works because the individual behaviors of different types of attackers are exposed showing the effects of each other on a network. The attack graph works seen in [44, 19, 52, 25] does have the capability to generate the attack paths that were generated with this model based off the vulnerability data but can not represent the critical weaknesses in a network configuration nor does it show the individual differences between attackers. This work not only shows the attack paths taken by the attacker but also can show the frequency at which the attacker selects certain machines and services. The game theory works from [50, 5] do have potential to show the differences between the attackers however to achieve this extensive expertise in game theory and cyber security is required to configure the model to show different behaviors. These models also assume that the knowledge of the attacker is static which limits the types of analysis that can be performed in the sense that the researcher would not know what the attacker needed to know to perform certain actions.
Chapter 6

Conclusion and Future Work

6.1 Conclusion

The primary contribution of this thesis is the introduction of knowledge-based behavior modeling to cyber attack adversary descriptions. By defining the key aspects of a cyber attack and attacker such as the attacker’s intentions, capabilities, opportunities, and preferences, it is possible to define a model that describes set of rules and attacker may have along with the probabilistic and random nature that cyber attacks have. Cyber attackers do not base their next actions based completely on mathematical models nor do they base their decisions on statistics as modeled by previous works. Instead the actions are based off of what they know about the field, the network, or goals, all of which are captured here in this work.

With the use of the simulator CASCADES, it was shown that the vulnerabilities on a network tell only one part of the story. Each different attacker has their own skills, personality, and style that plays a different role in how the network was attacked. Four different types of attackers were shown in this work: an amateur, an expert, a comprehensive attacker, and a random attacker. Despite the amateur’s limited skill set, it was found that the amateur attacker was equally effective at achieving the intent as a expert when successfully completing the simulation. When a misconfiguration was added into the network, the
amateur was unable to capitalize on the misconfigurations where the others saw improvements of up to 33%. The random attacker performed just as well as the amateur attacker on average when there was a misconfiguration and the best attack for the random was equal to the best attack for the expert attacker. This shows that as misconfigurations/vulnerabilities are introduced into the network the more naive attackers benefit which negatively impacts the overall security of the network.

This work proves to be useful for both network security analytics and cyber attack scenario analysis. Realistic and detailed cyber attack scenario for various types of attackers is difficult to find due to the secrecy of the field and the complexity of the data. This provides the potential to generate various scenarios that are based on someone else or a scenario that no one has seen before. However this model is not perfect as it can be difficult to configure properly requiring substantial trial and error to achieve various behaviors. It also lacks the flexibility to model extremely complex behaviors that require very precise control logic to model. Given these limitations, the concept of applying the attacker’s knowledge to model their behaviors has proven to be effective.

6.2 Future Work

This behavior model, although proven to be effective, certain aspects of the model could be improved upon, expanded on, or investigated further. The future work can be broken up into three major parts: updating the knowledge base, the fuzzy logic kill chain selection, and the CASCADES simulator. The knowledge base section focuses on extra knowledge of the network or the cyber attacker that the attacker could use to make decisions that was not presented here in this thesis. The fuzzy logic controller for the kill chain comes with its issues can be expanded on and investigated further. Then lastly the simulation environment performed well with this new ABM but it too could be fleshed out to provide more accurate results using this ABM.
6.2.1 Knowledge Upgrades

This design of a behavior model with its large focus on developing a knowledge base throughout the attack requires a solid foundation of possible information an attacker could learn about. This thesis provides this foundation of network knowledge needed to perform a realistic cyber attack simulation. Certainly not everything could be represented but at the moment the attacker is unaware of machines that are in the same subnet from one another. This is a critical piece of information because attacking another machine in the same subnet is unlikely to provide you with any new significant knowledge gain. Along with subnet modeling, the notion of assets on the network should be investigated to use as attacker motivation or intent. At the moment the types of information an attacker can learn is entirely based off of network configuration and not what they could gain in terms of sensitive information. In this model, the recon model must also be reconfigured as anytime an attacker performs a recon action perfect and complete knowledge is returned. This may not always be the case as some recon scanning may not provide an attacker with the most detailed information needed to perform an attack, like version detailed numbers of services for example.

The intent of the attacker is based of the types of actions they want to perform and has little to do with the actual reason they are attacking the network. For example, an attacker may want to steal all the customer information off of a database server on the network. The current model will consider this complete if a certain action is complete regardless if it was the correct machine. If the assets, like customer data, where modeled then the attack would not be complete until this information was exfiltrated. Lastly, the topic of social engineering and insider threats should be researched in addressing the critical information that can be gained. Most cyber attacks these days contain some sort of social engineering whether it is a phishing or reading a password off of a desk. Investigation should be done to understand the types of social engineering that could be performed and how it can be adapted into the model.
6.2.2 Fuzzy Logic

The fuzzy logic controller used here provided the ability to give some uncertainty to the membership of the rules defined for each of the kill chain stages. This uncertainty brings up its own issues as it made the simulation difficult to predict and sending the simulation into loops in certain situations. This could be due to the stage of the kill chain be selected at the beginning of the attack action process and it being ridged throughout the rest of the process. The defuzzification process should be examined in more detail to understand how it can be used to not aggressively pick one state over the other. A possible solution is to combine the kill chain membership determination with the preferences to weigh actions based on their membership to a stage instead of hard filtering out every action that is not apart of the state. In the case of where there is a draw between absolute membership of two states like recon and breach, this may prevent the times where the situation gets stuck selecting nothing but recon for example. Lastly the types of kill chain stages could be expanded on to allow for multiple different types of attacks like a phishing or automation. This allows for any different types of attackers to be modeled.

6.2.3 CASCADES Future Work

The CASCADES simulation environment can be expanded on to allow for some of these changes above. First more types of attacks should be addressed like phishing, trace removal, or privilege escalation. A main limitation of realism in CASCADES is that it assumes that every attack is a network and privilege escalation type attack. The information to handle DoS attacks and local attacks are contained in the vulnerability database as the CVSS data but not used at this moment. By implementing this information into the system, this would require the attacker in some cases to first gain access to a machine as a user and the locally gain access to the root user. Or the attacker may decide to just take down the machine for everyone like the comprehensive attacker and perform DoS attacks only. This sort of modeling of the network and vulnerabilities allows the capability to model
significantly more behaviors more directly than the current model.
Bibliography


Appendix A

Gaining Knowledge Through Reconnaissance

The main objective for an attacker is to develop the knowledge base to the point that the attacker can achieve their intent. Thus it is important to model different methods and tools that attackers commonly use to gain knowledge about their network. Tools like NMap, Nessus, or even social engineering techniques reveal certain aspects of a network like machines, services, and vulnerabilities. NMap is the most widely used and accepted tool by security professionals and attackers to perform general network reconnaissance and analysis. While this work does not try to emulate NMap, the functionality and the outputs are adapted to this application.

In this work, four functions of NMap are replicated. The four functions are: ping scan, port scan, TCP connect scan, and thorough scan. Ping scan reveals to the attacker which ip addresses are on the range given by the user as seen in Algorithm 1 and port scan is the same but with ports. TCP connect scan is similar to the port scan but will send a TCP packet to examine the response from the target. This reveals the services that are installed on the machine along with the open ports as seen in Algorithm 2. Then lastly the thorough/comprehensive scan reveals all the ports and services for a range of IP addresses, being the most effective but least stealthy methods for reconnaissance.

Algorithm 1 shows the general methodology of how the ping scan outputs are represented. The actual processes of performing the scan consists of retrieving a sorted list off all the IP addresses in the virtual terrain and finding the indices of the IP address at the beginning of the range given and at the end of the range. Then once the indices are found using a binary search, a sub list of all the values between these indices is created to represent the IP addresses in the virtual terrain that is within the bounds of the scan. Then to filter out the machines that would not respond to the scan (due to firewall permissions), the connection between the source and the IP’s in the new sublist are evaluated. This will examine whether or not the two machines can communicate with each other and if so it is added to the final list to be returned to the user.
**Data:** Range of IP addresses

**Result:** List of pinged machines

```python
sortedIPList = sorted list of IP’s from VT;
pingedList = sortedList.subList(range);
if pingedList empty then return ∅;
for each ip in pingedList do
    if Source machine is external then
        if ip.isExternallyVisible == false then Remove ip from pingedList;
        else
            if ip.canCommunicate(source) == false then Remove ip from pingedList;
        end
    end
return pingedList
```

**Algorithm 1:** Base pseudocode for a NMap Ping Scan implementation

As mentioned before the port scan is similar to the ping scan and only reveals the ports open in the system. The TCP scan is an extension of this as the basic port scan is used as seen in Algorithm 2. The logic for the TCP scan is also simple as a basic port scan is first performed over a range, if and only if the source can communicate with the target. Then for each of the ports that are open, the services that are associated on those ports are then added to the return list. This returns a mapping between the open ports and the services that respond on those ports which then get added into the knowledge base.

**Data:** A target machine and range of ports

**Result:** List of services on target

```python
if Source is external and target is not externally visible then
    return ∅
else
    if target.canCommunicate(source) == false then return ∅;
end
Perform port scan;
for each Open port do
    if Port is open on target then Record the services responding on that port;
end
return list of services
```

**Algorithm 2:** Base pseudocode for a NMap TCP connect port scan to reveal the services on the target

The thorough scan runs each of these methods to return one single data structure that represents all the open ports and services for a range of IP addresses. This yields the most information but takes a long time to run depending on the ranges and is also the most detectible by intrusion detection systems.