Towards Characterizing Adversarial Opportunity and Behavior in Critical Infrastructure Network

Md Sharif Ullah

Old Dominion University, mulla001@odu.edu

Follow this and additional works at: https://digitalcommons.odu.edu/ece_etds

Part of the Communication Commons, and the Computer Engineering Commons

Recommended Citation
https://digitalcommons.odu.edu/ece_etds/229

This Dissertation is brought to you for free and open access by the Electrical & Computer Engineering at ODU Digital Commons. It has been accepted for inclusion in Electrical & Computer Engineering Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.
TOWARDS CHARACTERIZING ADVERSARIAL OPPORTUNITY AND BEHAVIOR IN CRITICAL INFRASTRUCTURE NETWORK

by

Md Sharif Ullah
B.Sc. in 2013, University of Dhaka
M.Sc. in 2014, University of Dhaka

A Dissertation Submitted to the Faculty of Old Dominion University in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

ELECTRICAL & COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY
December 2021

Approved by:

Hongyi Wu (Director)
Sachin Shetty (Codirector)
Chunsheng Xin (Member)
Amin Hassanzadeh (Member)
The growth of information and communication technology is constantly revolutionizing various domains, e.g., energy utility systems, healthcare, the internet of things, etc. This inception of widespread cyber technology enables reliability and first operability of the system, yet simultaneously imposed a risk of significant impact due to disruption of safe and secure operation. The attack surface is expanding, creating a cyber exposure gap which indicates a higher threat landscape and increased risk of compromise. Motivated by this increased threat exposure, this dissertation investigates the attack surface as a static and dynamic indicator of adversary propagation, seeking an efficient modeling paradigm to initiate threat-informed defense. Our work explores multi-stage attack propagation within three different aspects such as opportunity, capability, and intent to characterize an attack successfully.

First, we proposed three criticality metrics for each host based on the opportunity it provides to facilitate cyber attacks. These metrics represent diverse interactions between attackers’ system components through social attack vectors, topological connectivity, and information flow dependency in the network. Following this analysis, we proposed an intrusion response system by considering the diverse strategy attackers could employ in their lateral movement within the target environment. The local alert information corresponding to the compromised host helps defenders to understand the security state and underlying plan of the attacker.

Next, we addressed the challenge of understanding the attacker’s capability and knowledge evolve throughout the attack surface for realistic threat modeling. A cyber threat analysis framework has been proposed based on characterizing adversarial behavior in a multi-stage cyber attack process. The framework extracts logical dependencies between stepping stones by leveraging technical indicator in the attack surface prior to enriching from multiple threat intelligence sources. The model reveals the meaningful insight of the attack phase in terms of the local and global threat landscape. Our analysis results in a novel path hardness metric that is leveraged to enumerate the risk posture of an agile security platform.

Finally, We proposed PatternMiner to mine threat intelligence reports for unified pattern
identification. These reports contain inclusive detailed information of the campaign but in an unstructured way. To classify and extract meaningful information from campaign reports, *PatternMiner* employs a two-fold framework. First, we design a multi-label ML classifier to identify TTP in campaign reports. Although the MITRE database is developed based on threat reports and expert knowledge, it is not possible to include all attributes in it. In the same way, security practitioners don’t often follow straight TTP keywords from the ATT&CK database. Thus we proposed a neural network architecture to capture these unknown behavioral artifacts in the cyber campaign. We call it Pattern Entity Recognition (PER), a framework that models the task of collecting adversarial attack pattern entities as a task of sequence labeling of natural language processing. By applying a sequence labeling model, each token in an unstructured campaign report is assigned with a label, and tokens assigned with attack patterns are then collected accordingly. With a two-fold structure, *PatternMiner* effectively capture known and unknown attack patterns from unstructured threat intel data.
ACKNOWLEDGEMENTS

I am thankful to many people for their support throughout this long Ph.D. journey. First and foremost, I am very grateful to have Dr. Sachin Shetty as my PhD advisor. He taught me the correct attitude to research, how to do good research in terms of reading, writing, presentation skill, and more importantly, how to think independently as a researcher. In addition, he also influences me a lot in my life. All of the things he endowed me and I have learned from him can not be simply stated in a few sentences. He is so positive and optimistic about life and is friendly to everyone. I am honored to have such a humble and knowledgeable person as my Ph.D. advisor.

I would like to thank my committee members Dr. Hongyi Wu and Dr. ChunSheng Xin for their valuable comments and time. I would like to express my sincere gratitude to Dr. Amin Hassanzadeh (Cyber Fusion Center, Accenture) for countless discussion and ideas as it sometimes helps me to get away the pressure of different research problems. My special thanks to my lab-mates for their support in making the lab a pleasant and interactive environment. I am thankful to all friends and families in the Bangladeshi community for extending their helping hands during crucial time in my Ph.D. journey and making my life enjoyable in Norfolk. I also wish to recognize the assistance and support of the Virginia Modeling, Analysis, and Simulation Center (VMASC) at Old Dominion University.

Last, but not least, I wish to acknowledge the support and great love of my parents. They always believed in me.
# TABLE OF CONTENTS

| LIST OF TABLES                          | viii |
| LIST OF FIGURES                        | ix   |
| Chapter                                |      |
| 1. INTRODUCTION                        | 1    |
| 1.1 ATTACK PROGRESSION ANALYSIS        | 3    |
| 1.2 PROPOSED APPROACH AND CONTRIBUTION: | 4    |
| 2. BACKGROUND                          | 9    |
| 2.1 ATTACK GRAPH:                      | 9    |
| 2.2 BAYESIAN NETWORK:                  | 11   |
| 3. MODELING ATTACKERS OPPORTUNITY FOR IMPROVING CYBER RESILIENCE IN ENERGY DELIVERY SYSTEMS | 13   |
| 3.1 PRELIMINARIES                      | 13   |
| 3.2 TOPOLOGICAL CONNECTIVITY BASED CRITICALITY METRIC (TCCM) | 15   |
| 3.3 SOCIAL VULNERABILITY BASED CRITICALITY METRIC (SVCM) | 18   |
| 3.4 INFECTION PROPAGATION BASED CRITICALITY METRIC (IPCM) | 21   |
| 3.5 SYSTEM DESIGN                      | 26   |
| 3.6 PERFORMANCE EVALUATION             | 28   |
| 3.7 DISCUSSION                         | 32   |
| 3.8 CONCLUSION                         | 33   |
| 4. ON THE EFFECTIVENESS OF INTRUSION RESPONSE SYSTEMS AGAINST PERSISTENT THREATS | 34   |
| 4.1 OVERVIEW OF THE APPROACH           | 34   |
| 4.2 ATTACK OPPORTUNITY ANALYSIS        | 36   |
| 4.3 INTRUSION RESPONSE                 | 38   |
| 4.4 IMPLEMENTATION AND RESULTS         | 41   |
| 4.5 CONCLUSION                         | 44   |
| 5. CYBER THREAT ANALYSIS BASED ON CHARACTERIZING ADVERSARIAL BEHAVIOR FOR ENERGY DELIVERY SYSTEM | 46   |
| 5.1 OVERVIEW OF THREAT INDICATORS      | 46   |
| 5.2 FRAMEWORK                          | 48   |
| 5.3 PATH COMPLEXITY AND EFFORT ESTIMATION | 52   |
| 5.4 IMPLEMENTATION AND RESULTS         | 56   |
| 5.5 CONCLUSION                         | 62   |
6. PATTERNMINER: TOWARDS AUTOMATED LEARNING AND ANALYSIS OF ADVERSARIAL ATTACK PATTERN FROM MALICIOUS CYBER CAMPAIGN

6.1 ATTACK PROGRESSION ANALYSIS AND DEFENSE STRATEGY

6.2 PRELIMINARIES:

6.3 TOOL ARCHITECTURE

6.4 EXPERIMENTS AND RESULTS

6.5 CONCLUSION AND FUTURE WORK

7. CONCLUSIONS AND FUTURE RESEARCH

BIBLIOGRAPHY

VITA
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Classification parameters for AVs</td>
<td>19</td>
</tr>
<tr>
<td>2. System calls with Events</td>
<td>28</td>
</tr>
<tr>
<td>3. Path Criticality Estimation</td>
<td>44</td>
</tr>
<tr>
<td>4. Vulnerable component risk</td>
<td>53</td>
</tr>
<tr>
<td>5. Examples of multi-word Expressions from our dataset</td>
<td>72</td>
</tr>
<tr>
<td>6. Dependency based context extraction</td>
<td>74</td>
</tr>
<tr>
<td>7. Pattern Entity to be Extracted from Campaign Data</td>
<td>75</td>
</tr>
<tr>
<td>8. Interaction Functions</td>
<td>79</td>
</tr>
<tr>
<td>9. Performance comparison of TTP extraction with BoW and Sentence Embedding(300d)</td>
<td>82</td>
</tr>
<tr>
<td>10. Performance comparison of TTP extraction with different word embedding dimension</td>
<td>83</td>
</tr>
<tr>
<td>11. Experimental Hyper-parameters</td>
<td>85</td>
</tr>
<tr>
<td>12. Performance comparison of PER with different hyper-parameter for flat corpora</td>
<td>85</td>
</tr>
<tr>
<td>13. Performance comparison with different word embedding and LSTM unit dimensions for flat corpora</td>
<td>86</td>
</tr>
<tr>
<td>14. Performance comparison with different word embedding for nested corpora</td>
<td>86</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Simple Bayesian Attack Graph illustrating Probability Computations [1]</td>
<td>12</td>
</tr>
<tr>
<td>2.</td>
<td>An example network attack scenario</td>
<td>14</td>
</tr>
<tr>
<td>3.</td>
<td>Object instance graph with system call log in time-order</td>
<td>22</td>
</tr>
<tr>
<td>4.</td>
<td>Infection propagation model</td>
<td>24</td>
</tr>
<tr>
<td>5.</td>
<td>System model</td>
<td>27</td>
</tr>
<tr>
<td>6.</td>
<td>Network diversity in terms of social attack vectors</td>
<td>29</td>
</tr>
<tr>
<td>7.</td>
<td>Standard deviation of TCCM due to multiple initial attack points (a) for each host (b) for hosts within layer</td>
<td>30</td>
</tr>
<tr>
<td>8.</td>
<td>Evaluation results for (a) Standard deviation of TCCM due to OS, vulnerable service and isolation pattern (b) risk of the network for particular attack points</td>
<td>31</td>
</tr>
<tr>
<td>9.</td>
<td>Infection propagation based CM for hosts</td>
<td>32</td>
</tr>
<tr>
<td>10.</td>
<td>An illustration of our threat model</td>
<td>35</td>
</tr>
<tr>
<td>11.</td>
<td>System Model</td>
<td>36</td>
</tr>
<tr>
<td>12.</td>
<td>Stealthy links between hosts</td>
<td>37</td>
</tr>
<tr>
<td>13.</td>
<td>Logical view of ICS testbed</td>
<td>42</td>
</tr>
<tr>
<td>14.</td>
<td>Attackers' evolving characteristic parameters in penetration steps</td>
<td>43</td>
</tr>
<tr>
<td>15.</td>
<td>Performance evaluation of IRS with regard to the attack propagation delay.</td>
<td>44</td>
</tr>
<tr>
<td>17.</td>
<td>Proposed cyber threat assessment framework</td>
<td>48</td>
</tr>
<tr>
<td>18.</td>
<td>A sample attack graph represented in Neo4j graph format</td>
<td>49</td>
</tr>
<tr>
<td>19.</td>
<td>ICS Test-bed</td>
<td>58</td>
</tr>
<tr>
<td>20.</td>
<td>Deviation of path hardness by effort correlation</td>
<td>60</td>
</tr>
<tr>
<td>21.</td>
<td>Distribution of attack paths within different hardness level</td>
<td>61</td>
</tr>
<tr>
<td>24.</td>
<td>(a) Techniques and (b)Tactics, exploited in the wild w.r.t the threat groups listed in the MITRE ATT&amp;CK [5] database</td>
<td>68</td>
</tr>
<tr>
<td>25.</td>
<td>Dependency of words in a sentence</td>
<td>70</td>
</tr>
<tr>
<td>26.</td>
<td>Adversarial Attack Pattern Learning pipeline</td>
<td>71</td>
</tr>
<tr>
<td>27.</td>
<td>Preprocessing States for PatternMiner</td>
<td>71</td>
</tr>
<tr>
<td>28.</td>
<td>Framework for Technique and Tactic Identification from Cyber Campaign data</td>
<td>72</td>
</tr>
<tr>
<td>29.</td>
<td>Example of adversarial pattern entity in threat intel data</td>
<td>75</td>
</tr>
<tr>
<td>30.</td>
<td>Character embedding for the word ‘TOKEN’</td>
<td>76</td>
</tr>
<tr>
<td>31.</td>
<td>Neural Network Architecture for Pattern Entity Recognition</td>
<td>78</td>
</tr>
<tr>
<td>32.</td>
<td>ATT&amp;CK data in STIX format</td>
<td>80</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

Today’s Critical Infrastructures (CIs), incorporated with Information and Communication Technology (ICT) form the lifeline of modern society. A variety of communication networks are interconnected for the purpose of real time monitoring, advanced sensing, analysis and control. The cyber integration with CI enables high reliability and fast operability of the system, simultaneously imposed a risk of significant impact due to disruption of safe and secure operation [6] [7]. Detection, prioritization, and mitigation of a cyber systems failure, unlike a physical system’s failure, is a very complex problem in CIs security.

The cyber resilience analysis of CIs is quite challenging, because their large scale, heterogeneity and interdependence make very hard to protect them adversarial cyber events [8]. While cybersecurity is mainly focused on reducing the risk of critical infrastructure by defense cyber measures to intrusion or attacks, the resilience includes the ability to withstand and recover from deliberate attacks, threats or incidents [9]. Insightful studies analyse resilience by quantifying system’s ability to 1) plan for adverse events 2) absorb 3) recover and 4) predict and prepare for future attacks in order to adapt to the potential threats [10]. There are four distinct factors for cyber resilience analytics: robustness, rapidity, resourcefulness and redundancy. In this work, we mainly focus on enhancing the robustness of the cyber network system prior understanding the attack pattern which enables a rapid response to hinder future intrusions.

All of the resilience factors discussed above aims to maintain the critical functionality of the system amid adversarial events. Most of the existing research assesses resilience, often focusing on a network comprised of homogeneous elements e.g., component, device, functionality etc. [11]. This analysis doesn’t cope with a real scenario where different elements hold different weights in terms of critical functionality. For example, a failure of a programmable logic controller (PLC) can cause more acute and far-reaching effects than a sensor and actuator [12]. A sensor or actuator generally responsible for the small part of the processing system where-as PLC interacts with multiple sensors and actuators throughout the system.
In order to make a system’s critical functionality more robust, Industrial Control System (ICS) network architecture must be compliant with ISA-99/IEC 62443 reference architecture [13] [14]. The critical targets are often not situated at the perimeter but are usually segregated and deployed with diverse security safeguards that make it challenging for the attacker to gain direct access to them. Despite this fact, it is true that recently executed more stealthy attacks took advantage of this architecture. Due to the potentially severe damage and cost of a nation, CI has been targeted intentionally suffered a significant loss. Determined attackers move laterally through the network zones, using them as stepping stones until they reach their target. For example, in 2015, in the Ukrainian power grid attack, termed ‘BlackEnergy,’ attackers followed a particular attack pattern [15]. In this incident, they first used spear-phishing emails to gain a foothold in the network and then used VPN and remote access tools to move laterally within the target network. Consequently, the attacker gained access to a Human-Machine Interface (HMI) and conducted a service outage. In 2016, CrashOverride, the first-ever designed and deployed malware for electrical grids, targeted the library and configuration files of an HMI to learn the operational processes and disrupt the OT system [16]. Similarly, well-known Stuxnet malware, executed by removable drives, moved node to node through the Operational Technology (OT) zones to sabotage the system facility by reprogramming the Programmable Logic Controllers (PLC) [17].

Thus, it is proven that defense-in-depth architecture is not enough for resilient critical infrastructure. Yet, adversaries need to propagate a long span of attack surface to fulfill their goal. The attack surface is defined as the “subset of the system’s resources that can be potentially used by an attacker to launch an attack” [18]. A larger attack surface indicates a higher cyber threat landscape and increased risk of compromise. Researchers have been working for a while in different adaptive cyber defense concepts such as Moving Target Defense (MTD) [19] and honeypots [20]. These efforts attempt to reduce and dynamically modulate the attack surface of the system by reconfiguring it in order to cause more uncertainty for the attacker. One of the major drawbacks of these approaches is it may introduce costly overhead for legitimate users as well as the potential for denial of service conditions. On the other hand, many CIs like Industrial Control System (ICS) use legacy technology and devices. It is fairly hard to periodically reconfigure the system and deploy decoys because
of high overhead, maintenance complexity, and high probability of the disruption of the service [21]. Moreover, the scale of the attack surface happens to be huge, essentially imposing high costs with these methods. Additionally, most of the existing work has many hypotheses about the attacker’s penetration without a detailed view of the attack surface. They also do not adequately consider the attacker’s behavior. In reality, there is asymmetric information of attack surface between defender and attacker imposing a big challenge for security. CI cyberattack surface is huge at this time, making it impossible for optimum defense without understanding the attack life cycle.

1.1 ATTACK PROGRESSION ANALYSIS

Attack life cycle can be modeled into multiple steps (also called attack phases [22] [23]. We adopt the definition of a cyber threat proposed in [24] and identify three aspects of each attack step such as opportunity, capability and intent in order to characterize an attack successfully. Furthermore, we analyze attack progression from the attacker’s and defender’s perspectives by considering these three aspects. It should be noted that despite having separate views, the defender needs to be able to determine the three aspects in order to choose an appropriate defense strategy. Consider the opportunity represents the first aspect of an attack: who is vulnerable and what can be done to their networks? What are the different ways of moving from the perimeter points throughout the network to reach to a target? Answers to these questions will lead us to identify the attack opportunity in the network. After understanding the available options for adversarial actions, the penetration relies on the adversary’s capability to perform those actions; what tools will be used, and how much effort is needed for the attack? The intent is another important aspect of an attack that influences the decision of the attacker; what assets will be targeted and what damage will be caused? These aspects are different in the attacker’s perspective versus the defender’s perspective and will be discussed below.

1.1.1 DEFENDERS PERSPECTIVE:

To examine attack surface efficiently, Attack Graph (AG) has been used for a while from quantifying the vulnerability impact to optimize the prioritized defense in security modeling
and analysis [25–27]. Attack graph helps defenders to model the opportunity in a given network. It provides the ability to grasp potential exploitation of vulnerabilities along the attack path and their respective consequences in the particular context. AG can identify the steps an attacker can take to penetrate the network, and this provides the defender with the ability to track the attacker’s options and capabilities required. The defender can do that by analyzing the technical exposure through paths. The defender assumes that during each step, the attacker can accumulate knowledge of the network given their own skill and progress further within the network. The defender can utilize the AG analysis techniques to identify the most probable paths that require similar capabilities. Defenders can conduct an end-to-end analysis by mapping different attack’s intents to the mission-critical targets. They only focus on the exposures helping attackers to those goals.

1.1.2 ATTACKER’S PERSPECTIVE:

The attacker can also benefit from AG; however, unlike defender, the attacker doesn’t have the global view of the network. Every successful attack step will expand the attacker’s view and provides more options and opportunities. The capability is defined by the potential exposure each step can provide. More specifically, it depends on two uncertain factors: intrinsic knowledge and evolving skill. For example, in dumping credential technique, the adversary will obtain all cached credentials that could aid in moving laterally throughout the network. But in practice, the adversary might have limited prior knowledge about those credentials. This phenomenon represents the uncertainty in characterizing the skill gain of attackers as they laterally move within the network and proceed towards final targets. Furthermore, attacker’s intent during the propagation is influenced by diverse motivations and constraints. An adversary may be persistent or goal-oriented (e.g., cyber espionage) whose only motivation is to reach the final target and cause severe damages. Sophisticated adversaries might modify exploits to avoid detection. Constraint (e.g. time, resource) bounded adversaries emphasize their sub-goals while choosing action in each step. These kinds of attacks are typically generated by pre-engineered tools.

1.2 PROPOSED APPROACH AND CONTRIBUTION:
Our analytics starts with investigating the attack surface through the defender’s perspective in terms of different aspects related to the attacker’s progression within the network. Defense-in-depth architecture forces attackers to conduct lateral propagation until the target is compromised. There is a need to conduct susceptibility analysis of intermediate nodes along the path to the target. The criticality of the nodes based on risk graph has been proposed by [28]. [29] defined assets as the attackers’ privilege and vulnerabilities and investigate their criticality in terms of dependency among them. Previous works on host criticality mainly depend on nodes connectivity, keeping many other attributes remain unaddressed.

In this dissertation, we investigate the criticality of the host based on the opportunity it provides to facilitate attack to the target. As criticality has a different meaning to different system players, a unique host metric can’t consider all these factors. Furthermore, these factors aren’t chosen/prioritized equally by the attackers and defenders. We proposed three criticality metrics for each host along the path to the target. The first metric refers to the opportunity of attackers before they penetrate the infrastructure. The second metric measure the opportunity a host provides by allowing attackers to propagate through the network. Along with vulnerability, we also take into account the attributes of hosts and links within each path. Then, we derive a third criticality metric from reflecting the information flow dependency from each host to the target. Each of these metrics provides the equal opportunity of attacker perceived by the defender.

Lateral propagation, often conducted by more sophisticated threats, also known as Advanced Persistent Threat (APT). In an APT scenario, after penetration (i.e., post-compromise phases in Mitre ATT&CK [30]), the adversary moves from host to host by exploiting vulnerabilities and leveraging legitimate functionality, and exploring more until he reaches to a critical target. An efficient situational awareness is needed to restrict an APT from penetrating deeper into the network, inflicting more damage. The research community has focused on devising Intrusion Response Systems (IRS) in the context of attack propagation in the network. In [31], the authors use an attack-response tree to analyze different security states of the network, based on the security alerts and their measurement uncertainty, model the response actions as a sequential stochastic game. Some of the prior work also focuses on modeling the defense decision by Q-learning [32] or a partially observable
Markov decision process [33] to essentially identify the most vulnerable paths and the critical future impacts an attacker can cause. On the other hand, Hasan et al. [34] orchestrated intrusion response by the practical application of software-defined networking (SDN) in EDS. In [35], the authors evaluated different IRS planning algorithms in terms of overhead and scalability.

Prior work only considers attack penetration by acquiring different hosts’ information, exploitable vulnerabilities, how they affect the probability of attack success, leaving aside other aspects of an attack strategy. As persistent attackers systematically probe the target network, static cyber defense, such as patching vulnerabilities or specific application security, will not provide an effective intrusion response. Furthermore, today’s network architectures are more heterogeneous based on implementation, integration, as well as devices with diverse operating systems, vendors, firmware, and service operations. A comprehensive understanding of network architecture, incorporated with the understanding of how an attacker could infiltrate the network, provides a way to identify different features of already compromised nodes and potential stepping stones for a particular threat. Security analysts can leverage this information to investigate the attackers’ capability in likely attack progression and interpret it into a response decision. In the second part of this dissertation, we model an intrusion response system by considering the diverse strategy an attacker could employ in his lateral movement within the target environment.

In practice, attacks are very diverse from one another in terms of types, propagation, and consequences. System administrator requires comprehensive analytics to assess defense against different cyberattack campaign. In almost all cases mentioned above, attacks are performed in multiple stages. Detecting a single intrusion doesn’t necessarily indicate the end of the attack, as the attack could have progressed far deeper into the network. Hence, individual attack footprint seems insignificant in an isolated manner since it is usually part of a more complex multi-step attack. It takes a sequence of steps to form an attack path towards a target in the network. Researchers have investigated several attack path analysis methods for identifying an attacker’s required effort, e.g., the number of paths to a target and the cost and time required to compromise each path, to estimate the level of risk in CI diligently.
The aforementioned path centric metrics and threat analysis methods do not consider attack complexity in an interconnected cyber infrastructure with a variety of attack paths which makes it impractical for real attack scenarios. More specifically, most of the analysis methods only consider the topological connection between stepping stones to measure the difficulty of reaching to a target. In addition, they only assume some predefined attacker’s skill set to estimate the path complexity. In reality, attacker’s capabilities and knowledge evolve throughout attack paths to the target. Thus, it is of paramount importance to perform additional inspections in order to identify adversary’s new opportunities obtained through an attack path. In this part of the dissertation, we proposed a methodology to assess cyber threats proactively by characterizing adversary behavior.

Modern cyber attacks often consist of a series of steps and generally part of a larger campaign. Large-scale field data includes low-level indicator of compromises (IoC) provides a quantitative measurement of a campaign. Commercial security products often rely on these IoCs for detecting security threats, forensic investigations, and attack attributions. As low-level IoCs are often susceptible to change, security tools based on IoC detection may not provide adequate defense against malicious campaigns. Security Operation Center (SOC) requires timely and relevant threat intelligence information, which needs to be maintained properly and accurately to secure cyberinfrastructure. These analyses, often done manually, results result in a tedious and extensive task by analyzing a large amount of irrelevant information an analyst needs to go through. On the other hand, security professionals identify and report qualitative campaign characteristics manually. These reports connect the dots between multiple attack identifiers and incorporate the semantics of corresponding campaigns. Extracting meaningful campaign information could help SOC to design its strategy proactively against cyber attacks.

To overcome both challenges discussed above, in the last part of the dissertation we presents a threat intelligence tool called PatternMiner that employs machine learning (ML), deep learning (DL) and natural language processing (NLP) architectures to process data stream, identifying relevant technique-tactic information and extracting adversarial pattern entities, thus developing valuable knowledge base for security analyst. We proposed a complete end-to-end architecture with no requirement of feature engineering or any other
additional component in the framework. Our approach focuses on high-level IoC such as techniques, tactics, and procedures (TTP), software tools, malware, etc, to understand attack patterns. Threat intelligence reports analyze the strategies employed by the campaign but failed to identify the structured pattern. The first-ever PER corpus was manually annotated with 6 types of adversarial pattern entities, which could act as a benchmark for different NLP tasks on threat intel data. With an efficient learning pipeline, PatternMiner effectively capture known and unknown attack patterns from unstructured threat intel data.
CHAPTER 2

BACKGROUND

In this chapter, we briefly discuss the background of different models and technology leveraged in our approach and methodology.

2.1 ATTACK GRAPH:

An attack is not an isolated event but a set of actions logically connected. An AG is the representation of a logical aspect of attack; more specifically, it is the mathematical abstraction of details of potential attacks leading to specific targets. In our case, the target is modeled as a critical service or assets whose unauthorized modification could result in an operational interruption. The term attack graph first introduced in 1998 by Phillips and Swiler [36, 37] and since then has been used in different domains. The node in the attack graph represents the probable states of a system during an attack, while the edge represents the corresponding changes of conditions due to the attacker’s action.

An attack graph is very useful for representing vulnerabilities and exploits. Also, they can be utilized for analysis such as path number enumeration and computing the loss and likelihood of successful attacks. Initially, generating AG is constructing the graph with several networks and host information, including vulnerability, connectivity, service etc. The researcher has been using extensively different AG models in different dimensions of security problems. The different graph model is constructed with various system parameters or behaviors based on the application it’s applied to. Mehta et al. [38] generate AG by leveraging Google’s PageRank algorithm to rank multi-stage cyber attack against a cyber network. AG is also very helpful in producing different quantitative metrics to enumerate the probability an attack occurs in a given network [27, 39]. The methods combine the causal relationship between vulnerability encoded in the attack graph. Social engineering exploits have been integrated with technical vulnerabilities into AG to do a quantitative analysis of a comprehensive defense strategy [40]. The AG could be used for hardening the network by examining
all hosts in the network and the relationship between them. For instance, security and network administrator can inspect how many ways a host could be compromised so they can act upon it and impose countermeasures [41]. In addition, potential intrusion can be easily detected by matching attack steps with real-time alerts. Many different representations have been proposed for AG, where these can be classified into two types.

2.1.1 STATE ENUMERATION ATTACK GRAPH

The method of model checking techniques has been first introduced by Ritchey, and Amman [42] to model the vulnerability of computer networks. Sheyner et al. [43] uses model checking to compute multi-host, multi-stage AG where each node represents the entire network state and the arcs represent state transition caused by attacker’s actions. This type is called state enumeration attack graph or also scenario graph. This state enumeration graph shows all possible attack paths to a particular goal attacker state. Each state is included with information on components of the network vulnerability model. The privilege level of the attacker on a host is also represented in each node. This type of AG suffers from a serious scalability problem commonly known as the state explosion problem. Thus, AG using this approach, faced redundant attack paths imposing lots of computational complexity [44].

2.1.2 DEPENDENCY ATTACK GRAPH

Given the scalability issues with the model checking approach to generate state enumeration AG, Amman et al. [45] proposed a more efficient representation of AG called exploit dependency attack graph. The node represents state conditions of system settings, where the edge represents a causal relationship between those conditions. The number of nodes in the exploit dependency AG scales linearly to the number of vulnerabilities in the network, in contrast, to exponentially as in the network state AG. Numerous tools have been developed to generate the exploit dependency AG for large networks.

Topological Vulnerability Analysis (TVA), an AG model, consists of two types of nodes, exploits nodes and security condition nodes [46, 47]. This approach models attackers as the transition between security conditions and computes combinations of atomic exploits that lead to given network resources. The model builds the dependency attack graph in
a two-step process. First, having the set of all exploits used by the attacker, the attacks graph starts with an initial condition exploit that moves from state to state by matching pre-condition and post-condition. Then, the model builds the forward dependency graph backward to contain all relevant exploits from the attacker’s goal to the starting point. Ou et al [48] presented an automated AG generation tool by adopting MulVal, a network security analyzer on networks with thousands of nodes. We use this form in our model, where for each vertex, instead of encoding all conditions in the network it encodes a single asset of the network. This characteristic allows it to scale well with the size of the network and makes it feasible for our application. The MulVal reasoning is done using XSB, a Prolog system evaluating the Datalog interaction rules on the facts given the system input. The generated graph is called logical AG, having two types of nodes called derivation and fact nodes. The details of the configuration with the expected attacker goal is used with related cost/benefit to construct the logical AG. Ingols et al. [41] proposed multi-prerequisite graph tool NetSPA that scales nearly linearly with the number of hosts in the network. NetSPA constructs AG with breadth-first technique and uses a method to model reachability of a network using typical information of hosts, IP address, open port numbers, and protocol.

### 2.2 BAYESIAN NETWORK:

A Bayesian network (BN) is a probabilistic graphical model representing the variables and the relationship between them. This type of network is often called a belief network or causal network originally adopted from artificial intelligence. A Bayesian network is usually represented as a directed acyclic graph (DAG) where nodes represent events or objects and are associated with probabilistic variables, and the edge represents causal dependency between nodes [49]. Different mathematical algorithms that have been developed for BN are suited to solve probabilistic questions on DAG structures. According to the Qin and Lee, BN “use probabilistic inference techniques to evaluate the likelihood of attack goals and predict potential upcoming attacks” [50].

A Bayesian attack graph (BAG) is an attack graph in the form of a Bayesian network [51]. An example of Bayesian AG is shown in Figure 1. The BN models the activity of an “remote attacker” (D), who is likely to use of the buffer overflow exploits (B, C) to get access to
Figure 1. Simple Bayesian Attack Graph illustrating Probability Computations [1]

a “root/FTP server” (C). The values on the edges reflect the probability of success of the associated vulnerability exploitation. The prior probability \( Pr(D)=0.7 \) has been assigned to the external attribute \( D \). This probability represents the administrator’s subjective belief of the chances of a remote attack.

The Bayesian network takes great advantage to analyze security under uncertainty. BN allows using efficient inference algorithms that compute a single query (variable elimination), as well as multiple queries (bucket-tree elimination), making it of great value since querying general graphs is an NP-hard problem [52,53]. BN is also able to update the model while new information is available and compute the new posterior distribution. Numerous efforts have involved attack prediction using BN [54], as well as real-time alert correlation citeramaki2015real. In offline mode, BAG is constructed from low-level alerts; on the other hand, in offline, the most likely step of the attacker is predicted. Different tools have been developed to implement BN, such as Graphical Network Interface (GeNIE) and BNlearn [55] for structural modeling and statistical learning. BN is also used to merge information from expert knowledge into information stored into a database into one graph [56].
CHAPTER 3

MODELING ATTACKERS OPPORTUNITY FOR IMPROVING CYBER RESILIENCE IN ENERGY DELIVERY SYSTEMS

In this paper, we model the attacker’s opportunity by developing criticality metrics for intermediate hosts to target. We model the opportunity into two categories: *pre-opportunity* and *post-opportunity*. The first category deals with how a host exposes opportunity to attackers prior to the penetration. We developed social attack vectors and used user interaction and diversity to develop the resultant metric. The second category refers to attackers’ opportunity after accessing the system. We proposed two metrics in this case. The first metric captures how the topological connectivity is exploited to propagate an attack in the network. We incorporate *effort-betweeness matrix* and *similarity-index* to estimate the path cost in this model. The second criticality metric captures information flow dependency between each host and the target. We capture the time-dynamic dependency of objects and investigate how a corrupted file initiates malicious propagation through the network. This metric also computes the probability of minimum and maximum damage a host could cause to the target within a specific time window.

3.1 PRELIMINARIES

Figure 2 illustrates an attack scenario in EDS multilayer network protected with three firewalls separating each layer that takes multiple hops to reach final targets, i.e., Remote Terminal Units (RTU) and (PLC). In this network, we would like to model the opportunities provided by intermediate hosts along an attack path to the target. The term *opportunity* can be categorized as *attack propagation*, *attack origin* and *damage propagation*.

3.1.1 USE CASES

*Use case 1 (Attack propagation):* To make our discussion more concrete, we will refer to the running attack scenario shown in Figure 2 from now on. Every compromised host of
control system has distinct impact in the EDS network, in both direct and indirect ways. Attackers residing in the internet or any host in the network can take different attack paths by escalating a series of malicious actions. Each intermediate host acts as a gateway to propagate these attacks to target hosts. It needs to find out which host could be leveraged as the most efficient gateway. This measurement is not so straightforward as ICS network is very large, having an extensive amount of network components. In Figure 2, it is shown that an attacker was residing in workstation 2 moves to HMI and then to RTU through vulnerability exploitation. There are several possible paths from the same host which has to pass through many other hosts to reach the target, but they don’t provide the same opportunity for the attacker.

**Use case 2 (Attack origin):** Apart from the external network shown in the previous use case, the cyber attack could originate from inside the network. The host can be infected with malware via email attachment, visiting a malicious website or via an infected USB stick or through stolen credentials. This infection plug a hole in the overall attack graph. Suppose an attacker gains access to the network by compromising a host through a spear-phishing attack. Compared to the attacker staying in the internet, this strategy enables the attackers to bypass lots of efforts through the target. Intrinsic system opportunity associated with human interaction should be analyzed for this type of attack.

**Use case 3 (Damage propagation):** According to NERC, “critical asset” is defined as “facilities, systems and equipment which, if destroyed, degraded or otherwise rendered unavailable, would affect the reliability or operability of Bulk Electric system” [57]. In a practical situation, it is very difficult to access inside the control system. So along with path
traversal difficulty, an attacker would take a different attack strategy; compromise a host containing information and data associated with reliable operation of EDS. Depending on the attacker’s privilege and characteristics of the compromised host, processes and files are exposed or modified, which could propagate to other hosts and might alter the operation of the system. Suppose an attack on the client access server is blocked or the attacker is unable to propagate his attack, then the damage on this host, which might be a malicious file, would have been propagated with different probabilities to target hosts. It is clear that not only a compromised host is used as a stepping stone for traversing to the target node, but also the damage that was already done on that one might affect the operation of the end system. Obviously, this phenomenon should be taken into account to measure host criticality.

3.1.2 OBJECT DEPENDENCY

OS-level objects are categorized within different types such as files, processes, filenames, socket, etc. A file object includes any data or metadata specific to that file, identified by the device, inode number and version number. The process, on the other side, is identified by process ID and version number [58]. A dependency relationship is specified by three parts: source object, sink object, and time interval. The time interval is used to reduce false dependencies. The dependency can be classified based on different events: process-process, process-file, process-filename, and process-socket. Process-process dependency occurs when one process directly affects the other process. One process can affect others directly by creating it, sharing memory with it, signaling it, etc., and indirectly by writing or reading files. A process affects or is affected by data or attributes associated with the file. The direction of dependency depends on the operation. A process depends on the file if it reads the file and the opposite by the write operation. Receiving data from a network socket can also be treated as reading a file but using the interface ID (socket ID) makes it easier to track the process-socket interaction.

3.2 TOPOLOGICAL CONNECTIVITY BASED CRITICALITY METRIC (TCCM)
In this section, we present the development of the Topological connectivity-based criticality metric (TCCM). We first analyze the cyber network topology along with firewall rules to automatically generate the network connectivity map encoding accessibility among different hosts. Each host has ports whose remote access is governed by organization policies and privileges. This connectivity map will be leveraged to generate the attack graph denoted by $G(V, E)$, where each vertex $v \subset V$ represents a host in a network and each edge $e_{ij} \subset E$ represents an attack path. Each edge encodes attackers’ privilege and security conditions, such as host access permissions, open ports, and vulnerabilities. This relation can be incorporated as a precondition of an attack with a tuple $\langle \text{privilege, systemconfiguration} \rangle \equiv \langle p_i, c_j \rangle$ signifying attack between host $i$ to $j$.

For a given EDS network, we model the attack graph $G^i_T(V, E) \subseteq G(V, E)$ as a directed acyclic graph, from a specific origin ($i$) to a set of targets ($T$), with attack paths comprise of exploitables. We model the opportunity each exploitable host provides. In order to measure the criticality metric for each exploitable host, we need to estimate the effort expended by the attacker. We first evaluate vulnerability exploitation by leveraging CVSS [59] exploitability score. Each host has multiple incoming and outgoing edges to and from other hosts. This is because a host containing multiple vulnerabilities could be compromised from other hosts as well as could exploit remote vulnerabilities connected to this host. This scenario lies with one of the major motivations of this metric. Multiple vulnerabilities in a host could share common misconfigurations, thus efficiently narrowing down the defense prioritization for the defender.

Next, we incorporate contextual information to identify the correlated risk, such as, vulnerable service (VS), operating system (OS) and isolation pattern (IP). The CVSS score provides a degree of exploitability irrespective of the environment. An attack path with repeatable vulnerabilities is more attractive than an attack path with only unique vulnerabilities. However, isolation can impact the attacker’s ability to choose these attack paths. The isolation can be achieved by access rules (firewall), authentication (IPSec), and payload inspection (IDS) and is included as ‘isolation pattern’ for our model. We only consider host-specific isolation (IPSec & IDS) but not layer-specific (firewall), which is treated as a unique host in our model. We also investigate the impact of diverse OS within each attack path.
and its effect on our metric. It is shown that the vulnerability of the same software on the
diverse operating system cannot be compromised with the same exploit [60].

An attack path is characterized by OS, isolation pattern and vulnerable service. We en-
closed these parameters in a set $Z$. Let us denote the set of attack path by $P = \{p_1, p_2, \ldots, p_k\}$
where each path consists of a set of hosts denoted by $H_y = \{h_1, h_2, \ldots, h_m\}$. Each path has
$q$ types of instances of a parameter $z \in Z$. For each parameter, we compute the relative
abundance of different instances in an attack path and define it as a similarity index:

\[
S_{index(p_y,z)} = w_z \times r_{p_y,z}
\]

where

\[
r_{p_y,z} = \prod_{j=1}^{q} \frac{m_j}{|H_y|}
\]

Here, $r_{p_y,z}$ is the effective richness of parameter $z$ (IP or OS or VS) instances in attack path
$p_y$ where $m_j$ is the number of instances of type $j$ and $|H_y|$ is the total number of instances
in whole attack path. Not all parameters are equally important, and the differences are
captured in our similarity index by a weight ($w_z$) factor.

This criticality metric also captures the impact of a target being compromised. We
assigned the damage potential of a target host $t$ by $D_t$, which refers to its highest direct
and/indirect operational capacity to the system. For example, a compromised HMI can
cause worse damage to the substation than a compromised data historian. Algorithm 1
shows the details of the formulation of our first criticality metric. It takes the attack graph
$G^i_T$, prior to setting up attack point $i$ and target host set $T$ and returns our first criticality
metric for all intermediate hosts. The algorithm considered all attack paths between $i$ and
set of targets $T$ (lines 1-3). For each attack path, the candidate host is updated, which
belongs to the set of intermediate hosts (line 4). Within a specific path $p_y$ towards target
point $t$, the algorithm calculates the cost of each candidate host $n$. It can be measured by
using the vulnerability exploitability ($x$) of different hops upto that host (lines 5-7). The cost
measurement for each target host residing in the control center is calculated in the same way
(line 8). In this stage, the algorithm identifies the host parameters for this path and compute
similarity index by using equation 1 & 13 (line 9). We modify the betweenness centrality, a
well-known graph centrality concept, to form an effort betweenness matrix \((EB^i_t)\) for target \(t\) (line 10).

**Input:** \(G^i_T(V,E), D^t\)

for all \(teT\) do intermediate host ← vertexes \((V \setminus V_{i,T})\)

for all \(py\) (vertex, edge) \(\epsilon G^i_T\) do

\(n ← \text{candidatehost}\)

\(c^i_{py,n} = \prod_i x_{10}\)

\(E^i_{py,n} = e^{-(c^i_{py,n})}\)

\(C^i_{py} = C^i_{py,n} \text{ compute } S_{\text{index}(py,j)} \text{ from equation (1) \& (2)}\)

construct \(EB^i_1, EB^i_2, \ldots \ldots, EB^i_t\)

repeat \(n \notin\) intermediate host compute \(C^m_{\text{top}} = \sum_{teT} \left( \sum_{py\epsilon G} x_{py,n} \times |e_{py,n}|_t \times \prod_j \frac{1}{1-S_{\text{index}(py,j)}} \right) \times D^t\)

**Definition** The effort-betweenness matrix \((EB^i_t)\) is a matrix of size \(M \times N\) describing the cost of each host \(n\) from source \(i\) to target \(t\) through path \(m\). Every row and column stand for different attack path and hosts in the network respectively. The element \(e_{mn}\) of such matrix specifies the ratio of cumulative cost of the host to the path cost.

For every target, a separate matrix is formed to store relevant information (line 11). Finally, the criticality metric is formulated for every intermediate host (lines 12-15). This equation simultaneously captures the target impact, path-cost, diverse attack paths, host position on the path, etc.

### 3.3 SOCIAL VULNERABILITY BASED CRITICALITY METRIC (SVCM)

We consider all attackers have the same skill set for our analysis, but attackers’ motivation can change in terms of their goal. Apart from that, insider attack is also a very common problem in cyber networks. Social vulnerability needs to be considered along with technical vulnerability. The probability of a social engineering attack can be addressed from two angles. The first one is attack vectors embedded within a host, and the other one is the operator’s behavior and characteristics influenced by different factors leading to exploit the vulnerability willingly or unwillingly. In general, successful exploitation of social vulnerability
depends on two factors: the attacker’s intention/capability and the possibility of action. We focus on the latter in our analysis.

In order to capture the opportunity for insider attacks, attack path analysis is not sufficient. The probability of social engineering attack is captured in the Social Vulnerability based Criticality Metric (SVCM). Successful exploitation of social vulnerability depends on two factors: attacker’s intention/capability and possibility of action. We focus on the latter in our analysis.

We assign scores to hosts based on their susceptibility to social engineering attacks, such as spear phishing, baiting, unauthorized VPN connections, etc. We first classify each attack vector (AV) in terms of the stages it spans by using the model given in [61]. Table 1 shows the alignment of each stage in a hierarchical manner. Every AV is parametrized with a category for each sub-stage given in the right-most column in the table. We mark these categories as classification parameters (l) for our model. It should be noted that for each sub-stage, classification parameters are ensured to be mutually exclusive. Now we give a brief overview of all these stages and parameters below.

**Orchestration** This stage focuses on the targeting, distribution method and automation of a social engineering attack.

### Table 1. Classification parameters for AVs

<table>
<thead>
<tr>
<th>6*Orchestration</th>
<th>2*Target chosen</th>
<th>Explicit target</th>
<th>Promiscuous target</th>
</tr>
</thead>
<tbody>
<tr>
<td>2*Method of Distribution</td>
<td>Local</td>
<td>Remote</td>
<td></td>
</tr>
<tr>
<td>2*Mode of Automation</td>
<td>Manual</td>
<td>Automatic</td>
<td></td>
</tr>
<tr>
<td>3*Exploitation</td>
<td>3*Deception vector</td>
<td>Cosmetic</td>
<td>Behaviour</td>
</tr>
<tr>
<td>4*Execution</td>
<td>2*Attack Persistence</td>
<td>One-off</td>
<td>Continual</td>
</tr>
<tr>
<td>2*Execution step</td>
<td>Single step</td>
<td>Multi-step</td>
<td></td>
</tr>
</tbody>
</table>

...
(l₁ & l₂): These parameters can help to identify the exposure of an attack. In case of explicit targeting, a particular individual or group of people with specific identity or company role, e.g., WiFi phishing attack where promiscuous targeting deals with maximum exposure.

(l₃ & l₄): This refers to the mechanism regarding the arrival of an attack to the target system. Local attacks are distributed within the target user’s local operating system. Remote attacks ordinate from other environments than the victim system.

(l₅ & l₆): This category helps us identify the activation and administration of an attack. A manual attack requires the action of the attacker to manually placing it in the target environment. The other category works in a pre-programmed manner without intervention from the attacker.

**Exploitation** This stage deals with tricks and tactics about how an AV bypass the information security.

(l₇, l₈ & l₉): The deception vector refers to the mechanism of how a victim is deceived into facilitating a security breach. The cosmetic attack exploits the trust of a user by the appearance of the GUI components. Behavioral deception is achieved by imitating a legitimate system’s behavior rather than it looks. Hybrid approaches combine the last two vectors to make a more convincing deception, e.g., phishing website, along with images and text copy from the legitimate website.

**Execution** This stage refers to the operational procedure being executing in attack runtime.

(l₁₀ & l₁₁): In the case of a one-off attack, once a victim triggers a payload the attack halts any further action. On the other hand, the continual attack doesn’t expire upon successful exploitation. Rather victim continues to be exposed to its deception attempts.

(l₁₂ & l₁₃): The single-step attack requires only one action from the victim to carry out its exploitation, where in the multi-step attack, a user needs to be deceived more than once to achieve its goal.

We see that every classification doesn’t hold the same importance in terms of attack probability. The network administrator can put a weight on each classification based on his policy and strategy. For example, an explicit targeting attack needs much more attention than promiscuous; similarly, a single step might be more probable than a multi-step. Let’s
denote this weight as $Z_k$ for classification parameter index $k$ and define a social vulnerability score (SVS) for AV $i$:

$$SVS_i = \frac{\sum_k (Z_k \times l_k)}{|L_i|}$$

(3)

Our classification showed in table 1 sets the value of $k$ within a range $[1,13]$. The value of $l_k$ is one of the parameters coincides with the AV $i$; otherwise, it is 0. The denominator $L_i$ denotes the set of classification parameters of attack vector $i$ where $l_k \epsilon L_i$. A host in a network might hold multiple social AVs. Our classification method allowed us to estimate the diversity of a host w.r.t social AVs. This diversity $d_{sv}$ is defined with the fraction of classification parameters a host’s AVs combined matches. The likelihood of an attack on a host also depends on how frequently the operator or user of that host encountered the AV. We denote this variable with $f_i$ as a fraction of time for particular $i$. Using all these parameters, we define our second criticality metric for each host $n$ having $m$ AVs.

$$C_{sv}^n = (\sum_{i=1}^{m} SVS_i \times f_i) d_{sv}$$

(4)

It is worth emphasizing that we only consider the number of AVs a host contains and their possible implications. We don’t consider the anomaly of user behavior. Likewise, we don’t differentiate the personal use of a computer while pretending to do legitimate work, commonly termed as ‘cyberloafing.’ Although all this information just mentioned can be tracked and stored in a log file, this analysis is beyond the scope of this paper.

3.4 INFECTION PROPAGATION BASED CRITICALITY METRIC (IPCM)

Interaction between hosts can be attributed to their asset values in terms of information flows between them. This interaction could be a new opportunity for the attacker, which can be identified by the dependency relationship between system objects. We classify three types of objects: process, files, and sockets for this analysis. The dependency between objects captures the system characteristics by showing the control and information flow between files and processes. For instance, a process can read from a file as input and then write to a socket.
Such interaction among systems objects enables intrusion, which has been created directly or indirectly by attackers, to propagate from one object to another. The intrusion seeds could be compromised service programs or files, corrupted data, etc. When these intrusion seeds interact with other system objects via system call operation, the object gets infected. This phenomenon is called infection propagation.

3.4.1 OBJECT INSTANCE GRAPH

To analyze information flow from compromised host, a dependency network is needed. The traditional form of this network treats objects as nodes and information flow between them as edges which is not suitable for a practical scenario as it misses temporal information. This temporal information is important for intrusion analysis because the state of the object is time-varying, and infection that happens in a specific state could spread differently. Here, we model this type of dependency by means of object instance graph [62]. The focus of this model is to estimate the probability of infection propagation to the end host to evaluate the impact on the EDS. So we consider only those objects whose change propagates through sockets.

**Definition** The Host Object Instance Graph is a directed acyclic graph $HOIG = (V, E, O_V, d_E)$ where the set of vertices represents object instances in a host and the set
of edges represents functional dependency between them. This graph is generated on a per-
host basis from system call trace $\sum_T$ within a time window $T$. A system call syscall is
parsed into two system object instances, src and sink in a timestamp $\tau \in T$. The function
$O_V : V \rightarrow \sum_T$ labels a vertex with a system call object given from a syscall trace $\sum_T$. The
function $d_E : E \rightarrow \text{dep}$ labels an edge with a specific $\text{src} \implies \text{sink}$ dependency between
object instances.

A HOIG in a network is connected to others through a socket based on information
dependency and forms a network-wide object instance graph (NOIG). Figure 3 shows a
simple object instance graph with time-ordered system log. It could be said that different
instances are different versions of the same object at different time points, which is defined
by the information flow and can thus have different infection statuses. Here, $src_i$ and $sink_j$
is the $i^{th}$ & $j^{th}$ instance of system object $src \in V$ and $sink \in V$ respectively. For a $src$
object, a new instance is created only when no instance of $src$ exists in the graph, where for
the $sink$, a new instance is created when a $src \in sink$ dependency occurs. In our example
scenario, the solid rectangles and eclipses are new instances of the new object in the graph.
On the other hand, dotted rectangles and eclipses denote the instances of the already existed
objects. Based on the conditions mentioned above, process A instance 1 is src and file 1
instance 1 is sink where the later one act as source having dependency relationship with
process B instance 1.

3.4.2 INFECTION PROPAGATION MODEL

Our focus is to estimate the probability of infection in the end objects related to control
operations in the EDS. To do so, we build our model using Bayesian Network (BN), whose
acyclic nature matches with our dependency graph [62]. Moreover, BN is an effective tool
to incorporate intrusion evidence in order to characterize the uncertainty relating to the
propagation of infection. Our first task is to formulate the Conditional Probability Table
(CPT) by specifying the proper interaction of objects. Using the principles given in [62],
our infection propagation model deals with two types of dependency relationships between
object instances. One is intra-object infection propagation and other is inter-object infection
propagation.
**Intra-object infection propagation** model captures the interaction between the different instances of the same object. An assumption is followed here that an object never returned to an uninfected state from the state of infected. So if an object is infected with its current instance, then it holds the same status in its future instances. The example shown in Figure 4 explains the situation in a better way where the CPT holds the probability of infection of a particular instance $PA_{j+1}$. Here, $PA_j$ and $PA_{j+1}$ are the different instances of the same object $PA$. If $PA_j$ is infected, then the total infection transfers to its subsequent instances, no matter what happens to the other source instances. The third row of CPT shows this perfect scenario where the probability of $PA_{j+1}$ is one given $PA_j$ is true and $F_i$ false.

**Inter-object infection propagation** model can be tracked by the propagation of infection between instances of different objects. In Figure 4, $F_i$ and $PA_{j+1}$ exhibit this category of model showing source and sink object respectively. Parameter $\rho$ in the CPT table represents that how likely $PA_{j+1}$ is infected given the only source $F_i$ is infected. This can be determined by the fraction of times during which information flows from $F_i$ to $PA_{j+1}$. A measurement prior to this propagation model is necessary for this purpose. We take Figure 3 as our example scenario, given this as a chunk of the whole system graph to explain where we learn the interaction by only considering objects, not their instances. Let’s assume that information flow from process $A$ happens in total 20 times to 4 different objects where file 1 and process $C$ get affected 7 & 1 times respectively. In this case, given process $A$ instance 1 as the source, the value of $\rho$ in the CPT is 0.35 for file 1 instance 1 and 0.05 for process $C$ instance 1.
The previous two sections discussed two different scenarios of object interaction. Another situation might happen where more than one source affects the sink at one time. The fourth row of the CPT in Figure 4 indicates a similar situation here. The probability of infection on a particular instance in the graph can be calculated with the CPT of that instance and using the following equation:

\[
Pr(X_i) = 1 - \prod_{j=1, i \neq j}^m [1 - Pr(X_i|X_j) \times Pr(X_j)]
\]  

(5)

Here \( i \) stands for sink object instance and \( j \) for source object instances. Our work deals with known vulnerabilities whose exploitation gives attacker a certain privilege on that particular host. This privilege can be attributed to the accessibility to different objects. Thus, our third criticality parameter considered three steps. First, how a host can be compromised by different attack paths from a single attack point. Then we need to determine what privilege can be achieved in terms of objects. Finally, given those objects are tainted, we determine how infection can be propagated to the end systems.

Let’s denote a set of objects with \( e_n \) accessible by the attacker for a compromised host \( n \). And a set of objects related to process control and operation denoted as \( O_c \). Certainly, these objects belong to direct ICS operation. Every tainted object in the set \( O_c \) has an effect on substation operation, which is denoted with \( I_c \). Based on their behavior, attackers can corrupt different combinations of objects at the same time. Thus, the criticality metric for each host is a set having different values based on the attacker’s action. Moreover, this metric dynamically changes over time because the socket communication between hosts is not the same. If an attacker at a certain time infects an object, it might not propagate to the target instantly. It depends on the reliability of the target on that infected host. By tracking this time variable and cascading dependency on target on every host in a network, we model our third criticality metric. We formulate our metric for two cases: the maximum case and the minimum case, respectively. In the case of maximum case, our metric finds the probability that all outgoing sockets from the target during the time window are infected. That means the maximum damage could be possible in this case. We multiplied the probability with the product of the impact of those sockets on substation operation:
\[ C_{inf(max)}^n(\eta) = \{ Pr[o_1, o_2, \ldots, o_i | e_1, e_2, \ldots, e_j] \} \prod_{o_c \in \{O_c\}} I_{oc} \] (6)

Here, \( \{e_1, e_2, \ldots, e_j\} \) is the set of infected objects of the host (n) under investigation. It can be obtained by IDS alerts. On the other hand, the minimum case defined the likelihood that at least one outgoing socket of the target host is infected. Thus, it can be shown as:

\[ C_{inf(min)}^n(\eta) = \{ 1 - Pr[o_1^c, o_2^c, \ldots, o_i^c | e_1, e_2, \ldots, e_j] \} \sum_{o_c \in \{O_c\}} I_{oc} \] (7)

The parameter \( \eta \) defines the time-window which is a time range the administrator sets to investigate the infection propagation to the target.

### 3.5 SYSTEM DESIGN

Figure 5 shows the overall system design of our proposed model. This design is comprised of three parallel methods to obtain different criticality metrics. Security-related information such as vulnerability info, host configuration, network configuration, etc. is extracted. Then these data are used to generate an attack graph and integrated into Algorithm 1, which generates the EB matrix and criticality metric TCCM.

The first step towards SVCM is finding a social attack vector of a host under consideration. The attack vectors can easily be identified by relevant services and port numbers which are accessible through network scanners. Then, it can be classified by our control stage classification given in Table 8. These categorized attack vectors are then used to compute the diversity of each host in terms of social AV. The frequency of user interaction can be tracked by different event logging information, which contains event ID for different operations, i.e., USB plugin, VPN connection, log-on & log-off, etc. All these data can be combined to measure our second CM.

The development of infection propagation-based CM includes online system call auditing and offline data analysis. System call auditing should be done through all hosts in the network in a timestamped order. As our metric only considers information flow events between host objects, filtering is necessary to prune the system calls. Table 1 gives the list
Figure 5. System model
Table 2. System calls with Events

<table>
<thead>
<tr>
<th>Events</th>
<th># System calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>process modifies files</td>
<td>write, pwrite, pwrites64, dup, dup2, rename, fchmodat, fchmod, chmod, fchownat</td>
</tr>
<tr>
<td>process uses but doesn’t modify files</td>
<td>stat64, lstat64, fstat64, open, read, readv, pread64, execve, write, writev, pwrites64</td>
</tr>
<tr>
<td>process used and modifies files</td>
<td>open, openat, creat, dup, dup2, rename, mmap, mmap2</td>
</tr>
<tr>
<td>process creation</td>
<td>vfork, fork, clone</td>
</tr>
<tr>
<td>process termination</td>
<td>vfork, kill</td>
</tr>
<tr>
<td>process writes socket</td>
<td>write, writev, pwrites64</td>
</tr>
<tr>
<td>file termination</td>
<td>close</td>
</tr>
<tr>
<td>process checks or reads socket</td>
<td>fstat64, read, pread64</td>
</tr>
<tr>
<td>process connects and writes socket</td>
<td>connect, accept, bind, semi, send, sendmsg, recvfrom, recv, recvmsg</td>
</tr>
<tr>
<td>socket reads or writes socket</td>
<td>connect, accept, semi, send, sendmsg, recvfrom, recvmsg</td>
</tr>
</tbody>
</table>

of system calls associated with data flow events [63]. This dependency is used to generate HOIG for each host. The Bayesian network (BN) is constructed by leveraging the instance graph topology leading to an infection propagation model for the network. For a given time, a trigger object is fed to the BN network to estimate the probability of infection on the target objects.

3.6 PERFORMANCE EVALUATION

In this section, we analyze the three proposed criticality metrics through simulation based evaluation. We focused mainly on the dynamic behavior of these metrics as well as investigating the characteristics of the whole network. All simulation results are collected using a computer equipped with a 3.6GHz CPU, 16GB RAM under Windows OS. The Bayesian network-based metric (IPCM) is implemented using MATLAB BNT toolbox [64].

3.6.1 SIMULATION SETUP

We design our simulation environment by considering every scenario and factor discussed in the last couple of sections. Our topological connectivity-based criticality metric mostly deals with reachability analysis i.e., how a host can be reached from another network host on the way to its target. Thus, our analysis considers a multi-host, multistage attack from the access point to the target of the system. The network is generated by randomly assigning nodes and edges between them, where the latter is imposed by the exploitation of vulnerabilities arbitrarily placed in hosts. We distribute other parameters (OS, Isolation pattern and vulnerable service) within hosts by α, the probability that at least two hosts follow the
similar parameter. The simulation starts with taking a host, assuming the attacker resides in and sees other hosts’ opportunities using TCCM. This starting host is called the ’attack point’ in the rest of the paper. For efficient analysis, we have taken small networks in the simulation model.

For the evaluation of SVCM, we have taken thirteen distinct real social attack vectors and classify them with our proposed method. Thus, each attack vector is given a social vulnerability score (SVS). Then we distribute the attack vectors randomly within the hosts in the network for further analysis. In order to evaluate infection propagation, we model object interaction between every host in a network. Moreover, we simulate every event shown in table 1. In each host, processes and files are generated randomly, and their instances are attributed to the operating time slot. Random socket connection between hosts initiates the network object instance graph. Our Bayesian network is learned with the model for total time; then, the simulation runs it for a specific time window to see how it propagates infection to the target.

3.6.2 RESULTS

The Impact of SVCM on defense deployment: Social engineering is a common attack vector in EDS. The SVCM metric, along with contextual knowledge, can be used for an efficient decision. For example, a host having high SVCM near the HMI is much important.
than a host with similar SVCM several layers away. The most interesting application of our metric is measuring network diversity w.r.t social AVs. In terms of defense mechanisms, It is sometimes important to know which patterns of AVs are more frequent in the network.

Figure 6 shows the characteristic view of the attack surface of our simulated network. It is seen that ‘remote’ AVs simultaneously ‘single step’ are very frequent within hosts in the network. This analysis plays a very important role in finding out the domain-specific importance of AVs. In an ICS network, a host having ‘manual’ and ‘explicit’ AVs is more critical because it exhibits a specific type of attacker who usually intends to do stealthy propagation to the target. The administrators can use our metric to take proper policy to minimize the opportunity of an attacker in their network.

**The Impact of TCCM on defense deployment:** The SVCM metric characterizes the attacker’s attempt to gain access to the network. But if the attacker successfully gets access into the system, then we should address the opportunity provided by the intermediate hosts in the network. We set up multiple attack points in the layer farthest from the target (layer 6) and examine the change of the host’s opportunity. To do so, we measure the standard deviation of TCCM for individual hosts and all hosts within a layer in Figure 7(a) and 7(b), respectively. Both figures follow a similar trend as the deviation is higher for
Figure 8. Evaluation results for (a) Standard deviation of TCCM due to OS, vulnerable service and isolation pattern (b) risk of the network for particular attack points

the hosts adjacent to the attack points (left in the figure) than the host near the target. The reason behind this is as we follow the targeted attack graph, attack paths are more concentrated to the target host. Thus, hosts that reside near the target share similar attack paths. In contrast, it is more distributed near the attack point.

For an ICS network, along with stealthy propagation, fast propagation is also important. To examine this phenomenon, we select a single attack point and change the scale of similarity from low to high for OS, vulnerable service, and isolation pattern, respectively. As shown in Figure 8(a), we can observe that deviation happens in almost every host due to fixed attacker’s path, and change in effort reflects diverse opportunities. Finally, we evaluate the risk of the network in terms of initial attack points. Figure 8(b) shows that for initial attack points in the upper layer, around 2% hosts are most critical in terms of opportunity, where 18% hosts don’t provide any opportunity. So this metric could be helpful for countermeasure cost-security tradeoff analysis.

**The Impact of IPCM on defense deployment:** Attackers don’t need only to target HMI or SCADA servers to disrupt substations via PLC and RTU. Similar damage can be conducted if a file server is connected with HMI and has information flow dependency. Figure 9 depicts this scenario. For every host similar number of random files are corrupted in a
specific time window. Then the infection propagation is tracked with regard to our metric in the next four time windows. We observed that each host’s criticality varies significantly with time. It is because each host has dynamic incoming and outgoing socket connections in each time window. And it only propagates infection if its incoming connection gets infected. Furthermore, the scale of propagation differs with the subsequent object’s reliance on malicious files. Another observation is some hosts reside relatively far (left in the figure) from the target have high criticality than host near the target due to the variation of multiple cascading dependencies through the network to the target. This metric can also be used to extract the most critical time window for infection propagation.

3.7 DISCUSSION

Our proposed framework does not require any additional devices and communication requirements than those already presents in the modern ICS network. Different security systems like IDS & Syslog servers are already in use in the real industry. A system administrator needs to aggregate all the log data, alert, scanning results to leverage our model and analysis. The system administrator could set the probable compromised host from IDS alerts as the initial point to observe critical hosts all the way to target hosts.

In opposed to most studies dealing with vulnerability criticality and ranking, we focused
on devising the host criticality in our paper. Because it is proven that exhaustive searching and patching all vulnerabilities are nearly impossible in large-scale computer networks. Some security controls targeted to critical hosts rather than patching vulnerabilities can significantly increase network resiliency against adversarial actions [1]. Our work does not deal with tracking the location of intrusion root, which could be done by different existing methods [65] [66] [67]. In our analysis, we assume that all operations regarding information flow have the same capability to propagate infection. For example, a process that reads an infected file is affected at the same level when an infected process writes a file. So infection does not confine due to the events shown in table 6.

3.8 CONCLUSION

In this paper, we have taken first steps towards modeling the attacker’s opportunity in order to derive host criticality metrics. Along with opportunity, our metric simultaneously considers the attacker’s strategy and service dependency for a successful intrusion. Our evaluation results effectively demonstrated the whole network’s security posture in terms of attacker’s opportunity.
CHAPTER 4

ON THE EFFECTIVENESS OF INTRUSION RESPONSE SYSTEMS AGAINST PERSISTENT THREATS

In this chapter we model an Intrusion Response System by taking into account the real-time security state of the network and the uncertainty of the attacker’s strategy and action in order to design an efficient defense mechanism. To the best of our knowledge, we have taken the first step forward in modeling an IRS by considering the diverse opportunities the attacker has, estimated in real-time. We argue that the attacker’s next opportunity not only reflects their path and strategy towards their target but also influences the defender’s policy in protecting the target system more efficiently. Our model intends to capture and analyze the dynamic behavior of the attacker throughout the attack path, thus devising the response according to that. First, we model an attacker’s opportunity based on network and host characteristics towards their potential path to the targets. Then, after gaining evidence of intrusion, a correlated footprint is formed as the suspicious chain of the persistent attacker. This chain will help the defender understand the underlying strategy of the attacker, which is continually extracted and used for the response decision for that particular threat. The security state of the network is continuously updating in terms of the attacker’s behavior in penetrating the network and the defender’s subsequent actions.

4.1 OVERVIEW OF THE APPROACH

4.1.1 THREAT MODEL

Our threat model focuses on the attackers who want to infiltrate the network and compromise the critical targets. This kind of attack often directs to the organization through vulnerability exploitation or different social engineering techniques. Once the attacker compromises a node (or multiple nodes), he has multiple options to carry out his plan. The
attacker can move forward by exploiting vulnerabilities, utilizing improper network configuration or policies, etc. Figure 10 depicts a typical multi-stage attack for our threat model, representing several attack paths to the target node. Each stepping stone creates a new set of opportunities depending on the characteristics of the network, different hosts’ architectural position, function, etc. In each penetration step of his path, the attacker explores more of the network, which affects his subsequent actions. Sometimes attackers escalate their privilege (shown as self linking edges for each node in the figure) in order to mark more impact on the target network. On the other hand, along with path traversal, persistent attackers could take a different attack strategy irrespective of their capability to penetrate further. Attackers could compromise a host that has information flow dependency with the reliable operation of the target. Tainted processes or files could propagate in a cascading manner and alter the operation of the system. An efficient intrusion response should consider all these attack strategies, understanding the different aspects of the potential opportunities to exploit, and their criticality to the active adversary in the system.

4.1.2 SYSTEM MODEL

Our approach illustrated in Figure 11 consists of two phases, the opportunity analysis phase, and the intrusion response phase. The attack opportunity analysis phase builds on the strategic view of the attack surface. It collects scanning data and system call logs in order to generate an attack graph (AG) and an object instance graph, respectively. The

Figure 10. An illustration of our threat model
Figure 11. System Model

AG is used for enumerating potential attack paths from the origin to the target. Likewise, the object instance graph captures the functional dependency between assets by tracking information flow among them. The *intrusion response* phase is responsible for decision making by considering all this data and incorporating it with alert evidence. First, we extract the suspicious chain by mapping correlated events from the security alert to the AG. This chain provides additional insight into the characteristics of attack propagation. The security administrator can then make an appropriate decision by using all this information prior to evaluating defense coverage (as explained later) for each response strategy. In the next subsections, we will discuss every phase of our system in detail.

4.2 ATTACK OPPORTUNITY ANALYSIS

The rapid defense deployment tends to focus on the narrow artifacts of attack rather than the strategic view, which has long term impact on any response decision. We intend to model attackers’ strategies based on their opportunity during penetration. More specifically, we investigate how a host provides an attacker opportunity following successful exploitation. In this paper, the opportunity is meant to facilitate the *lateral movement* and *infection*, two major aspects of persistent threat in our model.

4.2.1 PROGRESSION THROUGH EXPLOITATION

When an adversary penetrates through a network, each host acts as a gateway to expedite
the penetration process. Depending on the constraints, the attackers are always trying to figure out exploitable attack paths. In this section, we devise a score for each host based on the opportunity it provides for the attackers we derived in Chapter 3. For each path $p_k$ from attack point to a target $t \epsilon T$ the cost value $c_{t_{pkj}}$ of a node $j$ as well as the cost value $C_{t_{pk}}$ of target host $t$ is determined from Algorithm 1. We integrate the contextual information such as vulnerable service (VS), operating system (OS), and isolation pattern (IP) and compute their diversity $S_{index(p_k,x)}$ in a path $p_k$ by equations 1 and 3. By combining all phenomenon described so far the opportunity that node $j$ provides is formulated as exploitation score:

$$ES_j = D_t \sum_{t \epsilon T} \sum_{p_k \epsilon P} \left( \frac{1}{\xi_{t_{pkj}}} \times c_{t_{pkj}} \times \prod_{x} \frac{1}{1 - S_{index(p_k,x)}} \right)$$

A bias value $\xi_{t_{pkj}} = e^{-c_{t_{pkj}}}$ is included to incline a higher score towards the attack source. The damage potential $D_t$ of each target node $t \epsilon T$ determines the highest direct and indirect operational capacity to the system.

### 4.2.2 FUNCTIONAL DEPENDENCY ESTIMATION

Sometimes, a persistent attacker could follow a different strategy to initiate an operation failure in the target. It is possible for the attacker to conduct malicious propagation and avoid detection. The effect of an attacker’s exploitation on a host might propagate to other hosts without raising an alert. In a network, the hosts have functional dependency among
themselves by information flow, which could be a stealthy malicious link. This dependency could be estimated by the object interaction in a host. If an attacker compromises a host with some read/write privilege and inserts malicious data into a legitimate file, then it could spread through the network by the associated information flow dependency. We call this situation *infection propagation*. Intrusion detection alerts always deal with a local consequence of a specific action in a host/domain; however, they do not tell us how this consequence holds the current privilege, what it means in a larger context, or how it may spread further. Figure 11 shows the notion of stealthy links between hosts. Here, a corrupted file initiates a malicious propagation through the network, which passes from host to host in due time and could alter the operation of the system.

Functional dependency is modeled through object interaction which is derived by *Host Object Instance Graph* in Chapter 3. The opportunity created by a host, in terms of functional dependency, is derived by the infection propagation to the targets. Every host has some critical objects that should be listed prior to the analysis. Given those objects are infected, the *functional dependency score* for a time window $\tau$ is calculated as follows:

$$FS_j(\tau) = \{PR[o_{t1}, o_{t2}, ..., o_{ti} | o_{j1}, o_{j2}, ....o_{jz}]\} IO_t$$ (9)

Here, $O_t = o_{t1}, o_{t2}, ..., o_{ti}$ and $O_j = o_{j1}, o_{j2}, ....o_{jz}$ are the set of the objects from target node $t$ and the set of the objects from host node $j$, respectively. The objects from $O_j$ are assumed to be infected at the beginning of $\tau$, and the probability whether the target objects $O_t$ are infected is computed at the end. $IO_t$ is the impact of target objects on the control and operation of the system. The functional dependency score is dynamic because of time-variable socket communication and cascading dependency on the target of various hosts in the network.

**4.3 INTRUSION RESPONSE**

The IRS model we propose has three stages: the uncertainty of security state, the uncertainty of attacker’s behavior, and the response decision-making process. Identification of the security state of the network is the initial stage of response analysis. Security Information
and Event Management (SIEM) systems acquire real-time information from different events happening in the network. This information holds uncertainty because of missed detections and false alarms occurring widely in the Security Operation Center (SOC). Several ways could be examined to handle this situation. For fast detection, Naive Bayes binary classifier can be employed as a useful method [31]. The probability of a security event is computed given an observed data sources or triggered alert. The historical alert and corresponding event make use of the prior probability distribution to train this classifier. One of the major problems in this approach is that it cannot capture the unknown footprint of a malicious event. State-based approach [33] can overcome this drawback. Within AG, this approach can identify the potential available exploit to the attacker. It then merges with corresponding events and correlates between them to unfold the current security state of the network. As detection is not the scope of the paper, we omit the detailed description for the detection process.

Our response function triggers after identifying a stepping stone action labeled as a compromised node in AG. As for persistent threats, we deal with each compromised node and correlation information between them that forms a suspicious chain. The suspicious chain \((s_i)\) is the ordered sequence for connected compromised nodes. The last node of the chain or the closest node to the critical target is considered as the attacker’s current position to enforce our response. Monotonicity assumption [48], as the reason behind this consideration, states that the capability does not decrease by launching attacks and attacker does not need to go back to the privileges already gained. It is also very beneficial for analyzing large AGs by removing the cycles.

Section 4.2 discussed the attacker’s opportunity from the defender’s perspective without any active intrusion happening in the network. Additionally, unlike attacker, the defender has knowledge of the whole attack surface characteristics, e.g., the critical assets, their positions, and running security controls. On the other hand, attackers have a limited view of the network, thus penetrate based on their plan, strategy, and capability. Although it is hard to understand attackers’ behavior within a short period of time and with limited information available, the defender could get some idea about it by tracking the network footprint the adversary passes on. We intend to capture two behaviors of the attacker: 1)
diverse capability and 2) aggressiveness. In our analysis, we measure the diverse capability by three parameters (VS, IP, OS) and assign higher opportunities to the less diverse path by putting equal weight on each category. But with real intrusion information, the defense is designed by extracting the apparent capability from the suspicious chain. For each category $x$ the weight $w_x$ is defined as:

$$w_x^s = \frac{\sum_{l=1}^{q^s_l} (\frac{1}{\phi_l} \cdot \sum_{t_l} e^{\frac{t_l}{\lambda}})}{\sum_{z=1}^{n} e^{\frac{t_z}{\lambda}}},$$  \hspace{1cm} (10)$$

This weight determines the relative importance of each category for a particular threat (i.e., chain). Each category has a $q^s_l$ types of instances in a chain $s_i$ where $\phi_l$ is the number of instances of type $l$. Assuming attackers’ volatile strategy in their penetration, we emphasize the latest node’s characteristics than the root nodes. An exponential function is used for this purpose while $n$ is the number of nodes in the chain in an ordered sequence and $t_z$ is the time that node $z$ becomes compromised. The decay factor $\lambda$ resembles the steepness of exponential distribution. After plugging the weight, equation 16 becomes:

$$ES_j^{s_i} = D^t \sum_{t \in T} \sum_{p_k \in P} \left( \frac{1}{\xi} \cdot \frac{t^j_{pk}}{C^j_{pk}} \times \prod_x 1 + \left( \frac{1 - w_x^s}{1 - S_{index(p_k,x)}} \right) \right),$$  \hspace{1cm} (11)$$

The modified equation shows the exploitation opportunity for the host $j$ in chain $s_i$. Thus, each node could have a variable exploitation score exhibiting a particular opportunity for each threat. Along with the diverse capability, the aggressiveness is also tracked in our model by using the temporal information associated with each piece of evidence. The time slot per penetration is taken as the reference for the most aggressive attackers. Consequently, the parameter $\alpha_i$ shows how much the penetration in the chain $s_i$ deviates from most aggressive attacks that could happen in that chain

$$\alpha_i = \frac{\sum_{z=2}^{n} e^{\frac{t_z}{\lambda}}}{\sum_{z=2}^{n} (t_z - t_{z-1}) e^{\frac{t_z}{\lambda}}},$$  \hspace{1cm} (12)$$

Like before, we focus more on the latest action. It should be noted that in our model, we deal with aggressiveness as a unique characteristic of the attacker regardless of the skills and resources. Although in a path, different actions could take different times based on
the complexity, we argue that an aggressive attacker would choose a less difficult action to propagate. After getting some sense of all these behavior from the suspicious chain, the defender needs to find a node to enforce response action. Each candidate node has two scores such as exploitation and functional dependency score, revealing the attackers’ aggressiveness and stealthy strategy, respectively. Our model assumes that the attacker is constrained to take one strategy in a single time slot. Thus, the node $j$ for the optimum response could be found as follows:

$$\text{argmax}_{j \in G} \frac{df_{ES_j}^* + df_{FS_j}}{C_j}$$

subject to $df_{ES_j}^* \geq \alpha_i$

$df_{FS_j} \geq (1 - \alpha_i)$

$C_j \leq 1$

Here, $df_{ES_j}$ stands for defense coverage of node $j$ normalized by maximum $ES$ among all candidate nodes, i.e., $df_{ES_j} = \frac{ES_j}{\max_{i \in N} ES_i}; \forall i \in N$ where $N$ is the set of all candidate nodes. $df_{FS_j}$ can be computed in the same way. The action could be patching the vulnerability, disconnecting the vulnerable service, fixing the improper configuration, initiating redundant service, etc. The parameter $C_j$ can be determined as the cost value of the response action on node $j$ with respect to the maximum resource allocation ($A$), i.e. $C_j = \frac{\text{cost}_j}{\max_j A_j}$ in our model.

4.4 IMPLEMENTATION AND RESULTS

In this section, we analyze our approach with an ICS alike network environment. We utilize Accenture ICS testbed (Figure 13) to evaluate our proposed framework. The testbed has been designed based on widely deployed defense-in-depth standard architecture ISA/IEC-62443 that is comprised of three zones, i.e., Information Technology (IT), Operational Technology (OT) and Demilitarized Zone (DMZ). Vulnerabilities are intentionally injected to resemble real attack prone ICS networks. Our analysis starts with scanning the entire network and subsequently generates an attack graph. For infection propagation, we model object interaction between different hosts in the graph. We consider every event related to information flow between objects [68]. The end of the time window for estimating $FS$ is
the beginning of the response time-slot. This window is moving with the time-slot as it is updating with attack propagation. The inclusion of $FS$ assures the operational efficiency during selection of the response action.

After generating AG and object instance graph, we simulate the attacker’s penetration to the targets. We set multiple critical assets as targets from the deep OT zone of the network (in our case, PLC and RTU), where the initial attacker’s position is randomly selected from the IT zone. This is a reasonable assumption because persistent attackers systemically probe the network and escalate their privilege to laterally move towards the target. For simplicity, let’s take a hypothesis that the attack and defense actions are always successful. We set equal costs for defense action e.g, $C = 1$.

Like a real attack scenario, attackers have a limited view of the network and select a random node to expand their footprint. As the attacker penetrates further, the suspicious chain becomes longer and reveals more attack characteristics. Figure 14 shows such behavior. In the early stages of an intrusion, the aggressiveness and weight values for all categories

Figure 13. Logical view of ICS testbed
follow a similar trend, but after some steps, as the attacker moves further, it turns out to be more aggressive and less diverse. We take aggressiveness as the major decision parameter because finding whether an attacker infects a file in the current compromised node or injects malicious information is time-consuming. Furthermore, this takes more analysis and investigation. The aggressiveness and diversity are driven by attackers’ strategy and skill, respectively. Similarly, different suspicious chains in the network exhibit different combinations of threat characteristics. At the beginning of the attack, the defender has no information about the adversary, but as time goes on, the defender acquires more information. As such, the defender can estimate the whole network security posture for specific threats and design the corresponding response action.

Every attack propagation step normally associates with one or multiple intrusion alerts which triggers the corresponding defender action. Figure 15 shows another aspect of the performance evaluation of the IRS within this action-defense scenario. Multiple attack points have been set where each one initiates a separate intrusion chain to PLC or RTU. The figure shows the percentage increase in attack steps needed for an adversary to either reach the target or to be entirely blocked by the response action. The measurement is done with respect to the shortest path the attacker could pursue without any response activating. The formula is defined as: \( \frac{r-s}{s} \times 100 \) where \( r \) and \( s \) are the number of steps explained above and the length of the shortest path, respectively. We conduct two trials for the same set of
Figure 15. Performance evaluation of IRS with regard to the attack propagation delay.

Table 3. Path Criticality Estimation

<table>
<thead>
<tr>
<th>Path from different edge nodes</th>
<th>path-1</th>
<th>path-2</th>
<th>path-3</th>
<th>path-4</th>
<th>path-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.df</td>
<td>14.90</td>
<td>11.77</td>
<td>10.23</td>
<td>13.87</td>
<td>11.82</td>
</tr>
</tbody>
</table>

attack points, and due to attacker’s random penetration, it follows different chains. From the figure, it is evident that our IRS significantly increased the overall attack steps at least by 200%. Our analysis also helps defenders to harden the most critical paths to the target. Table 11 shows the paths with the highest cumulative defense coverage ($C.\text{df}$) from our test network. This parameter particularly refers to the normalized opportunity a path could provide to the attacker.

4.5 CONCLUSION

Unlike other cyber adversaries, persistent attackers conduct more sophisticated and targeted attacks. Understanding these attacker’s behaviors is very important, which takes some time and in-depth forensic investigation. Simultaneously, once an incident happens, we need the prompt to respond as delay could cause more uncertainty and risk as a consequence. To overcome this conflicting situation, we proposed an approach leveraging the maximum utilization of the available information within a limited time for efficient defense action.
Our model investigates the attacker’s opportunity through the exploitation and malicious information flow analysis. Then correlated intrusion evidence is augmented to obtain attacker’s behavior, which can be interpreted to evaluate the strategy he will likely follow, and based on it, the appropriate response action is taken. We implemented our approach using test data from an Accenture ICS testbed. Our performance evaluation results have shown that the response system can reveal a different combination of threat characteristics and how they evolve. It also helps the defender to identify critical paths which provide a higher opportunity for the attacker. In conclusion, our result shows that the response system can significantly delay and often block attacker’s penetration towards the target while maintaining operational efficiency.
CHAPTER 5

CYBER THREAT ANALYSIS BASED ON CHARACTERIZING ADVERSARIAL BEHAVIOR FOR ENERGY DELIVERY SYSTEM

In this chapter, we propose a cyber threat analysis framework based on characterizing adversarial behavior in a multi-stage cyber attack process. First, we investigate how a threat proceeds within the network by constructing adversarial Attack Graph (AG) and identifying all possible attack stages. Then we discuss how each stage can be associated with network attributes. Using a holistic view of threat’s exposure provided by attack graphs, our model incorporates attacker’s techniques and tactics into the stepping stones found in the AG. We propose to add more context to each attack stage using real-world knowledge base of adversary tactics and techniques, when characterizing the adversary’s progression in the attack path. Our attack path analysis model identifies the level of difficulty in taking a path by considering the complexity of the path, the attacker’s skill set, etc. The path hardness is measured in terms of the attacker’s capability and challenges. The insight into the level of difficulty of an attack path in the network helps security administrators to pinpoint critical paths and prioritize path hardening actions.

5.1 OVERVIEW OF THREAT INDICATORS

In order to investigate the attacker’s behavior, security administrators typically track low-level threat artifacts such as hash values, IP addresses, and domain names. These are often termed technical threat intelligence shown in the first three levels of the Pyramid of pain model [2]. Although it is relatively easy to integrate these artifacts within a defense system, a low-level indicator of compromise (IoC) is not much fruitful for defending against sophisticated adversaries. These indicators are susceptible to change over time as attackers use botnets, random domain names, or dynamically change hash values with low costs. On
the contrary, the attacker’s action follows some particular sequence that is being reused with little modification. Attributes related to actions are shown in the upper three layers of the pyramid, often termed as behavioral attack signatures of threats. These indicators are very hard to change for a particular group of attackers, such as script kiddies, hacktivists, cybercriminals, or state-sponsored attackers. Thus, defense systems that take into account the top three threat artifacts will present a tougher obstacle to the adversary.

**Network/Host Artifacts:** We can describe this artifact from the perspective of the defense mechanism design. For reactive defense, it represents the adversary traces in the network and host. These traces can clearly distinguish malicious activity from a legitimate one. File & directories, registry objects, distinctive transaction values, etc., are examples of host and network artifacts for this case. For proactive defense, the artifacts represent the conditions that allow the attacker to propagate. This can include network service information, running processes, network connection & access control, policy and principles, etc.

**Tools:** It refers to the software or utilities attackers use to accomplish their objectives. Some attackers rely on public exploits or open-source tools, while skilled attackers utilize proprietary tools and obfuscation. Some stealthy attackers utilize utilities that are part of the operating system or are present in the environment to carry out their malicious intent. Some of the examples include Metasploit, netstate, DustySky, Ettercap, etc. A tool can be part of multiple types of attacks as well as a group of attackers using similar types of tools.
Figure 17. Proposed cyber threat assessment framework

might share the same attack pattern in a particular network [69].

**Tactics, Techniques and Procedures (TTPs):** Attackers can take different strategies and paths to achieve their objectives. TTPs can capture this kind of latent behavior of the attacker. It describes the attacker’s approach at different levels of granularity within a cyber attack campaign. A tactic refers to how an adversary can operate part of the cyberattack campaign: what step to take next? The attacker’s tactic might follow a certain pattern for the entire campaign, or it could modify throughout the lateral propagation depending on the underlying security control or damage potential to the target. A particular tactic might have different ramifications depending on the adversary type. For defense evasion, some adversaries prefer to conduct silent cleanup, and some proceed to wipe the system services completely despite a higher chance of alert generation. The technique provides detailed description within the context of tactic: how to take the next step? In other words, the techniques are meant to facilitate the execution of different tactics. While technique consists of actions without specific direction, the procedure provides more low-level details correspond to a technique. It includes all the necessary steps to complete an action. A well-tailored procedure increases the success rate of a technique.

5.2 FRAMEWORK

The behavioral attributes discussed in previous section do not contain the level of information to understand attacker’s behavior comprehensively. For an attacker, learning a
new technique or adapting to a new method is much difficult than learning tools as well as learning tools is harder than learning network/host artifacts. We intend to identify these features and integrate them in a model for efficient threat analysis. Our proposed framework addresses the tasks illustrated in Figure 17. We will explain each component of this framework in the following subsections.

5.2.1 ATTACK GRAPH GENERATION

We use MulVal [70] to generate AG consisting of different node types to show how a set of network and system configurations result in unauthorized actions to specific targets. Figure 18 shows a small portion of an AG generated by MulVal and converted to a Neo4j graph database structure. An attack graph is a directed graph modeled as $G(V,E)$ with a set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E = \{e_1, e_2, \ldots, e_m\}$ connecting nodes together, where $|V| = n$ and $|E| = m$. Nodes in an AG are of three different types as also depicted in Figure 18.

![Figure 18. A sample attack graph represented in Neo4j graph format](image-url)
**Configuration:** The circular nodes in the network denote system or network configurations. These are the conditions that provide possibilities for actions by the adversary. Some of these nodes such as `hacl`, `nfsMounted` and `vulExists` shown in Figure 18.

**Rule:** The hexagonal nodes in the network represent the reasoning rules which usually represents the attack methodology leveraged by an adversary to achieve a particular goal.

**Impact:** The impact nodes indicate the sub-goal for a certain action attacker could take. In Figure 18, it is shown as rectangular nodes where `netAccess`, `access file`, `execCode`\(^1\) are examples of impacts ranging from benign to catastrophic. There are two types of edges in an attack graph: 1) configuration-to-rule edges that represent logically AND, meaning all configuration conditions have to be true to cause the impact; and 2) rule-to-impact edges that represent logical OR, meaning that the impact happens if at least one rule satisfies holds.

From the discussion above, we can say that an attack graph is created by taking into account the configurations directed by some rules in order to make some impacts on the target network. We enclosed all configuration, impact and rule nodes in sets \(C, I, R\), respectively. Therefore, \(C = \{c_j | c_j \in V, \forall c_j \text{ is a configuration}\}\) such as \(\{c_1 = v_{17}, c_2 = v_{18}, c_3 = v_{19}, c_4 = v_{20}\} \in C\) in Figure 2, \(I = \{i_j | i_j \in V, \forall i_j \text{ is an impact}\}\) which can be mapped to \(\{i_1 = v_{13}, i_2 = v_{15}\} \in I\) in Figure 2, and lastly, \(R = \{r_j | r_j \in V, \forall r_j \text{ is a rule}\}\) where \(\{r_1 = v_{14}, r_2 = v_{16}\} \in R\) in the example shown in Figure 18. Thus, the combination of these sets comprises all vertices of the graph \(G\) i.e. \(V = \{C, I, R\}\).

### 5.2.2 ACTION STATE MODEL:

AG allows us to inspect all possible sequences of exploits an attacker can take to infiltrate a network and reach its goals. Assuming some particular behavior of the attacker, we could simplify the AG and make it more scalable to investigate. The monotonicity assumption [48], states that attacker’s capability does not decrease by launching attacks, and the attacker does not need to go back to the privileges already gained. Using this assumption, our AG has been translated to a condition dependency graph which is very beneficial for analyzing

---

\(^1\)These MulVal terms will be further discussed in section 5.4.
large AGs by removing cycles.

By incorporating this phenomenon (i.e., conditional dependency), we further decompose the generated AG into a state graph where each state is called action state. The primary objective of this transitioned model is to represent the progression of the attacker in the network over distinct actions. An action state \((a_{si})\) is comprised of pre-condition(s), rule and associated exploit or impact. In Figure 2 two states have been designated for a better understanding of the model. Each action starts with enabling some pre-conditions for that action state. Pre-conditions are AND conditions depicted with red links to each rule. Thus, a rule holds if all configuration conditions already exist. An enabled pre-condition gives the attacker an opportunity to cause the impact; e.g., execute an exploit. More formally, \(P(e|\exists c_{pr} = F, (e, c_{pr}) \in a_{si}) = 0\): that is an exploit that can not be executed until all it’s pre-conditions are satisfied. A successful execution means that an attacker maliciously built a trust relationship between two hosts or gained a privilege in the same host. Post-conditions are OR conditions shown with green links from rule to impact. A post-condition can be treated as a pre-condition for another state. This action state model can efficiently track the attacker’s movement throughout the network. From now on, we use state and action state alternately in the rest of the paper.

5.2.3 MAPPING TO TECHNIQUE-TACTICS:

We need to map each attack state to a certain technique in order to unfold the current phase of the attack strategy deployed by the attacker. MITRE proposes a model called Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) [5] characterizing malicious behaviors for each step in a cyber attack campaign. Tactics provide the high-level objective of why an attacker follows a particular behavior in a system, and techniques provide more fine-grained information showing how an attack is performed. Our goal is to map each attack state to the distinct high level category defined in the ATT&CK model. By incorporating this, each attack path eventually exposes a sequence of tactics and techniques. Our evaluation finds that this sequence could form a Techniques, Tactic, and Procedures (TTPs) of the cyber Kill Chain [22] [71].

The ATT&CK model categorizes adversarial techniques into different tactics where each
technique might fall in multiple tactics. We define a tactic $l$ as $tcl = \{ta_1, ... , ta_t\}$ enclosing a set of techniques belonging to it. We intend to map each action state with a distinct Technique-Tactic (TT) pair. In our action state model, each rule represents a threat action associated with some primitives, which are ascribed as configuration nodes in our state graph. The configuration nodes are shared by multiple states in our model. Similarly, different network and system features are shared by multiple techniques. We use this phenomenon for our mapping methodology. For technique mapping, we use rule and configuration information which could map one state to several techniques. On the other hand, the tactic describes the sub-goal of a threat action which is analogous to the impact node in our state graph. Thus, using domain expert’s knowledge, we map each state to a TT pair. From a real attack history, it is evident that each technique often requires a pre-requisite technique to accomplish its goal [72]. For instance, in APT33 the technique T1078 (Valid accounts) cannot be performed without User Execution (T1204) before. Likewise, in APT1, technique T10005 (Data from Local System) and T1114 (Email Collection) are two pre-conditions for exfiltration of compressed data (T1002). We can use this finding to improve our mapping accuracy. This will be discussed in detail in our evaluation section.

5.3 PATH COMPLEXITY AND EFFORT ESTIMATION

The complexity of an attack path is essential to understanding an attacker’s behavior during attack progression. National Vulnerability Database (NVD) databases only provide insights into the exploitability of vulnerabilities. But determining the presence of exploitable vulnerabilities is not sufficient for computing the complexity of an attack path. Adversaries have to go through a chain of actions for the successful execution of an attack path. Each action has its own challenges based on different factors. Moreover, the attacker’s evolving skill should be considered for accurate path complexity calculation. Our framework, illustrated in Figure 17 incorporates contextual information in order to characterize the level of difficulty in taking an attack path.

5.3.1 VULNERABLE COMPONENT RISK:

Existing work focuses only on the vulnerability exploitability score to estimate the effort
required to attack a host. Although there exist several vulnerabilities in a cyber infrastructure, in reality, only a fraction of them are exploited widely e.g., only 15% of known vulnerabilities are exploited in the wild [73]. As exploitation of one vulnerability is more critical than others, a suitable assessment is needed to quantify the difference. Exploitability of vulnerabilities are typically computed using the Common Vulnerability Scoring System (CVSS) [59] scores that range from 0 to 10. We categorize these scores into easy (0-3), medium (4-7) and hard (8-10).

While default scoring of vulnerability only describes it’s technical aspect, we also focus on attempted vulnerabilities through real attack scenarios in the field. This will lead us to derive a unique vulnerability score for our network. In order to do that we distinguish the state of an exploit into three categories:

**Unproven:** refers to the unavailability of exploit code for exploiting the vulnerability. The vulnerability is identified but no potential full-fledged or general purpose exploit has been revealed. Vulnerabilities that fall in this category have a very low probability of being exploited.

**Proof of concept:** The development of proof-of-concept exploit happens to be part of penetration testing and the vulnerability disclosure process. Most relevant and updated information of this category can be acquired from the exploit database [74]. This is a Common Vulnerabilities and Exposures (CVE) [59] compliant archive for public exploit with associating vulnerable software.

**Exploited in the wild:** refers to the vulnerabilities extensively practiced in real attacks. Some vulnerabilities attract much attention and quickly get exploited. The information of this group is documented in different security reports and databases like Symantec’s AttackSignature dataset [75].

Table 4. Vulnerable component risk

<table>
<thead>
<tr>
<th>Exploitability level</th>
<th>Likelihood level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unproven</td>
</tr>
<tr>
<td>Easy</td>
<td>3</td>
</tr>
<tr>
<td>Medium</td>
<td>2</td>
</tr>
<tr>
<td>Hard</td>
<td>1</td>
</tr>
</tbody>
</table>

---

53
Table 4 shows the risk of a vulnerable component by combining these two factors. Each entry has been assigned with some numbers considering the relative risk of the component. We can see in Table 4 that the upper right index i.e. Easy-Exploited in the wild pair has a higher risk for the vulnerability fallen on that class. We note that the index score can be easily modified based on security personnel’s preference.

5.3.2 TECHNIQUE PRIORITY SCORE:

Along with vulnerability, we like to prioritize techniques employed by the attacker in the kill chain. Two factors have been considered to determine the priority of each technique independently: adaptability and exploitation. The adaptability of a technique depends on the environment and conditions that allow it to be exercised in addition to different goals the technique can attain. A technique usable for multiple OS allows attackers to carry their attacks out on various services and applications. Thus, a platform independent technique presents additional risks. An Adversary’s actions also require a particular privilege for successful exploitation. A normal user privilege can be easily achievable versus that of a superuser, but the latter could cause catastrophic damage. We have added both aspects by analyzing permission requirement of a technique. A technique’s effectiveness is also defined as the number of distinct goals it obtains. Expertise on a particular technique resulting in more tactics gives the attacker more opportunities to proceed in the kill chain. Therefore, the adaptability score (ASc) of a technique t is:

$$ASc(t_a) = p_{l_t} \times \sum_{i=1}^{p} p_{r_i} \times \tau_t$$

where $p_{l_t}$ and $\tau_t$ are the fraction of OS platforms and achievable tactics respectively. In addition, $p_{r_i}$ denotes the permission level i for technique t. This score is proportional to the privilege acquired by the permission. For instance, the SYSTEM permission has the highest rank in this context and User has the lowest one.

Technique’s priority also depends on how it has been manifested in the real world. The MITRE ATT&CK database maps each technique with it’s associated softwares which has been leveraged by threat actor/groups for their adversarial activities. Software is broken
down into three categories such as tool, utility and malware mostly referring to open source-commercial code, operating system utilities, custom or open source software for malicious purposes, etc. Groups use techniques in part of their attack campaign or intrusion activity over time. Hence, we measure how extensively a technique has been exploited by its exploitation score.

\[
ExSc(ta_t) = sf_t \times gr_t
\]  

Where \( sf_t \) and \( gr_t \) are the fractions of software and groups, respectively, that utilized technique \( t \) in real-world attacks, next, we combine these two parameters discussed above to find the priority score of a technique. This technique priority score depicts how beneficial a technique is for attackers and how likely they would use it regardless of network structure. Each technique’s score \( TSc \) is calculated independently using equations 13 and 14:

\[
TSc(ta_t) = \beta ASc(ta_t) + (1 - \beta)ExSc(ta_t)
\]  

The tuning parameter \( \beta \) is specified by the security administrator. This parameter refers to the relative importance of whether a technique gives more benefits to the attacker or is easy to learn from a real attack.

5.3.3 CORRELATION COEFFICIENT CALCULATION:

The attacker needs to follow a chain of actions to reach to a target in the network. It resembles a compromising chain of states in our proposed model. Each state is integrated with different database such as ATT&CK [5], NVD [59] and Common Weakness Enumeration (CWE) [76]. This knowledge base gives us the corresponding factors that influence an attacker to take a particular action. Multiple techniques from different or the same tactics might have overlapped factors that help attackers in their future actions. We track these attackers evolving skills from state to state by constructing a state correlation matrix in order to quantify how each state is correlated to others. The correlation matrix between state \( x \) and \( y \) is formulated as follows: \( CC_{x,y} = AMCC_{x,y} + ENCC_{x,y} \); where \( AMCC \) and \( ENCC \) are the attack method correlation and environmental correlation, respectively. We
determine AMCC by examining whether an attack follows the same technique or emerged from the same weakness. ENCC refers to the environmental and system features relevant to the attacks performed in the network. These features include platform, application or service, configuration, etc.

5.3.4 HARDNESS OF A PATH:

We define the state hardness $H_{as_i}$ as a function of two parameters, intrinsic state hardness $H_{as_i}(intr)$ and correlated hardness $H_{as_i}(corr)$. The intrinsic state hardness refers to the difficulty of state irrespective to the attack path it belongs to. The latter one is computed by correlating with the former states the attacker traversed in a particular path. The hardness of path $k$ from host $j$ to $j'$ is formulated as:

$$H_{P_{j,j'}}^k = \sum_{i \in AS_{P_{j,j'}}^k} H_{as_i}(intr) \ast H_{as_i}(corr)$$ (16)

The correlation between state $i$ and $q$ is denoted with correlation coefficient $(CC_{iq})$. For similar states, we introduce a decay factor $(\lambda)$ representing the effort reduction in similar actions; the larger the $\lambda$ is, the easier the same action will be at the next time. Thus, equation 16 yields:

$$H_{P_{j,j'}}^k = \sum_{i \in AS_{P_{j,j'}}^k} (\alpha_i^{-1} + TSc(ta_i)^{-1})e^{-\sum_{i=AS_{P_{j,j'}}^k} (0) \frac{CC_{iq}}{\lambda}}$$ (17)

Here, $(\alpha_i^{-1} + TSc(ta_i)^{-1})$ is the criticality of the state where $TSc(ta_i)$ is the technique priority score assuming state $i$ is mapped with technique $t$ and $\alpha_i$ is the vulnerable component risk for this state. The priority score reflects the defender’s priority alternatively, meaning it is less hard for the attacker. In our transformed state graph each path from a source to target can be represented with a set of states. Thus we denote $AS_{P_{j,j'}}^k = \{as_1, as_2, ......as_n\}$ as the set comprised of $n$ action states for the path from $j$ to $j'$.

5.4 IMPLEMENTATION AND RESULTS
We utilize Accenture ICS test-bed (Figure 19) to evaluate our proposed framework. The test-bed has been designed based on the ISA-62443 architecture comprised of three zones, i.e., Information Technology (IT), Operational Technology (OT), and Demilitarized Zone (DMZ), that simulate a power utility network. In the IT and OT zones, multiple workstations, servers, and security devices are embedded. On the other hand in the OT zone, I/O panels are controlled by PLC and RTUs, where these are subsequently interacting with the SCADA server and monitored by the human-machine interface (HMI) for active operation. We intentionally injected vulnerabilities in IT hosts, HMI, and SCADA servers. Vulnerable firmware is used in end devices like PLC and RTUs resembling real attack-prone ICS networks.

Our analysis starts with scanning the whole network in different stages. IT zone has been probed with active scanning tools such as Nessus, where passive scanning tools like Grassmarlin and ClarOty are used for OT networks. In addition, we used iDefense Intel-Graph threat intelligence database [77] for identifying vulnerabilities. This information has been used in order to generate logical AG for the network. The AG for the testbed is then translated into states by employing our methodology. For our analysis, we extracted attack paths terminating at multiple targets. The security administrator can determine the critical asset that needs to be protected and is set as the target accordingly. Multiple targets could also have shared attack paths that can influence cyber defense remediation strategies. Each path is comprised of a set of states from source points to different target hosts. Then the states are being mapped to the TT category. A sample state extracted from our attack path is shown below.

\[
\begin{align*}
\text{execCode}('192.168.15.124', \text{someUser}) : - \\
\text{vulExists}('192.168.15.124', 'CVE-2015-2808', \\
\text{safari}, \text{remoteExploit}, \text{privEscalation}) \\
\text{networkServiceInfo}('192.168.15.124', \text{safari}, \text{tcp}, \\
'1433', \text{someUser}) \\
\text{netAccess}('192.168.15.124', \text{tcp}, '1433') \\
\text{rule_desc}('remote exploit of a server program')
\end{align*}
\]

By taking into account the sample state’s configurations and the rule, our methodology primarily maps this state into two techniques: \textit{exploitation for client execution (T1203)} and
exploitation of remote services (T1210). These techniques are associated with two different tactics such as execution and lateral movement respectively. Our domain knowledge considers execution as the most appropriate tactic for this sample.

Moreover, we can acquire more confidence by introducing another characteristic phenomenon in our model. The cyber kill chain (CKC) describes different phases in the adversary life cycle in an attack campaign. It helps to understand the attack progress happening in variable forms. In our model, we can treat each attack path as a part of the superset of a CKC. Although it is true that adversary’s tactic doesn’t follow exactly the same linear order shown in [78], it ensures a causal relationship between actions. For instance, command and control (C&C) never happens before initial access. We introduce this property for tactic identification as the pre-state and post-state analysis. Referring to our example, one of the pre-conditions netaccess('192.168.15.124') can be represented as a prior state as shown below:

```prolog
netAccess('192.168.15.124',tcp,'1433'):--
attackerLocated(internet)
hacl(internet,'192.168.15.124',tcp,'1433')
rule_desc('direct network access')
```
Our initial analysis mapped this state to *initial access* tactic. Furthermore, the following state marked as `netaccess('192.168.15.123'..)` takes our sample state (i.e. `execCode('192.168.15.124'...)`) as a pre-condition and is represented as follow:

```
netAccess('192.168.15.123', tcp, '1433'):~
execCode('192.168.15.124', someUser)
hacl('192.168.15.124', '192.168.15.123', tcp, '1433')
rule_desc('multi-hop access')
```

The state shown above is mapped to the *lateral movement* tactic. Thus the sequence formed as: *initial access → execution → lateral movement*, is a perfect example of a kill chain phase. Consequently, it affirms our analysis and emphasizes the effectiveness of our method for TT mapping. After mapping to TT we assign a score for each selected technique by using equations 13, 14 and 15. Then we measure the vulnerable component score from 4 prior to discovering its appropriate category. This is an optional parameter; each and every state does not have to hold a vulnerability. From our evaluation, it is more evident that among the three states discussed above, only the first one has a vulnerable service. In contrast, TT is an inevitable feature of a state. Thus, TT embedded AG gives the defense planners more advantage to deploy their defensive techniques.

As we mentioned before, we generate attack paths for multiple targets. More specifically, targets are chosen from different zones like HMI from OT, file server (FileS) and web server (WebS) from DMZ and mail server (MailS) and workstation (WS) from IT network. It should be noted that our targets are heterogeneous in service and extracted from different architectural levels of the network. This allows us to measure the accuracy of our approach within different dimensions. The decay factor can be calculated from post-compromise real attack analysis. It can be estimated by matching our correlation method to the time-to-compromise of a state. The parameter could be improved gradually by the feedback method. For our convenience, we set the value 2 for decay factor means around 40% effort reduction if both states correlate perfectly.

After plugging all information into equation 17 we calculate the path hardness of all attack paths for each target host. Figure 20 depicts the effect of effort correlation in the hardness measurement. The X-axis represents a different target host we selected for our
Figure 20. Deviation of path hardness by effort correlation

analysis and Y-axis represents average path hardness considering all attack paths direct to each host. For each host, the hardness value is computed with and without correlation. Assuming the attacker goal is the target host, each case shows the least effort needed to traverse the path by taking into account correlation than without it. Indeed, this correlation reflects attack’s knowledge propagation in an attack path. It can be observed from the Figure that the scale of deviation is not same for all hosts. For instance, around 25% effort reduced for attack paths towards HMI regarding correlation while it is 15% for WS. Because HMI is located in OT zone, very low level in the network contrary to WS which resides in the IT zone, very close to the perimeter. Hence, attack paths in the direction to HMI have more states to pass, allowing attackers to learn more and help them to decide the following route.

On the other hand, by using our method, we can also determine the diversity of an attack path. A path having low hardness difference between correlation and non-correlation case has high diversity. Consequently, a system security administrator can devise an operative security policy by using this phenomenon in order to block future malicious activities. Prior to an attack, the administrator can harden the most impactful path by assessing the consequence ($C$) of the path along with its diversity ($D$). Attack paths having a high consequence and low diversity should be treated with high priority for this defense mechanism. This establishes a hierarchy among paths; the most critical paths of the network are those with the highest $\frac{C}{D}$ value.

Our path metric can also capture the total security posture of the network. Figure
Figure 21. Distribution of attack paths within different hardness level

21 shows the distribution of attack path within different hardness level for our selected target hosts. From the Figure, it is determined that the concentration of distribution varies from host to host in the network. In our analysis, HMI has large path distribution between hardness range 13—21, which is higher than other hosts. Although it is not the only deciding factor, the host’s architectural level is one of the reasons behind this scenario. EDS follows the defense-in-depth architecture where the control center is isolated with several security controls that make the majority of the route very tough for the attacker. In addition, hosts who remained in the same zone don’t need to follow the same hardness level. For instance, WebS and FileS both reside in the DMZ zone exhibits a noticeable difference in path hardness distribution. Previously discussed path diversity can be utilized to interpret this situation. Depending on the configuration of a certain network, the paths towards both hosts might have variable path length along with the different scales of correlation in each path. Assuming the attacker’s goal is to get into the DMZ zone, they might persist and explore the network to move further. In this case, WebS could be a very easier target for them.

The aforementioned path characteristics need to be examined thoroughly for proper defense deployment. The security administrator can assign a threshold as a particular path hardness level for critical assets in the network. As the attack graph is dynamically changing over time by zero-day vulnerabilities being discovered, regular security policy variation and countermeasure enforcement, the defender must investigate the hardness of critical node in
order to keep the risk of the network in an acceptable level.

5.5 CONCLUSION

In this chapter, we presented a proactive cyber threat assessment framework for EDS. We described the stages of attack progression from the attacker’s and defender’s perspectives. We identified the behavioral signature of an active state by incorporating the techniques and tactics within an attack graph. We developed two scoring metrics to quantify the likelihood of a state being compromised. We also evaluated the proposed framework in an ICS test-bed. The system security planner can use the path metric to apply the remediation policies to the most critical attack paths. It will also provide insights into balancing cyber risk and operational resilience. Additionally, the evaluation results have provided insights into mapping the tactics taken in an attack path to adversarial/threat characteristics. Some adversary groups would never change the tactic, while others would adapt, given the situation.
CHAPTER 6

PATTERNMINER: TOWARDS AUTOMATED LEARNING AND ANALYSIS OF ADVERSARIAL ATTACK PATTERN FROM MALICIOUS CYBER CAMPAIGN

6.1 ATTACK PROGRESSION ANALYSIS AND DEFENSE STRATEGY

Attack life cycle can be modeled into multiple steps (also called attack phases [78] [23]). We adopt the definition of a cyber threat proposed in [79] and identify different aspects to characterize an attack successfully. Understanding attack life cycle in campaign is important to identify the adversary’s behavior. The defender analyzed these captured behaviors, matching with their resources in order to determine appropriate defense strategy.

6.1.1 ATTACK LIFE-CYCLE IN THE CAMPAIGN

An Attack model is very crucial in order to understand targeted attacks and how they propagate in the campaign. The information which can be obtained within this process gives hints about how to protect the network and provide advanced planning by finding possible threats. Advanced persistent threats are well organized working together as part of a professional team, taking a slow-and-low approach to work their way into specific target companies. The goal of this deliberated “one-to-one” approach is to steal valuable intellectual property and money, such as, intercepting bank wire transfers, credit card data, authentication credentials, trade secrets, and other personal identifiable information as well disruption of services like power-grid, oil & gas pipeline, etc. The different steps in the attack are explained based on [5] [80] [3] as shown in figure 22.

1. Intelligence Gathering: The first step is gathering information and a crucial basis for the attackers to perform a successful attack. Attackers employ different social engineering
techniques and open-source intelligence tools to find possible entry points by getting to know
details about underlying IT infrastructure e.g. routers, anti-virus tools, firewalls, open ports,
etc.

2. Initial compromise The second stage includes malware delivery, including the
widely used direct delivery techniques, like Spear Phishing, and indirect techniques like a
Watering Hole Attack. Attackers only need to trick a victim into clicking a link that could
drive him to open a piece of malware in the form of a seemingly harmless attachment. This
exploits a zero-day vulnerability or takes in a malicious website, enabling them to control
the employee’s PC, gain access to the corporate network, and execute a cycle of difficult-
to-detect maneuvers to attain their ultimate goals. Infected USB drives or CDs/DVDS are
used to deliver malicious content in this attack type [81].

3. Establish foothold: This successful stage results in the installation of a backdoor,
allowing the attacker to command and control (C&C) the compromised machine and malware
for subsequent phases. Therefore, at this stage traffic is generated and file evidences are left
on targeted computers, giving defenders the chance to detect an APT in an early phase.

4. Escalate privileges: Legitimate credentials are collected in this stage. Accounts
with higher privileges are of major interest, since they enable a deeper access into the network

---

**Figure 22. Attack life cycle in APT progression [3]**
and are necessary for techniques and tools the attackers usually use. Thus attackers always try to escalate to advanced privilege through numerous techniques such as Brute-force, hash or password dumping, etc [82]. Access to these valid account enable them to penetrate further in the system.

5. Lateral Movement & Persistence: Once inside an attacker compromises additional machines to harvest credentials, escalate privilege levels and maintain persistent control. An Attacker uses them to hide between legitimate traffic and activities in order to stay undetected. Moreover, they usually leverage standard operating system methods and tools and perform similar actions as legitimate system administrators, which makes it harder to detect their actions.

6. Asset/Data Discovery, Exfiltration and Sabotage: Finally the main goal of the attacker is to steal sensitive data, sabotage or disrupt operations. Several techniques (ex. Port scanning) are used to identify the noteworthy assets and the services that house the data of interest [83]. Once sensitive information is gathered, the data is funneled to an internal staging server where it is chunked, compressed and often encrypted for transmission to external locations.

6.1.2 THE DETECTION MATURITY LEVEL MODEL

Defenders tailor their defense by observing threat activity and the information they could
acquire by observation. A bit extended model for the maturity of cyber threat detection has been proposed by Ryan Stillions [84] shown in Figure ???. The model is proposed to describe the maturity of an organization in terms of their ability to consume and act upon threat information [4]. Threat information includes indicator of compromise, technique, tactics, threat reports etc. The detection maturity level (DML) model emphasizes the increasing level of abstraction in the detection of cyber attacks while it assumes that security operation center (SoC) team of low maturity and skills would be able to detect attacks in terms of low level technical observations in a network, without necessarily understanding the significance of these observations. On the other hand, a SoC team of high maturity and high skills is assumed to be able to interpret technical observations to plan their defense.

The DML is a hierarchical model, has different levels depicting the various dimensions of threat patterns and their effectiveness for detection. Some information can be captured directly from different forms of data like logs and alerts while some information only appears after careful examination. Starting from level 0 (DML-0), the higher it goes, the more robust the pattern will be and the less precisely it can be identified. Adversarial pattern captured by PatternMiner effectively covers from DML-3 to DML-9 contribute multilevel detection capability. The final goal is to identify the attacker’s identity, which resides in the highest level (DML-9) can be obtained only if the lower levels are attributed correctly.

6.2 PRELIMINARIES:

6.2.1 INDICATOR OF COMPROMISES AND ADVERSARIAL ATTACK PATTERN:

Unlike measurement data, which is collected automatically, threat intelligence reflects a careful manual analysis of security threats, with conclusions drawn from the human knowledge and intuition of expert analysts. These conclusions can be encoded in indicators of compromise (IoCs). An IOC is an observable attack artifact more specifically composed of some combinations of virus signatures, IPs, URLs or domain names of botnets, MD5 hashes of attack files, etc. They are frequently described in cybersecurity articles, usually utilized
for early detection of future attack attempts by using intrusion detection systems and antivirus software. Thus, they play an important role in the field of cybersecurity. However, with the rapidly evolving cyber threats, the IoC data are produced at a high volume and velocity every day, which makes it increasingly hard for humans to gather and manage them.

We can describe these artifacts from the perspective of the defense mechanism design. For reactive defense, it represents the adversary traces in the network and host. These traces can clearly distinguish a malicious activity from legitimate one. File & directories, registry objects, distinctive transaction values, etc., are examples of host and network artifacts for this case. For proactive defense, the artifacts represent the conditions that allow the attacker to propagate. This can include network service information, running processes, network connection & access control, policy and principles, etc. It refers to the software or utilities attackers use to accomplish their objectives. Some attackers rely on public exploits or open-source tools while skilled attackers utilize proprietary tools and obfuscation. Some stealthy attackers utilize utilities that are part of the operating system or are present in the environment to carry out their malicious intent. Some of the examples include Metasploit, netstate, DustySky, Ettercap, etc. A tool can be part of multi-type of attacks as well as a group of attackers using similar types of tools might share the same attack pattern in a particular network.

Although it is relatively easy to integrate the low-level security artifacts within a defense system, a low-level IoC which mostly falls under DML-1 to DML-3 is not much fruitful for defending against sophisticated adversaries. These indicators are susceptible to change over time as attackers use botnets, random domain names or dynamically change hash values with low costs. In the contrary, an attacker’s action follows some particular sequence that is being reused with little modification. Attributes related to actions are shown in the upper levels of DML, termed as behavioral attack signatures of threats. These indicators are very hard to change for a particular group of attackers, such as script kiddies, hacktivists, cybercriminals, or state-sponsored attackers. Thus, defense systems that take into account the top three threat artifacts will present tougher obstacles to the adversary.

6.2.2 TACTICS, TECHNIQUES AND PROCEDURES (TTPS):
Figure 24. (a) Techniques and (b) Tactics, exploited in the wild w.r.t the threat groups listed in the MITRE ATT&CK [5] database.

Attackers can take different strategies and paths to achieve their objectives. TTPs can capture this kind of latent behavior of the attacker. It describes the attacker’s approach at different levels of granularity within a cyber attack campaign. The tactic refers to how an adversary can operate part of the cyber attack campaign: what step to take next? The attacker’s tactic might follow a certain pattern for the entire campaign, or it could be modified throughout the lateral propagation depending on the underlying security control or damage potential to the target. A particular tactic might have different ramifications depending on the adversary type. For defense evasion, some adversaries prefer to conduct silent cleanup and some proceed to wipe the system services completely despite a higher chance of alert generation. The technique provides a detailed description within the context of the tactic: how to take the next step? In other words, the techniques are meant to facilitate the execution of different tactics. While technique consists of actions without specific direction, the procedure provides more low-level details corresponding to a technique. It includes all the necessary steps to complete an action. A well-tailored procedure increases the success rate of a technique.

6.2.3 ATTACKER PROFILE WITH MITRE
With the surge of cyber attacks in recent years, a large number of attack artifacts have emerged, which has been extensively reported by public online sources and aggressively collected by different organizations.

Our framework utilizes this information from the existing knowledge base and uses it to profiling threat actors following similar behavior. Although APT groups are mostly state-sponsored, independent in their resources, skills certain TTPs get very popular and widely practiced in real attacks. Figure 24. shows the distribution of techniques and tactics with respect to APT groups listed in the MITRE ATT&CK [5] database. It is a standard knowledge base for attributes related to adversary techniques and tactics based on real world observations. Here, 244 techniques are listed so far for the enterprise system belonging to 12 tactics. We did our analysis on 93 threat groups from the database. From Fig. 24, it is evident that not all techniques-tactics are widely exploited in the wild. For instance, 80 techniques have never been practiced among the groups of our analysis where execution(Exc), delivery (De) and Discovery (Dis) tactics are extensively exploited by APT groups in their campaign. Our preliminary analysis discussed above ensures that not all attack patterns need to be treated with the same importance.

6.2.4 DEPENDENCY PARSING

Dependency parsing is an NLP technique for describing grammatical relations between words in a sentence. Such relations includes determinant, direct object, noun compound modifier and others. Our prototype extensively utilized dependency parsing in attack pattern learning modules. Dependency between actions and their respective goals has been inferred to understand adversary motivation. Figure 25 shows a sentence from threat reports and their dependency relation between words. Here we see a keylogger particularly related to Lazarus Group and called KiloAlfa has a compound relationship with the threat group. It also acts as a nominal subject (nsubj) for a specific action, e.g., obtaining a user token.

6.3 TOOL ARCHITECTURE

Our tool follows the high-level three-stage pipeline architecture depicted in Figure 26 to process data from the threat intelligence database and cyber campaign reports. The
first stage collects data from threat intelligence sources and threat reports such as security articles, blogs, posts, etc. After prepossessing, the data is fed into a multi-label classifier to identify techniques and corresponding tactics employed in a cyber campaign. Finally, in the information extraction stage, relevant attack patterns are processed by an PER network. To learn and collect adversarial pattern, we leveraged a set of NLP tools. Moreover, due to the unique characteristics of the open problem, the generic information extraction technique cannot achieve good performance for our task. Therefore we incorporated machine learning and deep learning in order to extract the correct pattern.

6.3.1 DATA COLLECTION AND PREPROCEESSING:

The first step before operating on texts is preprocessing. It’s the cleanup steps as we have parsed some noise during the data collection that can hinder our pattern identification process. We studied different ways to do the prepossessing and kept the results with the most accurate outcomes. The preprocessing phase shown in figure 19 is accomplished via multiple stages.

Converting to lowercase: First we convert every letter to its lowercase. This is based on the assumption that each letter should convey the same meaning regardless of its case.

Filtering stop words: We remove the words that which do not contain important significance to be used in the pattern extraction process. These words are defined as stop words listed in the Natural Language Processing Toolkit (NLTK) e.g., ‘for’, ‘the’, ‘to’, ‘in’ etc. These words are filtered out from search queries because they return a vast amount of unnecessary information.

Text cleaning and lemmatization: We remove punctuation and special characters which do not bring any valuable information e.g., ‘,’ ‘.’ ‘!’ ‘?’ etc. Hyphens and underscores
are kept because sometimes they convey special indications in the cyber domain. We also do Lemmatization, which is text normalization (sometimes called Word Normalization) technique that reduces the inflected words properly, ensuring that the root word belongs to the language. In Lemmatization the root word is called Lemma, which is the canonical form, dictionary form, or citation form of a set of words.

**Tokenization with multiword expression:** At this step, the module performs tokenization on cleaned text. This process generally extracts individual words as tokens but for the security domain it is not as straight-forward. Because security narratives often include multi-word expressions which cannot be analyzed as individual tokens. For example, Copy Cat is the name of malware where the words Copy and Cat are meaningless when considered separately. We identified multi-word expressions by using the method proposed in [85], and we consider such expressions as single token in the rest of our analysis. The intuition behind this method is that the joint probability of words is much higher than the product of the probability of individual words. For the above example, since the word “Copy” is likely to appear together with “Cat”, we identify “Black Hole” as a multi-word expression. Table 12 shows some examples of multi-word expressions from our dataset.

It should be noted that we only do the preprocessing steps mentioned above for the TTP extraction module, while PER module we keep the word in the original form. Because keeping all the information is important to understand the hidden pattern of a word in a sentence.
Table 5. Examples of multi-word Expressions from our dataset

<table>
<thead>
<tr>
<th>Multi-word Expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>white lambert, peppy trojan, moonwind rat, royal dns, metasploit stager, black lambert, sakula rat, googledrive rat, apt3 keylogger, havex rat, poison ivy, byebye shell, blue lambert, cobra carbon system</td>
</tr>
</tbody>
</table>

![Figure 28. Framework for Technique and Tactic Identification from Cyber Campaign data](image)

6.3.2 TECHNIQUE AND TACTIC IDENTIFICATION

**Feature Extraction:** The input of this stage is preprocessed tokens, and the output is feature vectors. The simplest way to represent text in a numerical format consists of transforming it to a vector where each element corresponds to a unique token and contains a value that indicates its “weight.” Such representation is known as bag of words (BoW) in the literature. Cleaned text from the raw document has been transformed to BoW, where each bag corresponding to a class represents the feature vector for that class. The next step concerns calculating the weight of the features, which is derived from the term frequency (TF) and the inverse document frequency (IDF) method. By $\text{TF-IDF}_{t,d} = tf_{t,d} \cdot \log \frac{N}{df_t}$, where $t, f_{t,d}$ is the number of times the term $t$ appears in the document $d$, $N$ is the number of documents in the corpus and $df_t$ is the number of documents containing term $t$. We also calculate TF-IGM [86] which is similar to TF-IDF but instead of IDF inverse gravity moment (IGM) is measured here with the formula as follows.
\[ igm(t_k) = \frac{f_{k1}}{\sum_{r=1}^{n} f_{kr}r} \] (18)

Where \( f_{kr} \) is the frequencies of the term \( t_k \) occurring in different classes, which is sorted in descending order with \( r(= 1, 2, ..., n) \) being the rank and \( n \) is the number of classes. Although suitable and efficient for many text mining tasks, BoW models pose considerable challenges to most classification algorithms. In particular, for our problem, it represents a very high dimension feature vector for each class and ignores the word order completely. In order to solve the issue, we employed sentence embedding resulting in a fixed feature vector corresponding to each sentence in the class. First, word embedding has been learned from text corpora and captures the semantic relationship between words. PatternMiner utilize Glove [87] and dependency-based word embedding [88] as both provide compatible performance in our analysis. GloVe embeddings are trained on the global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given corpus. This co-occurrence count matrix is processed by normalizing the counts and smoothing them, followed by factorization to get lower dimensional representations. Dependency-based word embedding (DepVec) utilizes skip gram model from Word2vec where contexts of a word is extracted by dependency instead of just surrounding words. Context of the example sentence from figure 28 is shown in Table 13. This type of context extraction captures different information than bag-of-words contexts as the former is more inclusive and focused than the latter. It can capture relations that happen to be out of reach with small window bag-of-words.

**Technique and Tactic Classification:**

The final goal of this module is to extract TTP from threat campaign information reported in security article. Figure 28 represents the framework for technique and tactic identification from cyber campaign data. The process consists of two phases: learning phase and detection phase. In the learning phase, labeled data from threat intel and security articles have been preprocessed and transformed into feature vectors. TF-IDF vectors for each class encompass all words in the corpus with TF-IDF weights where sentence vectors for each sentence in the class are formed by averaging word vectors belong to the sentence. Needless
<table>
<thead>
<tr>
<th>Words</th>
<th>Contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lazarus</td>
<td>Group/compound$^{-1}$</td>
</tr>
<tr>
<td>Group</td>
<td>KiloAlfa/compound$^{-1}$</td>
</tr>
<tr>
<td>Keylogger</td>
<td>KiloAlfa/compound$^{-1}$</td>
</tr>
<tr>
<td>KiloAlfa obtains</td>
<td>Group/compound, Keylogger/compound, obtains/nsubj$^{-1}$</td>
</tr>
<tr>
<td>user</td>
<td>KiloAlfa/nsubj, tokens/dobj$^{-1}$</td>
</tr>
<tr>
<td>tokens</td>
<td>tokens/compound$^{-1}$</td>
</tr>
</tbody>
</table>

Table 6. Dependency based context extraction

to say that each class is represented by a distinct *technique* in MITRE ATT&CK. In the detection phase, all the same operations have been done on the unlabeled data until classification. TF-IDF vector is in conventional one-hot representation and becomes very sparse in our analysis thus we use cosine similarity for TTP detection. The similarity is calculated between TF-IDF or TF-IGM vector of a class and a candidate sentence from security data. The class above a threshold value with the highest similarity score is selected as a potential TTP that appeared in the sentence.

We use supervised ML classification to determine the TTP (if any) in the candidate sentence. Our implementation utilizes logistic regression (LR) [89] and support vector machine (SVM) [90] to train the classifier. LR works well on the small labeled dataset by optimizing the log-likelihood function to determine a probability that is a logistic function of a linear combination of the training dataset, and every training point has a certain influence on the estimated logistic regression function. SVM works by maximizing the margin between the instance and the separation hyperplane.

**6.3.3 PATTERN ENTITY RECOGNITION**

Identifying semantic elements (Named Entity Recognition or NER [91]) and extracting relations between them have been extensively studied in the Natural Language Processing (NLP) community. However, existing NLP techniques cannot be directly applied for high level IoC and adversarial pattern discovery. NER systems are known to be brittle and highly
Table 7. Pattern Entity to be Extracted from Campaign Data

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>Does not contain useful information</td>
</tr>
<tr>
<td>Action</td>
<td>Action performed in cyber campaign</td>
</tr>
<tr>
<td>Intent</td>
<td>Motive of an action</td>
</tr>
<tr>
<td>Tool</td>
<td>Utilities and tools used in cyber campaign</td>
</tr>
<tr>
<td>Conf</td>
<td>Configuration facilitate malicious action</td>
</tr>
<tr>
<td>Action Object</td>
<td>Surface of action</td>
</tr>
<tr>
<td>Intent Object</td>
<td>Surface of potential action or objective</td>
</tr>
</tbody>
</table>

Corpus Annotation: In this section, we describe the construction of our PER corpus. There are 6 different entities listed in the corpus: *action*, *intent*, *tool*, *conf*, *action object* and *intent object*. The entity *action* tags any adversarial action of an attacker where *intent* tags motivation of that action. Tag *tool* represents utilities and tools used in a cyber campaign. *Conf* is a broad tag which means a certain configuration, network situation which facilitates adversary action or a certain path used for malicious action. *Action object* and *intent object* represents the surface of action and potential action respectively.

Embedding Layer: In this layer, we represent the text sequence with vectors corresponding to words. Word embedding is an expression that maps the word to low-dimensional space. 

**DCShadow** is a method of **manipulating Active Directory data**, including objects and schemas, by **registering** and simulating the behavior of a Domain Controller. Once registered, a rogue **DC** may be able to **inject and replicate changes into AD infrastructure** for any domain object, including credentials and keys. The adversary has used **CVE-2015-1701** to **access the SYSTEM token** and copy it into the current process as part of **privilege escalation**.

*Figure 29.* Example of adversarial pattern entity in threat intel data
vectors. The word representation cannot only be influenced by the surrounding words but also the characters they are composed of [93]. Character-level embedding is specifically useful for our model as threat intel, and security reports might consist of a good amount of out-of-vocabulary tokens, e.g., software, threat actors, malware. Figure 30 describes the architecture to generate word embedding for a word from its characters. The character embeddings corresponding to every character in a word are given in direct and reverse order to a forward and a backward LSTM. The LSTM model can be explained with the formulas below [94]-

\[
\begin{align*}
    i_t &= \sigma(W_i(h_{t-1}, x_t) + b_i) \\
    f_t &= \sigma(W_f(h_{t-1}, x_t) + b_f) \\
    o_t &= \sigma(W_o(h_{t-1}, x_t) + b_o) \\
    \tilde{c}_t &= \tanh(W_c(h_{t-1}, x_t) + b_c) \\
    c_t &= i_t \ast \tilde{c}_t + f_t \ast c_{t-1} \\
    h_t &= o_t \ast \tanh(c_t)
\end{align*}
\]
where $\sigma$ represents the \textit{sigmod} activation function. $tanh$ represents the hyperbolic tangent function. $x_t$ represents the unit input. $i_t$, $f_t$, $o_t$ represent the input gate, the forget gate, and the output gate at time $t$. $W$ and $b$ represent the weights and bias of the input gate, the forget gate, and the output gate respectively. $\tilde{c}_t$ denotes the current state of the input. $c_t$ denotes the update state at time $t$. $h_t$ denotes the output at time $t$.

The embedding for a word derived from its characters is the concatenation of its forward and backward representations from the bidirectional LSTM(BiLSTM). Then it is concatenated with word vector from pretrained word embedding dataset. This word representation is concatenated with head word representation and dependency relation to get the input representation of a word in position $i$ in the sentence- $e_i = [w_i, w_h; d_r]$. Here, $w_i$ and $w_h$ is the word representation of word in position $i$ and it’s parent respectively. $d_r$ is the embedding for the dependency relation. The above representation all together takes care of the direct long distance interaction in the input layer.

**Sequence Representation Layer:** After having the dependency encoded input representation $e$, we apply 2 BiLSTM layers to capture the contextual information as well as the interaction between words and their corresponding parents. After the first BiLSTM, the hidden states at each position are forwarded to the next BiLSTM layer and to its child in the dependency trees. An \textit{interaction function} is applied to capture the dependency relation between parent and child. For the root word, the hidden state at the specific position will forward itself. For a given sentence $X = (x_1, x_2, \ldots, x_n)$ containing $n$ words, each of them having a fixed dimensional vector, the hidden state of the first layer of BiLSTM is represented as-

$$H^1 = [f(h_1, h_{p1}), f(h_2, h_{p2}), \ldots, f(h_n, h_{pn})]$$

Here $p_i$ indicates more BiLSTM can be stacked to enable deeper reasoning between words in the sentence. The hidden states of the $(l + 1) - th$ layer of BiLSTM can be calculated from the hidden state of the previous layer

$$H^{(l+1)} = BiLSTM(H^l)$$

The 2 BiLSTM layers with interaction function effectively encodes long-distance relation
Figure 31. Neural Network Architecture for Pattern Entity Recognition
Table 8. Interaction Functions

<table>
<thead>
<tr>
<th>Interaction Function</th>
<th>$f(h_i, h_{pl})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self connection</td>
<td>$h_i$</td>
</tr>
<tr>
<td>Concatenation</td>
<td>$h_i \oplus h_{pl}$</td>
</tr>
<tr>
<td>Addition</td>
<td>$h_i + h_{pl}$</td>
</tr>
</tbody>
</table>

as well as parent-child dependency between words. The different interaction for the model is shown in table 6. *Self connection* returns the hidden state itself equivalent to the stacked LSTM layers. *Concatenation* and *addition* are a very straightforward way to model the interaction.

**Entity Extraction Layer:** We model our adversarial pattern entity extraction as a sequence labelling task. We introduce a CRF [95] layer to output the most likely sequence $y = (y_1, y_2, \ldots, y_n)$ of predicted labels given the input sequence $x = (x_1, x_2, \ldots, x_n)$. Following [93] the score of the label sequence $y$ is defined as the sum of the transition and emission probabilities from BiLSTM

$$score(y, x) = \sum_{i=1}^{n} p_i[y_i] + \sum_{i=0}^{n} T[y_i, y_{i+1}]$$  \hspace{1cm} (25)

where vector $p_i$ is the output of the BiLSTM layer and $p_i[y_i]$ is the probability of label $y_i$ at the $t$-th position. $T$ is a matrix that contains the transition probabilities of two subsequent matrices where $T[q, r]$ is the probability that a token with label $q$ is followed by the token with label $r$. The CRF layer took these scores and turned them into probabilities of label sequence by taking a softmax function over all possible label sequences.

$$P(y|x) = \frac{\exp(score(y, x))}{\sum_y \exp(score(y, x))}$$  \hspace{1cm} (26)

6.4 EXPERIMENTS AND RESULTS

In this section, we discuss the performance of *PatternMiner* with real world threat intel data. We run our experiments in Intel(R) Xeon(R) 2.5GHz with 128 GB RAM.
6.4.1 DATASETS:

Collection: The ATT&CK [5] framework is the central idea of our analysis and served as the ground truth for the model. The data defines different techniques and tactics we want to retrieve as part of the attack pattern. One of the main incentives to use the framework is that it characterizes attacker behavior in such an extensive manner and update frequently. We import data from MITRE repository \(^1\) structured in STIX 2.0 format [96]. Figure 32 represents the example of Enterprise ATT&CK data in STIX format. The object attack-pattern represents a technique (T1214) linked to a tactic called credential-access through kill_chain_phases inside the object.

The only data we could find already labeled was the content of ATT&CK website. Each of the techniques includes external_references which are usually threat intel data or technical descriptions linked with url where that specific technique has been reported. From the list of urls, we crawl 481 cybersecurity articles only connected to the attack-pattern which are published from 2010-2019. We only considered URLs are redirecting to the threat reports in HTML format not omitting PDF docs for our analysis.

Preparation:

\(^1\)https://github.com/mitre-attack
Each document is divided by sentences using Python NLTK [97] sent tokenizer. We build a set of regular expression\(^2\) to filter the data. TreebankWordTokenizer and WordNetLemmatizer from NLTK is used for tokenizing sentence into words and lemmatizing them respectively. Before tokenizing, we extracted multi-word expressions from our data to effectively distinguish the features. Python library newspaper3k [98] is for threat report scrapping from web. Dependency between words in a sentence is extracted by using SpaCy [99] which is broadly used for industrial strength NLP work. Glove\(^3\) has been utilized as pretrained word embedding for our model, which is trained on 6 billion words from Wikipedia and web texts. Moreover, we also utilized word embedding trained on only cybersecurity related resources \(^4\). Word2vec model has been applied to a collection of one million documents found on the Web relevant to cybersecurity.

**Tagging Schema:** We need to assign an adversarial pattern entity label to every word in a sentence. A single pattern entity could span multiple tokens within a sentence. For the flat entity, sentences are usually annotated with BIO encoding (which stands for “Begin”, “Inside” and “Outside”) where token in the beginning and inside of entity labeled as \(B-label\) and \(I-label\) respectively. \(O\) if it is not part of the any entity. This annotation is applied as CONLL-X format. There are 6 types of entities defined in our dataset. The corpora has been annotated manually with gate\(^5\) annotation tool. It is annotated by four annotators who are graduate students doing security research.

Our model is designed for flat pattern entity where it assumes each word has only one label. But in real data, although very small but some entities could overlap with others. In our dataset, more than 90% entities are flat entities. Nevertheless, we applied NER model from [100] for nested entity recognition. It is slightly different from our model in figure ???. A single BiLSTM layer is used; dependency and interaction function between words are not considered as well. The dataset has been prepared in a different way. It is still CONLL-X format but nested entities are linearized as given below.

\[
\text{PROMETHIUM O}
\]

\(^2\)https://docs.python.org/3/library/re.html  
\(^3\)https://nlp.stanford.edu/projects/glove/  
\(^4\)https://ebiquity.umbc.edu/resource/html/id/379/Cybersecurity-embeddings  
\(^5\)https://gate.ac.uk/sale/tao/split.html
### Table 9. Performance comparison of TTP extraction with BoW and Sentence Embedding (300d)

<table>
<thead>
<tr>
<th>Model</th>
<th>Features</th>
<th>Accuracy LR/SVM</th>
<th>Precision LR/SVM</th>
<th>Recall LR/SVM</th>
<th>F1-Score LR/SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>2* Bag of Words</td>
<td>TF-IDF</td>
<td>51.72 (cos sim)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>TF-IGM</td>
<td>53.83 (cos sim)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3* Word Embedding (Glove)</td>
<td>Universal</td>
<td>58.63/54.28</td>
<td>58.24/56.41</td>
<td>58.63/54.28</td>
<td>56.39/53.9</td>
</tr>
<tr>
<td></td>
<td>Universal (TF-IDF Weighted)</td>
<td>72.65/65.57</td>
<td>74.62/69.95</td>
<td>72.65/65.57</td>
<td>71.68/65.23</td>
</tr>
<tr>
<td></td>
<td>Pre-trained</td>
<td>54.69/53.60</td>
<td>53.50/54.55</td>
<td>54.69/53.60</td>
<td>52.31/52.17</td>
</tr>
<tr>
<td></td>
<td>Pre-trained (TF-IDF Weighted)</td>
<td>74.01/66.12</td>
<td>74.60/70.56</td>
<td>74.01/66.12</td>
<td>72.38/65.72</td>
</tr>
<tr>
<td>2* Dependency based Embedding</td>
<td>Pre-trained</td>
<td>49.79/45.17</td>
<td>48.95/47.24</td>
<td>49.79/45.17</td>
<td>46/44.46</td>
</tr>
<tr>
<td></td>
<td>Pre-trained (TF-IDF Weighted)</td>
<td>47.75/46.53</td>
<td>45.82/46.69</td>
<td>47.75/46.53</td>
<td>43.66/41.59</td>
</tr>
</tbody>
</table>

has O
used O
spearphishing B-tool
emails L-tool
to O
deliver B-action
BrainTest L-action | U-tool
as O
a O
malicious O
attachment O

The above encoding is called BILOU format\(^6\). The mapping from token to multilabel follows some rules. The entity mentioned earlier has priority over the entity starting later. On the other hand, for the same beginning, longer entities have priority over shorter ones. In our example, action entity deliver BrainTest has tool entity within it. It started before, and it gets prioritized in the nested labels.

### 6.4.2 RESULTS

The performance of PatternMiner is evaluated with multiple measurement parameters. We adopt four performance evaluation measures to represent our model performance: Accuracy, Precision, Recall and F-score. Accuracy computes the ratio of samples that are correctly classified in the total samples. Accuracy does not always provide the complete picture of the imbalanced dataset, as a large number of samples forms a bias towards majority

---

\(^6\)B- (beginning), I- (inside), U- (unit-length entity), L-(last) or O (outside) labels
Table 10. Performance comparison of TTP extraction with different word embedding dimension

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recalls</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>65.03</td>
<td>69.77</td>
<td>65.03</td>
<td>64.77</td>
</tr>
<tr>
<td>100</td>
<td>69.25</td>
<td>72.72</td>
<td>69.25</td>
<td>68.40</td>
</tr>
<tr>
<td>300</td>
<td>72.65</td>
<td>74.62</td>
<td>72.65</td>
<td>71.68</td>
</tr>
</tbody>
</table>

classes. Hence, precision(Pr), recall(R), and F1 measure are utilized as well. The calculation formulas are:

\[ Pr = \frac{TP}{TP + FP} \]  
\[ R = \frac{TP}{TP + FN} \]  
\[ F1 = \frac{2 \times Pr \times R}{Pr + R} \]

True positive (TP) indicates the correct classification where false positive(FP) and false negative(FN) address the miss-classification of negative and positive samples, respectively. In case of TTP extraction our problem is turned into multinomial classification problem where each sentence in threat intel has to be classified in either one of the 265 techniques or none. Needless to say, technique classification can automatically be mapped to tactics as it resides in the upper layer of the tree. We did a couple of experiments with different methods. Table 12 performance of our TTP extraction module with BoW and sentence embedding method. In the BoW method, TF-IDF and TF-IGM vector is formed for each class (i.e., technique) corresponding to all terms. To reduce the impact of more frequent words, we did the square root of each term’s tf weight. Vector of similar dimension is formed corresponding to each sentence in the threat report. Cosine similarity between two bags of words is measured and the highest similar TTPs are extracted. For both methods, we use the cut-off threshold to eliminate TTPs with low similarity scores to minimized falsely labeled TTPs. For BoW, accuracy is defined by the number of correct TTPs extracted from report, divided by the number of TTPs labeled for that report. Both TF-IDF and TF-IGM method the accuracy is 51% and 53%, respectively. A very sparse vector with lots of noise data could
be a probable reason for this low accuracy. Moreover, due to manual apprehension not all TTPs from an article are reported in ATT&CK database [101].

We used sentence embedding to solve this high dimensionality issue. Each sentence is classified with single techniques from ATT&CK. For Glove, we use embedding already trained on a large corpus\textsuperscript{7} called Universal for our analysis. We also trained the Glove vector in our corpus, which has been indicated as Pre-trained in our results. We use 6600 sentence for training and 1050 for testing. We generate sentence embedding by summing all word vectors within it. We use Scikit-learn library [102] to utilized different multinomial classifiers implemented in it. For SVM, we use ‘linear’ kernels as it performs better in our datasets compared to non-linear kernels such as Radial Basis Function (RBF) or Polynomial. From the result, we can see that both universal and pretrained embedding has a close classification metric score. Our classifier is trained also by multiplying the TF-IDF value with corresponding word vectors which results significant improvements in classification scores. Classification accuracy and f-score improve almost 20% for pre-trained embedding. For DepVec, we only use the vector trained on our corpus as no such vector existed for the universal corpus. The accuracy is much lower than Glove for TTP extraction. In every cases, logistic regression outperforms the SVM classifier with a slightly higher value.

We also did experiments on different dimensions of embedding vectors to see how it affects our classification model. Table 14 shows the performance of TTP extraction with vectors of three different dimensions- 50, 100, 300. With the LR classifier, the result shows that the classification score improves with rising dimensions even in a slight manner. The dimensionality in word embeddings represents the total number of features that it encodes. Larger vectors can store more information since they have more possible states. The module shows that our sentence embedding method is able to learn short corpus and imbalanced dataset with good performance. We assume that each sentence in security reports does not map to more than a single technique. It is not true all the time in the real scenario but we discard this for our analysis.

For PER, after annotation we train our neural network model for entity recognition. We use 4530, 750 and 527 fully annotated sentences for train, validation, and test set,\textsuperscript{7}https://nlp.stanford.edu/projects/glove/
Table 11. Experimental Hyper-parameters

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Flat</th>
<th>Nested</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word dim</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>LSTM dim</td>
<td>200</td>
<td>256</td>
</tr>
<tr>
<td>Dropout rate</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Initial learning rate</td>
<td>0.01</td>
<td>0.001</td>
</tr>
<tr>
<td>Optimizer</td>
<td>SGD</td>
<td>Adam</td>
</tr>
<tr>
<td>Batch Size</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Epoch</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Tag-Schema</td>
<td>BIO</td>
<td>BILOU</td>
</tr>
</tbody>
</table>

Table 12. Performance comparison of PER with different hyper-parameter for flat corpora

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>value</th>
<th>Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td>2*Word Embedding</td>
<td>Glove (100d)</td>
<td>88.61</td>
</tr>
<tr>
<td></td>
<td>Cyber-embedding (100d)</td>
<td>88.14</td>
</tr>
<tr>
<td>2*LSTM Layer</td>
<td>Layer-1</td>
<td>88.61</td>
</tr>
<tr>
<td></td>
<td>Layer-2</td>
<td>89.23</td>
</tr>
<tr>
<td>2*Optimizer</td>
<td>SGD</td>
<td>88.61</td>
</tr>
<tr>
<td></td>
<td>Adam</td>
<td>88.66</td>
</tr>
</tbody>
</table>

respectively. We use universal word embedding to make the model more adaptable for real world analysis. Moreover, training on our own corpus to generate word embedding is not feasible and realistic. For flat corpora, we train our networks using the back-propagation algorithm updating our parameters on every training example, one at a time, using stochastic gradient descent (SGD) with a learning rate of 0.01 and a gradient clipping of 5. Table 11 lists the hyper-parameter used in our model for flat and nested corpora. The dropout rate is set to 0.5 for both experiments. The main idea of dropout is to randomly disable some neurons with a certain probability, which can make the model have better generalization because it does not rely on some local features.

We use two layers of BiLSTM whose dimensions are set to 100. For dependency relation
Table 13. Performance comparison with different word embedding and LSTM unit dimensions for flat corpora

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td>3*Word Embedding</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>300</td>
</tr>
<tr>
<td>3*LSTM unit</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>200</td>
</tr>
</tbody>
</table>

Table 14. Performance comparison with different word embedding for nested corpora

<table>
<thead>
<tr>
<th>Embedding</th>
<th>Precision</th>
<th>Recalls</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glove</td>
<td>65.88</td>
<td>65.34</td>
<td>65.61</td>
</tr>
<tr>
<td>Cyber-embedding</td>
<td>64.67</td>
<td>64.05</td>
<td>64.35</td>
</tr>
</tbody>
</table>

we use the concatenation interaction function. There is no difference in results for addition as well. For dependency relation, for every word in a sentence, we inferred the parent and dependency relation between them. Table 12 lists the results with different hyper-parameter for flat corpora. We see that different word embedding does not provide any performance variance in test data. The same thing goes for the optimizer as well, where sgd and adam results in accuracy and an f1-score with a very slight difference. As we can see from the table, increasing the number of layers does not give us further improvements because long order dependency does not play an important role in finding entities.

We run our model with variable embedding dimensions, and hidden layers return very slight variation in every performance metric. Another observation from our experiment is that increasing the embedding dimension and LSTM layer turns the validation score significantly higher than the test score. Higher vectors and layers have more learning states compared to lower ones. This might cause an over-fitting problem for relatively small training data. Thus we prefer to go with a single layer and an embedding dimension equal to 100. For nested corpora, a bit higher performance is shown for entity recognition. It probably
happens because of high confidence propagation between nested labels.

6.5 CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel tool called PatternMiner to leverage the enormous amount of threat intelligence to understand adversarial attack pattern. Our attack pattern definition goes beyond just technique and tactic but more granular adversarial action. Based on the evaluation results of our experiment, our proposed model is proved effective on both TTP extraction and PER for known and unknown attack pattern, respectively. We observed that security documents can be learned satisfactorily by focusing on the topical similarity of words, not functional similarity. We further found that adding more learning states does not help in recognizing a pattern while having significant performance improvement in identifying TTP. Future work is integrating contextual embedding from bidirectional language model to our model. It is expected to improve the proposed model by contextual embedding and features to extract more accurate attack pattern.
CHAPTER 7

CONCLUSIONS AND FUTURE RESEARCH

While information theft and financial profit have been the main objectives of cyber attacks, another concerning trend has emerged in the last several years. Cyber attacks on critical infrastructure such as dams, power plants, and power grids with the intent to cause physical damage are on the rise. The NIST recommended cyber resiliency framework can be sought at multiple levels, including system elements, mission/ business functions, system-of-systems, etc. Defense-in-depth architecture imposes some additional safeguard, but it fails to secure the system, and recently more stealthy attacks took advantage of it. In Chapter 3 of this thesis, we model the attacker’s opportunity in the adversary propagation path, which should be the first parameter to estimate a systems risk. Our evaluation illustrates the effectiveness of metrics for efficient defense deployment in energy delivery system(EDS) cyber infrastructure.

The opportunity analytics gives a proactive estimate of the potential path criticality in the system. But the attacker’s motivation cannot be generalized as different attackers have diverse preferences on their goals and strategies. The uncertainty of an attacker’s behavior obstructs the response decision-making process. Chapter 4 designs an intrusion response mechanism by, first, analyzing prior steps of the attack sequence and their impacts, next, extracting attacker’s characteristics from them, and finally, incorporating their influence on the next adversarial options to efficiently restrict the follow-up opportunities. My work investigates on context-centric analysis for improving cyber resilience in energy delivery systems. Previous efforts focused more on operational and architectural contexts initiated from threat events. In the future, I would like to extend this work by emphasizing threat context. Grid cyber resilience can be apprehended properly without a guided and informed understanding of the threat landscape. The threat context identifies threat sources, threat events, and threat scenarios of concern for the system of interest. I plan to build a framework as a nexus of multiple contexts incorporating the characteristics and the evolving behaviors of
adversaries having distinct goals and procedures. The recently embedded MITRE ATT&CK for ICS could be a great resource to enhance those analytics.

Regardless of the environment, achieving consistent detection of malicious behaviors is one of the perennial challenges in cybersecurity. The continual cycle of evasion by adversaries drives cybersecurity defense to put less weight on the detection based on anomalies and IoCs but more on threat behaviors. Chapter 5 proposes a methodology to assess cyber threats proactively by characterizing adversary behavior. We describe the different levels of threat indicators and their effectiveness in understanding the adversary’s activity. Furthermore, we integrate static network information with dynamic attack strategy by mapping attack graphs into attacker’s techniques and tactics. This contextual integration provides insights into attacker’s stealthy behavior. While conventional IoC changes often, threat behavior persists, reused with little modification. To shed some light on this issue, in Chapter 6, we proposed PatternMiner which addressed the emerging challenge in the effective analysis of adversarial pattern analysis. Our evaluation shows that PatternMiner can effectively infer adversarial attack pattern with high accuracy from real world threat campaigns. In the future, we intend to explore the asymmetric view of the attack surface from attacker and defender’s perspective and initiate novel transformation by augmenting extracted attack pattern. Behavioral attack surface, a new way to attack scenario representation will enable us to explore the likely adversary sequence which could match the behavior of a particular threat group or campaign. Multiple stepping stones in the attack surface associated with the same attack pattern dictates the attacker’s strategic plan to move into the network. The framework will be able to generate a high-level graph and is expected to make an important contribution to attack comprehension.
BIBLIOGRAPHY


VITA

Md Sharif Ullah
Department of Electrical & Computer Engineering
Old Dominion University
Norfolk, VA 23529

Publications


Typeset using \LaTeX.