CHAPTER 1

INTRODUCTION

1.1 Introduction to Attack Graphs

Cybersecurity has been at the forefront of computing for decades, and vulnerability analysis modeling has been utilized to mitigate threats to aid in this effort. One such modeling approach is to represent a system or a set of systems through graphical means, and encode information into the nodes and edges of the graph. Even as early as the late 1990s, experts have composed various graphical models to map devices and vulnerabilities through attack trees, and this work can be seen through the works published by the authors of [21]. This work, and other attack tree discussions of this time such as that conducted by the author of [22], would later be referred to as early versions of modern-day attack graphs [20]. By utilizing this graphical approach, cybersecurity postures can be measured at a system's current status, as well as hypothesize and examine other postures based on system changes over time.

Attack Graphs are an appealing approach since they are often designed to be exhaustive: all system properties are represented at its initial state, all attack options are fully enumerated, all permutations are examined, and all changes to a system are encoded into their own independent states, where these states are then individually analyzed through the process. The authors of [23] also discuss the advantage of conciseness of attack graphs, where the final graph only incorporates states that an attacker can leverage; no superfluous states are generated that can clutter analysis. Despite their advantages, attack graphs do suffer from their exhaustiveness. As the authors of [20] examine, even very small networks with only 10 hosts and 5 vulnerabilities yield graphs with 10 million edges. When scaling attack graphs to analyze the modern, interconnected state of large networks comprising of a

multitude of hosts, and utilizing the entries located in the National Vulnerability Database and any custom vulnerability testing, this becomes infeasible. Similar difficulties arise in related fields, where social networks, bio-informatics, and neural network representations also result in graphs with millions of states [26]. Various efforts that will be discussed in Section 2.3 demonstrate methods and techniques that can mitigate these difficulties and improve performance.

1.2 Application to Compliance

1.2.1 Introduction to Compliance Graphs

As an alternative to attack graphs for examining vulnerable states and measuring cybersecurity postures, the focus can be narrowed to generate graphs with the purpose of examining compliance or regulation statuses. These graphs are known as compliance graphs. Compliance graphs can be especially useful for cyber-physical systems, where a greater need for compliance exists. As the authors of [13], [7], and [4] discuss, cyber-physical systems have seen greater usage, especially in areas like critical infrastructure and Internet of Things. The challenge of cyber-physical systems lies not only in the demand for cybersecurity of these systems, but also the concern for safe, stable, and undamaged equipment. The industry in which these devices are used can lead to additional compliance guidelines that must be followed. Compliance graphs are promising tools that can aid in minimizing the difficulties of these systems.

A few alterations are needed to attack graph generators to function as compliance graph generators, and these alterations are discussed in Section 1.2.2. Compliance requirements are broad and varying, and can function as safety regulations, maintenance compliance, or any other regulatory compliance. In the same fashion as attack graphs, compliance graphs are exhaustive, and future system states can be analyzed to determine appropriate steps that need to be taken for preventative measures [13].

1.2.2 Defining Compliance Graphs

The common features of attack graphs serve separate purposes in compliance graphs. The nodes of an attack graph typically represent the system state that includes the qualities and topologies of all assets in the network as they pertain to cybersecurity postures. Nodes of a compliance graphs also represent the system state, however they include the qualities and topologies of all assets in the network as they pertain to compliance regulation. For instance, a quality for a vehicle's maintenance compliance could be described as: *car:months_since_oil_change=6*, or *car:miles_since_oil_change=10,000*. Edges represent changes to a system state that inserted, modified, or deleted a quality or topology. Using the car example, an edge could represent the addition of more mileage or more time since the last oil change. One large differentiation of attack graphs and compliance graphs can be seen through topologies. For assets in attack graphs, topologies typically represent a connection of assets through a digital medium. For compliance graphs, topologies not only need to represent the digital connections of assets, but also need extensions to incorporate hardware devices such as sensors, actuators, or other equipment [13]. In addition, rather than using applicable exploits or vulnerabilities, compliance violation detections should be used. An attack graph generation engine would need to use compliance parameters rather than exploit files, but would otherwise function similarly in the generation process.

1.2.3 Difficulties of Compliance Graphs and Introduction to Thesis Work

Like attack graphs, compliance graphs suffer from the state space explosion problem. Since compliance graphs are also exhaustive, the resulting networks can grow to incredibly large sizes. Compliance regulations that need to be checked at each system state such as SOX, HIPAA, GDPR, PCI DSS, or any other regulatory compliance in conjunction with a large number of assets that need to be checked can very quickly produce these large resulting graphs. The creation of these graphs through a serial approach likewise becomes increasingly infeasible. Due to this, the high-performance computing space presents itself as an appealing approach. This work aims to extend the attack graph generator engine RAGE presented by the author in [9] to begin development for compliance graph generation. The example

networks in this work will also be in the compliance graph space, specifically examining vehicle maintenance compliance. This work will also examine approaches to leverage high-performance computing to aid in the generation of compliance graphs.

1.3 Objectives and Contributions

The objectives of this thesis are:

- Extend the utility of RAGE to:
    1. Reduce the complexity required for network model and exploit file creation
    2. Expand the complexity of attack modeling
    3. Allow for the creation of an infinite sized Attack Graph, assuming infinite storage
    4. Split Attack Graphs into subgraphs to simplify analysis of individual clusters
- Implement solutions to reduce state space explosion for inseparable features while remaining exhaustive and capturing all necessary information
- Extend RAGE to function for heterogeneous distributed computing environments
- Extend and utilize RAGE for compliance graph generation