

Mission-Centric Cyber Security Assessment of Critical Systems

Jeremy Pecharich, Suzanne Stathatos, Brian Wright, Arun Viswanathan, & Kymie Tan
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109

Space missions are supported on the ground by large, complex system consisting of several interconnected and interdependent cyber components such as servers, routers, switches, and applications. Cyber attacks against the underlying cyber components have the potential to ultimately affect the confidentiality, integrity and availability of high-level missions. A fundamental challenge for system designers and decision-makers in such complex, mission-critical environments is understanding how low-level cyber events propagate through the underlying interconnected and interdependent system to impact high-level mission objectives. We present a novel model-based, mission-centric approach to perform cyber security assessments for evaluating the impact of low-level cyber events on high-level mission objectives.

Traditional approaches to cyber-security assessment can be broadly classified as either threat-centric, where the focus is on modeling threat behavior, or system-centric, where the focus is on modeling system behavior, and consequences of attacks. We present a hybrid approach, in which we first build a multi-layered model of the cyber system, and model threats to the system via generic attack trees. Then, by incorporating specific vulnerability information about the nodes in the system, our approach allows us to visualize the propagation of multiple threat behaviors through the system model. This enables a more comprehensive assessment of the cyber risk to the high-level mission objectives. We demonstrate the benefits of our approach using a system model and attack trees specific to the command-and-control system of a spacecraft. Specifically, we demonstrate how our approach enables a decision-maker to assess the security posture of the system, identify necessary mitigations and prioritize their deployment.

Acronyms

CAVE	Cyber Analysis Visualization Environment
CVE	Common Vulnerability Enumeration
CWE	Common Weakness Enumeration
C&C	Command-and-Control
MBSE	Model Based Systems Engineering
SME	Subject Matter Expert
CNA	Computer Network Attack

I. Introduction

Space assets such as satellites, drones and spacecraft are increasingly attractive targets for adversaries ranging from individual hackers, to hacker groups and nation states.¹⁻⁸ Recent incidents such as the hijacking of the NASA *Global Hawk* drone by the AnonSec hacker group,¹ malicious control of U.S. Terra and Landsat 7 satellites by the Chinese military,⁶ and data outages in the National Oceanic and Atmospheric Administration (NOAA) weather system caused by Chinese hackers² demonstrate the significant consequences of cyber attacks against such assets, and the inherent challenges for cyber security. Investigations into several of the above incidents reveal that adversaries were able to gain an easy foothold into a ground system, use that to infiltrate deeply into the target network, and eventually gain unauthorized access to critical command and control functions. For example, in the case of the NASA drone hack, attackers first used a purchased

vulnerability to gain an easy foothold on a NASA server, and leveraged that to move further into the network by compromising easily compromised secure shell (SSH)^a passwords. They used the compromised machines to install network packet sniffers, and sniff out an administrator password from the packet capture. The password was then used to gain unrestricted (*root*) access to a storage system containing backups of aircraft flight logs. The attackers eventually gleaned enough information to carry out a man-in-the-middle attack to replace the drone route file, almost crashing the drone into the Pacific Ocean.¹

A fundamental concern today is the security of ground systems responsible for command and control of remotely controlled high-value assets. The command and control functions on the ground are typically supported by a large, complex system consisting of several interconnected and interdependent cyber components such as servers, databases, routers, switches, and applications. The large scale of the system implies an increased attack surface (i.e., more exploitable vulnerabilities), while the increased system complexity provides more opportunities for attacks to propagate through the system. A challenge today for system designers and decision-makers in such large, complex, mission-critical environments is to understand how low-level cyber events propagate through the interconnected and interdependent system to impact the confidentiality, integrity and availability of high-level missions.

One approach to address this challenge is to perform a cyber security assessment of the underlying system to evaluate the security posture of the system with respect to a set of high-level goals. There are several ways to perform such an assessment. The approach taken is dictated by the high-level objective of the assessment. Assessment approaches can be classified into two broad categories: (a) *threat-centric approaches*, which focus on analysis with respect to an adversary's goals and behavior; and (b) *system-centric approaches*, which focus on analysis with respect to system goals, system entities, missions and their interdependencies. We propose a hybrid framework called the *Cyber Asset Visualization Environment (CAVE)* which combines elements of both the threat-centric and system-centric approaches to enable more complete evaluation of the impact of cyber attacks on high-level mission operations. We present a detailed comparison of the above approaches, along with an explanation of the relevant work in Section II.

Our approach enables a cyber-focused assessment of a mission-critical system in three key stages, namely, (a) system modeling, (b) threat modeling, and (c) impact assessment. The *system modeling* stage consists of specifying a detailed multi-layered model of the system, comprising of entities such as hardware, software, data and workflow processes, and dependency relationships between them. The *threat modeling* stage consists of specifying one-or-more abstract models of an adversary's behavior and objectives as *annotated attack trees*. The special annotation of attack trees, discussed later in Section IV, is the key to our *impact assessment* stage, which consists of execution of the annotated threat models over the multi-layered system model to assess the impact of cyber threats to high-level mission objectives. The graphical environment of CAVE provides an interactive approach to visualization of the propagation of multiple threat behaviors through the system model, thus enabling a more comprehensive assessment of the cyber risk to high-level mission objectives. The insights gained assist a subject matter expert (SME) to develop appropriate mitigation strategies and deploy appropriate countermeasures to ensure the confidentiality, integrity and availability of high-level missions.

CONTRIBUTIONS This paper makes four key contributions towards automating cyber-focused evaluation of impact in mission-critical environments: (1) a hybrid methodology for cyber risk assessment based on executing abstract attack tree-based threat models over a multi-layered system model; (2) a simple, pragmatic approach to annotating attack trees to prepare them for automated execution over system models (Section IV.B), (3) an algorithm to execute the annotated attack tree models over a multi-layered system model (Section V), and (4) a visual perspective in our graphical framework called the *bedsheet layout*, which enables visualization of data and process flows layer-by-layer through the system model. In addition to the above technical contributions, we also introduce a taxonomy of cyber security assessment approaches in Section II.

The rest of the paper is organized as follows. Section II presents a comparison of our approach to related approaches using our taxonomy. Section III and Section IV discuss the system and threat modeling approach, respectively. Section V presents the algorithm for performing impact assessment by executing threat models over a system model. Section VI discusses the interactive visual environment used by cyber SMEs to combine the system and threat models and perform analyses. Section VII presents a detailed case study that uses CAVE to perform a real-world impact assessment of a spacecraft system to support the

^aSSH is a network protocol used to securely access remote systems.

deployment of effective mitigations. Finally, Section VIII concludes the paper.

II. Related Work

In this section, we set the context for our work by comparing our assessment approach with other prominent approaches in the literature. We first discuss our high-level methodology for comparison which results in a useful taxonomy of cyber security assessment approaches, followed by a comparison of our approach to other approaches.

A. A Taxonomy of Cyber Security Assessment Approaches

A cyber security assessment deals with the evaluation and estimation of the security posture of a system. There are several ways to perform such an assessment, and the approach taken is dictated by the high-level objective of the assessment. Nevertheless, any cyber security assessment approach has to deal with at least two fundamental challenges: (a) modeling the system under consideration (the *system model*), and (b) modeling the threats to the system (the *threat model*). Individual approaches differ in the complexity of abstractions adopted for modeling system and threats. We observe that one way to build a taxonomy of approaches to cyber security assessment is to organize them with respect to the complexity of their system and threat models. Table 1 presents such a taxonomy, and demonstrates the novelty of our approach with respect to the related work.

Our taxonomy organizes cyber security assessment approaches along two dimensions: (a) system model, and (b) threat model. The *system model* dimension captures complexity of system information contained in the models, and we divide the complexity into five categories as discussed below. The choice of these categories was made based on an understanding of the three key aspects necessary to be modeled to perform a comprehensive mission-centric cyber security assessment, namely, assets, countermeasures and high-level missions. We define these terms along with a description of the categories below.

(1) *Model of system assets*

In this case, the system model only captures information about the assets within a system. Assets may include entities such as hosts, networks, applications, protocols, users and data relevant to a system.

(2) *Model of system assets and countermeasures*

In this case, the system model captures asset information as in the previous category, but also includes information about countermeasures in the system. Countermeasures are threat mitigation strategies deployed within a system. These could be preventive measures, such as firewalls and authentication mechanisms, defensive measures such as intrusion detection, or responsive measures such as quarantine or isolation. For example, attack-defense trees (ADTrees),⁹ defense trees,¹⁰ and attack countermeasure trees (ACT)¹¹ explicitly include abstractions for modeling system countermeasures.

(3) *Model of assets and simple mission models*

In this category, the system model captures asset information as in the above cases, but also includes some information about the high-level missions running on the assets. A *simple mission* here refers to a high-level goal described using tasks, where tasks are described using required services and other assets. For example, Argauer & Yang 2008,¹² model missions as a task or process that is carried out by a computer network. Assets are tied to missions and have a criticality score between 0 and 1 indicating their importance to a mission.

(4) *Model of assets and complex mission models*

In this category, the system model captures asset information as in the above cases, along with a complex model of high-level missions running over the assets. A *complex mission* here refers to a more elaborate mission model capturing information such as sequence of tasks, parallelism between tasks, and data/control flows across the tasks. For example, in Breu et al.,¹³ missions (referred to as business processes in their work), are modeled as a sequence of activities leading to the accomplishment of a goal, along with information objects input/output from the business processes. Similarly in Musman et al.,^{14,15} mission models contain detailed information such as workflows, dependencies, uncertainty, fallback and failover activities, along with mission measures of effectiveness, and measures of performance.

(5) *Model of assets, complex missions and countermeasures*

In this category, the system model captures asset information, complex missions, and countermeasure information. For example, the system model in the SSARE system,¹⁶ consists of infrastructure information, mission information, along with countermeasures in the form of automated system responses.

We could have defined more categories based on combination of asset models, countermeasures and missions, but we could not find sufficient related work to justify any further categorization.

The *threat model* dimension captures the complexity of threat information contained in the models, and we divide the complexity into the following five categories. The choice of these categories was made based on an understanding of how threats have typically been modeled in literature.

(1) *No Attack Model*

This category was included to accommodate cases where assessment is done only with system models. For example, the CAMUS system proposed by Goodall et al.,¹⁷ enables building a detailed system model which maps low-level cyber assets to missions and users, but does not provide any abstractions to define threats to the system.

(2) *Single attack steps*

In this category, attacks are only modeled as point events, without any information about past or future attack events. For example, in Vigna et al.,¹⁸ single events are used to characterize the effects of an attack, such as maintenance, malfunction or compromise, on a cyber asset in their model.

(3) *Attack track (sequence of attack steps)*

In this category, attacks are modeled as a sequence of attack steps. For example, in Chen et al.,¹⁹ their attacker-specific templates contain a sequence of actions an attacker could use to achieve his objectives. Similarly, in Argauer & Yang 2008,¹² correlated intrusion detection alerts are used to define a sequence of attack actions, which are then used as an input for the assessment.

(4) *Attack graph-based*

In this category, threats are modeled using attack graphs. Attack graphs depict ways in which an adversary exploits system vulnerabilities to achieve a desired state.²⁰ Attack graphs are very detailed, system-specific representations of threat behavior as they incorporate very specific information, such as connectivity and vulnerability information from the underlying system. For example, the Cauldron²¹ system combines information in the attack graphs, alert observations, and mission workflows for performing attack impact analysis.

(5) *Attack tree-based*

In this category, threats are modeled using attack trees. These are multilevel conceptual diagrams that illustrate how a system may be attacked. They enumerate all possible paths that an adversary might follow to achieve a high-level objective. Unlike attack graphs, attack trees are more generic in nature, and once defined, can be easily adapted to a broader range of systems. Examples of attack tree based approaches include the original proposal by Schneier,²² boolean logic driven markov processes (BDMP),²³ attack-defense trees (ADTrees),⁹ defense trees,¹⁰ attack countermeasure trees (ACT),¹¹ and attack response trees (ART).²⁴

Our current framework (referenced as **CAVE-2016** in Table 1) relies on a special attack tree-based formalism, as discussed in Section IV, to model threats. We choose an attack-tree based formalism over other approaches primarily for two reasons. First, attack tree specifications are very generic in nature, and can be built without knowledge of the low-level system details. This is important because it allows us to leverage or reuse threat models across projects, and across organizations. Second, attack trees have very simple semantics, which allows us to easily extend and adapt them. In Section IV and V, we demonstrate how a simple extension to the attack tree formalism enables us to automatically execute attack trees over system models.

B. Comparison of CAVE with Other Approaches

In this section, we use the taxonomy developed previously to demonstrate the novelty of our approach as compared to other approaches. We populate our taxonomy with cyber security assessment approaches

Threat Model	Attack tree-based	Schneier 1999 ²² McLaughlin et al. 2010 ²⁵ McLaughlin & Podkuiko 2010 ²⁶ Lazarus et al. 2011 ²⁷	Roy et al. 2012 ¹¹ Kordy et al. 2011 ⁹ Somestad et al. 2009 ²⁸ Raugas et al. 2013 ²⁹ Zonouz et al. 2009 ²⁴ Piètre-Cambacédès & Bouissou 2010 ²³ Edge et al. 2007 ³⁰	<i>CAVE-2016</i>	Breu et al. 2008 ¹³	<i>CAVE-NG</i>	
	Attack graph-based	Phillips & Swiler 1998 ³¹ Liu & Man 2005 ³² Sheyner & Wing 2004 ²⁰	LeMay et al. 2011 ³³	Jajodia et al. 2011 ²¹	Cheng et al. 2012 ³⁴ Anwar et al. 2008 ³⁵		
	Attack track (sequence of attack steps)			Argauer & Yang 2008 ¹²	Chen et al. 2013 ¹⁹ Phan et al. 2012 ³⁶ Musman et al. 2010 ¹⁴ Musman et al. 2011 ¹⁵		
	Single attack steps				Vigna 2011 ¹⁸ Jakobson et al. 2011 ³⁷	Ambrosio et al. 2001 ¹⁶	
	No attack model			<i>CAVE-2015</i> Goodall et al. 2009 ¹⁷	Barreto et al. 2012 ³⁸ Agedal et al. 2002 ³⁹ Taubenberger & Jürjens 2008 ⁴⁰ Anita et al. 2010 ⁴¹ Cam & Mouallem 2013 ⁴²		
	Model of system assets	Model of system assets and countermeasures	Model of assets and simple mission models	Model of assets and complex mission models	Model of assets, complex missions and countermeasures		
	System Model						

Table 1: Comparison of the *Cyber Asset Visualization Environment* (CAVE) to related work with respect to the system and threat modeling approach.

directly relevant to our work. Specifically, we included relevant related work across each of the system and threat model dimensions described previously.

With respect to the taxonomy in Table 1, we observe that the approaches can be loosely grouped into three broad categories: (a) *threat-centric approaches*, (b) *system-centric approaches*, and (c) *hybrid approaches*.

THREAT-CENTRIC APPROACHES Approaches under this category are the ones in the top-left quadrant in Table 1, that is, in the top two rows over the first two columns; In general, with the threat-centric approach, the assessment is performed with respect to adversarial goals and behavior. The primary focus here is on modeling an adversary’s actions, and only relevant information about the system is incorporated into the threat model. Attack tree-based methods such as the original proposal by Schneier,²² attack tree-based analysis of smart meter vulnerabilities,^{25,26} and attract tree based analysis of election fraud,²⁷ are all examples of a threat-centric approach. Advanced attack-tree based approaches, which include abstractions for modeling system countermeasures such as the boolean logic driven markov processes (BDMP),²³ attack-defense trees (ADTrees),⁹ defense trees,¹⁰ attack countermeasure trees (ACT),¹¹ and attack response trees (ART)²⁴

are also examples of a threat-centric approach. Attack graph-based methods such as the original proposal for graph-based network vulnerability analysis,³¹ the work on attack graphs by Sheyner and Wing,²⁰ the bayesian attack graphs of Liu and Man,³² and the ADVISE system of LeMay et al.³³ are threat-centric approaches. We emphasize that in all the above approaches, although the focus is on modeling threats, system information is always captured implicitly to the extent required.

SYSTEM-CENTRIC APPROACHES Approaches under this category are the ones in the bottom-right quadrant of Table 1, that is, in the bottom two rows over columns three, four and five. In the system-centric approach, the focus is on the consequences of attacks on the system, and not how those consequences were manifested. For example, in this approach, the assessment cares that a *server is unavailable* but not the myriad ways in which an adversary could have made the server unavailable. This approach does away with modeling an adversary's behavior for estimating his current and future actions against the system, and only models the impact that any action can have on the system. Examples of the system-centric approach include ARGUS,³⁸ CAMUS,¹⁷ Missionary,¹⁸ SSARE,¹⁶ Cyber Security Incident Model,³⁷ CORAS,³⁹ and Business Process models with security requirements.⁴⁰ An earlier version of our framework, referred to as **CAVE-2015** in Table 1, and described in Kerzhner et al.,⁴³ was a purely system-centric approach. It allowed a cyber SME to perform assessments with only the graph-based model of the system, without incorporating any threat model.

HYBRID APPROACHES Approaches in this category are the ones in the top-right quadrant of Table 1, that is, the top-three rows over columns three, four and five. Focusing on either a threat-centric or a system-centric approach to assessment has its distinct advantages, but we observe that a comprehensive cyber assessment in critical environments requires modeling both the threat and the system. Our current approach (**CAVE-2016**) falls into the hybrid category of approaches which enable assessments by combining both the system and threat models. We distinguish our current approach, by comparing it with three of the most relevant approaches within this category: Breu et al.,¹³ Chen et al.,¹⁹ and Jajodia et al.²¹

Breu et al.¹³ also propose a model-based approach for quantitative assessment of an enterprise security system. Their objective is to identify which threats have the strongest impact on business security objectives, and how various security controls might differ with regard to their effect in reducing these threats. Their system models are multilayered, similar to CAVE. CAVE does not model complex missions yet, but adds more layers to the system model. For example, CAVE incorporates a file layer to reason about threats to data. CAVE fundamentally differs in the threat modeling and analysis approach. CAVE threat models are abstract attack tree-based models, which can be executed over any system model to perform assessment, whereas the attack tree models in Breu et al.¹³ are generated specific to the system model in question.

Jajodia et al.²¹ propose Cauldron, which is an integrated cyber situational awareness framework, and provides automated attack modeling, alert correlations and mission impact analysis. The framework correlates data from a variety of sources (asset inventory, vulnerability scans, firewall/router configurations, vulnerability databases, and intrusion detection alerts), builds a model of the network connectivity and attack vulnerability, and then maps out all possible attack paths through the network (attack graphs). CAVE and Cauldron fundamentally differ in their threat modeling approach. Cauldron generates detailed attack graphs from a system model, while CAVE uses generic annotated attack trees for execution over a system model.

Chen et al.¹⁹ present a workflow-oriented security assessment framework for design-time security analysis, and in that sense are very close to our high-level objectives. Their approach focuses on building a workflow-based model (similar to a complex mission model), by combining high-level security goals, workflow descriptions, system description, attacker models and quantitative evidence such as attack success probabilities. CAVE differs from their approach both in the system and threat modeling approaches. CAVE builds a detailed multi-layered model of the system, while they annotate their workflows with all necessary system information to create a single layer model. The advantage of a bottom-up multi-layered model is that it is not specific to a single mission or workflow. In their approach, an attacker model, which contains a sequence of attacker actions, is combined with the workflow model to produce a single model for analysis. CAVE uses independently generated, generic attack trees which are directly executed over a system model.

To summarize, CAVE is unique in the way it enables executing abstract attack tree models over a multi-layered system model to assess risks to high-level mission objectives. The current version of CAVE does not model complex missions or countermeasures, but we expect to incorporate these enhancements into the next version of our framework, referred to as **CAVE-NG** in Table 1.

III. System Modeling

In this section, we present our multi-layered approach to modeling the underlying networked system. We will discuss the basic entities and layers in our model, relationships between entities within each layer and across layers, followed by a discussion of the model properties which enables interesting analyses.

A. Model Entities and Layers

The idea of modeling a computer network as a graph is natural as it mimics the actual structure of the network. In order to understand the security risk of a mission one needs to expand beyond the network graph to include vertices such as applications and files.⁴⁴ This idea was further considered by Grimaila and Fortson⁴⁵ where they viewed the data on the hardware as the critical assets of the cyber domain. For example, to control a spacecraft it is not enough to compromise a server, but you must also affect the data contained on that server. We have expanded beyond their model to include the mission objectives that are supported by that data.

The model can be broken up into 4 abstract layer types: Hardware, Software, Files, and Workflow Processes. A Workflow Process represents an individual mission task that constitutes the larger mission goal. For example, if the mission goal is to command a spacecraft then a workflow process for that mission goal would be *create command files*. The Hardware layer can be further refined into the individual types that make up the hardware infrastructure of a typical enterprise network such as Servers, Laptops, Workstations, Switches, Routers, and Firewalls. Working from the top down, the Workflow Process layer consists of high level mission tasks or objectives, the File layer consists of data that support those Workflow Processes, the Software layer consists of applications that support those data products or a Workflow Process. For each unique Hardware, Software, File, and Workflow Process there exists a *vertex* in the graph. We discuss the *edges* of the graph below.

Associated to each vertex are *attributes* which make up the properties of the model. For example, to each server vertex we can associate the Hostname, IP address, OS, Groups, Users, and/or Authentication schema. The attributes can be adjusted to fit the unique needs of each different enterprise network. Table 2 lists typical attributes to the layers from above. It should be noted that in the Software layer we do not include all the applications installed on the servers, but only those that support other layers. However, the applications installed on the system do have an impact on the security posture, e.g., software vulnerabilities,⁴⁶ so third-party applications are included as attributes to Server vertices.

Type	Attributes
Laptop	Hostname, OS, IP address, User, Authentication, Encryption
Server	Hostname, OS, IP address, Groups, Users, Authentication, VirtualOrPhysical Machine, Third-Party Software
Application	Programming Language(s), Mission Function
File	Format, Permissions

Table 2: Typical Model Types with Attributes

B. Relationships

As discussed above the components of the network provide vertices of a graph, the edges of the graph are specified by connections between the different components. The edges between the hardware components are given by the direct physical or virtual connections. The connections between the other layers are compositional. For example, an edge will exist between a server and a software vertex if the software is installed on the server. A directed edge between a software vertex and a file vertex will exist if the file is either an input of the software or an output of the software, where the source of the edge is the input and the target is the output. Similarly, an edge exists between a workflow process and another vertex if that vertex is either an input to the workflow process or an output of the workflow process. In this way, the system model inherits a *directed graph* structure. See Figure 1 for a sample graph with a multilayered structure.

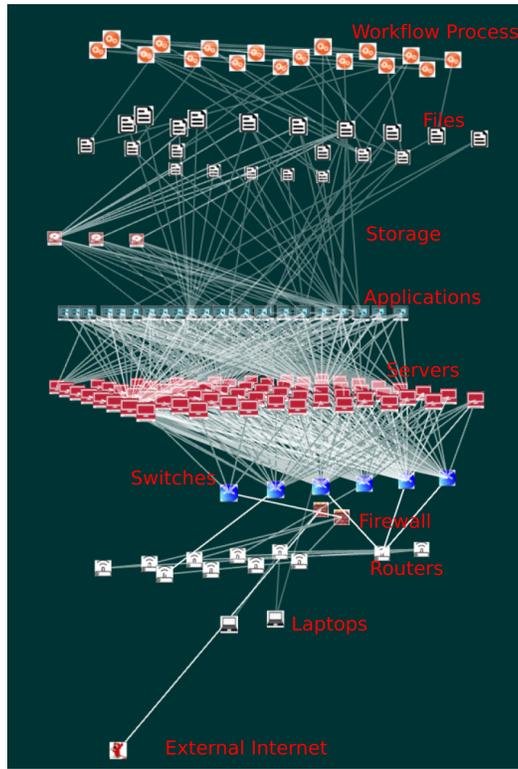


Figure 1: Multilayered system model as visualized with CAVE (with added labels)

C. Analyses using System Model

As the system model is by nature a directed graph, many different analyses can be performed using graph theory. Some basic analyses that can be performed are adjacency questions such as which mission applications are being supported by a given server. Some more complicated questions that can be answered through graph analysis are which servers are most connected based on degree centrality or generalizations such as eigenvalue centrality.⁴⁷ One can also perform reachability analysis such as does there exist a path from the External Internet to a certain mission application. If such a path does exist, then which path is the shortest path. This will allow a cyber SME to quickly analyze areas to place sensors or further defensive capabilities. For a detailed list of analyses that can be performed within CAVE see Section VI.

IV. Threat Modeling

In this section, we present our attack tree-based approach to modeling threats. We first provide a brief primer on attack trees, followed by a discussion of our threat modeling and analysis approach.

A. Introduction to Attack Trees

One approach to assessing the security of a system involves analyzing adversarial behavior and strategy. Threat modeling provides a means to understanding the security, or lack of security, in a system, along with the costs and risks associated with the levels of security. Through threat modeling, analysts can understand what the attack goals are, characterize who the attackers are (in terms of their motivations, levels of access, skill, and risk aversion), calculate the likelihood of certain attacks, understand security assumptions made when designing a system, and understand where to best allocate resources to alleviate security threats. Attack trees are a popular method of modeling cyber security threats.²²

Attack trees are multilevel conceptual diagrams that illustrate how an asset, target, or process may be attacked. They enumerate all possible paths that an adversary might follow to achieve a high-level objective. Attack trees allow one to build a knowledge base to describe the security of a system. They combine and

capture security and systems' engineers expertise, thus providing systems engineers with tools to make decisions regarding system security. Graphically, they are represented as a *directed rooted tree* such that the orientation of the edges is away from the root. We first discuss the terminology relevant to an attack tree followed by an explanation of how we use them to model threats.

TERMINOLOGY An attack tree broadly consists of one root, leaves, and children. The **root node** is an adversary's high-level objective; it is the only node within the tree that does not have a parent node. When an attacker has reached the root node, all sub-nodes of a path have been satisfied and the attack has completed successfully. The root node has an in-degree of zero. A node v is a **child node** of a node w if node w immediately precedes node v on the path from the root node to v . The node w is said to be the *parent* of node v . Each parent node contains a boolean expression (AND or OR) to describe the children nodes. This boolean expression will be called a condition. If the parent node is labeled AND, each child node state must be satisfied to reach the parent node, this is denoted in the attack tree by \square . If the parent node is labeled OR, only one child node state must be satisfied to reach the parent node, this is denoted in the attack tree by \sqcup . We have constructed the trees in such a way that every child of a parent node is of the same condition (i.e. AND: all children must be satisfied; OR: only one child must be satisfied). We assume, therefore, that a parent will never have a mix of child conditions to satisfy.

A group of nodes are **siblings** if they have the same parent node. A group of nodes are **independent siblings** if they are siblings and their parent is a conditional OR. A group of nodes are **conditional siblings** if they are siblings and their parent node is a conditional AND. An **internal node** is a node with out-degree greater than zero.

A **leaf node** is a node with no children. An **entry node** is a node from which an attacker can begin an attack path to the root node. Entry nodes are always leaf nodes (though not all leaf nodes are entry nodes). If the parent node of a leaf node is labeled OR, the leaf node is an entry node. If the parent node w of a group of siblings is labeled AND, such that every sibling is a leaf node, then the leftmost sibling is an entry node. No other siblings within that group are entry nodes.

B. Modeling Threats With Attack Trees

Attack trees have been used to understand several different aspects of risk, such as threats to physical systems, threats that tamper with electronics systems and threats on computer systems. We focus on attack trees that outline how to compromise computer control systems. In general, we follow a two-step process to model threats in our approach: (1) given a high-level attacker goal, construct an attack tree; and (2) annotate the attack tree for execution over the system model.

ATTACK TREE CONSTRUCTION The high-level attacker goal becomes the root of the attack tree. For example, if an attacker's ultimate goal is to compromise a spacecraft, the root of one attack tree may be titled "Compromise Spacecraft". The root node is then decomposed to construct broad subgoals from there. These subgoals, for example, may include nodes describing how to compromise the spacecraft, such as "Disrupt radiating dish" and "Tamper with commands files radiated to the spacecraft"; and nodes describing circumventing getting caught before successfully compromising the spacecraft, such as "Anonymize identity of spacecraft compromiser". Each of these subgoals have subgoals of their own, except the leaves of the attack tree. We make the nodes general for many reasons, chief among them being reusability. We intend to apply the attack trees to several spacecraft computer control systems, as well as to industrial control systems. For example, an attack tree to gain *command and control of spacecraft Y*, at a high-level will be similar to an attack tree to *command and control spacecraft X*.

ATTACK TREE ANNOTATION In the second step, we annotate the trees with properties that would link them to a more detailed graphical representation of the system. This will be further discussed below.

One of the main difficulties in executing an attack tree on a system model is that the language of an attack tree is written in conversational English. Moreover, the shorthand notation will differ between authors of the attack tree, cf., Section VII.D. Therefore, in the second step, we annotate leaf nodes of the attack tree with attributes contained in the system model. Recall, the *leaf nodes* of an attack tree represent the actions of the adversary on the system and the *internal nodes* represent the consequences of those actions, hence it is sufficient to annotate only the leaf nodes. The annotation will allow the leaves of the attack tree to be mapped to assets within the system.

C. Example Threat Model To Command & Control a Spacecraft

In this section, we present a fictitious ground system for a spacecraft, followed by an annotated attack tree-based threat model for command and control of the spacecraft by an adversary.

Example Spacecraft System As shown in Figure 2, the spacecraft computer control system consists of a collection of servers in various zones (a zone is a collection of subnets) with differing levels of protection.

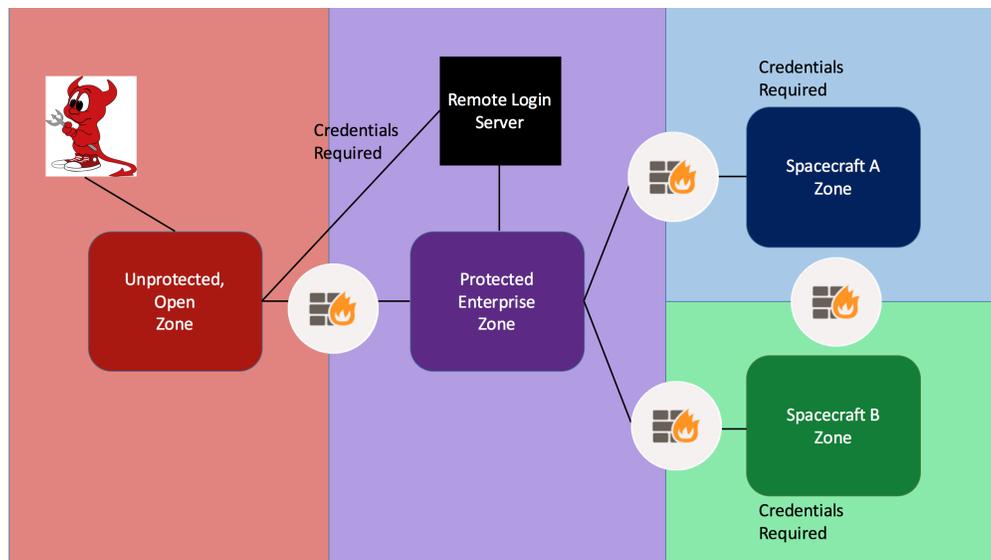


Figure 2: A fictitious spacecraft command & control system showing the key network zones, and firewalls to protect data flows between zones.

Spacecraft-specific zones - Servers located within spacecraft-specific zones (for example, “Spacecraft A”, and “Spacecraft B” in Figure 2), have varying functions. Some servers run applications to generate data needed for command generation and execution, some servers act as storage databases, and some servers exist to have redundancy in the control system. In this zone, some servers are inaccessible unless a user is physically within the organization, connected to the spacecraft-specific network zone, and the user has permission to access that server. These servers are protected by a zone firewall. Spacecraft command files for that mission are stored within a database on these protected servers. The spacecraft command files have access control and have multiple integrity checks. After verifying these checks, an operator will queue the spacecraft command files to be radiated to the spacecraft.

Protected enterprise zone - Servers within this zone are accessible through the enterprise network. Employees within the enterprise network have remote login capabilities.

Unprotected open zone - Servers located within this zone are internet-facing so that they can be accessible by foreign partners for collaborative purposes.

Example Threat Model The threat model demonstrates how an attacker tries to move through these system zones described above, to accomplish his malicious objectives.

As shown in Figure 3, there are four entry nodes: “Install malware at local internet café”, “CNA Server that Allows Remote Sign-on”, “CNA Firewall”, and “CNA External Facing Server”. An attacker can choose any of these as entry points to accomplish a computer network attack (CNA) to gain a toehold into the protected enterprise network. Let’s say the attacker chooses to “Install malware at local internet Café”. This attack preys on the fact that enterprise employees have and use mobile devices (e.g. laptops, phones, tablets) on untrusted networks outside of work. The attacker will need to install malware on an enterprise machine through an untrusted network. Once malware is installed onto the machine, the attacker will wait until the employee (owner of the mobile device) returns to work. There, the employee will connect to the protected zone, and malware will have permeated to a device on the protected zone. Once within the protected network zone, the attacker will try to access and attack a server in the spacecraft-specific zone. Specifically, the attacker will need to attack a server with access to a database that stores command

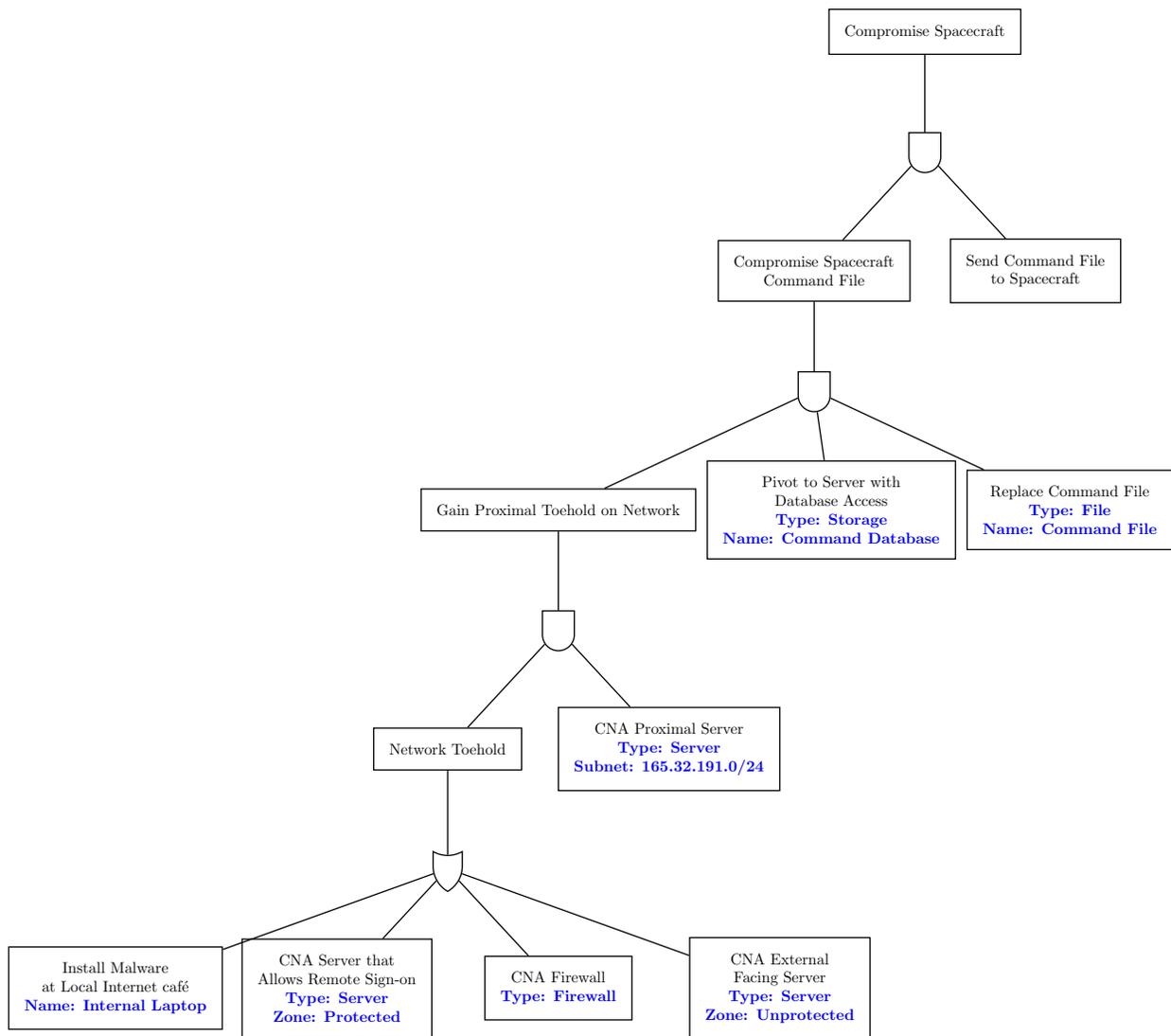


Figure 3: Annotated Attack Tree for Command & Control of Spacecraft

files. Once the attacker has completed this step, the attacker will have successfully reached the “Pivot to Server with Database access” internal node. Once the attacker has access to the database, (s)he will need to locate a spacecraft command file and manipulate the access control permissions to allow the attacker to modify the file. After compromising the spacecraft command file, the attacker will need to wait until the spacecraft command file is radiated to the spacecraft. After the compromised commands have been sent to the spacecraft, the attacker will have successfully compromised the spacecraft.

As discussed in the previous section the next step in constructing the threat model is to annotate the attack tree with attributes from the system model. For example, the phrase “*Install malware at local internet café*” is shorthand for installing malware on an internal computer which can be taken offsite. In this case we would annotate this node with the properties of (**type=Laptop, name=Internal Laptop**). The annotations of the leaf nodes of the attack tree discussed above are highlighted in blue in Figure 3.

The generality or specificity of the annotations is at the users discretion. For example, the designer of the attack tree could have meant that the malware to be installed would only target OSX operating systems with Firefox 43.0.1. In which case the annotation would be

(**type=Laptop, name=Internal Laptop, OS=OSX, app=Firefox 43.0.1**).

We list some attributes of the model types typically found in an enterprise network in Table 3.

Type	Attributes
Laptop	Name, OS, Application, Group, Username, CVE, Zero-Day
Server	Name, OS, IP address, Subnet, Groups, Application, Group Username, CVE, Zero-Day
Application	Programming Language, CWE, Function,
File	Format, Permissions

Table 3: Typical Model Types with Attributes

V. Algorithm for Mission Impact Assessment

Once the attack tree has been annotated it must be reconciled with the system model. The basic idea behind the algorithm is to match up the annotations of the attack tree with attributes of the system model. In more detail, a user will first select an entry node of the attack tree. The entry nodes are distinguished amongst the leaf nodes in the attack tree since they are the first actions that an adversary must take in order to gain access to the larger goal. The choice of an entry node will then determine a unique path in the attack tree with which the adversary can reach the root node. See Figure 4 for the path to the root node if the adversary executes the node “*Install malware at local internet café*”. For ease of the discussion, we are assuming that there is at most one internal vertex with a combinator of *AND* on each level. The above discussion still holds, but the choice of an entry point does not lead to a unique path in the attack tree. Instead the choice of an entry point will lead to n paths where n is the number of entry points on the subtree with root node equal to the other sibling vertices with a combinator of *AND*.

1. Algorithm to find the unique path

Once the entry node is selected, all unique paths from the root to that node will be computed. The paths are returned as a list of dictionaries. Each parent-child or parent-children tuple is captured in the following format:

```
{ parent: [ child(ren) ] }.
```

For example, looking at the attack tree in Figure 4, then, if “Install malware at Local Internet café” was selected as the entry point, the following attack path would be generated:

```
[
  { Network toehold                : [ Install malware at local internet cafe ] },
  { Gain Proximal Toehold on Network : [ Network toehold, CNA Proximal Server ] },
  { Compromise spacecraft command file: [ Gain Proximal Toehold on Network,
                                         Pivot to Server with Database Access,
                                         Replace Command File ] },
  { Compromise Spacecraft          : [ Compromise spacecraft command file,
                                         Send Command File to Spacecraft ] }
]
```

The function that generates these paths recursively traverses from the root node to the entry node. The function first checks if the node’s parent has condition *AND* or *OR*. Recall that the root will not have a parent, and will always have children, so it is automatically added to the path. If the node’s parent is *OR*, the node will be added to the path without its individual siblings. If the node’s parent is *AND*, each of the node’s conditional siblings, along with the paths to those conditional siblings, must be added to the path. The algorithm terminates when it has added all paths from all entry nodes to the root node. Finally, we prune the list of paths for those starting with the entry node that the user selected. Pseudocode to get all the paths from all entry nodes to the root node is shown below.

After the user has selected an entry node, the first step in the algorithm is to find all nodes within the system model whose attributes match the annotation of the selected entry node. If there do not exist any nodes in the system model with those attributes, then the attack through that entry point will fail and the

Algorithm 1 Get all paths from all entry nodes to the root node of Attack Tree

```
1: function GETPATHRECUR(currentNode, parent)
2:   if parent.condition == AND then
3:     add {parent : [node, child2, ..., childn] } to path in allPathsList
4:     add children to list of neighbors to explore
5:   else if parent.condition == OR then
6:     for child in parent.children do
7:       duplicate path and add {parent : [child] } to duplicated path
8:       GetPathRecur(child, leftmost child of child)
9:     end for
10:  end if
11:  if node is leaf node then
12:    if node is conditional sibling then
13:      add the paths from the other conditional siblings
14:    else if node is an independent sibling then
15:      the path has ended
16:    end if
17:  end if
18:  return list of neighbors
19: end function

20: function GETPATHS(entryNode)
21:  allPathsList = list()
22:  neighbors = GetPathRecur(currentNode=root, parent=NULL)
23:                                     ▷ Add subtrees from conditional siblings if conditional siblings were found
24:  for neighbor in neighbors do
25:    GetPathRecur(currentNode=neighbor, parent=neighbor.parent)
26:  end for                                     ▷ Prune allPathsList for those starting with entryNode
27:  entryNodePaths = list()
28:  for path in allPathsList do
29:    if Path begins with entryNode then
30:      add path to entryNodePaths
31:    end if
32:  end for
33:  return entryNodePaths
34: end function
```

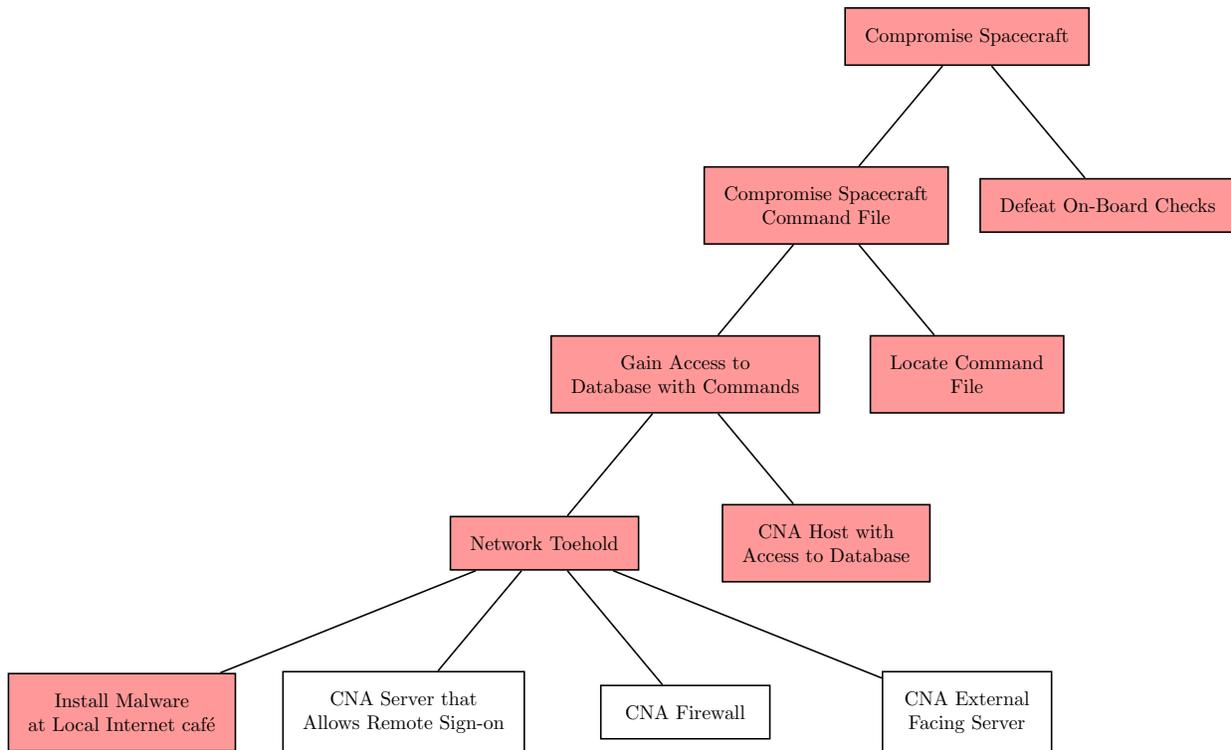


Figure 4: Attack path for command & control of spacecraft with a single entry node selected

user will be prompted with a message that the attack has failed. However, if the number of selected system model nodes is greater than zero then the user is prompted to make a selection in the visualization.

The algorithm will then proceed to the sibling directly to the right and begin the selection process based on the attributes of that node. If a node in the system model is selected by the user which is not equal to the current node in the system model, then the shortest path between the two nodes is taken. This can provide valuable insight to the user by showing which network devices the adversary must traverse in order to pivot from an internal laptop to a server. If all the siblings have been successfully traversed, then the algorithm will proceed to the parent node and repeat the search on the parent node's siblings. The algorithm will terminate when the root node is reached or there exists a leaf node with an empty selection in the system model.

It should be noted, if the leaf node has a Server or a Laptop as the annotated *type* then the algorithm by default will search for those nodes in the model with CVEs whose *Access Vector* is equal to *Network* and *Allows Access* is not equal to *None*. In other words, we search for hardware with vulnerabilities that allow an adversary to gain privilege via a network based attack. However, if the attribute *Zero-Day* is set to be *True* then all Laptops or Servers will be selected from the system model, and further selection from those can occur based on the other attributes such as Application or OS to which the Zero-Day applies. A similar search can occur if a specific CVE id number is entered as the attribute, e.g., CVE-2015-7112.

VI. Cyber Analysis Visualization Environment (CAVE)

In this section we discuss the CAVE graphical user environment, which allows an SME to combine the system and threat models, and perform cyber security assessment to understand how low-level cyber events propagate through the system to eventually impact the confidentiality, integrity and availability of high-level mission operations.

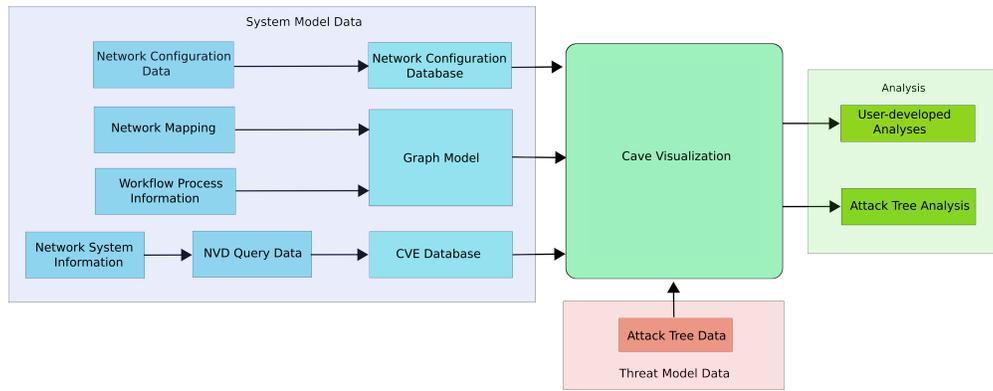


Figure 5: CAVE Architecture

A. Modeling in CAVE

CAVE provides an interface to combine the threat-centric model and the system-centric model to conduct cyber-focused evaluations on command and control systems. Network configuration data, CVE data, and the system graph model contribute to the system-centric model while the Attack trees provide the threat-centric model. While CAVE receives separate sources of data, they are closely related and contribute to a functional model. Figure 5 outlines the architecture of CAVE.

The core of the CAVE system model is based on network mapping and workflow process information. Network mapping data is retrieved programmatically, and consists of the hardware and software operating on the computer model. Workflow process information are gathered from SME's and converted to a machine-interpretable form. Network mapping data and workflow processes are associated with vertices in the graph model. Relationships and connections that are described between network machine information or process procedures are represented as edges in the graph model.

B. Model Navigation

Users navigate the graph primarily by a 3D camera system that supports common mouse behaviors. To spin the graph, one must click and drag. Zooming is supported by mouse wheel interaction, or a modifier key with a drag. A user can navigate to specific entities by double-clicking them directly in the 3D view. From the graph model, we use a 3D visualization system to visualize the directed graph. Vertices and edges have extensible representations that may be re-defined for any given network. The directed graph serves as a table top model used for discussion and exploration.

The visualization system renders the directed graph in 3D space to allow for greater separation of graph elements to accommodate large network models. Multiple layout algorithms are available to arrange the graph. Most layouts arrange the graph into 3D space, but 2D layouts are also supported, primarily for smaller data sets. Most graph layouts are provided through the graph library IGraph.⁴⁸ IGraph's layout algorithms depend largely on graph topology. Graph layouts that utilize network semantics must be configured in a custom fashion.

BEDSHEET LAYOUT One helpful layout we have developed is called the *Bedsheet Layout*. The Bedsheet layout separates the graph into a stacked view where each vertex type is isolated to a layer, as shown in Figure 6. The order of the layers serve to show how data and process flow through the current model. External entry points and unprocessed data occupy the lower layers, while project goals that utilize processed data tend to be present on the higher layers. This graph perspective also helps to emphasize dependencies of each layer, as data for a certain layer often needs some processing done by a lower layer. In our graph models, edges typically do not connect vertices of the same vertex type, so the bedsheets layout will also organize connections to be visually associated with their respective layers.

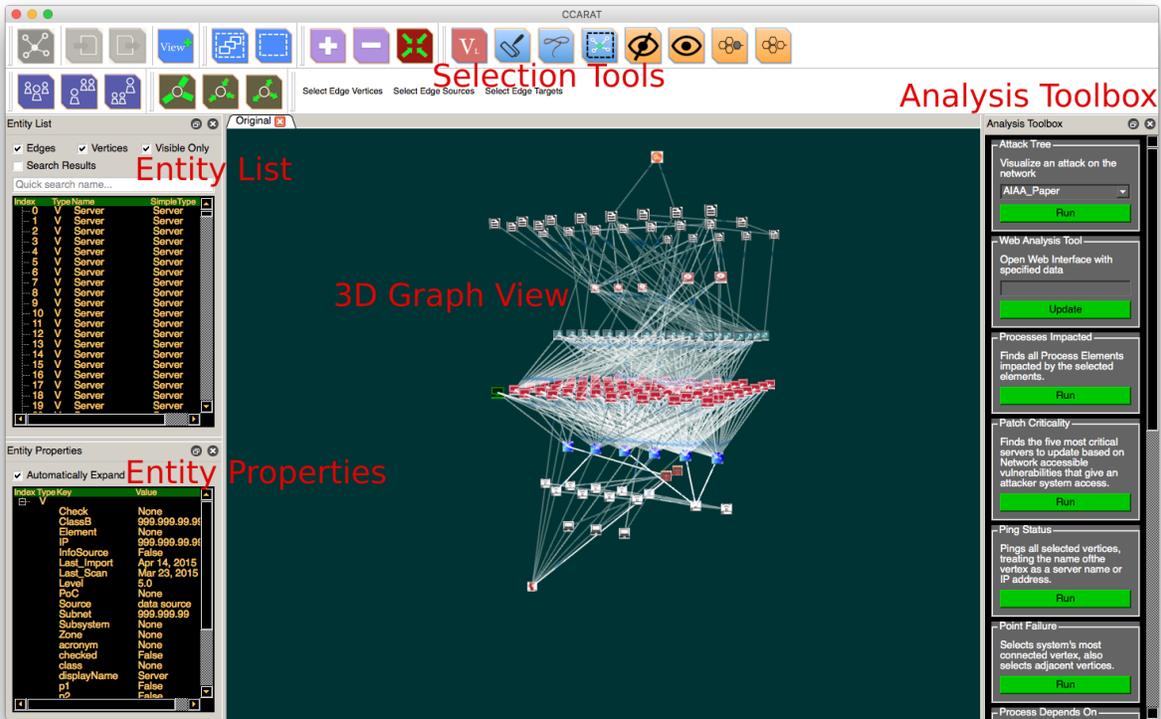


Figure 6: Default View of CAVE Interface with a Graph Model in the Bedsheet Layout

C. Searching and Selection

Graph entities encapsulate all available system properties, so it is important that there is a quick way to access any arbitrary entity. An *Entity List* interface, searchable by name, is provided to quickly find specific entities of interest. The list can also be filtered based on their graph entity type (vertex or edge), and whether or not an entity is visible. Entities can be selected directly from the list and highlighted in the 3D view to indicate correspondence. Whenever an entity is selected, the *Entity Properties* table is populated with attributes of a selected entity for quick viewing.

Selection serves as a way for a user to explore through the given network, and is performed in the 3D view by mouse-clicking on entities. CAVE also provides convenient graph selection functionality. From a vertex, all incident or outgoing edges can be quickly selected by menu buttons or a hot key combination. From an edge, source and target vertices can be selected. Graph selection based on network semantics are available, but they must be custom-fit to each network model. Selection can also be performed through Selection Tools such as a lasso selection, rectangle selection, or paint selection. Graph elements may also be hidden to increase focus on other elements.

D. Analytical Capabilities of CAVE

CAVE provides a programmatic interface for cyber-experts to develop analysis scripts to run on a network model. The CAVE interface allows for an analysis script to navigate the network graph, specify selectable inputs from the model, store/modify data in the graph, and specify how graph elements should be visualized.

Analyses are based on graph semantics, so extensive knowledge of the model is required to write an analysis script. The analysis scripts can be developed externally to CAVE, but the CAVE interface provides useful introspection into the network model for easy debugging. Analyses are run from the analysis toolbox on the right-side of the CAVE interface. The toolbox is auto-populated from available analyses that correctly accommodate the programmatic interface. Each analysis is executed concurrently in a separate thread of execution to help accommodate higher levels of computation. Currently, CAVE plugins exist to perform

analyses such as finding shortest paths and adjacent vertices, finding the most connected vertices (for assessing criticality of a Server), prioritizing server patching using CVE data, and cyber impact assessment on missions as described later in Section VII.

Scan_Type	Program	Version	CVE	Base_Score	Access_Vector	Complexity	Authentication	Confidentiality	
1	CREDENTIALLED	openssl	1.0.1i	CVE-2016-2842	10	NETWORK	LOW	NONE	COMPLETE
2	CREDENTIALLED	openssh	6.9	CVE-2016-1907	5	NETWORK	LOW	NONE	NONE
3	CREDENTIALLED	openssl	1.0.1i	CVE-2016-0800	4.3	NETWORK	MEDIUM	NONE	PARTIAL
4	CREDENTIALLED	openssl	1.0.1i	CVE-2016-0799	10	NETWORK	LOW	NONE	COMPLETE
5	CREDENTIALLED	openssl	1.0.1i	CVE-2016-0798	7.8	NETWORK	LOW	NONE	NONE
6	CREDENTIALLED	openssl	1.0.1i	CVE-2016-0797	5	NETWORK	LOW	NONE	NONE
7	CREDENTIALLED	openssh	6.9	CVE-2016-0778	6.5	NETWORK	LOW	SINGLE_INSTANCE	PARTIAL
8	CREDENTIALLED	openssh	6.9	CVE-2016-0777	4	NETWORK	LOW	SINGLE_INSTANCE	PARTIAL
9	CREDENTIALLED	openssl	1.0.1i	CVE-2016-0705	7.6	NETWORK	HIGH	NONE	COMPLETE
10	CREDENTIALLED	openssl	1.0.1i	CVE-2016-0704	4.3	NETWORK	MEDIUM	NONE	PARTIAL
11	CREDENTIALLED	openssl	1.0.1i	CVE-2016-0703	4.3	NETWORK	MEDIUM	NONE	PARTIAL
12	CREDENTIALLED	openssl	1.0.1i	CVE-2016-0702	1.9	LOCAL	MEDIUM	NONE	PARTIAL
13	CREDENTIALLED	openssh	6.9	CVE-2015-6565	7.2	LOCAL	LOW	NONE	COMPLETE
14	CREDENTIALLED	openssh	6.9	CVE-2015-6564	6.9	LOCAL	MEDIUM	NONE	COMPLETE
15	CREDENTIALLED	openssh	6.9	CVE-2015-6563	1.9	LOCAL	MEDIUM	NONE	NONE
16	CREDENTIALLED	openssh	6.9	CVE-2015-5600	8.5	NETWORK	LOW	NONE	PARTIAL
17	CREDENTIALLED	openssl	1.0.1i	CVE-2015-3197	4.3	NETWORK	MEDIUM	NONE	PARTIAL
18	CREDENTIALLED	openssl	1.0.1i	CVE-2015-3196	4.3	NETWORK	MEDIUM	NONE	NONE
19	CREDENTIALLED	openssl	1.0.1i	CVE-2015-3195	5	NETWORK	LOW	NONE	PARTIAL
20	CREDENTIALLED	openssl	1.0.1i	CVE-2015-3194	5	NETWORK	LOW	NONE	NONE
21	CREDENTIALLED	openssl	1.0.1i	CVE-2015-1792	5	NETWORK	LOW	NONE	NONE
22	CREDENTIALLED	openssl	1.0.1i	CVE-2015-1791	6.8	NETWORK	MEDIUM	NONE	PARTIAL

Figure 7: CVE Search Window

Vulnerabilities for a model can be analyzed directly with the *CVE database*. Figure 7 presents the interface that a user interacts with to query CAVE's internal *CVE database*. A user can use the *CVE Filter* search directly for CVEs based on their Name, Program, or Version, or they can select desired properties of a CVE. The attributes Access Vector, Access Complexity, and Authentication categorize the level of exploitability for a vulnerability. Confidentiality, Integrity, and Availability help illustrate a level at which a system could be compromised. The base score is calculated using the six aforementioned attributes, and provides a separate metric for the vulnerability of a program. Please refer to Kerzhner et al.⁴³ for details on base score computation. Gain Access serves to show what level of authentication a vulnerability may provide on a given machine. From these selections and/or searches, a CVE database query is processed, and the *CVE Result Table* is populated showing the resulting CVEs.

CVEs can be selected from the *CVE Result Table* for quick insights into hardware for a network model. By selecting a CVE in the table, we can directly highlight all hardware vertices in the 3D view that contain the selected CVE. Inversely, an SME can select hardware in the 3D view, and populate the table with all CVEs contained on the selected hardware.

E. Technical Implementation

The core CAVE application is built using Python 3.4.⁴⁹ Qt 4.8 is utilized through the python binding PyQt4, to build the user interface elements.⁵⁰ OpenGL through PyOpenGL is used to implement the 3D view into the network model. Qt provides an integration layer to work with OpenGL embedded into a Qt application. Python and Qt were chosen based on previous interface development expertise. Python also leverages an extensive standard library, ease of development, and cross-platform capabilities. CAVE is currently deployable on Mac OS X 10.7 and higher, Ubuntu 15, and Windows 7.

IGraph is used to store our graph representation of our network model. IGraph also provides basic network analyses, graph layout algorithms, and topology-based graph traversals. CAVE leverages a database for storing network data in a centralized location. The network data is stored as a Neo4j⁵¹ graph database object supporting queries that follow a graph paradigm. A large benefit provided by the database is the ability to support data requests from multiple platforms. Any platform ready to query a REST API can be given access to CAVE data with proper authentication. A central data store will also allow for ease of access of data for multiple users of CAVE.

VII. Evaluating the Impact of Cyber Attacks on Missions

In this section, we discuss how CAVE assists an SME in performing cyber security assessment of their mission critical system. We first describe a real use case, and then detail the overlay of the example attack tree from Figure 3.

A. Description of Use Case

THE BACKGROUND An SME knows that a mission critical system has weak points in its cyber security. The SME is trying to determine what mitigations can and should be put in place on that system. (S)he is constrained by a tight budget and inherited software/systems. Thus, the SME is limited to what mitigations can be put in place, and how many mitigations can be implemented. Therefore, the SME must be tactical about the mitigations added to the system.

GOAL The SME wishes to place in mitigations with the most impact. The SME hopes to block several attack paths with one mitigation. The SME must find where to add a mitigation, and know that it will have a positive effect on the cyber security of the system.

USING CAVE TO FIND THE RIGHT SOLUTION As discussed, CAVE provides a natural interface to view the entire system model, and CAVE has the analytical ability to map common computer control system attacks onto specific mission critical systems. An SME decides to use CAVE’s attack tree capabilities to identify the best places to add mitigations. The SME runs through several attack trees on the system model and identifies common vulnerable vertices. The SME also observes which differing attack entry nodes reach those vulnerable vertexes. One vulnerable vertex that the SME identifies is a server in the spacecraft protected zone. The server hosts a vulnerable SQL database, and does not filter any of its traffic. The majority of attack paths go through this server. Thus, the SME quantitatively identifies that this asset’s mitigation has high priority. The SME derives mitigations to best protect the server from the various threats along the attack paths. One mitigation for the identified server involves purchasing a firewall, so the SME chooses to do that. Therefore, CAVE identified a common point of attack in the system, and the SME derived a mitigation to prevent this attack path from successfully executing.

B. CAVE Visualization

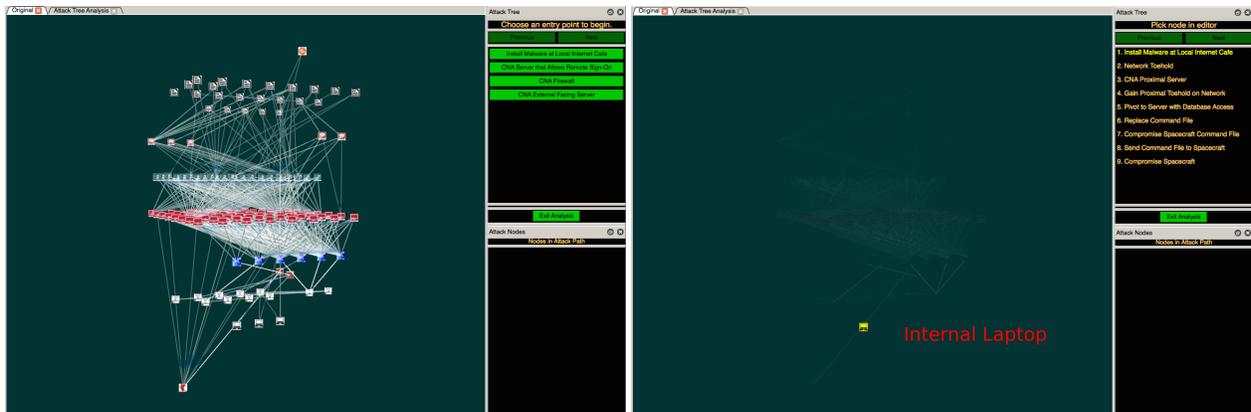
After an attack tree has been designed and annotated to fit the system model it is then ready to be visualized in CAVE. The attack tree is encoded as a comma separated file (csv) where the annotations along with the structure of the attack tree are contained in the columns.

Once the csv file has been inputted into CAVE the user will select an entry point from which to begin the attack (Figure 8a). In this example, we select the node in the attack tree corresponding to “*Install Malware at Local Internet Café*”. Then the unique path from the selected node to the root node is found using the algorithm from Section V. The nodes from the path in the attack tree are displayed in a panel on the right hand side of the user interface, cf., Figure 4. The algorithm will now search for those vertices contained in the system model matching the annotations contained on attack tree node “*Install Malware at Local Internet Café*”, i.e., **Name = Internal Laptop** (Figure 8b). If such vertices exist within the system model, then the user can select a vertex from those highlighted in the user interface. In the case that no such vertices exist with those attributes, the user is given an *Attack Failed* message and will need to choose another entry point in the attack tree.

The user will then step through the attack tree until the next *leaf node* with an annotation is selected. From example above this node is “*CNA Proximal Host*” with the annotations **Type=Server, Subnet=165.32.191.0/24**. Since this node has the type of Server only those server vertices with CVEs that allow access over the network are selected in the system model (Figure 8c). The red vertex indicates the server with the highest number of vulnerabilities allowing an adversary to gain access over via a network based attack. Moreover, the CVE search window in Figure 7 also displays the CVEs with the above properties for the user to investigate. The user can select either a server vertex in the UI or subset the server vertices by selecting a CVE. The shortest path from the Internal Laptop to the selected server vertex is then displayed to the user along with the names of the components in the right-hand side of the UI (Figure 8d). If no such path exists the user is prompted to select another server vertex until a path exists. In the case

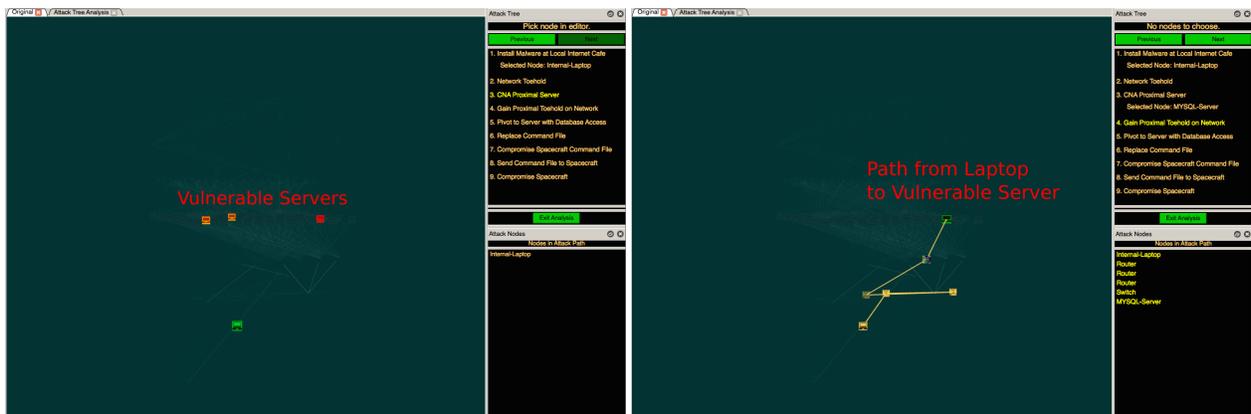
when all the server vertices have been exhausted, a message is displayed indicating that the attack was not successful.

The user will then continue through the rest of the attack tree until a leaf node is reached with an empty selection or the root node of the attack tree has been reached (Figure 8e,8f). If the root node has been reached, then the attack where the adversary's attack starts at the selected entry leaf node has been successful.



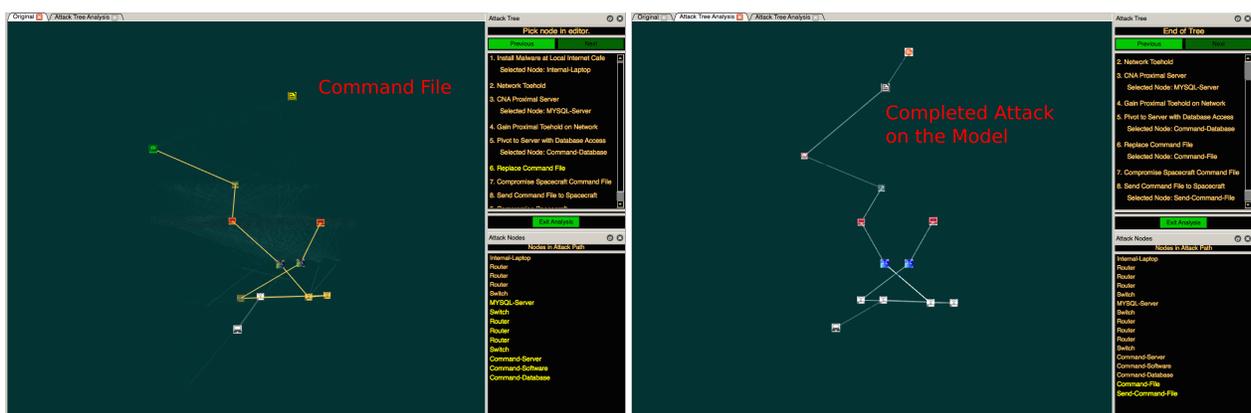
(a) Entry Nodes of the Attack Tree

(b) Selection of a Node



(c) Selecting a Server Vertex

(d) Traversing the Model through the Attack Tree



(e) Selection of a Node

(f) Completion of the Attack Tree

Figure 8: Visualization of Attack Tree

C. Impact Assessment With CAVE

We now discuss how CAVE improves the impact assessment process for an SME. As discussed in the introduction the success of an attack tree on a real system model is usually difficult to determine. For example, it could potentially take an SME sorting through tens of thousands of pages of mission design documents to learn enough to verify the success or failure of an individual attack tree. Moreover, the specifics of design documents will differ from mission to mission hence adding to the complexity of transferring an attack tree to a different mission. Additionally, the hardware, e.g., servers, routers, firewalls, are constantly changing throughout the life of the mission along with properties pertaining to configuration and security such as system vulnerabilities, OS, application, or authentication schemes. Hence, we sought to develop a method in which an SME does not need to sort through the design documents or trouble themselves with hardware specifics in order to verify an attack tree. The manner in which we accomplished this was to tie the leaf vertices of the attack tree to vertices in the system model. After this was performed, an SME could quickly perform an interactive impact assessment that provides valuable information, such as vulnerability information, hardware information, and mission software information pertaining to the attack. This allows an SME to develop countermeasures to protect against the attack.

D. Future Improvements to CAVE

In this section, we describe some of the current limitations, and the planned improvements to address those in the future versions of our tool.

PROBABILISTIC ATTACK TREES Our attack trees are related to the fault tree formalism in their boolean expressions to gate conditions when parent nodes are satisfied by their child nodes. However, fault trees often include a-priori probabilities with each node. Fault trees tend to calculate probabilities of higher parent nodes using Bayes' rule. However, with respect to computer security and the generality of our attack tree nodes, probability estimates are either unavailable, unreliable or too costly to gather. In addition, the probability distribution of events may not conform to a uniform distribution. Therefore, we did not do Bayesian probabilistic analysis. However, in a future version of the impact assessment in CAVE we will add the option for a user to attach probabilities to each of the nodes. The analysis will then indicate to the user which entry node is contained in the path with the highest probability of success.

AUTOMATED PARSING OF ATTACK TREES As we have discussed, all of the attack trees that we have generated have been generated by humans. We use the ambiguous English language to generate nodes of the attack tree. While the particular vocabulary and words of nodes may differ, some nodes may try to describe a common underlying concept. So, for example, if one person creates a node "*Hack the firewall*" and another person creates a node "*CNA firewall*", these two nodes ultimately mean the same thing. We are working toward generating a tool to recognize the nodes' similarity, and mapping those nodes to a common underlying concept. To do this, we are creating an intermediate, generic set of nodes that the leaf nodes of an attack tree can map to. We will try to use natural language processing tools and algorithms to calculate the similarity of the leaf nodes. In particular, we hope to parallel much of Yan Wu, Robin Gandhi and Harvey Siy's methodology from their natural language processing efforts on CWEs.⁵²

INCLUSION OF COUNTERMEASURES One immediate extension of this work is to include countermeasures into the attack trees to form an attack countermeasure tree, in a similar vein to work by Roy et al.¹¹ By attaching countermeasures to both the leaf nodes and internal nodes of the attack tree, the SME can better plan defensive mechanisms to thwart the attack. Attributes can be incorporated into the countermeasures such as investment cost, defensive probabilities, and return on investment. Further analysis can then be done across the tree to find the set of countermeasures that lowers the chance of a successful execution of the attack tree within a fixed budget using optimization algorithms such as simulated annealing or genetic algorithms.

VIII. Conclusion

The work presented in this paper is a part of a larger cyber security effort aimed at improving the confidentiality, integrity and availability of our missions. Our specific objective in this paper was to address the challenge of assisting a cyber SME in performing a cyber security assessment of mission-critical systems,

with the intent of evaluating the impact of low-level cyber events on high-level mission objectives. Toward our objective, we described a framework called the *Cyber Asset Visualization Environment (CAVE)*, which effectively combines both the system and threat centric perspectives to improve cyber impact assessments. Specifically, we demonstrated how our approach enabled the execution of abstract attack tree models over a multi-layered system model to assess risks to high-level mission objectives (Section V). We also presented an interactive visual environment that enables an SME to visualize the propagation of multiple attack behaviors through a system. This enables a more comprehensive assessment of the cyber risk to missions (Section VI, Section VII). We demonstrated the benefits of our approach with a real-world use case in which an SME used CAVE to assess security risks to high-level mission objectives, and to evaluate appropriate mitigation strategies to ensure confidentiality, integrity and availability of high-level mission objectives (Section VII).

Acknowledgments

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The authors would like to thank Bob Vargo, Sami Saydjari, Frank Kuykendall, Aleksandr Kerzhner, Marc Pomerantz, Brian Campuzano, Kevin Dinkel, Viet Nguyen, Robert Steele, and Bryan Johnson for discussions and feedback that helped develop the ideas and methods expressed in this paper.

References

- ¹Thalen, M., "Hackers Allegedly Hijack Drone After Massive Breach at NASA," Online: <http://www.infowars.com/hackers-allegedly-hijack-drone-after-massive-breach-at-nasa/>, Jan 2016.
- ²Flaherty, M. P., Samenow, J., and Rein, L., "Chinese hack U.S. weather systems, satellite network," Online: https://www.washingtonpost.com/local/chinese-hack-us-weather-systems-satellite-network/2014/11/12/bef1206a-68e9-11e4-b053-65cea7903f2e_story.html, Nov 2014.
- ³Fox News, "Chinese hackers took over NASA's Jet Propulsion Lab, Inspector General reveals," Online: <http://www.foxnews.com/tech/2012/03/01/chinese-hackers-nasa-jpl-lab.html>, Mar 2012.
- ⁴Roberts, P., "Hack Targets NASA's Earth Observation System," Online: <https://threatpost.com/hack-targets-nasas-earth-observation-system-051711/75242/>, May 2011.
- ⁵Sentementes, G., "Johns Hopkins APL cyber attackers got past the firewall," Online: http://www.baltimoresun.com/bs-mtblog-2009-06-johns_hopkins_apl_site_hacked-story.html, June 2009.
- ⁶Capaccio, T. and Bliss, J., "Chinese Military Suspected in Hacker Attacks on U.S. Satellites," Online: <http://www.bloomberg.com/news/articles/2011-10-27/chinese-military-suspected-in-hacker-attacks-on-u-s-satellites>, Oct 2011.
- ⁷Epstein, K. and Elgin, B., "Network Security Breaches Plague NASA," Online: <http://www.kepstein.com/2008/11/20/network-security-breaches-plague-nasa/>, Nov 2008.
- ⁸Albanesius, C., "RSA to Replace SecurID Tokens After Lockheed Cyber Attack," Online: <http://www.pcmag.com/article2/0,2817,2386512,00.asp>, June 2011.
- ⁹Kordy, B., Mauw, S., Radomirović, S., and Schweitzer, P., "Foundations of attack-defense trees," *Formal Aspects of Security and Trust*, No. C, 2011, pp. 80–95.
- ¹⁰Bistarelli, S., Fioravanti, F., and Peretti, P., "Defense trees for economic evaluation of security investments," *Proceedings of the First International Conference on Availability, Reliability and Security, ARES 2006*, 2006, pp. 416–423.
- ¹¹Roy, A., Kim, D. S., and Trivedi, K. S., "Attack countermeasure trees (ACT): towards unifying the constructs of attack and defense trees," *Security and Communication Networks*, Vol. 5, No. 8, 2012, pp. 929–943.
- ¹²Argauer, B. J. and Yang, S. J., "VTAC: Virtual terrain assisted impact assessment for cyber attacks," *Computer Engineering*, mar 2008, pp. 69730F–69730F–12.
- ¹³Breu, R., Innerhofer-Oberperfler, F., and Yautsiukhin, A., "Quantitative Assessment of Enterprise Security System," *2008 Third International Conference on Availability, Reliability and Security*, 2008, pp. 921–928.
- ¹⁴Musman, S., Temin, A., Tanner, M., Fox, D., and Pridemore, B., "Evaluating the impact of cyber attacks on missions," *International Conference on Information Warfare and Security*, Academic Conferences International Limited, 2010, p. 446.
- ¹⁵Musman, S., Tanner, M., Temin, A., Elsaesser, E., and Loren, L., "A systems engineering approach for crown jewels estimation and mission assurance decision making," *IEEE SSCI 2011: Symposium Series on Computational Intelligence - CICS 2011: 2011 IEEE Symposium on Computational Intelligence in Cyber Security*, 2011, pp. 210–216.
- ¹⁶D'Ambrosio, B., Takikawa, M., Fitzgerald, J., Upper, D., and Mahoney, S., "Security Situation Assessment and Response Evaluation (SSARE)," *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, Vol. 1, 2001, pp. 387–394.
- ¹⁷Goodall, J. R., D'Amico, A., and Kopylec, J. K., "Camus: Automatically mapping Cyber Assets to Missions and Users," *MILCOM 2009 - 2009 IEEE Military Communications Conference*, oct 2009, pp. 1–7.
- ¹⁸Vigna, G., "Missionary : A Formal Model for Cyber-Missions and its Application to Cyber Situation Awareness," Online: http://www.cs.ucsb.edu/~tim/cybaware_web/dwnld/yr2/vigna.pdf, 2011.

- ¹⁹Chen, B., Kalbarczyk, Z., Nicol, D. M., Sanders, W. H., Tan, R., Temple, W. G., Tippenhauer, N. O., Vu, A. H., and Yau, D. K., "Go with the Flow: Toward Workflow-Oriented Security Assessment," *Proceedings of the 2013 workshop on New security paradigms workshop (NSPW '13)*, 2013, pp. 65–76.
- ²⁰Sheyner, O. and Wing, J., "Tools for Generating and Analyzing Attack Graphs," *2nd International Symposium on Formal Methods for Components and Objects (FMCO'03)*, Vol. 3188, 2004, pp. 344–371.
- ²¹Jajodia, S., Noel, S., Kalapa, P., Albanese, M., and Williams, J., "Cauldron mission-centric cyber situational awareness with defense in depth," *2011 - MILCOM 2011 Military Communications Conference*, IEEE, nov 2011, pp. 1339–1344.
- ²²Schneier, B., "Attack trees," *Dr. Dobb's journal*, Vol. 24, No. 12, 1999, pp. 21–29.
- ²³Piètre-Cambacède, L. and Bouissou, M., "Beyond Attack Trees: Dynamic Security Modeling with Boolean Logic Driven Markov Processes (BDMP)," *2010 European Dependable Computing Conference*, 2010, pp. 199–208.
- ²⁴Zonouz, S. a., Khurana, H., Sanders, W. H., and Yardley, T. M., "RRE: A game-theoretic intrusion Response and Recovery Engine," *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, jun 2009, pp. 439–448.
- ²⁵McLaughlin, S., Podkuiko, D., Miadzevzhanka, S., Delozier, A., and McDaniel, P., "Multi-vendor penetration testing in the advanced metering infrastructure," *Proceedings of the 26th Annual Computer Security Applications Conference on - ACSAC '10*, Vol. I, 2010, pp. 10.
- ²⁶McLaughlin, S. and Podkuiko, D., "Energy theft in the advanced metering infrastructure," *Critical Information*, 2010.
- ²⁷Lazarus, E. L., Dill, D. L., Epstein, J., and Hall, J. L., "Applying a Reusable Election Threat Model at the County Level," 2011, pp. 12.
- ²⁸Sommestad, T., Ekstedt, M., and Holm, H., "The Cyber Security Modeling Language: A Tool for Assessing the Vulnerability of Enterprise System Architectures," *IEEE Systems Journal*, Vol. 7, No. 3, sep 2013, pp. 363–373.
- ²⁹Raugas, M., Ulrich, J., Faux, R., Finkelstein, S., and Cabot, C., "CyberV@R - A Cyber Security Model for Value at Risk," Tech. rep., 2013.
- ³⁰Edge, K. S., *A framework for analyzing and mitigating the vulnerabilities of complex systems via attack and protection trees*, Ph.D. thesis, Air Force Institute of Technology, 2007.
- ³¹Phillips, C. and Swiler, L. P., "A Graph-based System for Network-vulnerability Analysis," *Proceedings of the 1998 Workshop on New Security Paradigms*, 1998, pp. 71–79.
- ³²Liu, Y. and Man, H., "Network vulnerability assessment using Bayesian networks," *Defense and Security*, Vol. 5812, mar 2005, pp. 61–71.
- ³³LeMay, E., Ford, M. D., Keefe, K., Sanders, W. H., and Muehrcke, C., "Model-based security metrics using Adversary View Security Evaluation (ADVISE)," *Proceedings of the 2011 8th International Conference on Quantitative Evaluation of Systems, QEST 2011*, 2011, pp. 191–200.
- ³⁴Cheng, Y., Sagduyu, Y., Deng, J., Li, J., and Liu, P., "Integrated situational awareness for cyber attack detection, analysis, and mitigation," Vol. 8385, 2012, pp. 8385N–8385N–11.
- ³⁵Anwar, Z., Shankes, R., and Campbell, R. H., "Automatic security assessment of critical cyber-infrastructure," *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, 2008, pp. 366–375.
- ³⁶Phan, H., Avrunin, G., Bishop, M., Clarke, L. A., and Osterweil, L. J., "A Systematic Process-Model-Based Approach for Synthesizing Attacks and Evaluating Them," *Electronic Voting Technology Workshop*, 2012.
- ³⁷Jakobson, G., "Mission cyber security situation assessment using impact dependency graphs," *14th International Conference on Information Fusion*, 2011, pp. 1–8.
- ³⁸Barreto, A. D. B., Costa, P. C. G., and Yano, E. T., "A Semantic Approach to Evaluate the Impact of Cyber Actions on the Physical Domain," *7th International Conference on Semantic Technologies for Intelligence, Defense, and Security (STIDS 2012)*, 2012, pp. 64–71.
- ³⁹Agedal, J., den Braber, F., Dimitrakos, T., Gran, B., Raptis, D., and Stolen, K., "Model-based risk assessment to improve enterprise security," *Proceedings. Sixth International Enterprise Distributed Object Computing*, 2002, pp. 51–62.
- ⁴⁰Taubenberger, S. and Jürjens, J., "IT Security Risk Analysis based on Business Process Models enhanced with Security Requirements," 2008.
- ⁴¹Anita, D'Amico; Buchanan, Laurin; Goodall, John; Walczak, P., "Mission Impact of Cyber Events: Scenarios and Ontology to Express the Relationships Between Cyber Assets, Missions and Users," *2010 International Conference on Information-Warfare & Security*, Vol. 298, No. 0704, 2010, pp. 388–397.
- ⁴²Cam, H. and Moullem, P., "Mission-aware time-dependent cyber asset criticality and resilience," *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, 2013, pp. 0–3.
- ⁴³Kerzhner, A., Pomerantz, M., Tan, K., Campuzano, B., Dinkel, K., Pecharich, J., Nguyen, V., Steele, R., and Johnosn, B., "Analyzing Cyber Security Threats on Cyber-Physical Systems using Model-Based Systems Engineering," *AIAA SPACE 2015 Conference and Exposition*, 2015, p. 4575.
- ⁴⁴Burgess, M., Canright, G., and Engø-Monsen, K., "A graph theoretical model of computer security," *International Journal of Information Theory*, Vol. 3, No. 2, 2004, pp. 70–85.
- ⁴⁵Grimaila, M., Mills, R., and Fortson, L., "Improving the cyber incident mission impact assessment (CIMIA) process," *Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead*, 2008.
- ⁴⁶NIST, "National Vulnerability Database," Online: <https://nvd.nist.gov>.
- ⁴⁷Katz, L., "A New Status Index Derived from Sociometric Index," *Psychometrika*, 1953, pp. 39–43.
- ⁴⁸"iGraph," Online: <http://igraph.org>, July 2016.
- ⁴⁹Python Software Foundation, "Python 3.4," Online: <https://www.python.org>, July 2016.
- ⁵⁰"QT 4.8," Online: www.qt.io, July 2016.
- ⁵¹"Neo4j," Online: <https://neo4j.com>, July 2016.

⁵²Wu, Y., Gandhi, R., and Siy, H., "Semi-Automatic Annotation of Natural Language Vulnerability Reports," *International Journal of Secure Software Engineering*, Vol. 4, No. 3, July 2013, pp. 18–41.