Software and Attack Centric Integrated Threat Modeling for Quantitative Risk Assessment

Bradley Potteiger
Vanderbilt University
2201 West End Ave
Nashville, TN 37235
bradley.d.potteiger@vanderbilt.edu

Goncalo Martins
University of Denver
2199 S University Blvd
Denver, CO 80208
Goncalo.Martins@du.edu

Xenofon Koutsoukos
Vanderbilt University
2201 West End Ave
Nashville, TN 37235
Xenofon.Koutsoukos@vanderbilt.edu

ABSTRACT

One step involved in the security engineering process is threat modeling. Threat modeling involves understanding the complexity of the system and identifying all of the possible threats, regardless of whether or not they can be exploited. Proper identification of threats and appropriate selection of countermeasures reduces the ability of attackers to misuse the system. This paper presents a quantitative, integrated threat modeling approach that merges software and attack centric threat modeling techniques. The threat model is composed of a system model representing the physical and network infrastructure layout, as well as a component model illustrating component specific threats. Component attack trees allow for modeling specific component contained attack vectors, while system attack graphs illustrate multi-component, multi-step attack vectors across the system. The Common Vulnerability Scoring System (CVSS) is leveraged to provide a standardized method of quantifying the low level vulnerabilities in the attack trees. As a case study, a railway communication network is used, and the respective results using a threat modeling software tool are presented.

Categories and Subject Descriptors

I.6.5 [**Simulation and Model Checking**]: Model Development—*modeling methodologies*; K.6.m [**Management of Computing and Information Systems**]: Miscellaneous—*Security*

Keywords

Quantitative risk assessment, threat modeling, cyber-physical systems

1. INTRODUCTION

Designing secure cyber-physical systems (CPS) is becoming a major issue in many areas. It is widely accepted that designing secure CPS is a difficult problem. Moreover, security has always been treated as an add on. An application is assumed to be secure if it uses cryptography, security protocols, etc. While claiming that a system uses "Hash-based message authentication code" or "128-bit keys" sounds effective, these statements mean very little when taken out of context. The important question to ask is "Are the security features of the system necessary, and do they meet the system's security needs?" [8].

Traditionally, security has been an art, relying on the judgment of highly trained professionals to qualitatively assess vulnerability states of systems. The largest benefit of a qualitative approach is rapid implementation due to the lack of necessity for large datasets and statistical analysis since risk is determined by expert opinions and personal experience. However, one problem is that since the risk assessment is largely associated with personal experience, risk models are subjective and are largely inconsistent among organizations. Since an attacker only needs one chance to succeed, this inconsistency could prove to be a significant problem. With the introduction of a quantitative approach, risk can be more objectively defined, leading to more consistent risk assessment results. With the increased consistency, the probability of organizations missing critical vulnerabilities is decreased.

A challenge presented in the quantitative risk assessment of CPS deals with developing security metrics and models capable of analyzing a system in respect to a set of security properties in a given context. This problem additionally dives into the question of whether the scientific method and "definitive measurement techniques" can be applied to the area of security which has traditionally been a qualitative field.

This paper attempts to make a step towards solving this problem by providing a standardized, quantitative approach to model the security of components as well as the system as a whole by taking into account the inter-connectedness of each component. This goal is accomplished through the use of a two step composition approach. A high level system model is developed illustrating the various components as well as their interconnectedness to other components. Additionally, for each component as well as the system as a whole, attack trees are developed. Component attack trees are associated with illustrating attack vectors that are contained within the specific component domain, while system attack graphs represent attack vectors that are created from component vulnerability propagation. This paper further

HotSoS '16, April 19-21, 2016, Pittsburgh, PA, USA

DOI: http://dx.doi.org/10.1145/2898375.2898390

considers the problem of quantifying model input to confirm that system security analysis can be as realistic and effective as possible.

Threat modeling involves understanding the complexity of the system and identifying all possible threats, regardless of whether or not they can be exploited. Identifying threats helps develop realistic and meaningful security requirements. Security requirements are then analyzed based on their criticality and likelihood, and a decision is made whether to mitigate the threat or accept the risk associated with it. Proper identification of threats and appropriate selection of countermeasures reduces the ability of attackers to misuse the system. Threat modeling is implemented using one of three common approaches independently: Asset centric, attacker centric, and software centric [13].

The security designer can start thinking about threats using each of these approaches. [13] supports the direction that if there is a methodology that provides the ability to merge these three approaches in a common modeling environment, the number of discovered threats and vulnerabilities will be maximized.

Securing systems is about tradeoffs; often finding the ideal balance is a challenge. It is impossible to guarantee 100% security. However, it is possible to work towards 100% acceptance. A good security system strikes a balance between what is possible and what is acceptable through a risk management process. Risk management is comprised by risk assessment, risk reduction, and risk acceptance [14]. Another goal of the proposed approach is to include quantitative risk assessment on the identified threats and respective countermeasures. This enables the analysis of the suitability of common requirements to the system. The security designer will be able to visualize the impact of different mitigations for the identified threats.

To provide a standardized method of modeling the probability of success for the risk assessment, the Common Vulnerability Scoring System (CVSS)[1] can be incorporated. CVSS provides the ability to have a formal rating system for attacks. Currently, the usefulness of CVSS is a widely debated topic in the metrics field due to the mysteriousness surrounding the score calculation algorithms as well as the possible score inconsistencies created from different judging "experts." However, due to its wide use and the assumption that the standardized method will reduce the amount of inconsistencies in the scoring process, we believe that CVSS is useful for illustrating our modeling approach. It is important to note that our work provides the flexibility to use any quantitative scoring methodology so in the future different metrics can be inserted in the place of CVSS.

A software research prototype is developed to illustrate the proposed methodology. A domain specific modeling language is developed for the system and attack models using the Generic Modeling Environment (GME) [5]. Additionally, the ADTool graph generation software suite [4] is integrated to perform automated risk propagation on the generated attack models in GME. Furthermore, this tool allows for an automated method of observing changes in the quantitative risk values in attack models based on the implementation and redesign of attack models.

The contributions of this work include (1) integrating, in a common modeling approach, software and attack centric threat modeling approaches, (2) including quantitative risk assessment on the identified threats and respective countermeasures, (3) including a standardized method to model the probability of success for the risk assessment, and (4) providing a composition methodology for integrating system and component models.

This paper focuses on presenting a threat modeling approach with a case study for illustration purposes. As a part of the approach, the components and network layout are first modeled in a system model, depicting the structural design of the system. Then, qualitative attribute templates are developed for each specific component and network connection, while quantitative risk values are assigned to attributes. Furthermore, component attack trees are created through a combination of component attributes, illustrating attack vectors to compromising system components. Finally, component attributes are inserted into system attack trees, conveying how individual component vulnerabilities and mitigations effect the overall system security.

The remainder of this paper is organized as follows. Section 2 presents the related work. In Section 3 the problem formulation and case study are presented. Section 4 describes the component modeling and risk assessment methodology. In Section 5, the system modeling and risk assessment methodology is presented. Finally, the paper concludes with a summary in Section 6.

2. RELATED WORK

Typically, threat modeling has been implemented using one of three approaches independently, asset centric, attacker centric, and software centric.

2.1 Asset Centric

Asset centric threat modeling encompasses a defense strategy (blue team) in protecting the internal infrastructure of a system. This approach is typically popular in information technology and business applications where an "asset" such as health data, monetary funds, or personally identifiable information needs to be protected from outside intruders, similarly to the functionality of a bank vault in the physical domain. In 2013, the SANS Institute developed a list of the 20 most impactful security measures relevant to network security after analyzing the most common offensive exploits. This is very beneficial in that it focuses only on the most practical "proven to work" approaches in guiding companies to better security [1]. However, one drawback of this approach is that it is constrained to the cyber domain, mainly focusing on general network security information.

With the recent trend in Internet of Things (IOT) devices, OWASP at Oxford University has conducted research on IOT compromises to produce a list of the top 10 IOT attack surfaces and mitigations in what is called the "Internet of Things Top 10 List" [9]. This document goes one step further than the critical security controls in providing paths to execute an attack in addition to the mitigations that can lead to system protection.

For providing a general modeling structure, NIST has developed a cyber security framework, allowing businesses to customize their security measures in categories relating to identifying threats, providing protective measures, detecting intrusions, responding to infiltration, and recovering after an attack [2]. This framework encompasses a wide spectrum of a compromise, not only providing security measures for pro-

[1]https://nvd.nist.gov/cvss.cfm

tecting systems from intrusions, but also covering how to recover from attacks to minimize damage.

2.2 Attacker Centric

In 1998, Phillips et al. developed an attack graph generation tool which relied on a collection of variables representing the state of a network and edges representing actions between states [15]. The authors noted that exponential state explosion becomes a problem as the number of states increased. However, a search engine was developed to to aid in analysis and a partial order reduction algorithm was implemented to eliminate duplicate paths contributing to explosion.

In 2003, Scheyner took this tool a step further in utilizing model checking techniques to compute attack graphs [12]. The developed tool could be integrated with a vulnerability scanner to conduct a real time analysis of a network. Similar to Phillips, Scheyner's tool has problems with state space exponential explosion resulting in a high simulation time for large models. It can additionally be noted that both of the tools developed by Phillips and Scheyner are heavily constrained to the network domain, not being easily extendable to the cyber physical domain due to a significant reliance on network vulnerability scanners.

In 2013, Kordy et al. developed a tool at the University of Luxembourg called ADTool. ADTool uses a similar structure to the previous tools, except, it provides capabilities to insert mitigation actions to counteract the attack process. This allows for the defender role to be modeled in addition to focusing on the attacker, resulting in a holistic overview of a security game. Furthermore, both actions and mitigations can be assigned risk values, allowing for a quantitative risk to be computed from the model.

2.3 Software Centric

Microsoft has developed the SDL threat modeling tool for focusing on modeling software related system vulnerabilities [7]. SDL focuses on using the STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) methodology to analyze system vulnerability to these different categories of attacks. Further, if there is any vulnerability in a respective category of STRIDE, recommended mitigations will be provided and the model can be adjusted accordingly to reflect the changes.

DREAD (Damage Potential, Reproducibility, Exploitability, Affected Users, Discoverability) is another threat modeling methodology encompassing an acronym referring to several threat assessment categories [10]. However, instead of focusing on a boolean variable representing the vulnerability of a category, DREAD uses a numerical approach in assigning one of three values to the first four categories (0, 5, 10) and one of four values (0, 5, 9, 10) to the last category. These assignments allow for an average value to be calculated to represent the risk of the entire system.

Martins et al. developed a tool similar to SDL that models the software centric state of a system based on STRIDE [6]. Instead of providing mitigation recommendations based off of Microsoft's security approach, NIST guidelines are utilized. However, there still exists a problem in all three of the above applications in that the individual attack processes are not analyzed. By integrating attack processes, a more detailed model could be developed illustrating recommendations based off of more specific circumstances.

2.4 Risk Management

NIST has developed the Common Vulnerability Scoring System (CVSS) that provides the ability to have a formal rating system for attacks [3][10]. This allows security research to have a reliable, normalized representation of an attack. With this data, the opportunity for quantitative risk analysis is enhanced. However, some drawbacks noted include not reducing the attack surface, high complexity compared to the STRIDE and DREAD approaches, and requiring a high overhead.

TRIKE is an open source threat modeling framework similar to the Microsoft methodologies of STRIDE and DREAD, but focuses on a risk based approach. While STRIDE and DREAD models consist of a representation of attacks, threats and weaknesses, TRIKE focuses on the impact of system stakeholders [11][10]. One benefit of this tool is the ability to integrate both software centric and attack centric approaches by autonomously generating attack graphs based off of the requirements model and implementation model input. However, it seems this modeling process is extensive with several diagrams required to comprise the input models.

Ucedavélez et al. has developed the PASTA (Process for Attack Simulation and Threat Analysis) framework for threat modeling in respect to addressing the most viable threats to an application target [16]. This framework provides 7 layers for more detailed modeling capabilities than traditional threat modeling tools. However, similar to TRIKE, the framework modeling process seems to be extensive with several layers of modeling required. This framework seems appropriate to large corporations with a vast security department with thorough knowledge of system attributes.

3. PROBLEM FORMULATION

The problem that this paper addresses is that security is heavily dependent on the qualitative judgment of system designers. Additionally, it is hard to model component dependencies so that the propagating impact of individual vulnerabilities on the rest of the system behavior can be analyzed. Furthermore, this problem is extended to include the challenge of composing a model such that the quantitative risk metrics of components can be connected to illustrate risk propagation and interdependent attacks in a system.

The objectives of this paper include developing a methodology aimed at modeling component quantitative risk through the use of software and attack centric modeling approaches, as well as connecting component threats together for the purpose of composing system wide attack propagation models. From this standpoint, an important goal is to translate component level composition to system level attack composition by developing an interface for applying component specific risk to the rest of the system.

For the first objective, component attack trees are developed for the five threat categories of the software centric STRIDE methodology. These attack trees illustrate the threat dependencies of the higher level STRIDE attack categories on the lower level component attributes. Additionally, a goal is to introduce a formal, standardized method of assigning risk and mitigation values to the attributes included in the component attack tree models. This is accomplished by using the CVSS scoring methodology to assign risk values to component attributes in the attack tree and executing a

risk propagation algorithm to recursively model the risk of higher level STRIDE threat categories.

For the second objective, attack ports are generated for individual components illustrating actions that can translate risk to connected components. Attack ports are additionally dependent on a combination of the component attack trees, allowing STRIDE risk to transfer to connecting components. With a generated system attack graph for a targeted component, the composed attack paths in the graph represent a series of attack ports executed to reach the target. With each attack port dependent on the risk of its respective STRIDE attack category, component risk values can be propagated throughout the graph to illustrate the effect on the target component.

3.1 Case Study

A railway communication network is utilized as a case study to illustrate the effectiveness of the modeling approach.

There are two different locations in the system: Central operating stations and the field. The central operating station is where the train operator remotely communicates with various train switches and signals. This location is modeled with operator, network switches and router components. This infrastructure allows the main headquarters of the train operating agency to communicate to the outside field. The next location is the field, which consists of basestations, railway signals and railway switches, along with a repeater in the case of far distances away from the main operating center. Basestations, railway signals and railway switches can be connected wirelessly while the rest of the components are connected through wired connections. However, there are some constraints to component interconnections. Railway signals and switches have to be connected to a basestation, the basestation has to be connected to a router or repeater, the operator has to be connected to a network switch and a network switch has to be connected to a router. The constraints and components are listed below. Additionally, Table 1 defines commonly used symbols used in the sections below.

$$Components = \left\{ \begin{array}{c} operator \\ network\ switch \\ router \\ repeater \\ basestation \\ railway\ signal \\ railway\ switch \end{array} \right\}$$

Connection Constraints
$operator \rightarrow networkswitch$
$networkswitch \rightarrow (router \vee operator)$
$router \rightarrow (repeater \vee basestation)$
$repeater \rightarrow (router \vee basestation)$
$basestation \rightarrow (railwaysignal \vee railwayswitch \vee repeater)$
$(railwaysignal \vee railwayswitch) \rightarrow basestation$

In the following sections, the process of modeling component threats with the goal of composing a system attack propagation model is illustrated using the railway case study. First, a component model is developed for a railway signal (Rsignal) to illustrate the process of composing component attack trees for each category of STRIDE. In this example a Rsignal attack tree is developed for a tampering threat, conveying the different attack vectors possible for the threat. Additionally, the quantitative risk methodology is described by using the CVSS approach to assign risk values to the Rsignal vulnerabilities in the tampering attack tree. Furthermore, a risk propagation algorithm is implemented to illustrate how the risk of lower level Rsignal vulnerabilities effect the overall risk of a tampering attack occurring.

Symbol	Description
SM	System Model
CM	Component Model
c	Components
w	Communication Channel
SAG	System Attack Graph
CAT	Component Attack Tree
s	States
t	transitions
s_g	Attacker Goal State
p	Risk
AT	Attribute Template
a	Attribute
AP	Attack Ports
dep	Attack Port Dependencies

Table 1: System Model Symbols

Finally, the process of translating component specific threats to system wide attack paths is shown by looking at the Rsignal tampering attack again. However, in this case, the risk of a Rsignal attack is dependent on external component threats in the connecting basestation, repeater, router, switch, and operator components. By using specific STRIDE threats in these components, attack paths are developed using a series of connecting attack ports to generate a system attack graph. After completion, a comparison of the Rsignal tampering attack risk is shown in respect to the internal component risk defined in the component attack tree, versus the external propagating risk defined in the system attack graph. This allows for prioritizing strategies for implementing mitigation measures.

4. COMPONENT RISK ASSESSMENT

4.1 Component Modeling

The component model is a tuple $CM = (AT, CAT, AP)$ where $AT \subseteq vulnerabilities, mitigations$ is the set of attribute templates describing the component, CAT represents the set of component attack trees illustrating the various attack vectors that can be utilized to compromise the component and AP represents the set of attack ports conveying how component threats can propagate to external components. An attribute template ($AT \subseteq a$) represents various component descriptors (vulnerabilities and mitigations) that describe the security state of components in the system. As such, every component in the system contains a set of attribute templates as a part of its formal component model.

Definition 1. *Attribute Template*
An attribute template is a generic property of the hardware or software configuration of the system which includes but is not limited to component vulnerabilities and mitigations.

Additionally, the assignment of a value to an attribute template results in an attribute.

Definition 2. *Attribute*
An attribute is an assigned instance of an attribute template.

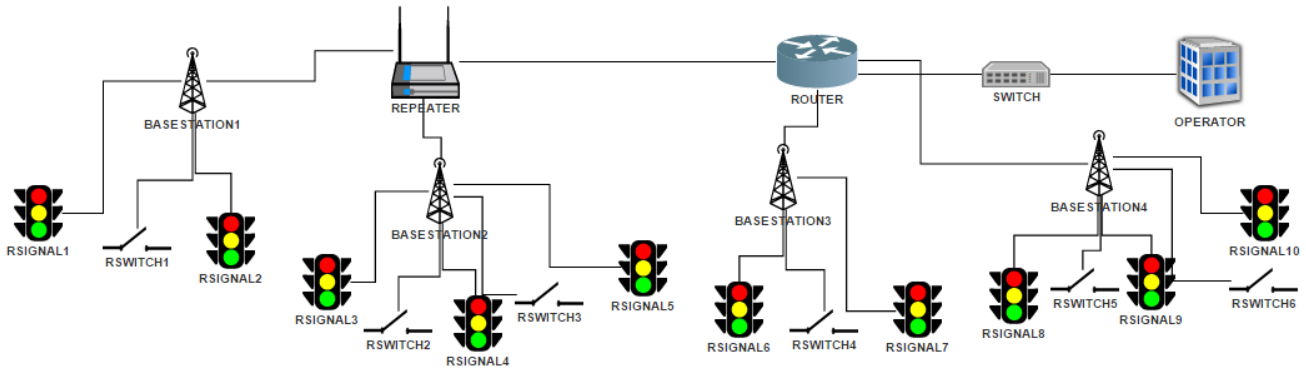

Figure 1: Railway Communication Network Case Study

It can either be assigned a true or false value, or a weighted quantitative value.

The value p assigned to each attribute is associated with the security risk. For this purpose, CVSS is utilized to provide a standardized methodology of quantifying attribute risk.

The component model tuple contains a component attack tree (CAT) representing various possible attack vectors. A component attack tree is comprised of a tuple $CAT = (s, s_g, t, p)$. A state s represents a threat or mitigation, while s_g represents the high level threat category (STRIDE). A transition $t \subseteq s \times s$ connects two states together in the tree. Finally, $p \in \Re$ represents risk probabilities assigned to initialize respective attributes. The value of p can be quantified with non negative real numbers to represent the risk level of vulnerabilities and mitigations.

Finally, the component model tuple contains a set of attack ports (AP). Attack ports represent an instance of a component attack that spreads risk to external components. Each attack port contains dependencies (dep) of threat categories based on the attack tree classes.

Before generating component attack trees, it is important to note that each component consists of 5 individual attack trees correlating to the 5 separate threat categories of STRIDE. As such, each component has a generated attack tree representing a Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Escalation of Privilege threat.

4.2 Component Attack Trees

For every component, a CAT is generated representing each respective category of STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege). This allows generic software centric principles to be applied more specifically to an application domain. An example of a CAT is shown in Figure 2.

There are 4 different atomic objects that make up the attack tree states in the modeling language. These are the root node, intermediary node, leaf node, and mitigation node. An attacker starts at a leaf node, and iterates through an intermediary node path until it reaches the root node. At this point an attack is considered successful. To model the success of an attack, each leaf and mitigation node is assigned a value. Through "and" and "or" logic propagation, these values are applied to iteratively calculate the parent intermediary node probability of success until the root node probability of success is calculated. In this modeling language leaf nodes correspond to the entry exploit attributes' likelihood of success, while the mitigations correspond to a percentage value that reduces the probability of success of a leaf node. The attack model has a set of nodes with specific connection constraints. In this situation, it is assumed the directional connections occur starting from the leaf node and propagate in an upwards direction to the root node.

$$AttackTreeNodes = \left\{ \begin{array}{c} Leaf\ Node\ (Risk) \\ Mitigation\ Node\ (Risk\ Reduction) \\ Intermediary\ Node \\ Root\ Node \end{array} \right\}$$

Connection Constraints

$leaf \rightarrow (mitigation \vee intermediary \vee root)$
$mitigation \rightarrow (intermediary \vee root)$
$intermediary \rightarrow root$

The CAT in Figure 2 represents a Rsignal Tampering attack. This attack tree consists of all 4 types of nodes: Leaf (gold), mitigation (blue), intermediary (black) and root (red). By iterating from the leaf nodes (Attribute Templates) to root nodes, different attack vectors of implementing an tampering attack are illustrated. In this case, the set of leaf and mitigation nodes are a subset of the Rsignal attribute templates (AT). As such $AT \subseteq$ {Power Drain, Memory Access, Untrusted Code, Logs Access, Resource Availability, Access Control} The possibility of a tampering attack on a Rsignal is directly dependent on the system becoming corrupt, the call chain and logs access. Looking further, one of the nodes that corruption is dependent on is power drain. When analyzing the CAT with non-nullified mitigations, if a high risk value exists for a power drain exploit, it propagates upwards and increases the total chance of a tampering attack on the Rsignal. However, this vulnerability can be mitigated with resource availability measures (NIST guidline SC-6 [6]). When the resource availability measures are in place, the probability of success of a power drain exploit is very low resulting in the risk becoming low. Then, this value causes a decrease in the corruption state attack category and propagates through to decrease the tampering attack category as a whole. As an example, let the risk reduction value of the resource availability mitigation be 25%. That means that the updated risk value of a power drain exploit is now 75% of the original risk value or 25% less.

4.3 CVSS Scoring

The common vulnerability scoring system is used to provide a standardized method of modeling the risk value of

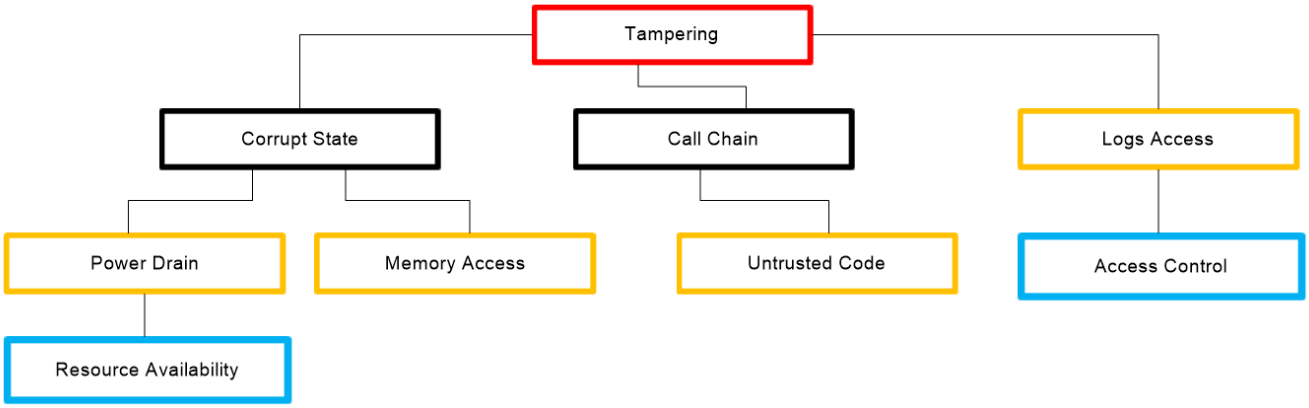

Figure 2: Rsignal Tampering Component Attack Tree

Rsignal Tampering Risk Parameters				
Category	Entry Vulnerability	Risk	Mitigation	Risk Reduction
Corrupt State	Power Drain	8	Resource Availabilty	25%
	Memory Access	4		
Call Chain	Untrusted Code	2		
Miscellaneous	Logs Access	6	Access Control	50%

Table 2: RSignal Tampering Risk and Mitigation Parameters

leaf nodes (vulnerability attributes) in attack trees [3]. In this system there are three types of categories representing a base score, temporal score, and environmental score. The base score represents the intrinsic characteristics of a vulnerability that are constant over time and across user environments. This metric includes two sub-metrics: exploitability, and impact. These subgroups represent characteristics of the vulnerable components as well as reflect the magnitude of the consequences of the action. The temporal metric group reflects the characteristics of a vulnerability that may change over time but not across user environments. This score reflects whether there are any available patches, or if an exploit is easy to implement. Finally, the environmental metric group represents the characteristics of a vulnerability that are relevant and unique to a particular user's environment [3]. After each metric is assigned a value, the base score, temporal score and environmental score are calculated and averaged to get a total score within a range of 0 to 10.

From the generated quantitative score, a qualitative category can also be associated. For example, a score of 0 to 4 represents a low rating, 4 to 7 represents medium and 7 to 10 represents high.

To illustrate the process of calculating a CVSS score, the logs access vulnerability is used from the table of Rsignal tampering vulnerabilities in Table 2. The CVSS score is calculated from an average of the base score, temporal score and environmental score. The base score for logs access includes a network attack vector, low attack complexity and high integrity and availability impact, resulting in a score of 6.4 out of 10. The temporal score is represented as the logs access vulnerability having an unproven exploit maturity, temporary fix for the remediation level and a reasonable report confidence, resulting in a score of 5.4 out of 10. Lastly, the environmental score includes a low confidentiality requirement, high integrity and availability requirement, a network modified attack vector and a high modified attack complexity. This score calculates to 6.4 out of 10. By averaging these three scores together, the total score is 6.06 out of 10, rounding to the nearest integer score of 6 (logs access p risk assignment). This methodology is used to calculate scores for the rest of the vulnerabilities in Figure 2.

4.4 Risk Propagation

The attack tree structure is initialized with leaf node attribute values while the intermediary and root nodes start as uninitialized. As such, the parent node risk value can be calculated based off of a series of logical equations taking into account whether the children nodes have "and" or "or" relations as well as whether the children attributes are mitigations or vulnerabilities.

Taking into account the risk and mitigation values of nodes, the risk value for a parent is calculated by the equations below.

For "or" relation of two leaf nodes x, y
$p = x \cup y = x + y - xy$
For "and" relation of two leaf nodes x, y
$p = x \cap y = xy$
For parent p with mitigation percentage m from 0 to 1
$newp = p * (1 - m)$

As an example with two child nodes with an "or" relation, if one child node (x) has a CVSS score of 5 out of 10, while the second child (y) has a score of 4 out of 10, the parent node will have a risk score of $p = (.5 + .4 - (.5) * (.4)) = .7$

By recursively applying these equations starting at the bottom leaf nodes and working up through the tree structure, the root node risk is calculated conveying the probability and impact of an attack occurring.

4.5 Risk Assessment and Mitigations

After risk propagation, the modeler is able to identify the overall risk for individual components based on a CVSS vulnerability score for each respective STRIDE component attack tree. If the score is above 7, the attack can be classified as being probable to occur and having a large impact on the system. By iterating through the model, the modeler is able to identify the specific components as well as their corresponding attacks that have a high associated risk value.

By analyzing the attack tree, the specific vulnerabilities contributing to the attack risk can be identified. This allows the modeler to look deeper in the system model and determine not just that there is a high risk but what specific component vulnerabilities and attacks are contributing the most to this risk. By comparing the risk between active and inactive mitigation scenarios, the modeler can prioritize vulnerabilities to patch as well as look at the most effective mitigation measures.

Risk is defined as the probability of success of an attack. In the case of the attack tree model, the root node, intermediary node and leaf nodes all have risk. In the modeling framework, the user assigns probability of success values to each individual leaf node of a CAT, allowing for individuality of different component models. For example, one type of component may be more vulnerable than others in respect to a set of exploits. In this case, the component leaf nodes have an increased risk value symbolizing that there is a greater chance of an attack.

With a high risk value in a component attack tree, mitigations can be put in place. These mitigations are represented as a percentage decrease in the overall risk of a child leaf node. Mitigations represent patching a problem, lowering the probability of an attack and consequently the risk to the system. If a high risk value exists for a specific exploit attribute, the risk propagates upwards and increase the total chance of an attack on the set of parent nodes. However, if a mitigation is put in place for the vulnerability attribute, the risk of that exploit is reduced consequently cascading to the parent nodes above in the attack tree, resulting in a lower overall attack tree risk value.

The security of an individual Rsignal component is looked at more specifically by modeling a tampering attack. This attack consists of several leaf vulnerability attributes (Power Drain, Memory Access, Untrusted Code, Logs Access), along with corresponding mitigation attributes (resource availability measures, access control). When analyzing the vulnerabilities through the CVSS method, the vulnerabilities for untrusted code and memory access are in the low to medium range. Even though these attacks are common and easy to occur, they are easily detectable and mitigated making them not as effective as other vulnerabilities. Power draining and accessing the logs of the system are rated as the highest risk vulnerabilities. Power draining has a high risk because once a successful attack occurs, the Rsignal loses availability and no longer is able to perform critical functions. Therefore, power is the most important resource on the Rsignal and would be the most impactful to the component. Logs access is also a fairly high risk vulnerability because it is fairly easy for an attacker to implement without a great amount of suspicion to reverse engineer the Rsignal. This could result in an attacker possibly gaining administrative access or introducing malware. The good news is there are mitigations that can protect against these high risk vulnerabilities. For log access, access control measures can be implemented and for power drain, resource availability protection measures can be implemented. These reduce the risk by 50% and 25% respectively.

From Table 2, the vulnerability attribute with the highest risk is when there is a power drain, inferring that the Rsignal component has power restrictions. This vulnerability is rated an 8 out of 10 conveying it is very probable to occur and has a large impact on the system if implemented by an attacker. However, a mitigation attribute is recommended to "add resource availability." After recalculating the CVSS score with this mitigation taken into account, the risk is reduced by 25% to approximately 6 out of 10. Additionally, another mitigation attribute "access control" is recommended to reduce the logs access vulnerability by 50%. The CAT in Figure 3 illustrates the makeup of this component attack.

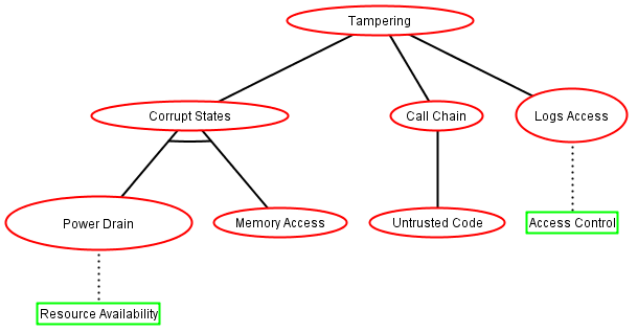

Figure 3: Rsignal Tampering Attack Tree

Figure 4 illustrates the risk value of a Rsignal tampering attack occurring as .78 meaning that the CVSS vulnerability score is a 7.8 out of 10, lying in the high range.The leaf nodes referring to logs access and power drain are rated as .6 and .8 respectively. It is important to remember these values correspond to the CVSS scores of the vulnerabilities, meaning that logs access is rated a 6 out of 10 and power drain is rated as a 8 out of 10. Thus, logs access is a medium security threat while power drain is a high security threat.

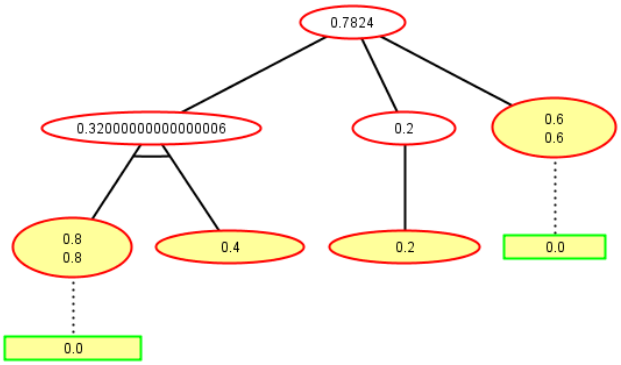

Figure 4: Rsignal Tampering Attack Tree With No Mitigations

Finally, after activating the mitigations, the new overall risk value of a tampering attack is .57 (5.7 CVSS Score), downgrading by 27% from the non-mitigation score to a medium threat range (Figure 5).

4.6 Attack Propagation

After risk assessment is conducted on the component attack trees, attack ports need to be generated for the component to start the process of conducting system risk assessment. Attack ports represent attacker actions that spread risk between components. It is important to remember that each instance of an attack is dependent on the threat levels in a component. As such, each attack port is dependent on the five threat categories of STRIDE (Spoofing, Tampering,

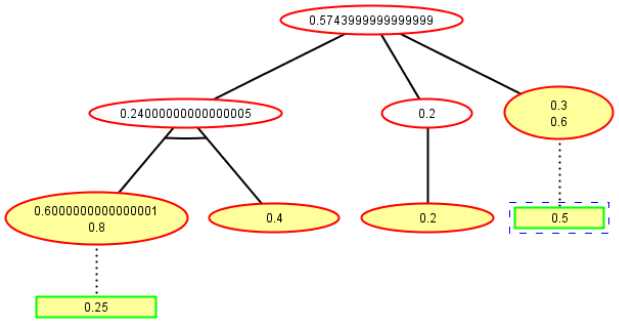

Figure 5: Rsignal Tampering Attack Tree with Mitigations

Attack Ports	
Name	Dependencies
Disrupt BS1 Communications	BS1 DOS, BS1 EOP
Rswitch1 Integrity Attack	Rswitch Spoof
Repeater Integrity Attack	Repeater DOS, Repeater Spoof
Router Relay	None
Switch Relay	None
Operator Send Malicious Packet	Operator EOP

Table 3: Attack Port Examples

Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege). It is also the case that these threat categories serve as the root nodes of the five component attack trees in the CAT. In this case, the attack port dependencies are the root nodes in the CATs. Since risk propagation has already taken place, the root nodes are already assigned a risk score from 1-10. This means that if an attack port is dependent on a CAT root node with a high risk score, that attack port also has a high risk score and has a high probability of being executed. Furthermore, if the dependency has a low risk score, that risk is translated resulting in the corresponding attack port having a low magnitude of risk translated to external components.

Table 3 provides a list of component attack ports for the railway case study example. As noted above, each attack port dependency is related to a CAT root node category, providing the ability to correlate the risk score of the tree root node. Additionally, it can be noted that there can be multiple dependencies for an attack port, as well as no dependencies. In this case, the attack port won't be dependent on the specific component attack trees, but could be dependent on another connected attack port. Furthermore, a component can have multiple attack ports and a single CAT root node can be established as a dependency for more than one attack port.

Looking at Table 3, let's use "Disrupting BS1 Communications" as an example. In the right hand column, "BS1 DOS", and "BS1 EOP" are assigned as dependencies. This infers that the action of disrupting the basestation communications is correlated to whether there is a high risk for a denial of service or elevation of privilege attack. In the case of a denial of service threat, the basestation could either not have enough resources to process the data to transmit or the wireless transmitter could be jammed. Additionally, in the case of an elevation of privilege threat, an attacker could have root access and inject code that would prevent the basestation from transmitting data. In either one of these cases, the data communications transmitting from the basestation becomes non-reliable increasing the chance of malfunction in connected devices such as the railway signals and switches.

5. SYSTEM RISK ASSESSMENT

5.1 System Modeling

It is very common that successful attacks encompass multiple steps ranging over various different portions of the system. The attacker target may not be possible to infiltrate directly. However, especially in cases where only peripheral defenses exist, the target may trust various internal components in the network assuming that the other respective components are not malicious. Therefore, if an attacker can gain entry into another more vulnerable component in the network, it could be possible to compromise the target through that trust link. It is important to not only model individual component vulnerabilities but also develop a threat model for the system as a whole, linking respective component dependencies together.

The system model can be considered a tuple $SM = (c, w, SAG)$ where c is a set of components, $w \subseteq c \times c$ is a set communication channels representing transition connections between components, and SAG is a set of system attack graphs.

The system attack graph SAG is included in the system model to illustrate possible multi-component attack vectors. The system attack graph is modeled by a tuple $SAG = (s, s_g, t, p)$ where s represents states, $s_g \subseteq s$ represents a goal state, $t \subseteq s \times s$ represents transitions between states and $p \in \Re$ represents the risk values assigned to each state. Even though the system attack graph is similar to the CAT, the composition style is different. In a tree structure, the component threat category is organized in a hierarchical composition with the root level consisting of the highest level, while the lower leaf level consists of the specific vulnerability makeup. However, in a graph, the composition is organized based on paths, where the root level corresponds to the target state and the children levels represent specific paths to get to that target state.

5.2 System Attack Graph

Similar to the component attack tree modeling process, the System Attack Graph encompasses a root node, intermediary nodes, and leaf nodes. The root node corresponds to a target component STRIDE threat category, also representing that component CAT root node. An intermediary node represents a component attack port illustrating attacker actions throughout the system network that translate risk between components. As mentioned earlier, these attack ports are dependent on respective CAT root node risk levels, leading to the STRIDE threat categories represented by the CAT root nodes to be assigned as children leaf nodes of the attack ports. One difference between the component attack tree modeling process, and the SAG process is that the connections between nodes represents a path to reach a target instead of a hierarchical relationship.

When developing the SAG, the scoring assignment methodology is similar to the CAT process. For the leaf nodes in the SAG, which are CAT root nodes, the score achieved from the CAT risk propagation process is used. Therefore, the score of the SAG leaf node should be the same as the CAT root node for the respective component STRIDE threat category.

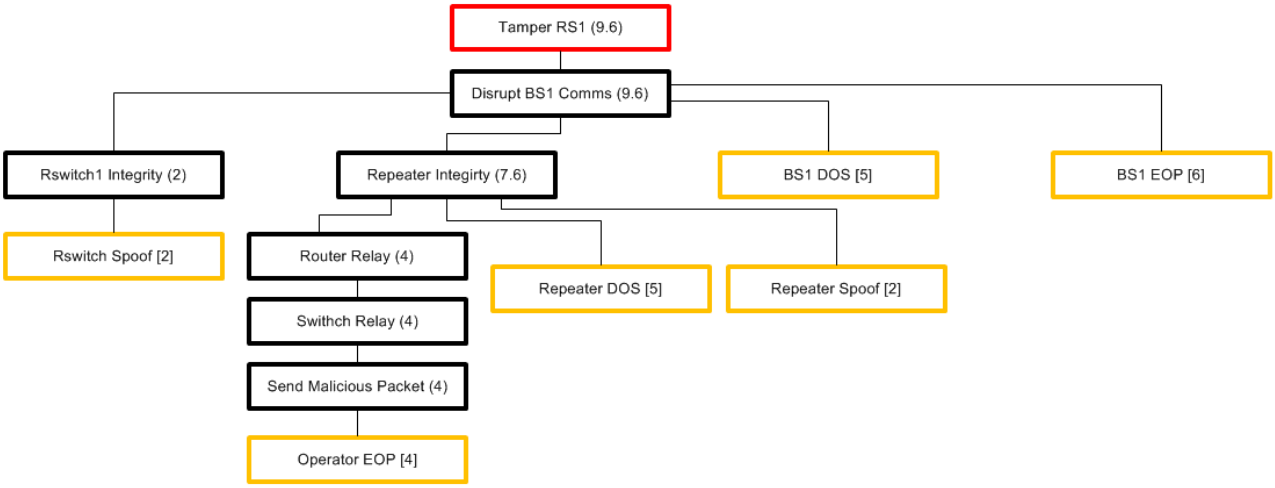

Figure 6: Railway System Attack Graph for Signal Tampering

Since the component attack ports don't have assigned risk scores, the intermediary nodes start as unassigned. Furthermore, the root node begins as unassigned. From this point, the risk from the SAG is propagated to the intermediary nodes until the root node has an assigned risk score. At this time, the SAG score for the root node can be compared to the relating CAT root node score to analyze whether the biggest component threat is from an internal or system wide propagation perspective. This allows the threat modeler to choose mitigation measures to implement.

To illustrate an example SAG from the case study described in Figure 1, the target threat analyzed is tampering in the Rsignal1 component. This threat category is assigned as the root node of the SAG. Additionally, for the purposes of this example, it is assumed that the branches of Basestation2, Basestation3, Basestation4, and Rsignal2 have no effect on RSignal1. Therefore, they have no attack ports assigned. However, Rsignal1, is still effected by the Basestation 1, Rswitch1, Repeater, Router, and Operator components. As such these components have generated attack ports that present an effect on the SAG. These attack ports are defined in Table 3.

When creating the System Attack Graph, the generation algorithm starts with labeling the target threat category as the root node. As such, Tamper Rsignal1 is defined as the root node in this instance. The next step is to move to the next connected components and determine if there are any attack ports. In this case BS1 has an attack port named Disrupt BS1 communications. This attack port is inserted as a child intermediary node of the root node Tamper Rsignal1. Then the algorithm looks for dependencies of the attack port as well as other components that are connected to BS1 with attack ports. Since two dependencies of disrupting BS1 communications are BS1 Denial of Service and Elevation of Privilege, these are assigned as child leaf nodes with their respective risk scores of 5 and 6 generated from their CATs. Since two more connected attack ports are a Rswitch1 Integrity Attack and Repeater Integrity attack, these are assigned as child intermediary nodes of disrupting BS1 communications. This process is repeated until no more new connected attack ports can be found and there are no new current attack port dependencies. The SAG generation algorithm is illustrated in Algorithm 1 and the final system attack graph is in Figure 6. In Figure 6, the leaf nodes have numbers in brackets next to them. This means that each of the numbers correspond to the risk scores of the respective CAT root nodes. On the other hand, the intermediary nodes have scores in parenthesis. This means that these node scores are a result of risk propagation from children nodes and do not have their own CAT models. Finally, the root node has a score in parenthesis indicating that it was calculated by risk propagation instead of taken from the associated CAT model. This however, allows the modeler to compare the risk score calculated from the SAG model with the CAT score. By comparing the two values, the modeler is able to prioritize mitigation efforts to reduce risk.

Algorithm 1 SAG Generation Algorithm

```
1: Initialization: Pick a Target CAT Root Node
2:                 setSAGRoot(CAT Root)
3: Repeat for connectedAttackPort
4:     setIntermediaryChild(connectedAttackPort)
5:     generateChildNodes(connectedAttackPort)
6: End Repeat
7:
8: generateChildNodes(Node)
9:     foreach dependency
10:        setLeafChild(dependency)
11:    foreach connectedAttackPort
12:        setIntermediaryChild(connectedAttackPort)
13:        generateChildNodes(connectedAttackPort)
14: End generateChildNodes
```

5.3 Risk Assessment

After the SAG is generated, the risk score in the root node can be compared to the CAT score. In Figure 6, the root node in the SAG representing tampering the Rsignal1 component has a risk of 9.6 out of 10, representing a very high threat. Additionally, in Figure 4 the CAT root node risk is rated a 7.8 out of 10 which is a high threat but not as high as the SAG. Furthermore, when mitigations are implemented in the CAT, the risk of a tampering attack decreases

to 5.7 out of 10, entering the medium severity range. Comparing the SAG value with the CAT values, the modeler can identify that the component is more susceptible to external propagating attacks than internal threats. With this knowledge, the modeler is better equipped to localize the biggest external threat source and implement the most effective mitigating measures. Additionally, if the CAT has a higher risk value than the SAG, the modeler can conclude that the component is more at risk from an internal standpoint. Therefore, a mitigation should be implemented that deals with the internal security of the component such as implementing access control. By comparing internal versus external component risk, the modeler is be able to implement more effective mitigation measures for critical systems.

6. CONCLUSION

By integrating software centric and attack centric threat modeling methodologies, security specialists are able to analyze vulnerability risk as well as specific paths that lead to compromise. By assigning a risk value to each attribute in a CAT, a standardized, quantitative analysis of a system component is made possible. With mitigations added to CATs, models can be developed to analyze the effectiveness of implemented security guidelines.

Additionally, by incorporating component risk in SATs, an analysis can be conducted to illustrate the overall effect that individual component vulnerabilities and mitigations have on the rest of the system. The modeling framework includes the flexibility to connect external software packages for the purpose of enhancing the visualization and analytical experience for the modeler. By providing a systemic, quantitative method of modeling system risk, this paper serves as one step towards providing a definitive, scientific approach towards measuring the security of CPS systems.

Acknowledgments

This work is supported in part by the Air Force Research Laboratory (FA 8750-14-2-0180), the National Science Foundation (CNS-1238959), and by NIST (70NANB15H263). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL, NSF, NIST.

7. REFERENCES

[1] Critical security controls. SANS Institute.

[2] Preliminary cybersecurity framework. nist, october 29, 2013. Cybersecurity, Improving Critical Infrastructure.

[3] First. Common vulnerability scoring system v3.0 specification document.

[4] B. Kordy, P. Kordy, S. Mauw, and P. Schweitzer. Adtool: security analysis with attack–defense trees. In *Quantitative Evaluation of Systems*, pages 173–176. Springer, 2013.

[5] A. Ledeczi, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason, G. Nordstrom, J. Sprinkle, and P. Volgyesi. The generic modeling environment. In *Workshop on Intelligent Signal Processing, Budapest, Hungary*, volume 17, page 1, 2001.

[6] G. Martins, S. Bhatia, X. Koutsoukos, K. Stouffer, C. Tang, and R. Candell. Towards a systematic threat modeling approach for cyber-physical systems. In *Resilience Week (RWS), 2015*, pages 1–6. IEEE, 2015.

[7] C. Möckel and A. E. Abdallah. Threat modeling approaches and tools for securing architectural designs of an e-banking application. In *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, pages 149–154. IEEE, 2010.

[8] S. Myagmar, A. J. Lee, and W. Yurcik. Threat modeling as a basis for security requirements. *In Proceedings of the Symposium on Requirements Engineering for Information Security (SREIS'05), Paris, France*, 2005.

[9] OWASP. Oxford iot top ten. https://www.owasp.org/images/7/71/Internet_of_Things_Top_Ten_2014-OWASP.pdf, 2014.

[10] OWASP. Threat modeling. https://www.owasp.org/index.php/Threat_Risk_Modeling, 2015.

[11] P. Saitta, B. Larcom, and M. Eddington. Trike v. 1 methodology document [draft]. *URL: http://dymaxion. org/trike/Trike_v1_Methodology_Documentdraft. pdf*, 2005.

[12] O. Sheyner and J. Wing. Tools for generating and analyzing attack graphs. In *Formal methods for components and objects*, pages 344–371. Springer, 2004.

[13] A. Shostack. *Threat Modeling - Designing for Security.* Wiley, 2014.

[14] G. Stoneburner, A. Goguen, and A. Feringa. Risk management guide for information technology systems. *Recommendations of the National Institute of Standards and Technology (NIST), Special Publication 800-30*, 2002.

[15] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer-attack graph generation tool. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, volume 2, pages 307–321. IEEE, 2001.

[16] T. Ucedavélez and M. M. Morana. Intro to pasta. *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*, pages 317–342.