Spring 2018

DETERMINING VULNERABILITY USING ATTACK GRAPHS: AN EXPANSION OF THE CURRENT FAIR MODEL

Beth M. Anderson
Eastern Washington University

DETERMINING VULNERABILITY USING ATTACK GRAPHS:

AN EXPANSION OF THE CURRENT FAIR MODEL

A Thesis

Presented To

Eastern Washington University

Cheney, Washington

In Partial Fulfillment of the Requirements

for the Degree

Master of Science in Computer Science

By

Beth M. Anderson

Spring 2018

THESIS OF BETH M. ANDERSON APPROVED BY

DAN LI, GRADUATE COMMITTEE	DATE
STUART STEINER, GRADUATE COMMITTEE	DATE
JASON ASHLEY, GRADUATE COMMITTEE	DATE

DETERMINING VULNERABILITY USING ATTACK GRAPHS: AN EXPANSION OF THE CURRENT FAIR MODEL

By

Beth M. Anderson

Spring 2018

ABSTRACT

Factor Analysis of Information Risk (FAIR) provides a framework for measuring and understanding factors that contribute to information risk. One such factor is FAIR Vulnerability; the probability that an event involving a threat will result in a loss. An asset is vulnerable if a threat actor's Threat Capability is higher than the Resistance Strength of the asset. In FAIR scenarios, Resistance Strength is currently estimated for entire assets, oversimplifying assets containing individual systems and the surrounding environment. This research explores enhancing estimations of FAIR Vulnerability by modeling interactions between threat actors and assets through attack graphs. By breaking down the scenario into more representative and quantifiable parts, more detailed and precise analyses are possible.

Keywords—Factor Analysis of Information Risk; Attack Graphs; Resistance Strength; Vulnerability;

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor Dr. Dan Li of the Computer Science Department at Eastern Washington University. Her unwavering support and trust in me was stronger than I felt for myself. She consistently allowed this paper to be my own and gave encouragement and counsel whenever I steered off course.

I would also like to thank my colleagues and advisors at RiskLens who supported my work: Bryan Smith, Isaiah McGowan, Forrest Ireland, and the rest of the engineering team. Without their encouragement, support, input, and review, this thesis would not be what it is today.

Thank you.

Beth Anderson

TABLE OF CONTENTS

LIST OF FIGURES, TABLES, AND EQUATIONS

DETERMINING VULNERABILITY USING ATTACK GRAPHS:

AN EXPANSION OF THE CURRENT FAIR MODEL

Beth Anderson

Factor Analysis of Information Risk (FAIR) [1] provides the ability to understand and quantify risk and the components that comprise it. FAIR breaks from traditional risk assessments where risk is measured with qualitative ratings such as high, medium and low. Instead, FAIR provides a model to quantify risk into dollars and cents. FAIR accomplishes this by breaking down risk into understandable measurable components that combine in the FAIR ontology. However, there is still work to be done to develop FAIR into a more realistic and robust ontology. One component within the FAIR ontology, Vulnerability, lacks a realistic estimation model that captures the complexities of networks surrounding assets, vulnerabilities, and other nuances. This research expands upon the current FAIR model to address limitations of FAIR Vulnerability by expanding the ontology to include attack graphs.

This paper is structured as follows: To begin, the current FAIR model is explained and areas for improvement within FAIR Vulnerability are highlighted. Then, a solution to limitations of FAIR Vulnerability is presented, followed by a description of attack graphs and vulnerabilities. Following that, an example network is introduced, the corresponding attack graph analyzed, and FAIR is applied to exploratory scenarios using this network. Finally, future work is discussed, and the paper concludes.

1. FACTOR ANALYSIS OF INFORMATION RISK

Factor Analysis of Information Risk (FAIR) [1] places risk into an ontology (Figure 1) where lower components combine together to determine higher components until the root of the ontology, Risk, is determined. Each factor within the FAIR ontology represents a probability distribution [1]. Components lower in the ontology are defined using minimum, most likely, and maximum. By estimating values of measurable components, probabilities that are difficult or impossible to measure can be determined using the probability distributions of measurable components further down the ontology. The FAIR ontology is used bottom up, where measurements are provided at more granular components at the bottom and then combined to determine components higher up the ontology. However, when explaining structure of the FAIR ontology, it is easiest to explain from the top down, starting with Risk and breaking it down into smaller components.

1.1 Risk

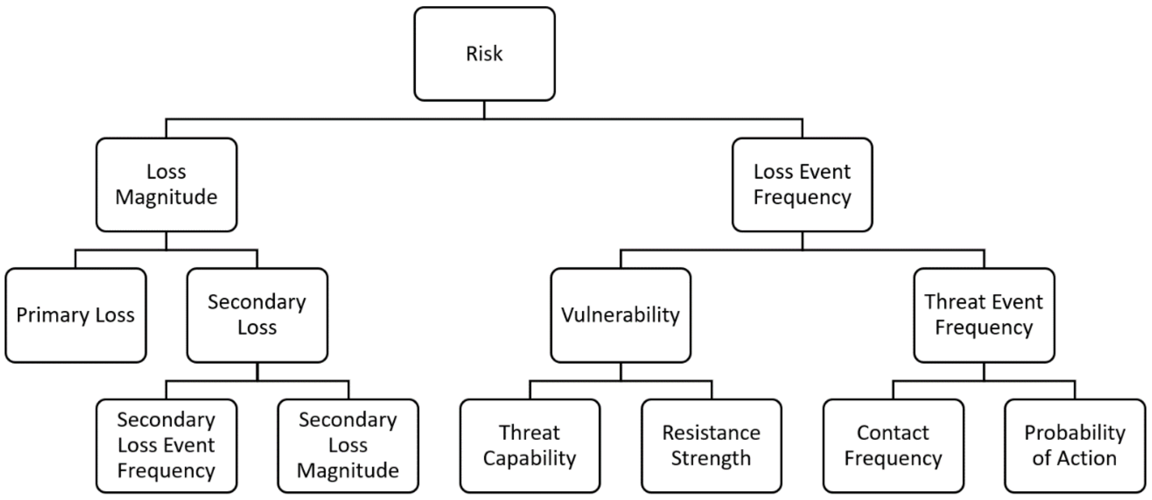


Figure 1. FAIR Ontology [1]

Within FAIR, Risk defined as “the probable frequency and probable magnitude of a future loss” [1]. Risk is the combination of how likely it is that a loss will occur (Loss Event Frequency) and how much is likely to be lost (Loss Magnitude). These two components, Loss Event Frequency and Loss Magnitude comprise the first layer of the FAIR ontology and are the direct descendants of Risk.

1.2 Loss Magnitude and Subcomponents

Loss Magnitude represents the magnitude of incurred loss — i.e. how much loss is likely to result. This is measured in currency units, such US dollars, or Euros, and is related to the asset, the thing of value. The more valuable the asset is, the larger Loss Magnitude is likely to be. Loss Magnitude is defined as “the probable magnitude of primary and secondary loss resulting from an event” [1]. However, determining a probability distribution for the entirety of Loss Magnitude is difficult as there are many different factors that determine how much money will be lost. Because of this, Loss Magnitude is broken down into two subcomponents: Primary and Secondary Loss.

Primary Loss is “primary stakeholder loss that materializes directly as a result of the event” [1]. Primary Loss is measurable as companies can determine how much money it will take to get operations back up and running, as well as how much an asset is worth. Examples of Primary Loss include: lost revenue from outages, wages paid to workers when work is not being completed, and replacement of tangible assets such as cash. This node is represented by a probability distribution with values for the minimum, most likely, and maximum amount of money that would be directly lost due to a loss event.

Secondary Loss is the other component that comprises Loss Magnitude. Compared to Primary Loss, Secondary Loss is the fallout of the event rather than the direct result [1]. Examples of Secondary Loss include results from reputation loss, fines, and judgements. Secondary Loss is more nuanced than Primary Loss and is more difficult to measure. Because of this, Secondary Loss is broken down further into Loss Event Frequency and Loss Magnitude in the same way that Risk is broken down at the top on the ontology. Secondary Loss Event Frequency is defined as “the percentage of primary events that have secondary effect” [1]. As not all scenarios have the potential for Secondary Loss; it is important to measure how often the primary event will result in a secondary event. Secondary Loss Events can include fines and judgements being levied because of negligence. The other half of Secondary Loss is Secondary Loss Magnitude, the “loss associated with secondary stakeholder reactions” [1]. This would include notification cost, credit monitoring, legal defense costs, and diminished stock prices.

Measurements taken on Secondary Loss Event Frequency and Secondary Loss Magnitude are combined to create a probability distribution that represents Secondary Loss. The probability distribution of Secondary Loss is then combined with the distribution representing Primary Loss to determine Loss Magnitude, the amount of damage an event could inflict.

1.3 Loss Event Frequency and Subcomponents

Where Loss Magnitude tells us how much can be lost, Loss Event Frequency is how frequent a loss is expected to happen, often measured in events per period such as twice per year. It is important to note that Loss Event Frequency is how often an event completely comes to fruition, not how frequently an event is attempted. For example, Loss Event Frequency would measure how often a hacker successfully infiltrates a database (once a year) not how often hackers attempt to infiltrate (many times a day). In the same way that Loss Magnitude is further broken down into more understandable and measurable components, so is Loss Event Frequency. Loss Event Frequency is comprised of Threat Event Frequency and Vulnerability.

Threat Event Frequency is the “probable frequency, within a given time-frame, that threat agents will act in a manner that may result in loss” [1]. Threat Event Frequency includes all potential events including those that result in loss as measured by Loss Event Frequency and those that do not. In the earlier example, Threat Event Frequency is how often hackers tried to infiltrate the database, which may be many times a day. In this case, the probability distribution can be decided at the Threat Event Frequency level. If there are logs that measure how often threat actors attack, there is no need to measure at lower levels of the ontology. However, there are some situations where measuring at Threat Event Frequency is difficult. Thus, Threat Event Frequency is broken down into two other components, Contact Frequency and Probability of Action.

Contact Frequency is “the probable frequency, within a given time-frame, that threat agents will come into contact with assets” [1]. A threat agent is anything that can cause a loss event, from a hacker to a tornado. Contact Frequency includes both logical contact, such as a scan of a web server, or physical contact, such as an employee stealing sensitive documents. Several types of contact include a threat agent randomly encountering an asset, contact during regular business activities, or a threat agent purposeful contacting an asset. It is useful to measure at Contact Frequency if there is an asset that is contacted for reasons other than the loss event. For example, having a database manager access the database to check logs every day would be a type of contact. However, it is unlikely that each time the manager accesses the logs is a loss event.

Contact happens all the time; what determines if an event becomes a threat is the Probability of Action. Probability of Action is “the probability that a threat agent will act upon an asset once contact has occurred” [1]. Once contact has been made, there is a probability that the threat agent will or will not do something negative to the asset. In the example of Contact Frequency mentioned earlier, a database manager is checking the daily logs, meaning that he or she has access to information about the database. The Probability of Action would measure if he or she attempts to do damage to the database, whether it be to breech confidentiality or degrade the integrity of the data. Most of the time, Probability of Action in this case would be near 0%. Unless the employee is disgruntled, the chances of them doing anything negative during these checks is small. On the other hand, a hacker purposely contacting a web server would have a Probability of Action of a 100%. The combination of how often an asset is contacted and the probability that action will be taken upon contact determines Threat Event Frequency.

Loss Event Frequency is comprised of both Threat Event Frequency, made of Contact Frequency and Probability of Action, and Vulnerability. Even if the Threat Event Frequency of an asset is high, with events happening constantly, this does not mean that the asset will have high Loss Event Frequency, the frequency that something damaging has happened. This is because not every attempt at an asset is successful. Vulnerability is the probability that an event involving a threat will result in a loss. Note that this definition of vulnerability is different from typical definitions of vulnerability. When speaking of vulnerability as defined by FAIR, “FAIR Vulnerability” will be used. When speaking of vulnerability as a flaw in the defenses of an asset, such as an exploitable software bug, “vulnerability” will be used. FAIR Vulnerability is “the probability that a threat agent’s actions will result in a loss” [1]. FAIR Vulnerability is further broken down into Threat Capability and Resistance Strength.

Threat Capability is the level of ability of the threat actor measured on a percentile scale from 0-100. Threat Capability is determined based on a population of threat actors also known as a threat community. For example, the probability distribution for the Threat Capability of Cyber Criminals may place the least capable threat actor in the 60th percentile, the most capable actor at the 100th percentile, and a majority of threat actors within the 90th percentile [1]. The rationale behind this measurement is that Cyber Criminals are more skilled and have better resources than an average cyber threat agent. Threat Capability is directly compared against Resistance Strength to determine FAIR Vulnerability.

The Resistance Strength of an asset is measured on the same scale as Threat Capability. Resistance Strength is defined as “the level of difficulty that a threat agent must overcome” [1]. This could be how strong the brick and mortar of a house is against a hurricane or how advanced an anti-virus software is against malicious software. The Resistance Strength of an asset is how capable the asset protects itself against threats. Being on the same scale as Threat Capability, Resistance Strength is directly compared to Threat Capability. If Threat Capability is higher than Resistance Strength, the threat event would become a loss event – i.e. The threat actor would successfully complete his or her goal. If Resistance Strength is higher than Threat Capability, then the asset is safe, and no loss would occur.

Measurements of Resistance Strength and Threat Capability are combined to determine FAIR Vulnerability. Threat Event Frequency can be broken down into Contact Frequency and Probability of Action or measured directly at Threat Event Frequency. The probability distribution resulting from the combinations of Threat Event Frequency and FAIR is Loss Event Frequency which combined with Loss Magnitude quantifies Risk.

2. PROBLEMS WITH FAIR VULNERABILITY

The current method of determining FAIR Vulnerability has room for improvement. The current estimation model for FAIR Vulnerability as described by an industry expert [2] places the burden of estimating Resistance Strength on the analyst who must determine values for minimum, maximum and most likely, for the Resistance Strength of the entire asset. For example, if an analyst discovers that an asset has a SQL injection vulnerability, it is up to him or her to decide how much impact that vulnerability has on the strength of the asset. Introduce new analysts and the resulting Resistance Strengths of the same asset can vary. Additionally, assets can be made up of intricate systems and parts, all of which have their own strengths and weaknesses. An analyst would have to determine and then aggregate the Resistance Strength of a complicated connection of different subsystems. Without detailed knowledge of all the aspects of an asset, an analyst cannot make accurate judgements about the asset’s Resistance Strength. Furthermore, assets can live within larger systems where threats can elevate privileges outside of the asset, allowing access to the asset that would be difficult to model through FAIR in its current form. For example, a web application is connected to many other systems ranging from servers to databases. A vulnerability within any of the connected systems has the potential to expose the web application to attackers. Looking at just the vulnerabilities of one part of the highly connected network potentially misses interesting and important attack pathways. Assets are not always single units, they can be multi-layer systems with each part having various levels of Resistance Strength.

A solution to these limitations is to create a model of FAIR Vulnerability that can account for the complexities and intricacies of real assets. Determining the Resistance Strength at the asset level oversimplifies highly complex assets that may be involved in larger systems. A model that can make use of information about the internals of an asset, or its relationships to external systems will better represent the real world.

3. SOLUTION

The proposed solution introduces an expansion to the calculation of FAIR Vulnerability by employing an attack graph to model paths a threat actor can take to successfully harm an asset. Along these paths, the threat actor will exploit vulnerabilities in sequence, elevating his or her privilege along the way. However, it should be noted that this expansion to FAIR will not fit all risk scenarios modellable by FAIR. For example, a tornado destroying a server warehouse would not be an applicable scenario. This expansion is suited for situations that would be modellable by an attack graph – e.g. a threat actor is attempting to hack into a database to gain personally identifiable information.

This expansion to FAIR introduces the idea that FAIR Vulnerability can be improved to take into account the features of an asset. This allows for the environment surrounding the asset as well as components of the asset to be described in terms of concrete components and connections via an attack graph. Using an attack graph ensures that only the relevant aspects of an asset will be used in a FAIR scenario. There is no use in determining Resistance Strength for parts of the asset that would not be touched within a scenario.

This expansion can use attack graphs in two different ways. The first would be to determine Resistance Strength for each attack step in the attack graph. These determinations are made by analysts informed by system configuration, known vulnerabilities, and Common Vulnerability Scoring System (CVSS) scores as well as expertise in the asset. This way requires more input; however, this could be beneficial if there are intricacies of the asset that are not completely encapsulated by the information provided to the attack graph. Analysts highly familiar with the configuration of an asset may be better able to determine the viability of an attack step than blind probability.

The second way that attack graphs can be used is with Bayesian Networks to determine the probability of a successful attack. This would allow for the probability of an actor completing each step within the attack graph to be determined based on the probability of previous steps. Information such as vulnerabilities will be included to form probabilistic relations between the different steps in the attack sequence. However, this way does not inherently take into account differences in the Threat Capabilities of threat actors.

The first and second way of using attack graphs within FAIR can be used in conjunction. There is no strict need to adhere to one technique or the other. Sometimes there will be situations where a step in the attack graph is more suited for an analyst to determine the Resistance Strength compared to the Threat Capability of the threat actor.

Other times this will not be needed, and Bayesian probability models the situation accurately. The general approach of the examples within this paper is to rely on a Bayesian network to provide probabilities that a threat actor would be able to make it to a particular step in the attack graph but analyze and adjust the Bayesian network based on knowledge about the model and the threat actor.

4. ATTACK GRAPHS

Attack graphs are models that represent information about vulnerabilities and interaction in a network to show different paths that an attacker can follow to reach a given goal. Along each of these paths, the attacker will exploit vulnerabilities in sequence, elevating his or her privilege along the way. There are two main types of attack graphs: state-based representations and logical attack graphs. Within state-based representations each node represents the state of the whole network after a single atomic attack. The number of state nodes increases dramatically as the network increases, thus limiting the applicability of this type of graph to very small networks. Logical attack graphs on the other hand are defined as bipartite graphs that represent dependencies between security conditions and exploits [3]. This research uses logical attack graphs as the size of assets can range from small networks to dynamically large systems.

4.1 MulVal

Constructing an attack graph is a “tedious, error-prone and impractical for attack graphs larger than a hundred nodes” [4]. This tediousness can be alleviated by using tools such as MulVal, the Multi-host, Multi-stage Vulnerability Analysis Language tool [5]. MulVal is a scalable end to end framework that models the interaction of vulnerabilities with system and network configurations. MulVal takes information encoded as Datalog facts to create an attack graph. Datalog is “a declarative logic language in which each formula is a function-free Horn clause, and every variable in the head of a clause must appear in the body of the clause” [6].

Using the information provided, as well as logical rules defined within the MulVal engine, MulVal creates a logical attack graph that shows steps an attacker can take to reach a specified goal and the configurations used in order to execute attack steps. Information is entered into the MulVal reasoning engine as Datalog facts. Interaction rules are also supplied to the reasoning engine. The engine then creates attack simulation traces that determine all different attack paths that could be used to reach the goal. The trace is then passed to a graph builder which creates a logical attack graph, which is then outputted to the user [7].

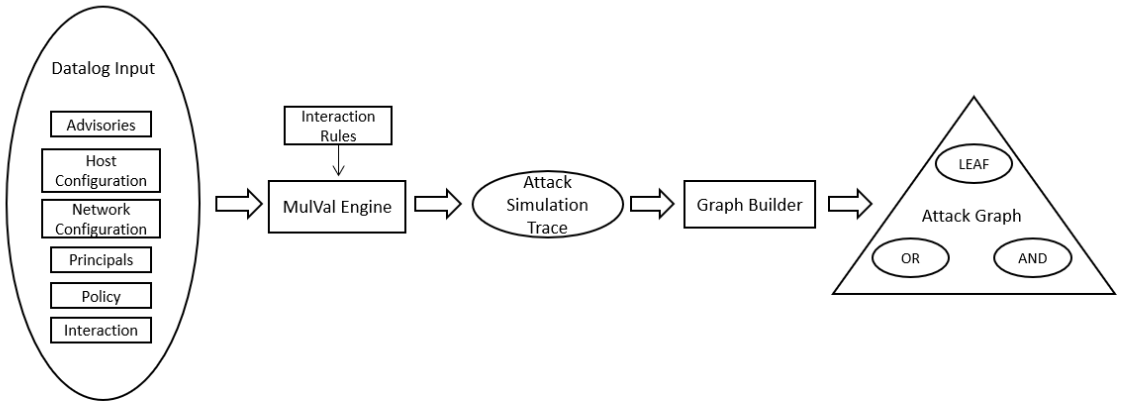


Figure 2. Graphical representation of the MulVal tool [5]

The logical attack graph consists of three different types of nodes: *AND*, *OR* and *LEAF* nodes. An *AND* node represents an attack step within the attack graph. An example of an *AND* node is “RULE 6 (direct network access)”. This *AND* node is determined based on interaction rules that are inputted into the MulVal tool as well as preconfigured rules that are already included in the tool. *OR* nodes are privilege nodes. An example of a *OR* node would be “netAccess (webServer, tcp, 80)”. This means that an attacker has privileges to access the web server using tcp on port 80. The final type of node is a *LEAF* node which represents the configuration of the system. An example of a *LEAF* node would be “hacl(internet, webServer, tcp, 80)”, which means that the web server can connect to the internet using tcp on port 80. All three of these nodes types comprise the attack graph created by MulVal [7].

The following sections describe each type of information provided to the MulVal engine in order to create the attack simulation trace and eventually an attack graph [5].

4.1.1 Advisories

Advisories relate to what vulnerabilities exist in the network. Vulnerabilities can be determined by using a vulnerability scanner such as an OVAL (Open Vulnerability Assessment Language) scanner or one of the many other available options [8-9]. The original MulVal research used an OVAL scanner which uses OVAL formalized vulnerability definitions to test a machine to see if any of those definitions are matched [10]. The results from the OVAL scanner are translated into a Datalog fact and supplied to the MulVal engine. An example of a Datalog fact of a vulnerability is

```
vulExists(webServer, ’CAN-2002-0392’, httpd).
```

This fact tells that there is a vulnerability involving the server program httpd on a web server with the ID of CAN-2002-0392. Another Datalog clause is used to capture the effect of the vulnerability.

```
vulProperty(’CAN-2002-0392’, remoteExploit, privilegeEscalation).
```

This Datalog fact shows that the vulnerability CAN-2002-0392 enables arbitrary code execution by a remote attacker with all the program’s privileges. This information can be determined from sources such as the National Vulnerability Database (NVD) provided by the National Institute of Standards and Technology (NIST) [11]. However, many vulnerability scanners already correlate this data and provide information about the impact of the vulnerability which can be fed into the MulVal reasoning engine.

4.1.2 Host Configuration

Host configuration is information about the software and services that are running on hosts as well as the configuration of those hosts. This information can also be determined using scanners. For example, one could use the OVAL scanner to extract configuration parameters such the port numbers and privileges of a program on a host. This data is converted to Datalog clauses like

```
networkService(webServer, httpd, TCP, 80, apache).
```

This clause describes a web server with a program httpd listening on port 80 using TCP protocol and running as the user ‘apache’.

4.1.3 Network Configuration

Network configuration is information about how network routers and firewalls are configured. MulVal models network configurations as abstract host access-control lists (HACL). This information can be found by looking at the configuration of network routers and firewalls. This information is converted to Datalog clauses like the following which shows that TCP traffic is allowed from the internet to port 80 on the web server.

```
hacl(internet, webServer, TCP, 80)
```

4.1.4 Principals

Principals refer to the users of the network and their accounts on the network hosts. For example, a system admin has an account on the web Server with root permission. This is written in Datalog as

```
hasAccount(sysAdmin, webServer, root).
```

4.1.5 Interactions

Interactions model how components within a network interact. General rules can be established for the network using Horn clauses. In Horn clauses, the first line is the conclusion, or result, and the remaining lines are the conditions that enable the conclusion. An example of this is

```
execCode(Attacker, Host, Priv) :- vulExists(Host, VulID, Program),
vulProperty(VulID, remoteExploit, privEscalation), networkService(Host,
Program, Protocol, Port, Priv), netAccess(Attacker, Host, Protocol, Port),
malicious(Attacker).
```

The earlier Horn clause describes that an attacker would be able to execute arbitrary code on the machine under the privilege *Priv*, if there was a program running on a host that allowed for privilege escalation, that program was listening on Protocol and Port while running under *Priv* privilege, and the attacker can access that Protocol and Port through the network. Words capitalized within this clause (Host, Program, VulID, ect.) are Prolog variables that matched, meaning that this rule can be applied to any set of actual parameters that match this pattern. Using the extendibility of these rules, general sets of rules have been created and special network specific rules can be added to the MulVal engine as needed.

4.1.6 Policy

Policy refers to access permissions on data. If a policy is not explicitly allowed, then it is assumed to be prohibited. For example,

```
  allow(Everyone, read, webPages)
allow(systemAdmin, write, webPages)
```

This policy says that anyone can read web pages and that the user *systemAdmin* is allowed to write web pages. Like the Horn clauses of configuration, Everyone is capitalized meaning it is a Prolog variable that represents any user.

4.2 Bayesian Attack Model

Attack graphs, such as the one created by the MulVal tool, provide information about the paths that a threat actor can take to reach a particular goal. However, Attack graphs are not well suited to model ongoing attacks as they cannot represent the progression of an attacker through the system [12]. Adding a Bayesian network to an attack graph adds the probability that an attacker would be reach a step within the attack graph. A Bayesian network is a probabilistic graphical model based on a directed acyclic graph where nodes represent random variables and edges represent the probabilistic dependences between those variables.

Within the context of an attack graph, nodes represent exploits *e* and conditions *c* each of which have two probabilities, p(*e*) and p(*c*) for individual scores and P(*e*) and P(*c*) for cumulative scores respectively. Individual score p(*e*) is the probability that the exploit *e* will be successfully be executed given that all other conditions for exploit are already met. This is the intrinsic likelihood of an exploit in isolation without the influence of any other factors. The same is for p(*c*), but with conditions rather than exploits. P(*e*) and P(*c*) are cumulative scores that measure the overall likelihood that exploit *e* will be successfully executed or that condition *c* will be met. These probabilities are based on probabilities earlier in an attack sequence [13]. Conditional probabilities are of the main interest within attack graphs as those probabilities model the likelihood of an attacker reaching a particular node within the graph.

5. VULNERABILITY DATA

As mentioned earlier “vulnerability” has a different meaning within the context of FAIR. FAIR vulnerability is “the probability that a threat agent’s actions will result in a loss” [1]. However, the definition of vulnerability within this section is a “weakness in an IS (Information System), system security procedures, internal controls, or implementation that could be exploited” [14]. More specifically, a vulnerability is “a specific bug in a specific software which can be abused in an unintended manner to potentially cause a negative impact to the user of the software” [15]. Information about vulnerabilities in software and hardware is available, enumerated, and cataloged though Common Vulnerabilities and Exposures, Common Weakness Enumeration, and the Common Vulnerability Scoring System.

5.1 Vulnerability Information

5.1.1 CVE

Common Vulnerabilities and Exposures (CVE) is list of publicly known common identifiers for cyber security vulnerabilities [16]. After being launched in 1999 by MITRE Corporation, CVE is now the industry standard for identifying vulnerabilities. A CVE identifier (CVE-1999-0400) maintains a one to one relationship with a vulnerability and a standardized description for each vulnerability or exposure. While CVE is a dictionary, not a database, it does feed into NVD [11]. This database is very useful for this work as MulVal engine needs information about the vulnerability beyond the name.

5.1.2 CWE

Common Weakness Enumeration (CWE) [17] is similar to CVE, except that it is a list of common software weaknesses that can occur in software architecture, design, code or implementation that can lead to exploitable security vulnerabilities. According to MITRE, the difference between weaknesses and vulnerabilities is that a software weakness is an error that can lead to software vulnerabilities. Examples of software weaknesses include buffer overflows, authentication errors, and code injection (e.g. SQL Injection (CWE-89)). CWE can be useful for analysts to determine the type of vulnerability they are looking at. Seeing that a vulnerability has a CWE of SQL Injection allows for quick understanding about what type of attack would be able to exploit this vulnerability. Additionally, CWEs can be used to categorize sets of vulnerabilities, leading to further research into how the skill needed by an attacker to exploit vulnerabilities can be related to CWEs.

5.1.3 CVSS

Common Vulnerability Scoring System (CVSS) is the industry standard for measuring the severity of vulnerabilities in software [18]. This system provides a way to translate the characteristics of a vulnerability and produce a numerical score that reflects the severity. Version 2 of the CVSS was released in June 2007 and has been replaced by Version 3 in June 2015. Both version 2 and 3 are included in databases such as the National Vulnerability Database and some older vulnerabilities are only scored using version 2, including the vulnerabilities that were found on the Model Network.

The CVSS calculates three different scores: a base, a temporal, and an environmental score. The base score is calculated using intrinsic and fundamental characteristics of the vulnerability that do not change over time or in different environments. The temporal score is based on characteristics that change over time, but not between user environments. Finally, the environmental score is calculated using characteristics that are dependent upon the user’s environment. The CVSS score expressed on a scale from 0 to 10, with 10 being the most severe.

Of interest in this research are the exploitability metrics included in the base score. The exploitability metric group measures how difficult a vulnerability is to exploit. Table 1 describes the metrics for exploitability as well as the associated numerical value for each. This table describes version 2 of CVSS scoring system as the vulnerabilities within the Model Network used in this paper are scored using version 2.

Exploitability Metric	*Value*	*Value Description*	*Numeric*
Access Vector: The minimum level of access needed to exploit the vulnerability	Local	An attacker must have physical access to the system or a local (shell) account	0.395
	AdjacentNetwork	An attacker must have access to either the broadcast or collusion domain of the vulnerable software	0.646
	Network	An attacker does not need local network access or local access	1.0
Access Complexity: The complexity of the attack needed to exploit the vulnerability	High	Specialized accesses conditions. For example, an attacker must employ DNS hijacking.	0.35
	Medium	Somewhat specialized access conditions. For example, information must be gathered before the attack	0.61
	Low	No specialized access conditions	0.71
Authentication: The number of times an attacker must authenticate to exploit a vulnerability	Multiple	An attacker must authenticate two or more times	0.45
	Single	An attacker must authenticate once	0.56
	None	Authentication is not needed	0.704

Table 1. Exploitability metric details [19]

5.2 Application of Vulnerabilities

Understanding the severity of a vulnerability allows for informed decisions to be made about how a vulnerability will impact FAIR vulnerability. Previous research [20] demonstrates a way to determine the effort and skill that a threat actor would need to possess to successfully exploit a vulnerability. This is based on the exploitability metric group of CVSS scores described earlier. One important thing to note is that CVSS base scores of vulnerabilities are calculated independently. However, in an attack graph, vulnerabilities are not in isolation. Because of this, the base CVSS score in some situations is not accurate.

An example of the base CVSS score not being accurate would be two vulnerabilities that exist on two separate hosts connected to each other (Figure 3). Host 1 is connected to the internet, where the attacker is residing, as well as Host 2 with firewalls between both. Host 2 is only connected to Host 1. Both Host 1 and Host 2 have one vulnerability, Vuln 1 and Vuln 2 respectively. The first vulnerability, Vuln 1, present on Host 1, when successfully exploited allows an attacker to gain access to the local network of Host 2. Vuln 2, which is present on Host 2 has a base Access Vector metric of *AdjacentNetwork*. However, within this example, because a threat actor can gain access to the local network of Host 2 using Vuln 1, the Access Vector of Vuln 2 should be changed from *AdjacentNetwork* to *Network*. Changing this metric would increase the overall score of the vulnerability. The dependent relationships of vulnerabilities need to be taken into account when determining the Resistance Strength of an asset.

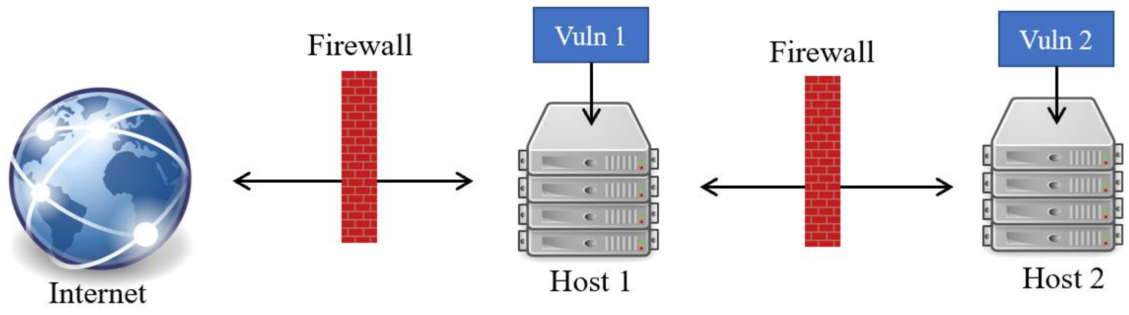


Figure 3. Example network with 2 vulnerabilities

Once new base metrics of each vulnerability within the attack graph have been determined, the effort and skill needed to exploit a vulnerability can be calculated. This is calculated by using the Access Vector (*[AV]*), Access Complexity (*[AC]*), and Authentication (*[Au]*). Equation 1 shows how to calculate both the effort score (*es(e)*) and the skill score (*ss(e)*)

$$es(e)\ \&\ ss(e) = \frac{0.6395}{[AV] * [AC] * [Au]} - 0.2794$$

Equation 1.

The larger the score is, the more effort or skill is required to successfully exploit the vulnerability. This scoring system has been found to accurately match the success rates of simulated attacks [20]. The skill score required by a threat actor can theoretically be mapped directly to Threat Capability, as Threat Capability is the skill of the threat actor. However, in a broader sense, Threat Capability also includes the time and resources that a threat actor has access to. This means that Threat Capability also includes the effort that a threat actor is willing to go through to achieve a goal. The distinction between skill and effort of the threat actor in relation to Threat Capability is a subject that calls for further discussion and inquiry beyond the scope of this paper. However, within the context of this paper, effort and skill will be used interchangeably.

6. MODEL NETWORK

As a proof of concept that attack graphs can add value to FAIR Analyses, a simple model network [5, 21-22] is used as an example (Figure 4).

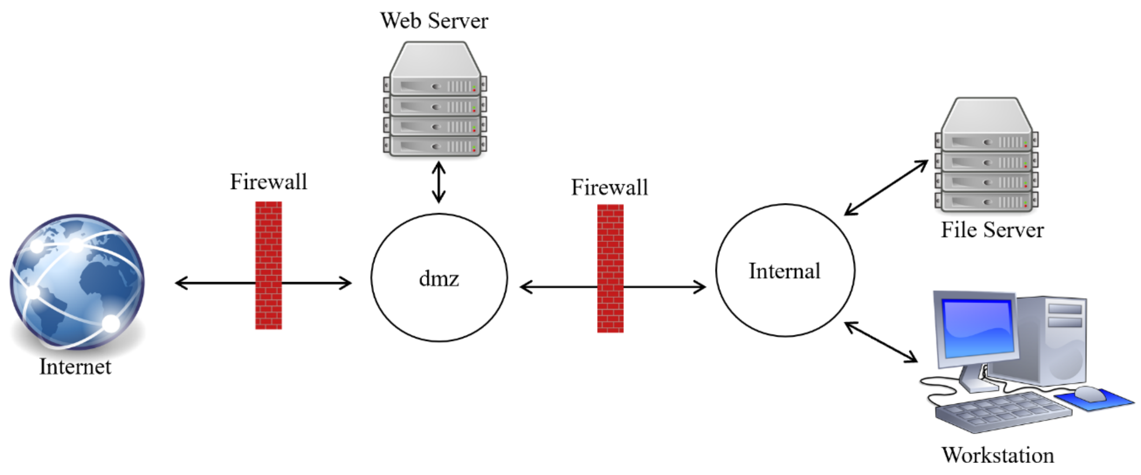


Figure 4. Model Network

6.1 Description of Model Network

This Model Network has three zones: *internet*, *dmz*, and *internal*, separated by two firewalls. The network also has 3 hosts: a web server named *webServer* located within the *dmz* zone, a workstation named *workStation*, and a file server named *fileServer*, both of within *internal*. All three of these hosts are managed by administrators using the username *root*. Users, modeled as a single user with the username *userAccount*, have access to *workStation*, which is many different machines. These many workstations run the same software configuration maintained as a collection of application binaries on the file server and are treated as a single entity. The binaries for the workstations are exported from ‘export/share’ through NFS from the file server. The file server also exports ‘/export/www/’ to the web server.

The MulVal tool also includes a scanner that detects vulnerabilities. The scanner found CVE-2002-0392 and CAN-2003-0252, which has been upgraded to CVE-2003-0252. CVE-2002-0392 on the web server and CVE-2003-0252 on the file server of the Model Network. Both vulnerabilities can result in privilege escalation and are remotely exploitable. Table 2 describes the two found vulnerabilities and their characteristics.

	CVE-2002-0392	**CVE-2003-0252**
Location	*webServer*	*fileServer*
Description	Apache 1.3 through 1.3.24, and Apache 2.0 through 2.0.36, allows remote attackers to cause a denial of service and possibly execute arbitrary code via a chunk-encoded HTTP request that causes Apache to use an incorrect size.	Off-by-one error in the xlog function of mountd in the Linux NFS utils package (nfs-utils) before 1.0.4 allows remote attackers to cause a denial of service and possibly execute arbitrary code via certain RPC requests to mountd that do not contain newlines.
Access Vector	Network	Network
Access Complexity	Low	Low
Authentication	None	None
Confidentiality	Partial	Complete
Integrity	Partial	Compete
Availability	Partial	Complete
CVSS v2.0 Base Score	7.5 (High)	10.0 (High)
Additional Information	- Provides unauthorized access - Allows unauthorized disclosure of information - Allows disruption of service	- Provides administrator access - Allows unauthorized disclosure of information - Allows disruption of service

Table 2. Descriptions of the two found vulnerabilities [23-24]

6.2 Creation of the Attack Graph

The primary concern within the Model Network is a remote attacker infiltrating the system. Therefore, a reasonable goal for this scenario would be a remote attacker executing arbitrary code on one of the hosts within the network. For this example, the goal of the attacker will be to execute code on the workstation. The attacker is located outside of the network on the internet. To achieve this goal, the attacker must navigate into the *dmz* zone, where the web server is located and then move into the *internal* zone where both the file server and the workstation are located. Once inside the *internal* zone, the attacker must gain privileges in order to execute code on the workstation. This process can be modeled using an attack graph. The previous information describing the network is formatted and fed into the MulVal tool. See Appendix A for the input given to the MulVal tool for this Model Network. The MulVal engine runs through the many pathways that an attacker may try and create an attack graph that contains all of the pathways that an attacker can take to successfully infiltrate the system and be able to execute arbitrary code on the workstation. The attack graph produced by the MulVal tool is seen as Figure 5 and nodes are seen in Table 3.

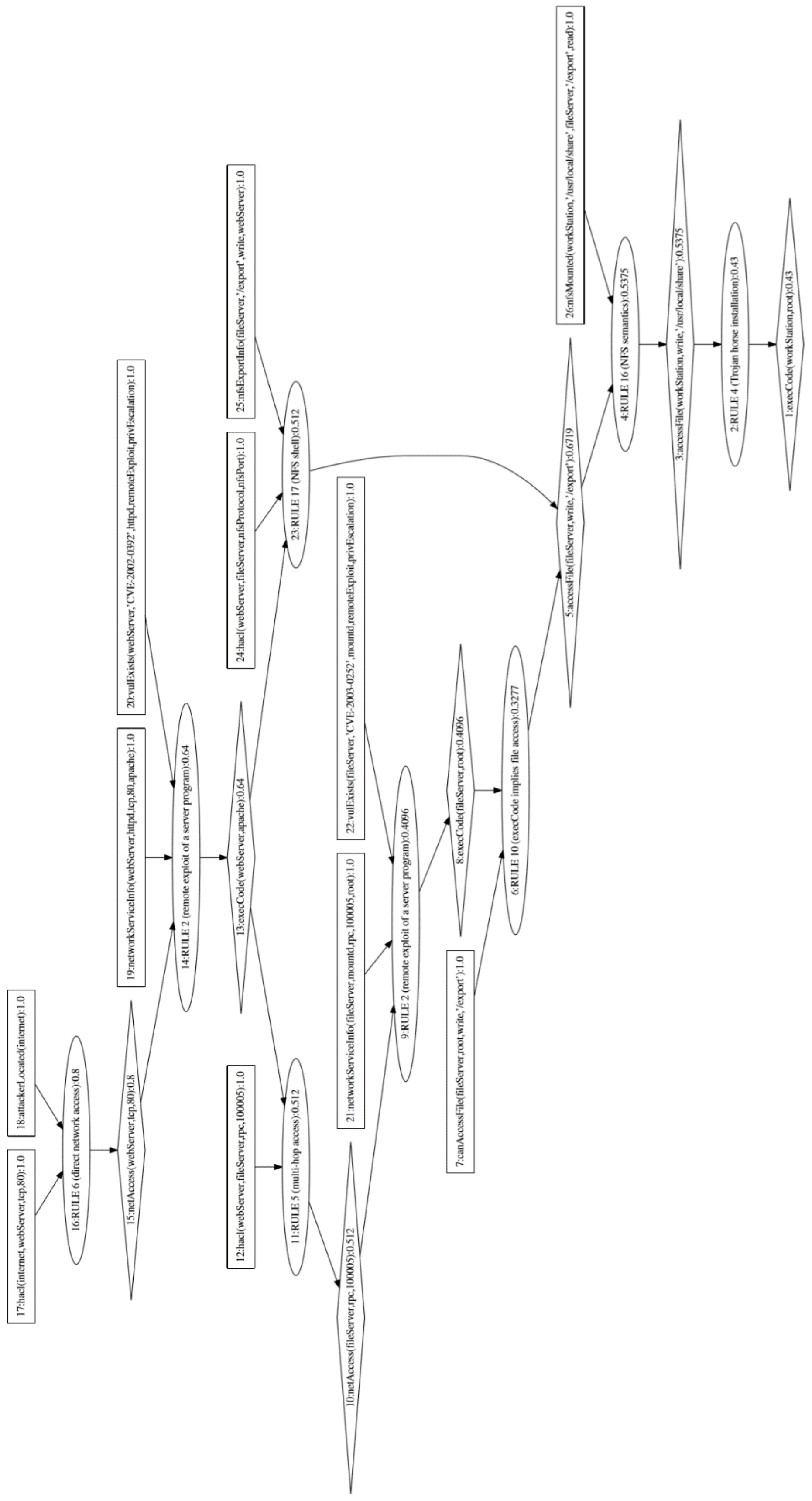

17:hacl(internet,webServer,tcp,80):1.0
18:attackerLocated(internet):1.0
16:RULE 6 (direct network access):0.8
15:netAccess(webServer,tcp,80):0.8
19:networkServiceInfo(webServer,httpd,tcp,80,apache):1.0
20:vulExists(webServer,'CVE-2002-0392',httpd,remoteExploit,privEscalation):1.0
14:RULE 2 (remote exploit of a server program):0.64
13:execCode(webServer,apache):0.64
12:hacl(webServer,fileServer,rpc,100005):1.0
11:RULE 5 (multi-hop access):0.512
10:netAccess(fileServer,rpc,100005):0.512
21:networkServiceInfo(fileServer,mountd,rpc,100005,root):1.0
22:vulExists(fileServer,'CVE-2003-0252',mountd,remoteExploit,privEscalation):1.0
9:RULE 2 (remote exploit of a server program):0.4096
8:execCode(fileServer,root):0.4096
7:canAccessFile(fileServer,root,write,'/export'):1.0
6:RULE 10 (execCode implies file access):0.3277
24:hacl(webServer,fileServer,nfsProtocol,nfsPort):1.0
25:nfsExportInfo(fileServer,'/export',write,webServer):1.0
23:RULE 17 (NFS shell):0.512
5:accessFile(fileServer,write,'/export'):0.6719
26:nfsMounted(workStation,'/usr/local/share',fileServer,'/export',read):1.0
4:RULE 16 (NFS semantics):0.5375
3:accessFile(workStation,write,'/usr/local/share'):0.5375
2:RULE 4 (Trojan horse installation):0.43
1:execCode(workStation,root):0.43

Figure 5. Attack graph for Model Network

Node #	*Node Name*	*Node Type*
1	execCode (workStation, root)	OR
2	RULE 4 (Trojan horse installation)	AND
3	accessFile (workStation, write, ’/usr/local/share’)	OR
4	RULE 16 (NFS semantics)	AND
5	accessFile (fileServer, write, ‘/export’)	OR
6	RULE 10 (execCode implies file access)	AND
7	canAccessFile (fileServer, root, write,’/export’)	LEAF
8	execCode (fileServer, root)	OR
9	RULE 2 (remove exploit of a server program)	AND
10	netAccess (fileServer, rpc, 100005)	OR
11	RULE 5 (multi-hop access)	AND
12	hacl (webServer, fileServer, rpc, 100005)	LEAF
13	execCode (webServer, apache)	OR
14	RULE 2 (remote exploit of a server program)	AND
15	netAcess (webServer, tcp, 80)	OR
16	RULE 6 (direct network access)	AND
17	hacl (internet, webServer, tcp, 80)	LEAF
18	attackerLocated (internet)	LEAF
19	networkServiceInfo (webServer, httpd, tcp, 80, apache)	LEAF
20	vulnExists(webServer, ‘CVE-2002-0392’, httpd, remoteExploit, privEscalation)	LEAF
21	networkServiceInfo(fileServer,mountd,rpc,100005,root)	LEAF
22	vulExists(fileServer, ‘CVE-2003-0252’, mountd, remoteExploit, privEscalation)	LEAF
23	RULE 17 (NFS shell)	AND
24	hacl (webServer, fileServer, nfsProtocol, nfsPort)	LEAF
25	nfsExportInfo(fileServer, ‘/export’, write, webServer)	LEAF
26	nfsMounted(workStation, ‘usr/local/share’, fileServer, “/export’, read)	LEAF

Table 3. Attack graph nodes for Model Network

6.3 Analysis of Attack Graph

Based on the attack graph, there are several ways that an attacker could move through this network in order to reach the final goal of executing code on the workstation. To begin, the attacker is based on the internet, which is represented in the attack graph as node 18. From the internet, the attacker can compromise the web server by remotely exploiting the vulnerability CVE-2002-0392. As the web server is allowed to access the file server, the attacker can then exploit the vulnerability CVE-2003-0252 and become *root* on the file server. Once the attacker is *root* on the file server, he or she can modify files on the file server. Since the executable binaries for the workstation reside on the file server, the attacker can change executable programs on the workstation. If a user then executes an attacker compromised program, the attacker would have the ability to execute code on the workstation, thus reaching his or her goal.

The MulVal engine also implements a Bayesian network to determine the conditional probabilities of a threat actor moving through the attack graph. These values can be found on Figure 5 as the number after the description of the node and a colon. For example, Node 23, Rule 17 (NFS shell) has the probability of 0.512. Based on the outputted attack graph (Figure 5), the probability that a threat actor would be able to execute code on the workstation, the goal of the attack graph, is 0.43. That means that 43% of threat actors would be able to infiltrate the workstation and execute code.

6.4 FAIR Analysis

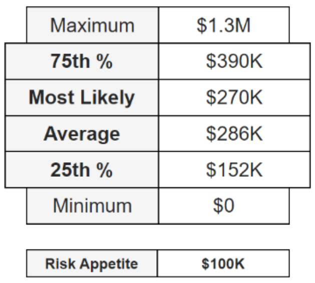
Now that an attack graph has been made, a FAIR analysis can be defined. Realistically, an organization would define the scope of a scenario that FAIR will be applied to and then an attack graph would be created. In this case, an attack graph and model network have already been defined, so instead, the scope of the scenario will be decided based on the attack graph. The end goal of the attack graph, an attacker executing code on the workstation, is not a scenario in FAIR. However, by being able to execute code on the workstation, the attacker has compromised the integrity of the asset, which is modellable by FAIR. Thus, this FAIR scenario will calculate risk, expressed as Loss Exposure (how much money a company could lose), due to Cyber Criminals harming the integrity of the workstation. Cyber Criminals are chosen in this case as they were discussed earlier.

Based on the attack graph created earlier, the probability that a threat actor would have a high enough Threat Capability to compromise the Resistance Strength of the workstation in order to execute code was 43%. This is also a measurement for FAIR Vulnerability, as FAIR Vulnerability is the probability that an event involving a threat will result in a loss. It is important to note that defining FAIR vulnerability using this single attack graph increases the specificity of the FAIR Analysis. It would not be correct to assume that this scenario would also cover a threat actor stealing information from the file server. While an actor could steal information from the file server through similar attack paths as an actor attempting to execute code on the web server, the attack graph was created with the latter goal and the probability reflects that particular goal, not all goals. To reflect this, the scope of the scenario should be updated to represent the risk of Cyber Criminals harming the integrity of the workstation by executing arbitrary code on the workstation.

There are other aspects to a FAIR scenario besides FAIR Vulnerability, like Loss Event Frequency and Loss Magnitude. Since this is a proof of concept, much of this information is unavailable and estimates are used instead. The Threat Event Frequency of Cyber Criminals attempting to comprise the integrity of the workstation within this scenario is estimated at a minimum of once per year, a maximum of 10 times per year with a most likely of 5.5 times per year. This Threat Effect Frequency may be high for this asset. From the little information known about this network, there is no indication that the workstation stores valuable information that would be targeted by Cyber Criminals. But for the sake of having results, (if there are no threat events, there is no risk) it is assumed that Cyber Criminals attempt to comprise the Model Network between 1 and 10 times a year.

As for Primary and Secondary Loss, estimations will also be used. Within Primary Loss, the replacement cost for the asset is estimated between $100 and $100,000 with a most likely of $50,050. If the integrity of the machines was compromised and valuable software corrupted, it may cost a large amount money to replace. However, it is difficult to estimate without more information, hence the broad range. Primary Loss also includes the amount of money spent on fixing the problem. Estimated, it would take a minimum of 50 hours to a maximum of 100 hours to fix damage to the asset, with a most likely of 75 hours. This value is multiplied by the average wage of the employee, which is set at $60 per hour, to determine the cost of man hours if the integrity of the workstation was compromised. Within Secondary Loss, it is estimated that between 20% to 70% of integrity compromises would have an adverse effect on secondary stakeholders. In this case, the secondary stakeholders could be the users of the workstation which are estimated to have a worth of $100 to the organization. Within a real FAIR analysis, all of these values would have defensible evidence to support the validity of these estimations.

Now that all aspects of a FAIR scenario have been established, an analysis is run on the scenario. Figure 6 is a graphical representation of the results of the FAIR analysis. This shows that the potential Loss Exposure is as high as 1.3 million dollars with an average loss of $286,000. This output was created using the RiskLens Cyber Risk Quantification Engine [25]. The output histogram is a binning of the results of a Monte Carlo simulation of the FAIR Scenario. An interesting feature of the RiskLens output is the red line present on the histogram. This corresponds with the Risk Appetite set by the organization. The Risk Appetite, also known as Risk Tolerance, is the "the amount and type of risk that an organization is willing to take in order to meet their strategic objectives" [26]. Within this example, the Risk Appetite of the organization is $100,000. Looking at the output, one can see that the Risk Appetite is lower than a majority of the distribution. This indicates that there is more risk in this scenario than the organization is willing to have and there should be steps taken to mitigate this risk.



Maximum	$1.3M
75th %	$390K
Most Likely	$270K
Average	$286K
25th %	$152K
Minimum	$0
Risk Appetite	**$100K**

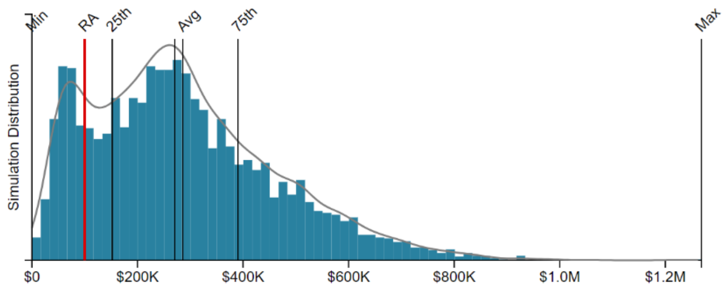


Figure 6. Loss Exposure of Model Network

By adding an attack graph to the calculation of FAIR Vulnerability, the results of this analysis are more defensible. Rather than having an analyst set the Resistance Strength of the asset, the attack graph can show the exact path an attacker would be able to take to achieve the goal of executing code on the workstation, allowing for a more granularized approach to the analysis. Using Bayesian probabilities further increases the defensibility of the results. Using attack graphs within FAIR is flexible; if there was a situation that the Bayesian probabilities do not accurately represent the situation, these probabilities can be adjusted. Additionally, since the asset is now situated within an attack graph, a risk analyst completing a FAIR analysis will also have a better idea of what sort of vulnerabilities and configurations are important. This provides more insight into potential remediation methods and ways that risk can be mitigated.

7. EXPERIMENTATION

Another advantage of adding attack graphs to FAIR scenarios is the ability to experiment and adjust the attack graph based on what-if analyses. For example, removing a vulnerability from a network and determining the new attack graph would show the effect of patching. For the Model Network, the machines have long uptimes and any patching would result in loss of availability [5]. Any downtime is best avoided, so the ability to theoretically patch vulnerabilities and see the change in resulting risk is very valuable. Within this section, different experiments will be ran on the Model Network to highlight helpful analyses that are available if attack graphs are integrated into FAIR. Additionally, research on CVSS scores will be integrated into the current MulVal engine to increase the realism of the attack graphs.

7.1 Original Model Network with CVSS Calculation

In the original calculation of the MulVal engine, all Leaf nodes, including nodes representing vulnerabilities are set with a probability of 1. This is not accurate as there are differences in the difficulty of vulnerabilities; not all vulnerabilities require the same level of skill of effort to exploit. Because of this, it is important to integrate CVSS scores into the calculation of the probability of steps requiring vulnerability exploitation. Research discussed earlier [20] was employed to calculate the skill and effort needed to successfully exploit each vulnerability. A vulnerability found in the network is CVE-

2002-0392, which is located on the web server. The exploitability metrics for CVE-2002-0392 are *Network* for the Access Vector, *Low* for Access Complexity, and *None* for Authentication. Completing Equation 1 with the corresponding numerical values in Table 1, the resulting score for skill and effort is 1. 70% of simulated attackers sampled from a normal distribution were able to successfully exploit a vulnerability of that score [20]. Thus, the probability of an attacker successfully exploiting that vulnerability is also 70%. CVE-2003-0252, which is on the file server has the same exploitability metric, and thus also has 0.7 probability. The MulVal Bayesian engine was adapted to allow for differing probability of leaf nodes and was run with nodes 20 and 22 with probabilities of 0.7 corresponding to the two vulnerabilities. The attack graph with the probability scores is seen in Appendix B.

The probability of 0.7 for both vulnerabilities is based on an entire population of threat actors. In the FAIR scenario defined earlier, the analysis was specifically calculating the risk of Cyber Criminals affecting the integrity of the workstation. Most Cyber Criminals are in the 90th percentile of threat actors. This means that while only 70% of threat actors in the general population would be able to exploit these vulnerabilities, the threat capability of Cyber Criminals exceeds the general population. Thus, the original probability of the MulVal tool, with the probability of exploiting a vulnerability is 1, is more similar to the actual probability of a Cyber Criminal being able to exploit a vulnerability than the general population. However, knowing the FAIR Vulnerability of an asset for the general threat population is also useful. Therefore, both the original MulVal tool with the probability of vulnerabilities being exploiting being 1 and the probability of the vulnerability being exploited based on the effort and skill score of the vulnerability determined by CVSS scores will be explored.

The corresponding Loss Exposure of this experiment is the Loss Exposure of a general threat population, not specifically Cyber Criminals. Here it can be seen that the FAIR Vulnerability of the general threat population being able to reach the end goal of executing code on the web server is 0.2953 compared to 0.43 when the probability of exploiting a vulnerability was 1. This highlights the importance of understanding who the threat actor is in a scenario. Within this particular scenario, if an organization was concerned about a general threat population the average Loss Exposure would be $204K. On the other hand, If the organization was concerned about Cyber Criminals targeting their assets, the threat capability of Cyber Criminals on average is higher than the general threat population, so the average Loss Exposure of that scenario is $286K as Cyber Criminals are more likely to be able to succeed in their goals than the general population.

Maximum	$892K
75th %	$282K
Most Likely	$55K
Average	$201K
25th %	$75K
Minimum	$0

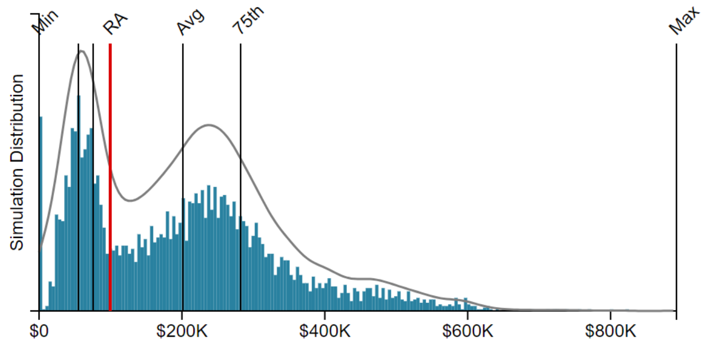
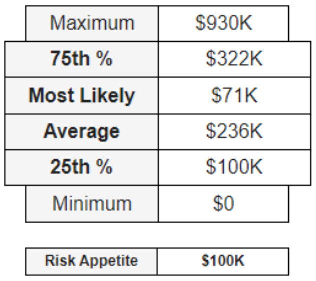


Figure 7. Loss Exposure of Model Network with CVSS calculation

7.2 Remove Vulnerability on File Server from Model Network

An interesting prospect of using attack graphs within FAIR Analyses is the ability to determine return on investment of patching vulnerabilities. To demonstrate this, the same network as describe earlier was run with CVE-2003-0252, originally present on the file server, was removed. Once this vulnerability was removed, a new attack graph was calculated (Appendix C). This attack graph shows that the FAIR Vulnerability of the asset is 32.77%. Compared to the 43% FAIR Vulnerability found when the network has 2 vulnerabilities, the FAIR Vulnerability of the asset was reduced by 10.23%. That seems like a lot, but it is impossible to know how that affects risk without further analysis. The previous FAIR Analysis was run with the same inputs as the original except FAIR Vulnerability was updated from 43% as with 2 vulnerabilities, to 32.77% with just one vulnerability. The results are seen in Figure 8.



Maximum	$930K
75th %	$322K
Most Likely	$71K
Average	$236K
25th %	$100K
Minimum	$0
Risk Appetite	**$100K**

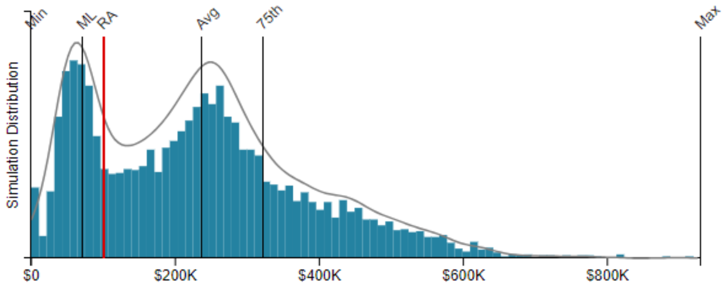


Figure 8. Loss Exposure of Model Network with CVE-2003-0252 removed from file server

Comparing the two Loss Exposures, we can see that removing one vulnerability reduced the average Loss Exposure from $286K to $236K, a difference of $50K. If the patching of CVE-2003-0252 on the file server cost the organization $1,000 due to the availability of the file server being down, this would be well worth the investment. This scenario can be seen in Figure 9. If patching the vulnerability cost the organization significantly more, then it might not be worth it as the organization is spending more money fixing something than if a loss event was to occur.

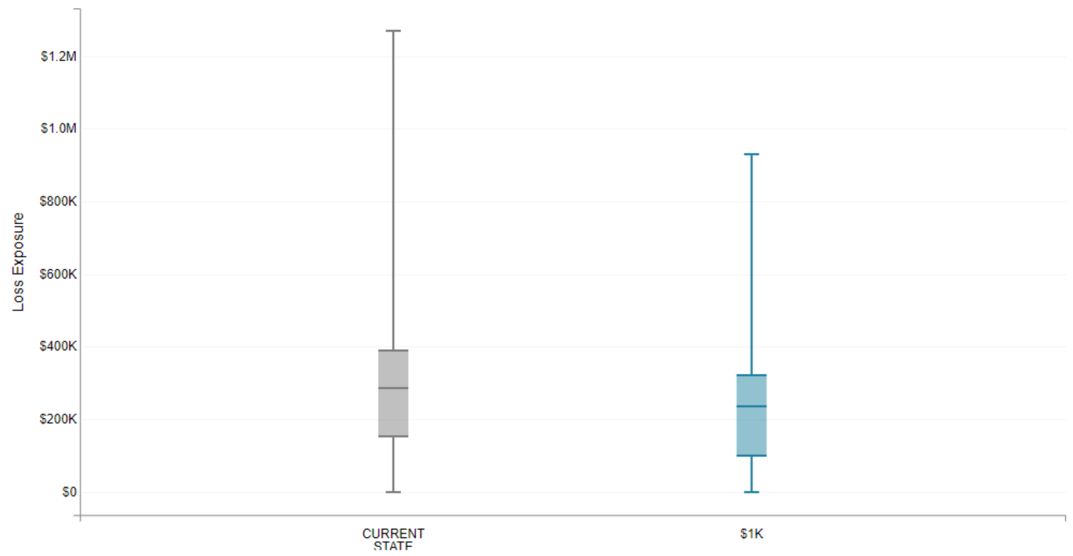
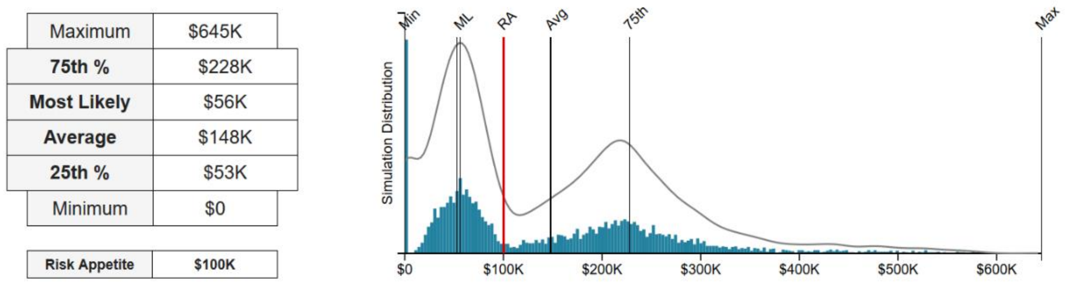


Figure 9. Comparison of Loss Exposure of the original Model Network compared to the Model Network without CVE-2003-0252 on the file server

It is also interesting to explore how threat community impacts this single vulnerability's probability. The same scenario, but with probability of a threat actor successfully exploiting the vulnerability based on the entire threat actor population using the CVSS score, was run. The corresponding attack graphs has the probability of the threat actor executing code on the workstation at 0.2294 (see Appendix D). The same scenario was run with the vulnerability of 22.94% and the results are shown in Figure 10.



Maximum	$645K
75th %	$228K
Most Likely	$56K
Average	$148K
25th %	$53K
Minimum	$0

Risk Appetite	**$100K**

Figure 10. Loss Exposure of Model Network with CVE-2003-0252 removed from file server and CVSS calculation.

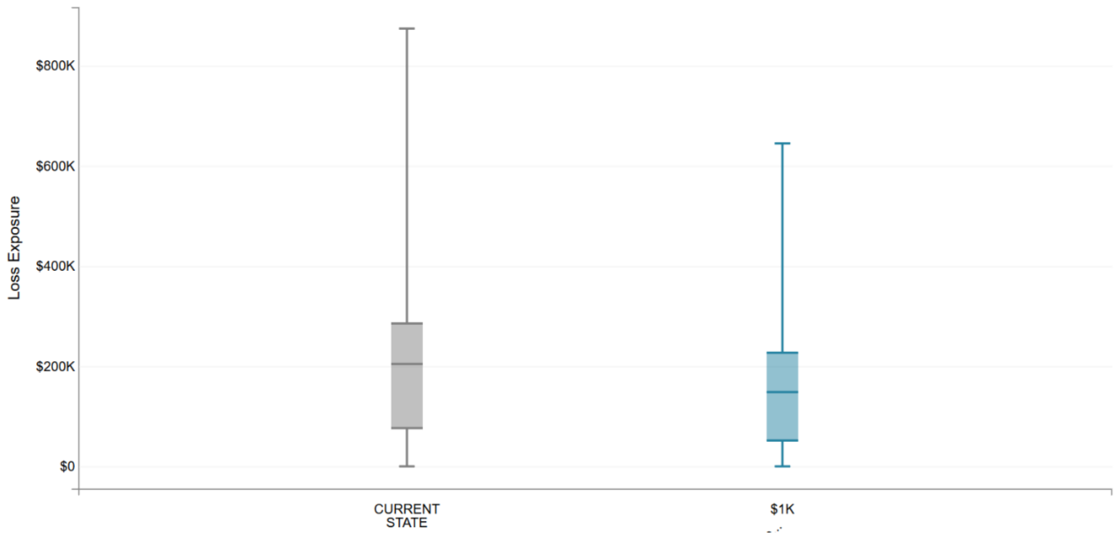


Figure 11. Comparison of Loss Exposure of the original Model Network compared to the Model Network without CVE-2003-0252 on the file server with CVSS calculation

7.3 Remove Vulnerability on Web Server from Model Network

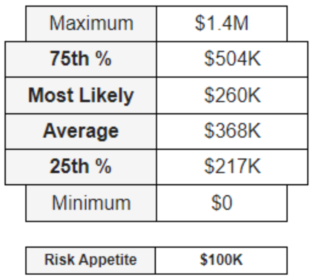
For this experiment, CVE-2002-0392, present on the web server, was patched. The result of the MulVal tool was surprising. Patching the vulnerability on the file server caused no viable attack path to be found, meaning that an attacker would not be able to reach the end goal of executing arbitrary code on the workstation once this vulnerability was patched. This provides interesting insight that patching a single vulnerability can render an entire attack graph useless. This also highlights that there can exist choke points within the attack graph that every pathway is required to go through to get to the end goal. By recognizing these choke points, particular machines and vulnerabilities can be monitored and patched more efficiently. However, it is important to note, that just because there was not an attack graph for this very particular goal, that does not mean that the entire asset is safe. There could be other goals of attackers that would be used within different FAIR Analyses.

7.4 Remove Both Vulnerabilities from Model Network

Next, MulVal was run with no vulnerabilities present in either the web server or the file server. Similarly, to when the CVE-2002-0392 was removed from the network, there is no pathway for an attacker existing on the internet to reach the workstation in order to execute code.

7.5 Add Hypothetical Low Complexity Vulnerability to Model Network

Another interesting experiment is to see how risk is affected if a new vulnerability is added to system. For this, a low complexity hypothetical vulnerability was added to the network. This hypothetical vulnerability can be remotely exploited on the web server to escalate privilege using the program httpd. The attack graph with this hypothetical vulnerability is found in Appendix E. This attack graph shows that the probability of an attacker being able to execute code on the workstation is 53.23%, when the probability of exploiting all vulnerabilities is 1. Compared to the vulnerability of 43% of the original situation, 53.23% is quite an increase. This increase can be seen in the Loss Exposure of these two scenarios. Figure 12 shows the result of the Model Network with 3 vulnerabilities and a vulnerability exploitation probability of 1. The addition of a 3rd vulnerability into the system increased the average loss exposure from \$286K to \$368K, an increase of \$82K.



Maximum	\$1.4M
75th %	\$504K
Most Likely	\$260K
Average	\$368K
25th %	\$217K
Minimum	\$0
Risk Appetite	**\$100K**

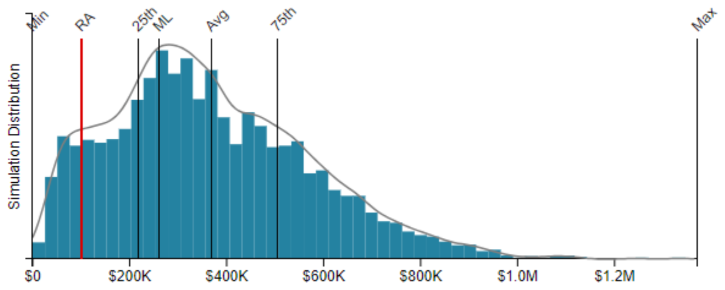


Figure 12. Loss Exposure of Model Network with hypothetical low complexity vulnerability

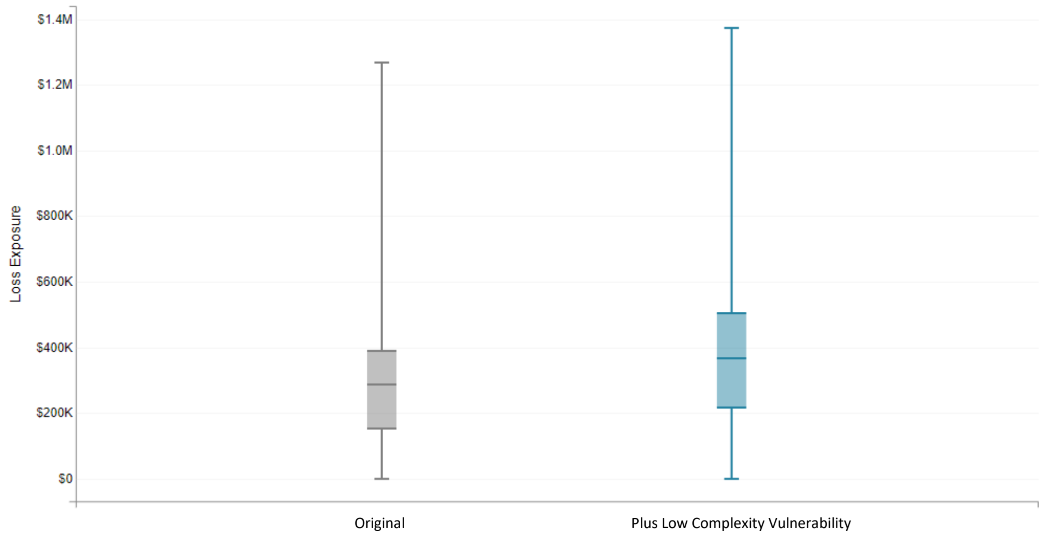


Figure 13. Comparison of the Loss Exposure of the original Model Network compared to the Model Network with hypothetical low complexity vulnerability.

This situation could be seen when the scenario is using Cyber Criminals as the threat actor, as the probability of cyber criminals being able to exploit the vulnerabilities present on the network is near 1. However, there are also interesting implications to explore if the threat actor is not a Cyber Criminal and instead was the general threat population.

Using the CVSS calculated probabilities (0.7) of the 2 actual vulnerabilities, the hypothetical vulnerability was added with the same metric for exploitability as the two known probabilities (Access Vector: *Network*, Access complexity: *Low*, and Authentication: *None*). This means that the hypothetical vulnerability also has a probability of exploitation of 0.7. Appendix F shows the attack graph for this situation. Figure 14 shows the results for the loss exposure of the Model Network including a hypothetical vulnerability with a probability of exploitation of 0.7 along with the two known vulnerabilities similarly with a probability of exploitation of 0.7.

Maximum	$1.1M
75th %	$410K
Most Likely	$243K
Average	$297K
25th %	$163K
Minimum	$0

Risk Appetite	**$100K**

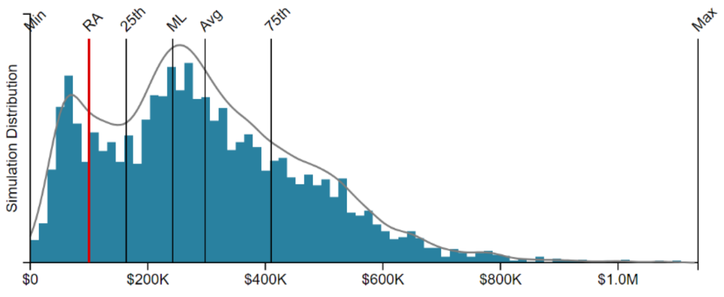


Figure 14. Loss Exposure of Model Network with hypothetical low complexity vulnerability and CVSS calculation.

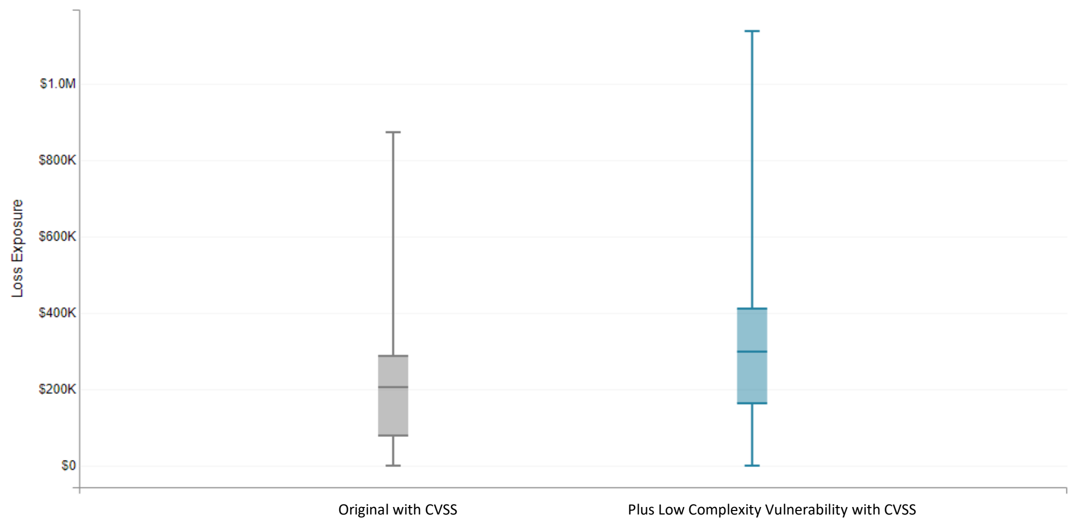


Figure 15. Comparison of the Loss Exposure of the original Model Network compared to the Model Network with hypothetical low complexity vulnerability with CVSS Calculations.

7.6 Add Hypothetical High Complexity Vulnerability to Model Network

This experiment covers a situation where the third vulnerability added to the Model Network is more complex than in the previous experiment. In this situation, the exploitability metrics of the hypothetical vulnerability are Access Vector: *Network*, Access complexity: *High*, and Authentication: *Multiple*. In a real FAIR scenario, the viability of this sort of vulnerability can be determined. However, without much information, it is difficult to say if the exploitability metrics of this vulnerability could be possible within this network, so it will be assumed that they are. The Access Vector of this vulnerability has stayed the same since attacker will have access to the web server from the internet via the attack graph. Adjusting metrics based on these situations was discussed in section 5.2. The effort and skill score for this hypothetical vulnerability is 3.98, which corresponds to about 30% of attackers selected from a normal distribution being able to exploit this vulnerability. Thus, the probability of the general threat community being able to exploit this vulnerability is 0.3. Appendix G shows the attack graph and Figure 16 shows the results of this FAIR analysis. The result of this experiment shows that risk when a high complexity vulnerability is added to the Model Network is less than the risk with a low complexity vulnerability. That is because a low complexity vulnerability is easier to exploit than a high complexity vulnerability.

Maximum	$983K
75th %	$342K
Most Likely	$235K
Average	$250K
25th %	$117K
Minimum	$0

Risk Appetite	**$100K**

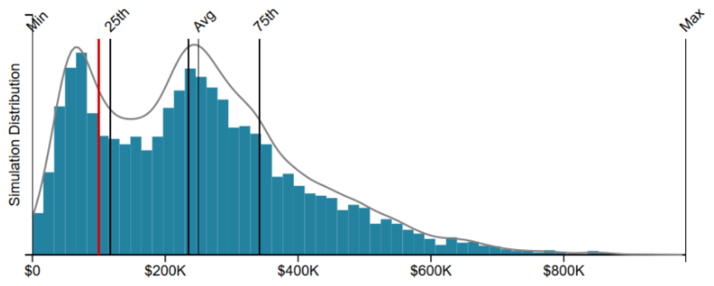


Figure 16. Loss Exposure of Model Network with hypothetical high complexity vulnerability and CVSS Calculation.

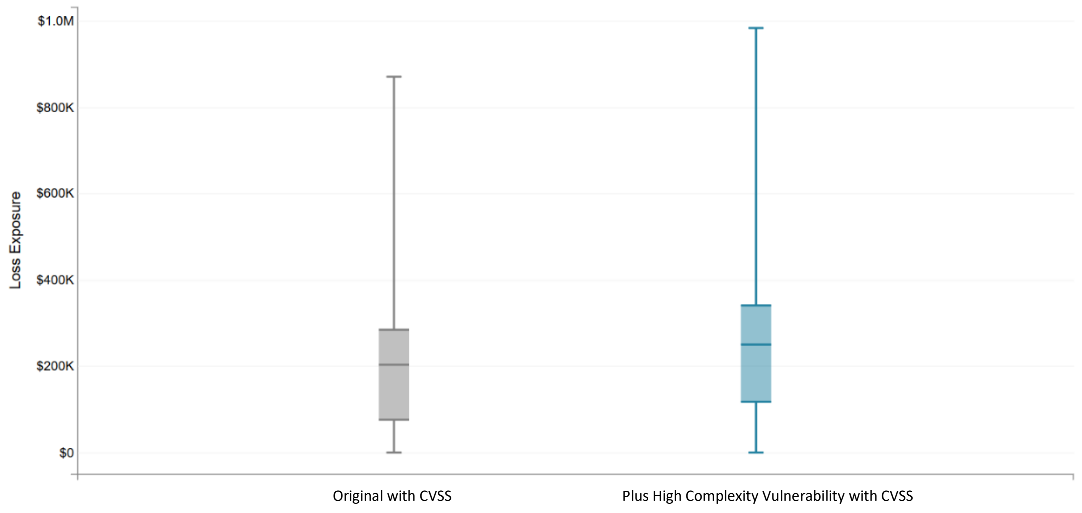


Figure 17. Comparison of the Loss Exposure of the original Model Network compared to the Model Network with hypothetical high complexity vulnerability with CVSS calculation

8. FURTHER RESEARCH

This paper is the first step in integrating attack graphs into the realm of FAIR. As such, there is much more research that can be done to integrate attack graphs successfully into FAIR. In general, FAIR analyses can take a large amount of time since analysts have to determine the estimations of ontology components. Adding the time and resources needed to determine network configuration and other information needed for attack graph creation increases this pain point within FAIR. There has already been research on using scanners and other tools to speed up attack graph generation [27]. These tools should be explored in greater depth and added into information gathering for a FAIR analysis. In addition, using attack graphs in more realistic and complex FAIR scenarios will be able to determine how this process scales when an analysis includes numerous assets.

Research has been done to add intrusion detection systems to the Bayesian networks of attack graphs. These networks can become sophisticated enough to distinguish between stealthy and detectable attacks [12] and model non-perfect intrusion detection systems [28]. Complexities such as these and other controls can be added to the attack graph to even better represent real life systems. This also opens the opportunity for what-if analyses to determine when adding intrusion detection systems to an asset would be worthwhile and impactful.

One limitation of the MulVal engine is that the set of interaction rules prepackaged with the engine are limited. They have not yet created exploit rules for vulnerabilities whose exploit consequences are confidentiality or integrity loss [5]. Further research can expand these rules and the engine to include a wider variety of rules, perhaps employing CWEs to characterize different sets of vulnerabilities and the attack pathways that normally accompany those vulnerabilities. Other extensions have been proposed for the MulVal tool; these can also be employed to increase the modeling abilities of MulVal [29-30].

9. CONCLUSION

Revised FAIR enhances the calculation of FAIR Vulnerability from being based on analysts' opinions, to being situated within a Bayesian network that models how a threat event would happen. Revised FAIR also offers flexibility in the calculation of FAIR Vulnerability as analysts can step in and change probabilities of nodes within the attack graph to better reflect knowledge about the network not captured by the original attack graph. Demonstrated within this paper, the probabilities of particular nodes can be determined by the effort and skill required to exploit a vulnerability based on CVSS scores.

Adding attack graphs to FAIR analyses allows complex environments and assets to be broken down into more understandable and quantifiable pieces providing interesting analyses and new insights. Using Revised FAIR allows analysts to conduct granular what-if analyses to determine return on investment of actions such as patching a software vulnerability. By running a FAIR analysis before and after removing a vulnerability within the network, how much the risk has been reduced can be compared with the money spent to fix the vulnerability. Additionally, an attack graph provides the ability to model the impact of new vulnerabilities within a system. This was demonstrated by adding new vulnerabilities of both low and high complexity to the network and determining how this new vulnerability impact the loss exposure of the asset. Overall, adding attack graphs to the calculation of FAIR Vulnerability provides greater precision, dependability, and detail to FAIR analyses.

REFERENCES

[1] J. Jones, and J. Freund. Measuring and Managing Information Risk: A FAIR Approach. Oxford, UK: Elsevier, 2015.

[2] Discussion with Bryan Smith, CTO of RiskLens. November 2017.

[3] L. Munoz-Gonzalez, D. Sgandurra, M. Barrere, and E. Lupu. “Exact inference techniques for the analysis of bayesian attack graphs,” IEEE Transactions on Dependable and Secure Computing, 2017.

[4] C. Wang, N. Du, and H. Yang. Generation and analysis of attack graphs. Procedia Engineering, 29, 4053-4057, 2017.

[5] X. Ou, S. Govindavajhala, and A. Appel. “MulVAL: a logic-based network security analyzer,” 14th USENIX Security Symposium, August 2005.

[6] J. Ramsdell. Datalog user manual. http://www.ccs.neu.edu/home/ramsdell/tools/datalog/datalog.html. Access on May 26, 2018

[7] J. Sembiring, M. Ramadhan, Y. Gondokaryono, and A. Arman. "Network security risk analysis using improved MulVAL bayesian attack graphs." International Journal on Electrical Engineering and Informatics 7, no. 4, 2015.

[8] Vulnerability Scanning Tools. https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools. Access on May 26, 2018

[9] S. Ghaffarian and H. Shahriari. Software vulnerability analysis and discovery using machine-learning and data-mining techniques: a survey. ACM Computing Surveys (CSUR), 50(4), 56, 2017.

[10] OVAL Language Overview. https://oval.mitre.org/language/about/overview.html. Access May 26, 2018.

[11] National Vulnerability Database, https://nvd.nist.gov/vuln/detail/ CVE-2002-0392, Accessed on May 4, 2018.

[12] A. François-Xavier, B. Olivier, B. Grégory, C. Vania, and D. Hervé. "Bayesian attack model for dynamic risk assessment," arXiv preprint arXiv:1606.09042, 2016.

[13] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric. In IFIP Annual Conference on Data and Applications Security and Privacy (pp. 283-296). Springer, Berlin, Heidelberg, July 2008.

[14] CNSS. National information assurance glossary. 2003.

[15] M. Tornquist, M. Analysis of software vulnerabilities through historical data. Lund University Libraries, 2017

[16] Common Vulnerabilities and Exposures. https://cve.mitre.org/about/. Accessed December 07, 2017.

[17] Common Weakness Enumeration. https://cwe.mitre.org/. Accessed December 07, 2017.

[18] Common Vulnerability Scoring System SIG. https://www.first.org/cvss/. Accessed December 07, 2017.

[19] P. Mell, K. Scarfone, and S. Romanosky. A complete guide to the common vulnerability scoring system version 2.0. June, 2007

[20] P. Cheng, L. Wang, S. Jajodia, and A. Singhal. Aggregating CVSS base scores for semantics-rich network security metrics. In Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on (pp. 31-40). IEEE, October 2012.

[21] X. Ou, W. Boyer, and M. McQueen. A scalable approach to attack graph generation. 13th ACM Conference on Computer and Communications Security (CCS 2006), October 2006.

[22] J. Sembiring, M. Ramadhan, Y. Gondokaryono, and A. Arman. Network security risk analysis using improved MulVAL bayesian attack graphs. International Journal on Electrical Engineering and Informatics 7, no. 4, 2015.

[23] NVD. CVE-2002-0392 detail. https://nvd.nist.gov/vuln/detail/CVE-2002-0392. Accessed May 28, 2018.

[24] NVD. CVE-2003-0252 detail. https://nvd.nist.gov/vuln/detail/CVE-2003-0252. Access May 28, 2018.

[25] RiskLens. www.app.risklens.com. Accessed May 28, 2018.

[26] Institute of Risk Manamgment. Risk appetite and tolerance. https://www.theirm.org/knowledge-and-resources/thought-leadership/risk-appetite-and-tolerance.aspx. Accessed May 28, 2018.

[27] O. Sheyner and J. Wing. Tools for generating and analyzing attack graphs. In International Symposium on Formal Methods for Components and Objects(pp. 344-371). Springer, Berlin, Heidelber, Novermber 2003.

[28] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. Automated generation and analysis of attack graphs. Proceedings 2002 IEEE Symposium on Security and Privacy. doi:10.1109/secpri.2002.1004377, 2002.

[29] E. Bacic, M. Froh, and G. Henderson. Mulval extensions for dynamic asset protection (No. DRDC-ONTARIO-CR-2006-251). Cinnabar Networks Inc Ottawa (Ontario), 2006.

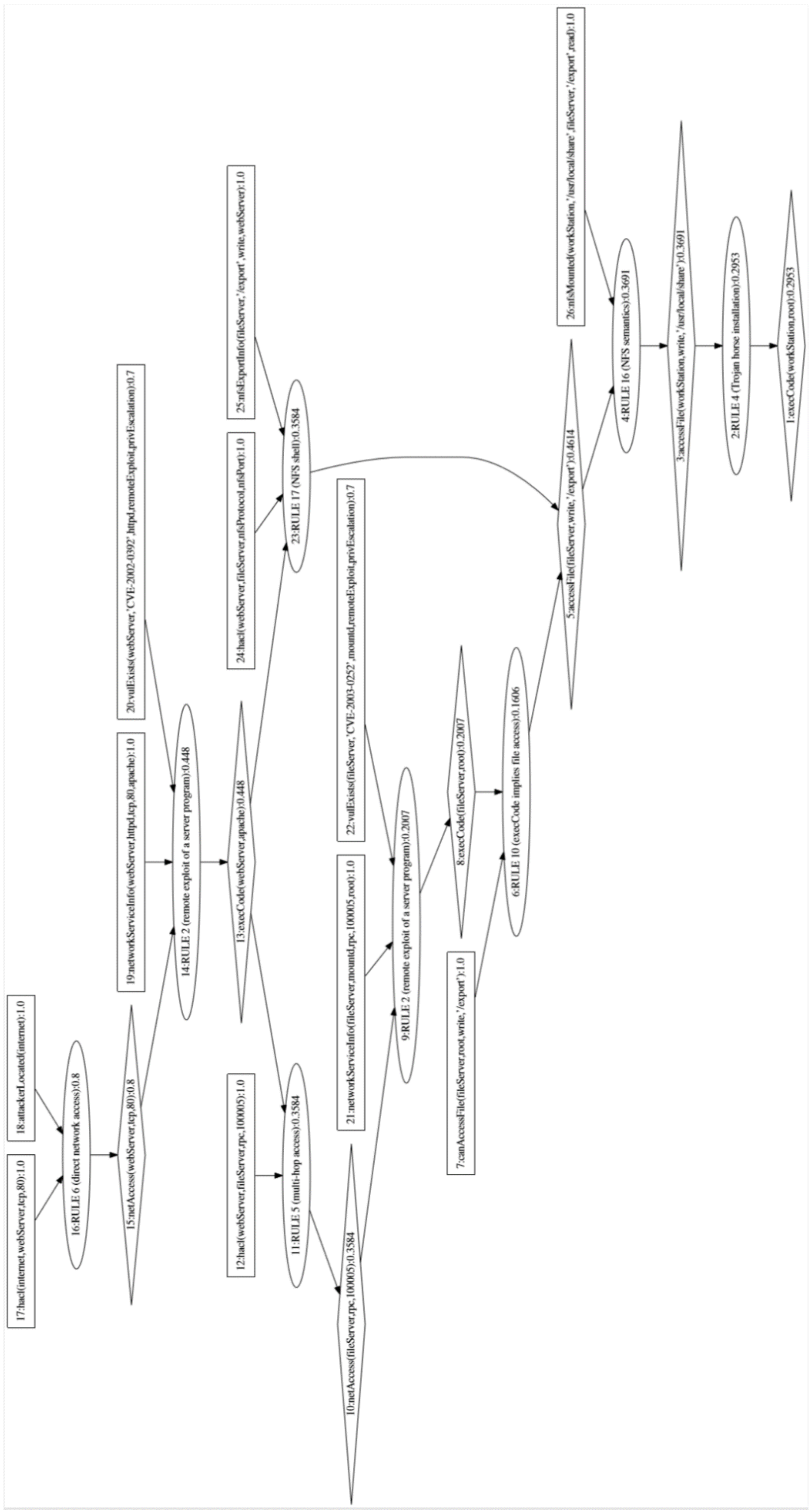
[30] M. Froh and G. Henderson. MulVAL extensions II. Defence R & D Canada-Ottawa, 2009.

APPENDIX A

```
attackerLocated(internet).
attackGoal(execCode(workStation,_)).

hacl(internet, webServer, tcp, 80).
hacl(webServer, _,  _, _).
hacl(fileServer, _, _, _).
hacl(workStation, _, _, _).
hacl(H,H,_,_).

/* configuration information of fileServer */
networkServiceInfo(fileServer, mountd, rpc, 100005, root).
nfsExportInfo(fileServer, '/export', _anyAccess, workStation).
nfsExportInfo(fileServer, '/export', _anyAccess, webServer).
vulExists(fileServer, 'CVE-2003-0252', mountd).
vulProperty('CVE-2003-0252', remoteExploit, privEscalation).
localFileProtection(fileServer, root, _, _).

/* configuration information of webServer */
vulExists(webServer, 'CVE-2002-0392', httpd).
vulProperty('CAN-2002-0392', remoteExploit, privEscalation).
networkServiceInfo(webServer , httpd, tcp , 80 , apache).

/* configuration information of workStation */
nfsMounted(workStation, '/usr/local/share', fileServer,
'/export', read).
```

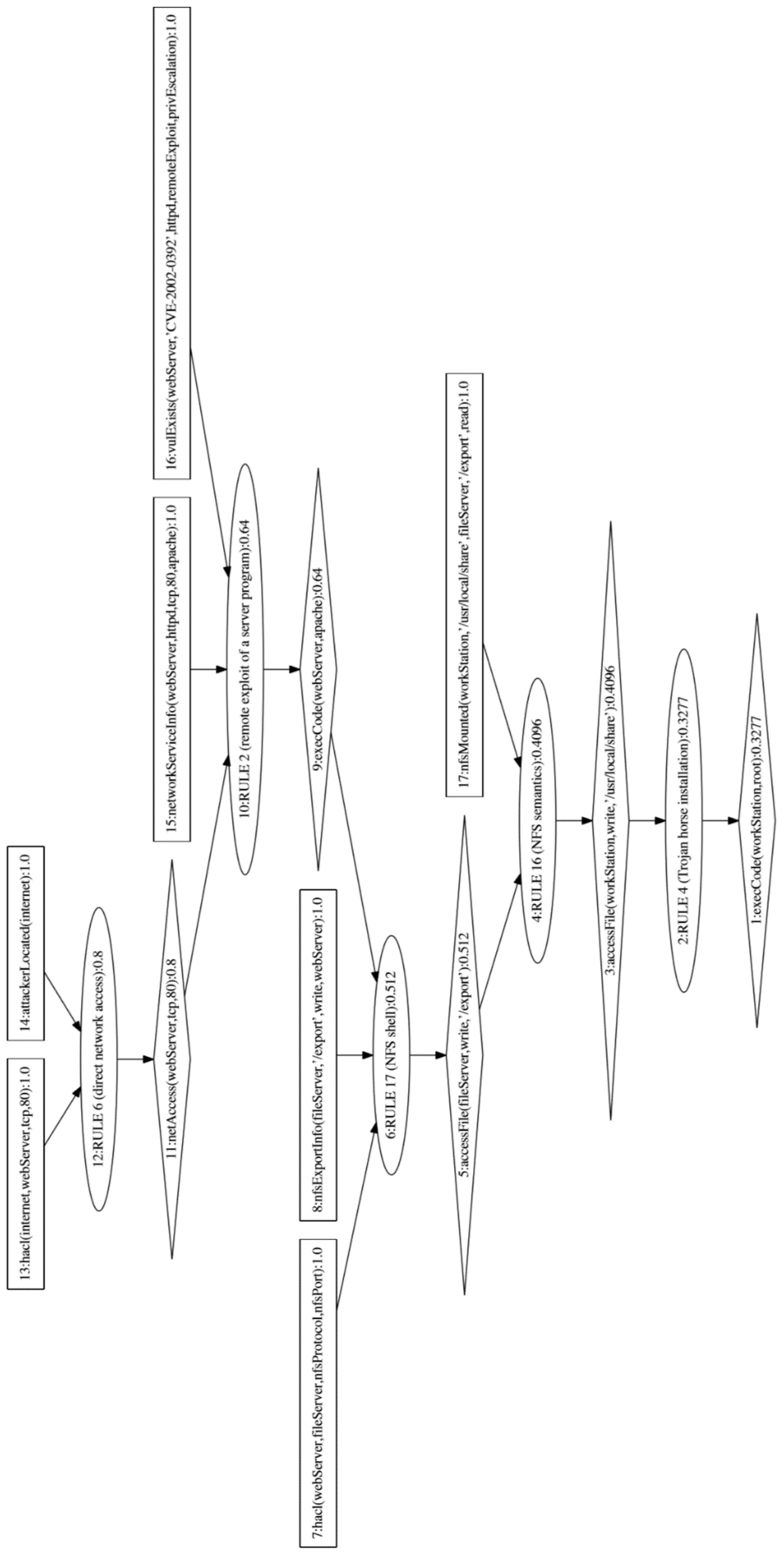
Figure 18. Information fed into MulVal for the Model Network based on [5].

APPENDIX B


17:hacl(internet,webServer,tcp,80):1.0
18:attackerLocated(internet):1.0
16:RULE 6 (direct network access):0.8
15:netAccess(webServer,tcp,80):0.8
19:networkServiceInfo(webServer,httpd,tcp,80,apache):1.0
20:vulExists(webServer,'CVE-2002-0392',httpd,remoteExploit,privEscalation):0.7
14:RULE 2 (remote exploit of a server program):0.448
13:execCode(webServer,apache):0.448
12:hacl(webServer,fileServer,rpc,100005):1.0
11:RULE 5 (multi-hop access):0.3584
10:netAccess(fileServer,rpc,100005):0.3584
24:hacl(webServer,fileServer,nfsProtocol,nfsPort):1.0
25:nfsExportInfo(fileServer,'/export',write,webServer):1.0
23:RULE 17 (NFS shell):0.3584
21:networkServiceInfo(fileServer,mountd,rpc,100005,root):1.0
22:vulExists(fileServer,'CVE-2003-0252',mountd,remoteExploit,privEscalation):0.7
9:RULE 2 (remote exploit of a server program):0.2007
8:execCode(fileServer,root):0.2007
7:canAccessFile(fileServer,root,write,'/export'):1.0
6:RULE 10 (execCode implies file access):0.1606
5:accessFile(fileServer,write,'/export'):0.4614
26:nfsMounted(workStation,'/usr/local/share',fileServer,'/export',read):1.0
4:RULE 16 (NFS semantics):0.3691
3:accessFile(workStation,write,'/usr/local/share'):0.3691
2:RULE 4 (Trojan horse installation):0.2953
1:execCode(workStation,root):0.2953

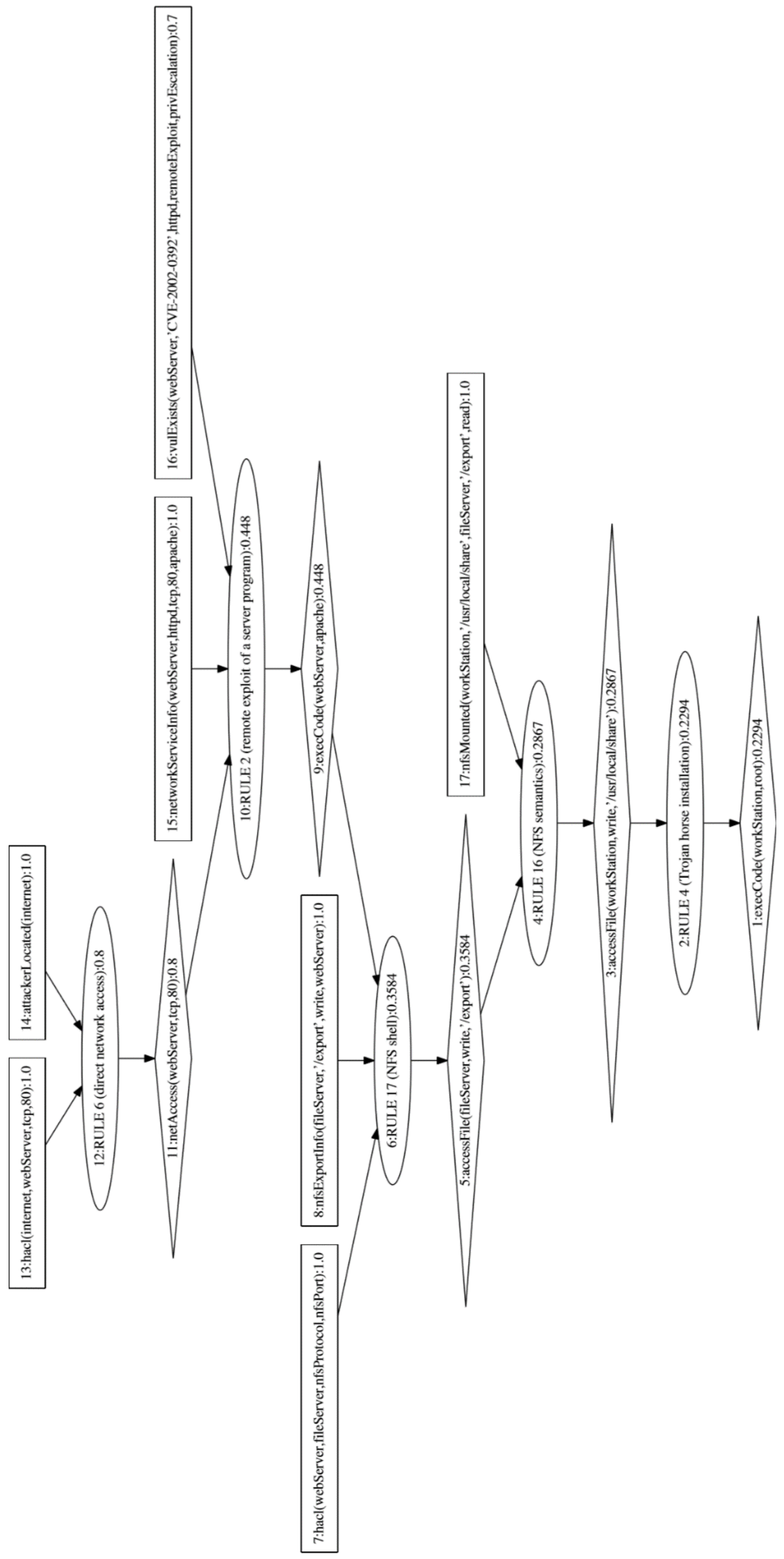
Attack graph for Original Model Network with CVSS calculation

APPENDIX C


13:hacl(internet,webServer,tcp,80):1.0
14:attackerLocated(internet):1.0
12:RULE 6 (direct network access):0.8
11:netAccess(webServer,tcp,80):0.8
15:networkServiceInfo(webServer,httpd,tcp,80,apache):1.0
16:vulExists(webServer,'CVE-2002-0392',httpd,remoteExploit,privEscalation):1.0
10:RULE 2 (remote exploit of a server program):0.64
9:execCode(webServer,apache):0.64
7:hacl(webServer,fileServer,nfsProtocol,nfsPort):1.0
8:nfsExportInfo(fileServer,'/export',write,webServer):1.0
6:RULE 17 (NFS shell):0.512
5:accessFile(fileServer,write,'/export'):0.512
17:nfsMounted(workStation,'/usr/local/share',fileServer,'/export',read):1.0
4:RULE 16 (NFS semantics):0.4096
3:accessFile(workStation,write,'/usr/local/share'):0.4096
2:RULE 4 (Trojan horse installation):0.3277
1:execCode(workStation,root):0.3277

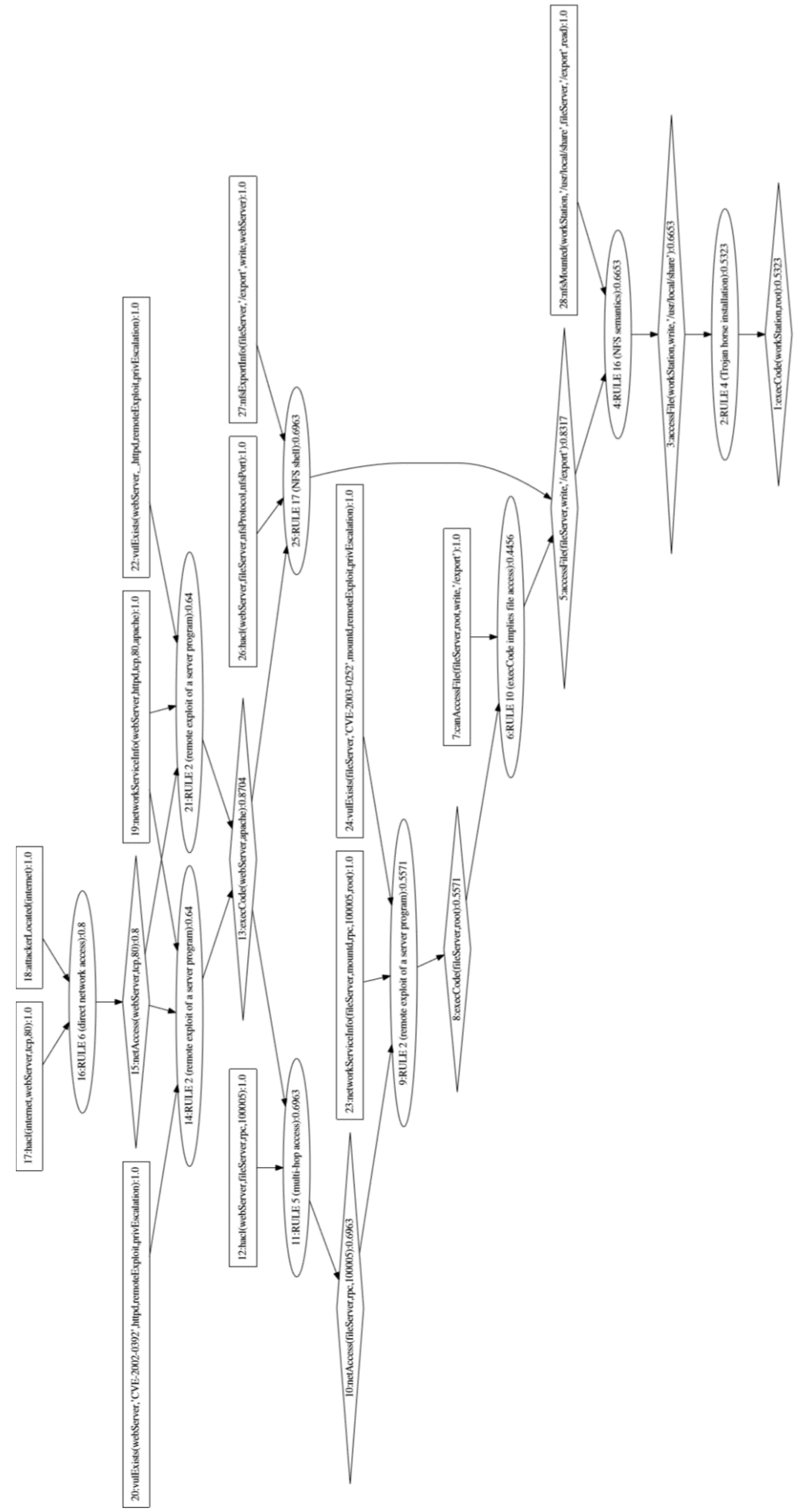
Attack graph for Model Network without CVE-2002-0392

APPENDIX D


13:hacl(internet,webServer,tcp,80):1.0
14:attackerLocated(internet):1.0
12:RULE 6 (direct network access):0.8
11:netAccess(webServer,tcp,80):0.8
15:networkServiceInfo(webServer,httpd,tcp,80,apache):1.0
16:vulExists(webServer,'CVE-2002-0392',httpd,remoteExploit,privEscalation):0.7
10:RULE 2 (remote exploit of a server program):0.448
9:execCode(webServer,apache):0.448
7:hacl(webServer,fileServer,nfsProtocol,nfsPort):1.0
8:nfsExportInfo(fileServer,'/export',write,webServer):1.0
6:RULE 17 (NFS shell):0.3584
5:accessFile(fileServer,write,'/export'):0.3584
17:nfsMounted(workStation,'/usr/local/share',fileServer,'/export',read):1.0
4:RULE 16 (NFS semantics):0.2867
3:accessFile(workStation,write,'/usr/local/share'):0.2867
2:RULE 4 (Trojan horse installation):0.2294
1:execCode(workStation,root):0.2294

Attack graph for Model Network without CVE-2002-0392 with CVSS calculation

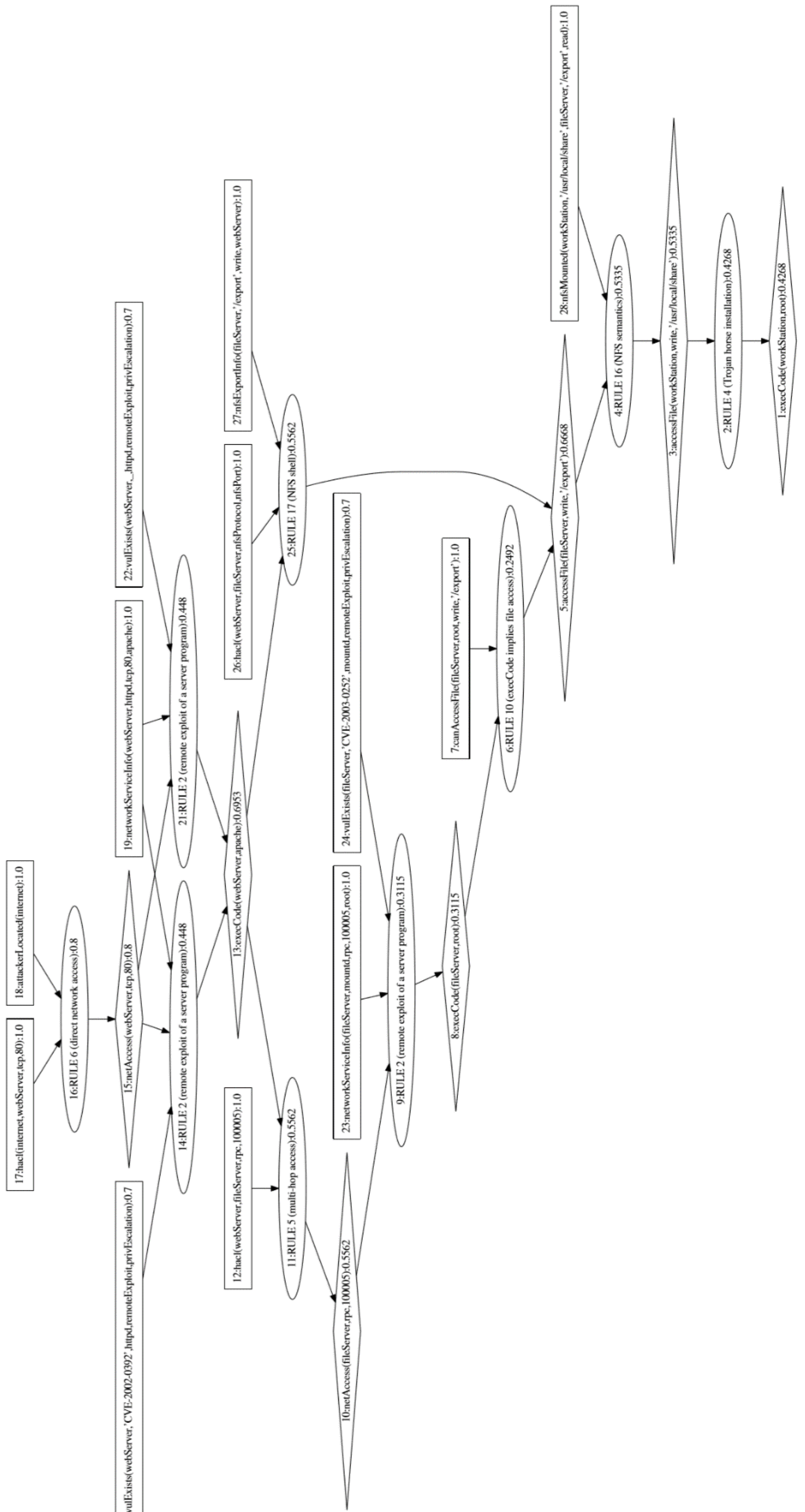
APPENDIX E



17:hacl(internet,webServer,tcp,80):1.0
18:attackerLocated(internet):1.0
16:RULE 6 (direct network access):0.8
15:netAccess(webServer,tcp,80):0.8
20:vulExists(webServer,'CVE-2002-0392',httpd,remoteExploit,privEscalation):1.0
19:networkServiceInfo(webServer,httpd,tcp,80,apache):1.0
22:vulExists(webServer,_,httpd,remoteExploit,privEscalation):1.0
14:RULE 2 (remote exploit of a server program):0.64
21:RULE 2 (remote exploit of a server program):0.64
13:execCode(webServer,apache):0.8704
12:hacl(webServer,fileServer,rpc,100005):1.0
26:hacl(webServer,fileServer,nfsProtocol,nfsPort):1.0
27:nfsExportInfo(fileServer,'/export',write,webServer):1.0
11:RULE 5 (multi-hop access):0.6963
25:RULE 17 (NFS shell):0.6963
10:netAccess(fileServer,rpc,100005):0.6963
23:networkServiceInfo(fileServer,mountd,rpc,100005,root):1.0
24:vulExists(fileServer,'CVE-2003-0252',mountd,remoteExploit,privEscalation):1.0
9:RULE 2 (remote exploit of a server program):0.5571
8:execCode(fileServer,root):0.5571
7:canAccessFile(fileServer,root,write,'/export'):1.0
6:RULE 10 (execCode implies file access):0.4456
5:accessFile(fileServer,write,'/export'):0.8317
28:nfsMounted(workStation,'/usr/local/share',fileServer,'/export',read):1.0
4:RULE 16 (NFS semantics):0.6653
3:accessFile(workStation,write,'/usr/local/share'):0.6653
2:RULE 4 (Trojan horse installation):0.5323
1:execCode(workStation,root):0.5323

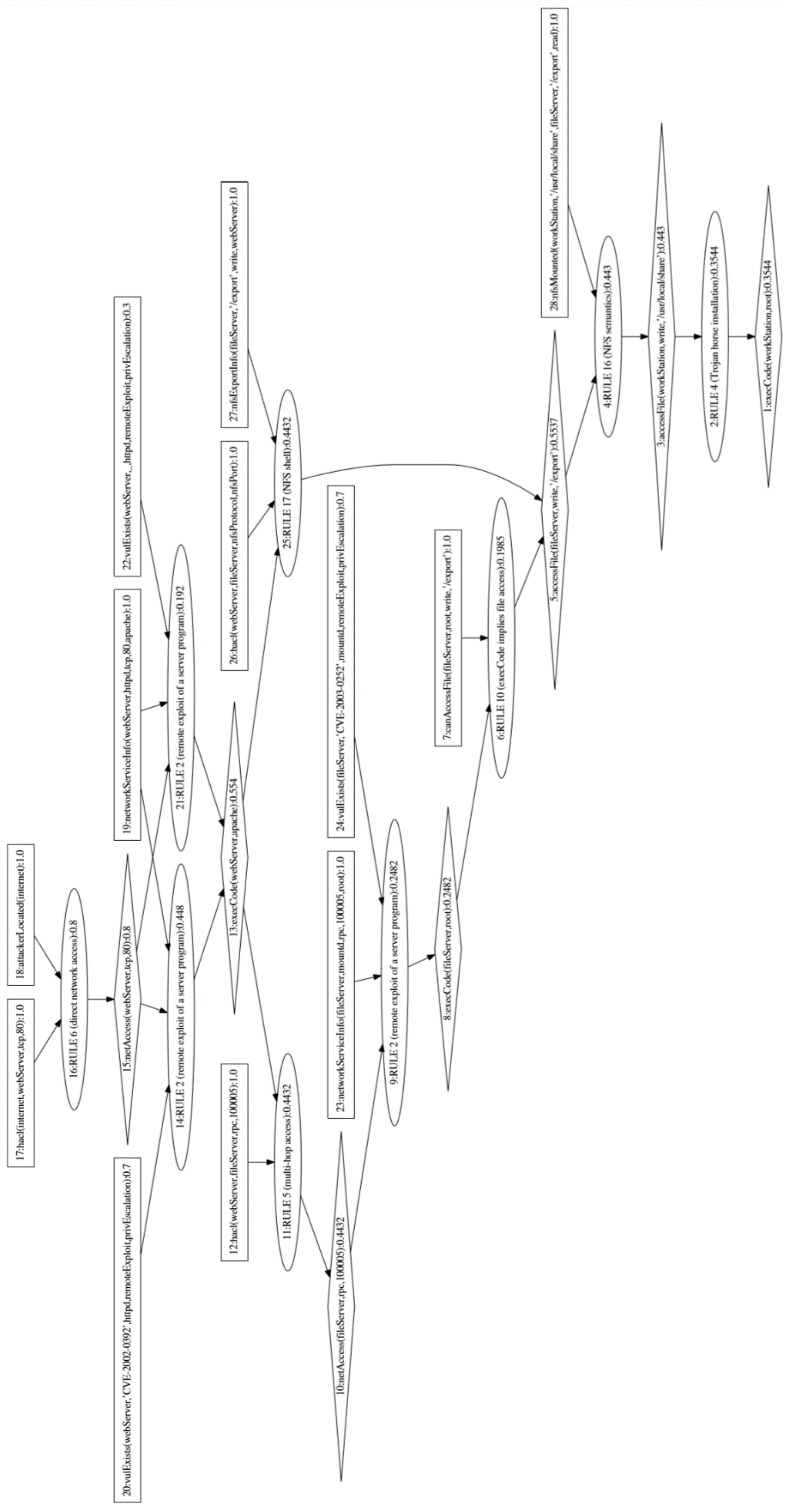
Attack graph for Model Network with low complexity vulnerability

APPENDIX F


17:hacl(internet,webServer,tcp,80):1.0
18:attackerLocated(internet):1.0
16:RULE 6 (direct network access):0.8
15:netAccess(webServer,tcp,80):0.8
20:vulExists(webServer,'CVE-2002-0392',httpd,remoteExploit,privEscalation):0.7
19:networkServiceInfo(webServer,httpd,tcp,80,apache):1.0
22:vulExists(webServer,_,httpd,remoteExploit,privEscalation):0.7
14:RULE 2 (remote exploit of a server program):0.448
21:RULE 2 (remote exploit of a server program):0.448
13:execCode(webServer,apache):0.6953
12:hacl(webServer,fileServer,rpc,100005):1.0
11:RULE 5 (multi-hop access):0.5562
26:hacl(webServer,fileServer,nfsProtocol,nfsPort):1.0
27:nfsExportInfo(fileServer,'/export',write,webServer):1.0
25:RULE 17 (NFS shell):0.5562
10:netAccess(fileServer,rpc,100005):0.5562
23:networkServiceInfo(fileServer,mountd,rpc,100005,root):1.0
24:vulExists(fileServer,'CVE-2003-0252',mountd,remoteExploit,privEscalation):0.7
9:RULE 2 (remote exploit of a server program):0.3115
8:execCode(fileServer,root):0.3115
7:canAccessFile(fileServer,root,write,'/export'):1.0
6:RULE 10 (execCode implies file access):0.2492
5:accessFile(fileServer,write,'/export'):0.6668
28:nfsMounted(workStation,'/usr/local/share',fileServer,'/export',read):1.0
4:RULE 16 (NFS semantics):0.5335
3:accessFile(workStation,write,'/usr/local/share'):0.5335
2:RULE 4 (Trojan horse installation):0.4268
1:execCode(workStation,root):0.4268

Attack graph for Model Network with low complexity vulnerability with CVSS calculation

APPENDIX G


17:hacl(internet,webServer,tcp,80):1.0
18:attackerLocated(internet):1.0
16:RULE 6 (direct network access):0.8
15:netAccess(webServer,tcp,80):0.8
20:vulExists(webServer,'CVE-2002-0392',httpd,remoteExploit,privEscalation):0.7
19:networkServiceInfo(webServer,httpd,tcp,80,apache):1.0
22:vulExists(webServer,_,httpd,remoteExploit,privEscalation):0.3
14:RULE 2 (remote exploit of a server program):0.448
21:RULE 2 (remote exploit of a server program):0.192
13:execCode(webServer,apache):0.554
12:hacl(webServer,fileServer,rpc,100005):1.0
26:hacl(webServer,fileServer,nfsProtocol,nfsPort):1.0
27:nfsExportInfo(fileServer,'/export',write,webServer):1.0
11:RULE 5 (multi-hop access):0.4432
25:RULE 17 (NFS shell):0.4432
10:netAccess(fileServer,rpc,100005):0.4432
23:networkServiceInfo(fileServer,mountd,rpc,100005,root):1.0
24:vulExists(fileServer,'CVE-2003-0252',mountd,remoteExploit,privEscalation):0.7
9:RULE 2 (remote exploit of a server program):0.2482
8:execCode(fileServer,root):0.2482
7:canAccessFile(fileServer,root,write,'/export'):1.0
6:RULE 10 (execCode implies file access):0.1985
5:accessFile(fileServer,write,'/export'):0.5537
28:nfsMounted(workStation,'/usr/local/share',fileServer,'/export',read):1.0
4:RULE 16 (NFS semantics):0.443
3:accessFile(workStation,write,'/usr/local/share'):0.443
2:RULE 4 (Trojan horse installation):0.3544
1:execCode(workStation,root):0.3544

Attack graph for Model Network with high complexity vulnerability with CVSS calculation

VITA

Beth M. Anderson

EDUCATION

Post-Baccalaureate Studies in Computer Science
Eastern Washington University, Cheney, WA
January 2016- June 2016

Bachelor of Arts in Psychology
University of Puget Sound, Tacoma, WA
Graduated May 2015

- Minors in Math and Computer Science
- Coolidge Otis Chapman Honors Scholar

PROFESSIONAL EXPERIENCE

Software Engineer
RiskLens, Spokane, WA
June 2017 – Present

Computer Literacy Graduate Instructor
Eastern Washington University, Cheney, WA
September 2016 – June 2017

Software Engineer Intern
RiskLens, Spokane, WA
June 2016 – Jun 2017

Risk Analyst Intern
Global Business Analysis, Gig Harbor, WA
June 2015 - December 2015