3-14-2014

A Recommender System in the Cyber Defense Domain

Katherine B. Lyons

A RECOMMENDER SYSTEM IN THE CYBER DEFENSE DOMAIN

THESIS

Katherine B. Lyons, Second Lieutenant, USAF

AFIT-ENG-14-M-49

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-14-M-49

A RECOMMENDER SYSTEM IN THE CYBER DEFENSE DOMAIN

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Science

Katherine B. Lyons, B.S.C.S.

Second Lieutenant, USAF

March 2014

AFIT-ENG-14-M-49

A RECOMMENDER SYSTEM IN THE CYBER DEFENSE DOMAIN

Katherine B. Lyons, B.S.C.S.
Second Lieutenant, USAF

Approved:

//signed//	25 Feb 2014
Kenneth Hopkinson, PhD (Chairman)	Date
//signed//	25 Feb 2014
Maj Kennard Laviers, PhD (Member)	Date
//signed//	25 Feb 2014
Timothy Lacey, PhD (Member)	Date

AFIT-ENG-14-M-49

Abstract

In the cyber domain, network defenders have traditionally been placed in a reactionary role. Before a defender can act they must wait for an attack to occur and identify the attack. This places the defender at a disadvantage in a cyber attack situation and it is certainly desirable that the defender out maneuver the attacker before the network has been compromised. The goal of this research is to determine the value of employing a recommender system as an attack predictor, and determine the best configuration of a recommender system for the cyber defense domain. The most important contribution of this research effort is the use of recommender systems to generate an ordered list of cyber defense actions.

Table of Contents

Page

List of Figures

List of Tables

List of Symbols

Symbol	Definition
IQR	Inner Quartile Range
I	set of all items
$r_{a,i}$	rating for item i by user a
$r_{b,i}$	rating for item i by user b
$\overline{r_a}$	average rating for user a
$\overline{r_b}$	average rating for user b
t	threat level
o	occurrence of event
n	number of ratings of events
x	rating for event

Subscripts

a	user a
b	user b
i	item i
A	event A
B	event B
C	event C
1	true value

List of Acronyms

Acronym	Definition
ADVISE	ADversary-driven VIew Security Evaluation
AFRL	Air Force Research Laboratory
AFIT	Air Force Institute of Technology
AI	Artificial Intelligence
ANOVA	ANalysis Of VAriance
COTS	Commercial-Off-The-Shelf
CPU	Central Processing Unit
CUT	Component Under Test
IDS	Intrusion Detection System
IP	Internet Protocol
INFERD	INformation Fusion Engine for Real-time Decision-making
nDCG	normalized Discounted Cumulative Gain
OODA	Observe Orient Decide Act
OS	Operating System
RAM	Random Access Memory
RMSE	Root Mean Square Error
SNT	Stratified Node Topology
SSARE	Security Situation Assessment and Response Evaluation
SUT	System Under Test
TANDI	Threat Assessment for Network Data and Information
TF-IDF	Term Frequency-Inverse Document Frequence
WPAFB	Wright Patterson Air Force Base
VLMM	Variable Length Markov Model

Acronym	Definition
XML	EXtensible Markup Language

A RECOMMENDER SYSTEM IN THE CYBER DEFENSE DOMAIN

I. Introduction

1.1 Overview

In the cyber domain, network defenders have traditionally been placed in a reactionary role. Before a defender can act they must wait for an attack to occur and identify the attack. Clearly this places the defender at a disadvantage in a cyber attack situation and it is desirable that the defender outmaneuver the attacker before the network has been compromised. The defender must have some insight into how the attacker will execute their attack to close off that attack vector. Attack predictors provide that information to the defender by analysis of current known attack methods, the state of the network, and the previous actions of the attacker.

1.2 Problem Statement

The Air Force relies on the availability, integrity, and confidentiality of the network in order to function. Cyber defense has become a major concern for the Air Force. The cyber defense domain has been accepted by the Air Force as another domain in which we fight. Therefore, it is reasonable to apply conventional warfare strategic methods to cyber defense. John Boyd's OODA loop, shown in Figure 1.1, aligns with current cyber defense practices. In order to develop better cyber defense systems new approaches are being explored applying concepts from other domains. In this research effort, the Intrusion Detection System (IDS) functions as the Observe and Orient aspect of the OODA loop. Using recommender system techniques that have previously been used for helping customers find products of interest, a new decision making technique can be used for a

cyber defense system. The human cyber defender performs the action step [2]. The IDS and recommender system augment the human by providing increased situational awareness and ranked recommendations for actions.

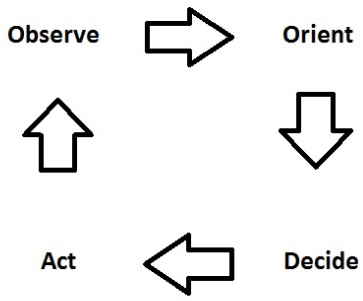

Figure 1.1: John Boyd's OODA loop

Cyber attacks occur in stages (1) Information Gathering, (2) Scanning and Vulnerability Assessment, (3) Intrusion, (4) Maintaining Access, and (5) Clearing Tracks. Understanding and anticipating the actions of the attack gives the cyber defender their opportunity to overcome the advantages of the attacker. The cyber defender must create a faster OODA loop process in order to out maneuver the attacker. The time between the start of the attack and the counter action taken by the cyber defender must occur on the order of seconds. Many cyber attacks often strike quickly, before the defender can counter the attack.

1.3 Goals and Approach

This research effort focuses on speeding up the defender's OODA loop, specifically the decision aspect. Already research has been done to create efficient IDS. In order to aid the cyber defender in making their decision a recommender system can be implemented to provide recommended actions. Recommender systems have commonly been used to suggest items of interest to consumers. For example, Netflix uses a recommender system

to suggest movies for users to watch. Netflix takes in information about what movies the user enjoys by asking them to rate movies on a 1 to 5 scale and what types of movies the user enjoys based on genre. The exact algorithm implemented by Netflix is proprietary, but it is easy to understand that if you give *Die Another Day* a high rating and enter that you enjoy action spy movies that the algorithm would suggest other James Bond films. The recommender system behind that decision uses large matrices that predict the rating a user would give to a movie based on the information they have gathered about that user.

The recommender system uses the information about the network from the IDS to predict the likelihood of certain events for particular nodes on the network. The recommender system then presents a ranked list of actions to the cyber defender with the highest ranked action mitigating the most events which have been predicted to occur. By giving a cyber defender a list of recommended defensive actions they are able to chose which action best suites their specific network. The recommender system is meant to enhance human cyber defenders.

1.4 Contribution

Many different attack predictors have been created in the past. Previous attack predictors present the cyber defender with a prediction of the attackers next action. The recommender system is an attack predictor, but taken to the next level by presenting the cyber defender with how to counter act the predicted action. Recommender systems have been used extensively in the domain of recommending items to customers but have not been applied to the cyber defense domain. The insight supplied by a recommender system holds great potential for acting as recommending defensive cyber actions. No one has taken a purely recommender system algorithm and implemented it to recommender cyber defense actions.

1.5 Summary

Cyber defense is a technical issue that concerns all organizations which rely on networks to function. Appropriately and quickly reacting to cyber attacks is the main option for today's cyber defender. The implementation of a recommender system as a cyber defense decision making tool is an area that merits being explored. Recommender systems have been studied for decades, but only in their original domain of retail customer suggestions. The same algorithms and techniques could have value for other domains.

II. Literature Review

2.1 Overview

The four main types of recommender systems include collaborative, content-based, knowledge-based, and hybrid. In this chapter the background of each type of recommender system will be explored. Recommender systems are often used in the world of customer buying preferences. The suggestions made by recommender systems are a form of Artificial Intelligence (AI) which makes predictions based on previous actions of the customer, knowledge of the problem domain, or survey of customer preferences. The majority of the literature approaches recommender systems within the problem domain of suggesting items to a customer.

2.2 Collaborative

The actions of existing users form the foundation of collaborative filtering for recommendations. The system observes the actions of a new user and compares them with the actions of existing users in order to find their nearest neighbor. Creating a suggestion uses the ratings or actions of a previous user as a predictor for future actions of the new user. Most collaborative filtering techniques employ a user-item matrix in order to generate preferences for a user. A user-item matrix is shown in Figure 2.1. Collaborative systems were first created in the 1990's and since then their capabilities have been thoroughly explored [10]. But the majority of these research efforts have been confined to the domain of customer purchases [15].

One of the most common methods to calculate a rating is the Pearson's correlation coefficient. It assigns pairs of users a value from -1 to +1, with +1 being a very strong positive correlation and a -1 is a very strong negative correlation. Users with high positive correlation values are very similar in the items that they bought. Using the basic assumption

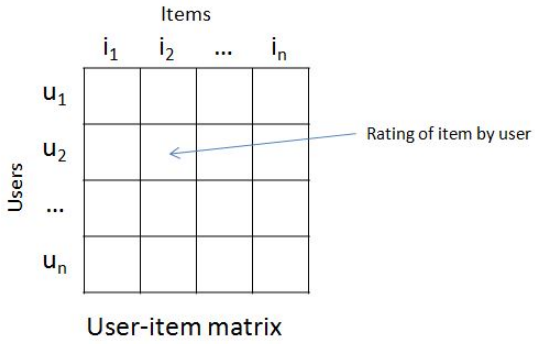

Figure 2.1: User-Item Matrix

for collaborative recommender systems, it follows that highly correlated users will be interested in similar items. The items bought by one similar user and not by the other should be recommended. The prediction value for the item is calculated based on how close that neighbor is to the user's average rating [15]. Pearson's correlation coefficient is one way to calculate similarity between users shown in Equation (2.1). The values are summed for each item i from the set of all items I. The symbol $r_{a,i}$ is the rating for item i by user a and the symbol $\overline{r_a}$ is the average rating for user a.

$$similarity(a,b) = \frac{\sum_{i \in I}(r_{a,i} - \overline{r_a})(r_{b,i} - \overline{r_i})}{\sqrt{\sum_{i \in I}(r_{a,i} - \overline{r_a})^2}\sqrt{\sum_{i \in I}(r_{b,i} - \overline{r_b})^2}} \tag{2.1}$$

The concept that users will buy similar items to the items they have showed an interest in and avoid items that they have no interest in drives item-based recommendation. Item-based systems take input as a user-item matrix to determine relationships between different items. Scalability is a challenge for user-based systems. When the user-item matrix becomes very large, the matrix computations become time-consuming [25]. Analyzing the interaction of items generates a recommendation for a user. Item-based algorithms outperform user-based algorithms, because the relationship between items is more stable than the relationship between users. With a more stable relationship the time consuming task of computation can be performed offline while still supplying accurate

predictions [25]. Using the similarity value calculated from the Pearson's correlation coefficient and the N nearest neighbor a predicted rating can be calculated, shown in Equation (2.2).

$$prediction(a, i) = \overline{r_a} + \frac{\sum_{b \in N} similarity(a, b) * (r_{b,i} - \overline{r_b})}{\sum_{b \in N} similarity(a, b)} \tag{2.2}$$

Real world data sets often lack large amounts of information which results in a problem known as sparsity. Without enough information it is difficult to make accurate recommendations for users. A similar problem, known as cold start, occurs when a new user has only rated a few items and there is not enough ratings in the matrix to make good predictions. Cold start also happens when there is a high item to user ratio, meaning that it is difficult for the user to rate a large enough number of items for the recommender to be able to make meaningful predictions [20]. Both problems stem from the difficulty in matching a user with few ratings to similar neighbors. A pure collaborative approach is to turn the matrix information into a graph and use a technique called spreading activation to find relationships between users and items. The recommendations are based on how close an item is from a user in the graph where distance is the number of edges from the item to the user [12]. Another method uses probability and similar user ratings to generate suggestions. By exploiting the information that does exist for the user the accuracy of predictions improve [29]. The most straight-forward solution is to use a hybrid system which gains insight into the recommendation using other sources of information such as item attributes or demographic information.

2.3 Content-based

Content-based recommendation uses information about items and past actions of users in order to make predictions. The thought process behind content-based recommenders is that a user will buy future items which are similar to items they have bought in the past. In order to determine the similarity of items certain attributes or features must be

associated with those items. Collecting the attributes about items, especially if the attributes are qualitative, can be a difficult task. When an item lacks information it falls into a similar issue of sparsity as found in collaborative recommendations. Content-based recommenders do not need the large number of users or ratings of items by a user that collaborative recommender systems demand, but must have item information.

Content-based recommendation not only uses the meta-data of features, but also the actual content of documents. A vector is constructed using a 1 to represent if a word is contained in a document and a 0 when the word is absent. Then the recommender system compares the vector with other documents. The process is simple, but there are many issues to this basic approach. The vector favors longer documents and does not take into consideration the significance and frequency of the words in the document [15]. To overcome these issues documents are analyzed using the Term Frequency-Inverse Document Frequence (TF-IDF) technique. Term frequency takes into account how often a word appears in a document. Inverse document frequency helps to highlight the difference between documents by giving higher weight to words which only appear in that document [24].

To determine if an item would be of interest to a user, the recommender system needs to have information of items that were previously of interest to the user and the similarity between potential items. The recommender system tracks items the user found useful and gathers a rating for items. The recommender system collects items into neighborhoods using a similarity calculation. Each item the user bought, associated with a particular neighborhood, counts as a vote for that neighborhood. If k of the nearest neighbors were rated highly by the user, then that item should be recommended. In order to create neighborhoods new items are compared to current items in the inventory taking into account attributes and similar terms. Long-term profiles of users can be kept and analyzed in order to give the best recommendation [1]. Short-term profiles can be used when a user

has frequent changes of interest. The k-nearest-neighbor technique is straightforward, adaptable, and requires little data in order to give a good recommendation [15].

Content-based recommender systems perform well with many items and users, but there are some issues when implementing a system. The information that can be associated with items limits the abilities of content-based recommender systems. With text based items, such as documents, news articles, and websites, recommender systems simply analyze the words, but this neglects other attributes of items which are more subjective. Features are difficult to extract when items are not text based. Manual entry of attributes requires too high of a cost to be a viable solution. With few attributes for items, content-based recommender systems suffer from their own form of sparsity. Content-based recommenders focus on suggesting similar items to previous items of interest, but they can begin to make suggestions which are too similar. A user is not interested in buying an item if they already bought the item from a different company. To provide interesting by not too similar items some diversity can be introduced to the system or a filter can be used to eliminate items which are too similar [31]. Lastly, the cold start problem does effect content-based recommender systems, but not as severely as collaborative recommender systems. The content-based recommender requires only a few ratings or past action information in order to make a prediction, unlike collaborative systems which require mostly complete user-item matrix [15].

2.4 Knowledge-based

Knowledge-based recommendations do not require user-item data to make recommendations for users. Instead knowledge-based recommenders use explicit rules about the problem domain and attributes of items to generate predictions for users. Knowledge-based recommender systems interact more intimately with users. Unlike collaborative and content-based recommendations which track user actions and collect ratings, knowledge-based recommendations collect specific requirements from the user in order to bring them

closer to items of interest. Therefore, with items that are not bought often, knowledge-based recommenders can direct a user to an item without dealing with sparsity issues [3].

Constraint-based recommenders perform recommendations as a solution to a constraint satisfaction problem. By using the requirements given by a user as the constraints, the attributes of products are compared in order to find items which are within the given parameters [9]. A simple constraint solver is capable of finding items which satisfy the requirements supplied by the user. Another way to perform constraint-based recommendations is to execute a conjunctive query over the database of items. The system constructs a conjunctive query by connecting together the requirements for attributes given by the user, then performs a database query which returns items which meet the constraints. Both conjunctive query and constraint satisfaction solvers view the requirements from the user as constraints, categorizing them as constraint-based recommenders.

Case-based recommendations are made from the similarities between items and the requirements. McSherry defines a distance similarity for an item depending on the sum of all the similarities of attributes weighted by the requirements. Other items are of interest purely dependent on how far an attribute is from the given requirement. To find similar items local similarity can be calculated based on the distance of the attribute of the item from the desired attribute divided by the total range of the attribute [19]. Usually, the similarity calculations for case-based recommendations are used as an aspect of utility-based recommender systems. Other case-based recommenders rely on a more query-based paradigm. A purely query-based system can be very difficult if the user does not have specific requirements. A different form of query-based recommendation guides users to explore the space and give incremental critiquing in order to key into items of interest. The user gives adjustments to the requirements throughout the search in response to the items recommended to the user [3]. Both types of case-based recommendations

depend heavily on finding items which do not fulfill all the requirements of a user, but instead offer items which are close to the desired traits.

Utility-based recommender systems find the overall utility of an item for a user after gathering the weight or interest level the user has in a particular attribute. The weight can also be determined by the system which significantly decreases the load on the user. The total utility is calculated as the sum of all the item values, which is the weight multiplied by the similarity function used in case-based recommendations. The result is a ranking of items based on how similar the items are to the requirements set by the user [19].

Often knowledge-based recommenders are unable to satisfy all the requirements given by a user. Instead of providing a null response the recommender system should guide the user or automatically relax the constraints. By incrementally relaxing the constraints the recommender system can find an item which is close to meeting the original requirements. Another method is to identify conflicts between the requirements and potential items. Using a divide-and-conquer algorithm QuickXPlain finds conflicts for given constraints [16]. After determining constraints that cannot be satisfied the recommender system must suggest ways to repair requirements so that they can be met by the available items.

2.5 Hybrid

Hybrid recommender systems attempt to exploit the strengths of each of the three main types of recommender systems. By pooling information from different approaches a better representation of the problem can be created. The algorithms implemented and the method of hybridization factor into the results. The algorithm's solutions can be combined at the end of separate calculations or the results of one algorithm can feed into the input of a second algorithm [15].

Both collaborative and content-based recommenders share the problem of sparsity, which makes them more suited for problems with a high density of information. Knowledge-based recommenders do not suffer from sparsity, because the focus is

more on the problem domain than the people or items in the domain. Unfortunately, knowledge-based recommenders are not capable of developing associations between users and items, but instead function based on their understanding of the problem domain [4]. Because of the dependence on users and items, collaborative and content-based recommenders can react and learn as the needs of users change. Collaborative techniques function well in domains where content-based techniques suffer, such as situations with few attributes for items or where attributes of items are difficult for a machine to analyze [20]. Content-based recommendations require very few user-item ratings to make accurate recommendations.

Weighted hybrid recommender systems perform multiple techniques on the same set of data. Then post computation combines the information from the various techniques in order to present the overall recommendations. Static weights must be able to give valid results for all possible combinations. Some weighted hybrids adjust weights depending on the shifting situation. Specifically, the P-Tango system employs a hybrid recommender which adjusts the weights for each technique in order to find the most accurate recommendation [6]. As users give feedback in the form of rating, the weights are altered to reflect the best recommendations for users. A weighted hybrid recommender attempts to balance the strengths and weakness of many different techniques by simply giving a weighted value to the output.

A switching hybrid determines the best technique to use in a given situation and uses that to compute a recommendation. The recommender determines which technique to execute based on the user-item matrix and the acceptability of the technique's results [15]. Sparsity could be addressed by using a knowledge-based recommender with new items, then once enough users have bought or rated the item collaborative recommendation would be chosen. Some switching hybrids function based on a default and only employ other

techniques when the default fails to give a valid result [27]. The decision of when to switch can also be based on a probability which is determined by the quality of the prediction [4].

Feature combination combines collaborative and content-based recommendations in another way to address the issue of sparsity. The collaborative results are used merely as an additional feature for an item. The hybrid recommender relies mostly on content-based, but does not lose the information that collaborative recommendations capture. Content-based recommenders often cannot identify the connections between items which are not explicit attributes, but that a collaborative recommendation would uncover because of the actions of users [4]. Implementation of knowledge-based recommenders with another technique using feature combination has not been widely explored [15], but may lead to a better solution to the problem of sparsity.

Instead of running different recommendation techniques in parallel then combining the output, cascade uses the output of one recommendation as the input for a different technique. One combination uses a knowledge-based recommendation to sort the items into different categories, then the second technique is applied to the items within each category to give a more complete recommendation. Certain categories could be ignored for the second search, if they are infeasible. For large data sets it could be very useful to remove infeasible items from the pool, but in contrast it may not return enough items to be useful. The second technique for a cascade recommender could simply be used to break ties for the results of the first technique [4]. Cascaded hybrid recommenders prioritize items and then aggregate the items into an overall recommendation.

Feature augmentation allows a different recommendation technique to refine the results of the main technique. The first technique alters the attributes for the item input before the second technique is applied. For example, in the content-boosted collaborative filtering created by Melville et al. sparsity and first-rater problem are overcome using both content and collaborative techniques [20]. They use a content recommender to create

pseudo user-ratings for items the user has not rated, then simply apply a collaborative recommender using the Pearson correlation. The content recommender was used to enhance the ability of the collaborative results.

By combining different methods of recommendation weaknesses in certain techniques can be mitigated. Because both collaborative and content-based recommenders suffer from the sparsity problem it may be redundant to use a hybrid of those two techniques. Generally, using a knowledge-based recommender is good for countering the sparsity problem. The order in which certain hybrid methods are implemented impact the outcome. For example, feature combination using a collaborative then a content-based would be very different than using a content-based then a collaborative recommender. Hybrid recommender systems exploit the strengths of other recommendation methods by finding a balance between different techniques.

2.6 Attack Predictors

The recommender system is used to anticipate the actions of an attacker and suggest a course of action to mitigate the degradation of the network capabilities. Understanding current methods to predict attacker actions plays a key role.

2.6.1 Attack Graphs.

Attack graphs create a Bayesian representation of attacker actions which can be performed to compromise the network. Defenders use attack graphs to increase their situational awareness of vulnerabilities in the network. By anticipating attacker course of action a defender can counter the attacker before the attack exploits the network.

The ADversary-driven VIew Security Evaluation (ADVISE) method developed by Lemay et al. analyzes attack graphs and returns results applicable to the specific attacker. In order to answer a decision question about the network security a discrete event simulation is run on a modeled system. The attacker is characterized from the utility of their attack, the skill level the attacker is capable of executing, goal of the attacker, system knowledge,

and system access. The utility of potential attacks are calculated based on the cost for the adversary, reward for the adversary, probability of a successful attack, and probability of the adversary of being detected during the attack. Attacks are represented in different states in a graph where each step of an attack is a different state. The system performs the analysis based on a simulation which is only a representation of the actual system. In a simulation only well understood attacks can be analyzed. The system does take many attacker characteristics into consideration. Overall, ADVISE performs a comprehensive analysis, but does not have the capability for analyzing complex systems [17].

The causal network developed by Qin and Lee takes attack graphs to the next level. The approach has a more strategic level view which attempts to determine the goals and intensions of the attacker. The system first correlates all the alerts clustering them into attacks. Then the alerts are sorted based on priority. The priority is calculated taking into account the defender's mission objectives, and the severity of the attack. Using Bayesian techniques the probability of different attacks are determined with consideration for the configuration of the network. Additionally, statistical analysis is applied to find relationships between alerts. Applying the alert information to the creation of the attack graph forms an attack tree with branches including possible intrusions. The attack tree is used to create a causal network with each node with a given probability. An example of an attack graph in Figure 2.2 generated from an attack tree in Figure 2.3. Each node has a binary value which updates as more alerts occur in the network. The Bayesian network supplies the attack prediction for the defense network. Using the data set from DARPA's Grand Challenge Problem, the system was able to predict the correct attacker action and goal. The major goal of this effort was to develop a systematic way to generate attack plan graphs based on alerts from the network. Cyber defense has begun to shift in the direction of understanding the strategic objectives of the attacker. The system created by Qin and Lee

accomplishes that task simply by analyzing intrusion alerts. By learning the objective of the attacker, the cyber defender gains a clear advantage when choosing defensive actions [22].

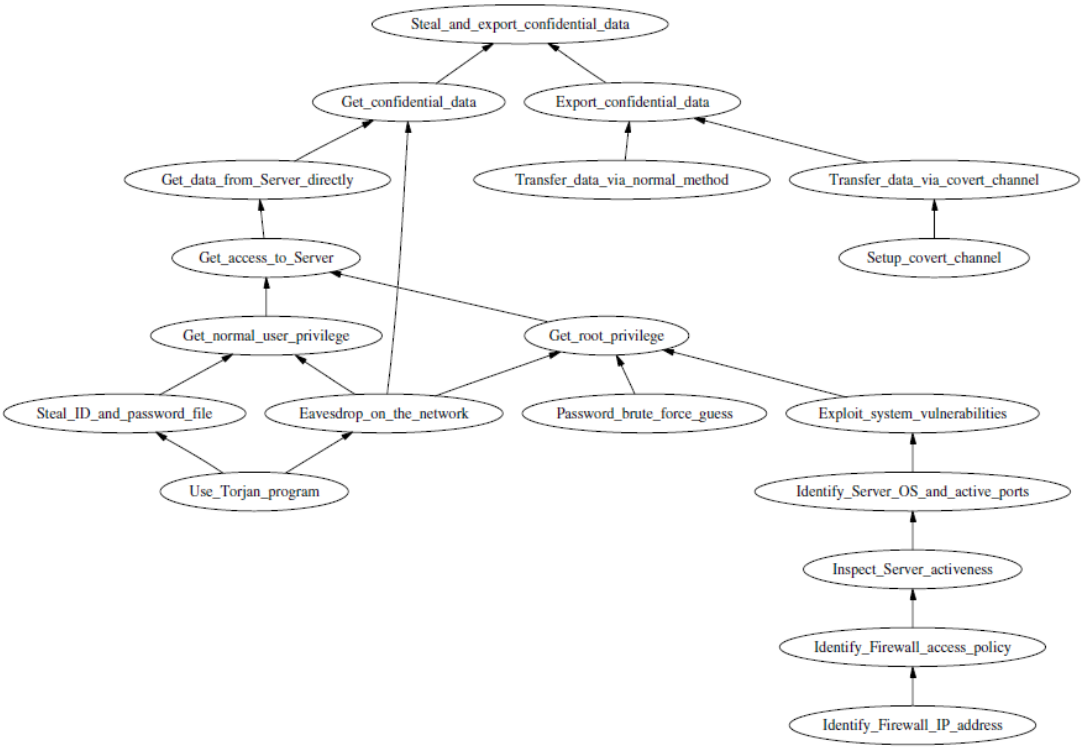

Figure 2.2: Attack Graph [22]

For attack graphs to be practical they must scale well. The NetSPA system uses a new type of attack graph called the multiple-prerequisite graph which enables it to linearly increase to the size of any network. The system has significant capabilities such as creating an attack graph to represent the attacker's ability to maximally intrude into the network, model attackers from different locations in the network, and quickly builds the multiple-prerequisite graph. The multiple-prerequisite graphs forms a graphical representation of an attacker's possible sequence of actions to compromise the network. An example of a multiple-prerequisite graph is shown in Figure 2.4. Information about

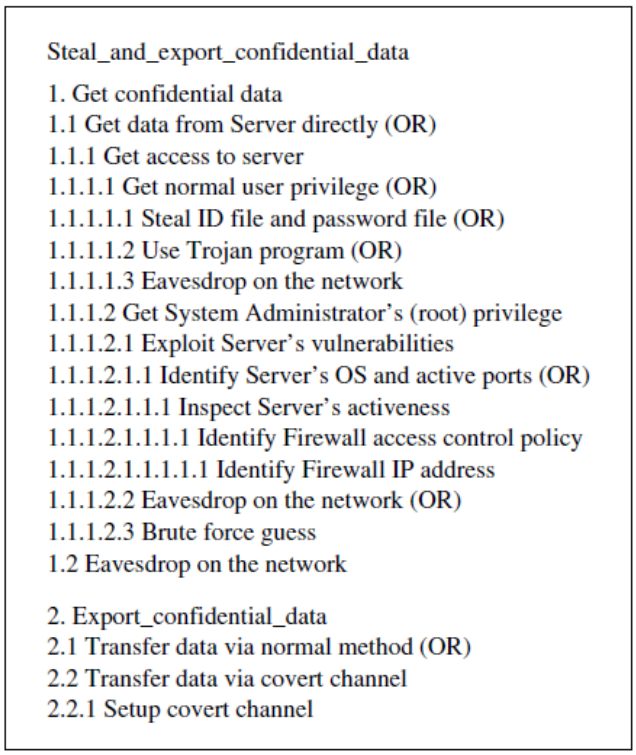

Figure 2.3: Attack Tree [22]

the vulnerability of the network is gathered through Nessus scans, system configuration data, and reachability calculations. The vulnerability information feeds into the NetSPA algorithm which using a breadth-first search forms the multiple-prerequisite graph. In order to deal with scalability the system consolidates similar states and groups of nodes behind bottlenecks. Recommendations are computed based on the key vulnerabilities which allow the attacker to progress through the attack. Testing of the NetSPA system was done with a simulation of 50,000 nodes and a field test of 250 nodes. For the field test, the attack graph was generated in a matter of 0.5 seconds, but was too large for a person to understand quickly. The system's recommendations were more useful with such a large graph. NetSPA is one of the few systems which addresses the issue of scaling and presents

recommendations for the defender instead of just predicting the next attack; which makes it more practical for use in a real network [13].

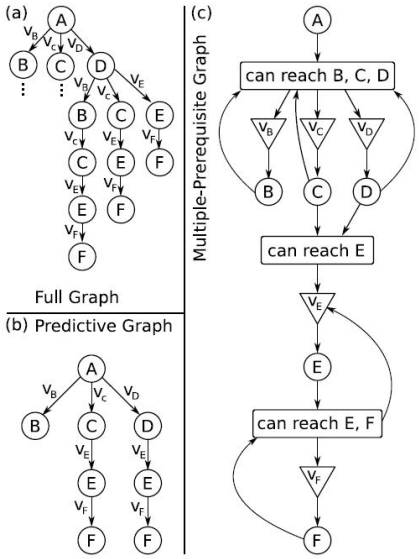

Figure 2.4: Example of (a) Full Graph, (b) Predictive Graph, and (c) Multiple-Prerequisite Graph [13]

2.6.2 Attack Trees.

Attack trees present a goal oriented attack with a multi-level graph. Attack trees represent similar information to attack graphs in either an outlined textual format or a graphical format with nodes, edges, and dependencies. Attack trees have additional conditions of AND and OR for parent-child node relationships. Because of the distribution and dependency, attack graphs which are analyzed using Bayesian techniques are not possible. Therefore, other methods must be used for analysis of attack trees.

The framework that Daley et al. developed uses modified attack trees. The nodes in the attack tree follow a certain classification called Stratified Node Topology (SNT). The SNT labels nodes based on functionality, including application exploits, abstract attacks,

and attack goals. The graph has both implicit and explicit edges which represent an indirect and direct connection to other exploits respectively. An example of SNT is show in Figure 2.5. The attack trees are constructed in reference to specific host vulnerabilities on the network. The SNT forms the framework for analysis of attacks. No specific technique for determining likely attacks has been developed for the framework. SNT does present the potential cyber attacks in a step by step format which enables the defender to observe possible attack vectors [7].

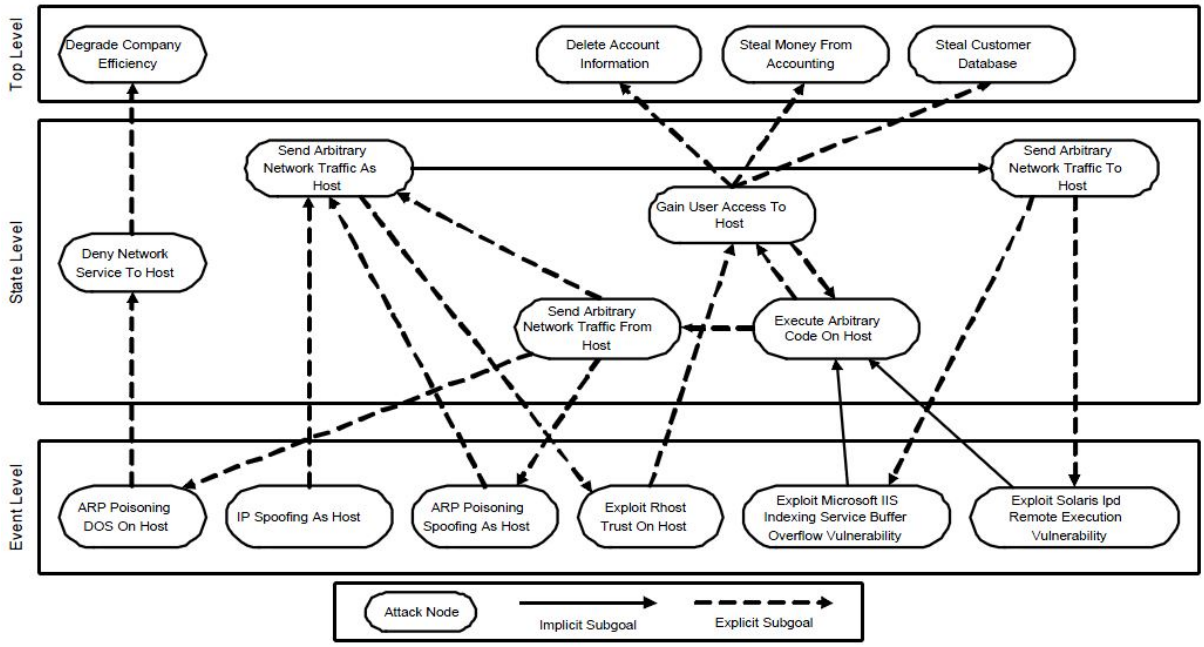

Figure 2.5: Example of Stratified Node Topology [7]

The network is represented by a logical model which includes the system configuration, vulnerabilities, and potential attackers. The nodes on the cyber terrain graph represent hosts and include specific services and data metrics. The value of the node is represented by a utility given to each service and data metric assigned 0 to 1, with 1 representing a high valued service or data. A service tree is generated which maps vulnerabilities, alerts,

and privilege levels to specific services on the network. The edges of the terrain graph include an access and band list. The behavior prediction process begins with filtering the IDS alerts. Next a probability model is built based on the observed alerts using the finite context model previously constructed from network analysis. Different orders of probabilities, which are different levels of depth, are created with the blending of these orders called Variable Length Markov Model (VLMM). Testing of the VLMM system was performed on a virtual network with scripted attacks. The VLMM significantly outperformed the individual orders of probability with a 90% accuracy rate. The cyber terrain and VLMM system take two very different approaches to predicting cyber attackers but provide a good representation of an attacker's possible behavior [8].

Yang et al. have designed a cyber fusion engine called INformation Fusion Engine for Real-time Decision-making (INFERD) and Threat Assessment for Network Data and Information (TANDI) to predict stages of a multi-state cyber attack. INFERD performs the IDS alert correlation while TANDI determines threats to the network based on the output from INFERD and network configuration. As alerts are generated on the network INFERD associates each alert with an existing attack, then updates the severity of each on-going attack. Influencing the decisions of both INFERD and TANDI is the Guidance Template which is viewed as the expert for attacker modeling. TANDI determines threat levels taking into account hacker's goals, vulnerabilities, value of the target, and skill of the attacker. The threat assessment algorithm weighs the information and determines likely targets for the next attack. The system is limited by not being able to observe insider threats and being unable to correlate coordinated attacks from different sources. INFERD was tested on a virtual network by Air Force Research Laboratory (AFRL), while a set of cyber attacks were designed by the researchers to test TANDI. Both demonstrated good performance, but with limitations for understanding multi-stage cyber attacks. Both systems are limited in their knowledge of possible attacker actions by the Guidance Template, which must be

created by someone with domain knowledge. Because of the dependence on the Guidance Template the system has limited adaptability [30].

2.6.3 Machine Learning.

Machine learning encompasses a wide range of techniques which focus on presenting the system with information and enabling the system to incorporate this information to make future decisions. The tools included in this section did not fit into the previous categories; therefore they are included in the wider discipline of machine learning.

Exploiting the knowledge about the previous actions of an attacker is a common approach. The tool called *Nexat* uses machine learning techniques to predict the actions of attackers on a cyber network. *Nexat* executes in phases starting with data extraction phases where attack sessions from the IDS are placed in hash tables. Then the training phase is performed which creates sets of targets with associated probability. The training phase is followed by the prediction phase where *Nexat* uses the probabilities and weighted sums to determine the most likely next attack. The *Nexat* tool was evaluated using a dataset from a cyber competition and it performed with over a 94% average accuracy. *Nexat* requires a workload to train and is limited in its knowledge based on the training. With a good training dataset the system performs well and is able to scale to any size network [5].

Another technique was developed by Ning et al. to compute the strategy of an attack based on the IDS alerts triggered by the attack. The technique is based upon an attack strategy graph with nodes representing attacks, and edges representing order of attacks. Using a subgraph isomorphic method the similarity to other attacks based on the IDS alerts can be calculated. A correlation model is built from the IDS alerts, which shows the dependencies of different attack vectors. The system is able to identify the attack strategy when the IDS alerts are generalized and computes the similarity between different attacks. The information obtained for the cyber defender gives them a strategic view on the cyber attack, but little practical information. Understanding the goals of the cyber

attacker is important, but understanding exactly what vulnerabilities will be exploited is more valuable [21].

2.7 Summary

Recommender systems process large martices of information in order to make predictions. A wide range of methods can be used to obtain a utility of an item for a user. Collaborative recommenders focus on grouping similar users, while content-based recommenders create neighborhoods from the attributes of items. Knowledge-based recommenders extract information from users then through relaxing requirements arrive at a recommendation. Hybrid recommenders bring the other three techniques together in order to create a well-rounded recommender. Each recommender must be carefully designed to fit their specific problem domain.

The current methods for attack predictors: attack graphs, attack trees, and machine learning, are well developed. The major problem with using current methods of network analysis centers on scalability. A large network with many nodes makes the nodal model very complex. The information presented to the network defender is overwhelming. Many of the systems attempt to present the information in such a way as to aid the defender. But these systems fall short compared to the capability of a recommender system, which focuses on making meaningful suggestions to a user. The majority of the work in predicting attacks has been with the goal of improving predictions. Knowing the next action of the cyber attacker is useful but only if the defender knows how best to counteract the attacker. A recommender system would be a more complete tool for the defender and holds the possibility to act autonomously.

III. Recommender System Design

3.1 Overview

In this chapter the design of the recommender system and IDS are explained. The IDS used in this experiment was designed by Captain Evan Raulerson. In the context of the Observe Orient Decide Act (OODA) loop the IDS functions as the Observe and Orient while the recommender system functions as the Decide. Finally, the Action part of the OODA loop is performed by the cyber defender and has the potential to be automated.

3.2 Network Model

The Network Modeler functions as an IDS from the perspective of the recommender system, but it gives significantly more information about the network than the average Commercial-Off-The-Shelf (COTS) software. For the purpose of simplicity when referring to the Network Modeler of the experimental system it will be called an IDS. The IDS has three major sections: sensors, database, and modeler algorithm. The sensors are located on each node of the network and send updates to the database. The modeler pulls information from the database to classify the information. The database and COTS software run on the sensor machine

3.2.1 Sensors.

There are a variety of types of sensors used to collect information about the network status. To understand the overall network health the sensors view the network from different aspects. Almost all of the sensors are COTS with the exception of the host monitoring which was custom written. The creator of the IDS intended to use only COTS, because of the availability and known success, but a compatible host monitoring system could not be found for the scope of the IDS. Snort is used to observe the network traffic and alert to known signatures of attacks. The rule set for snort is customized for the attacks that will be

executing on the network to ensure that an alert is generated. The Snort rules in Figure 3.1 was designed to alert to the netbios smb buffer overflow exploit. Another tool observing

```
alert tcp any any -> \$HOME_NET 445 (msg:"'ET NETBIOS Microsoft
Windows NETAPI Stack Overflow Inbound - MS08-067 (15)"';
flow:established,to_server; content:"'|1F 00|"'; content:"'|C8 4F
32 4B7016 D3 01 12 78 5A 47 BF 6E E1 88|"'; content "‘|00 2E 00 2E
5C 00 2E 00 2E 00 5C|"'; reference: url, www.microsoft.com/technet/
security/Bulletin/MS08-067.mspx; reference:cve, 2008-4250;
reference: url, ww.kb.cert.org/vuls/id/827267; reference:url,
doc.emerginingthreats.net/bin/view/Main/2008705; classtype:
attempted-admin; sid:2008705; rev:5;)
```

Figure 3.1: Custom Snort rule [23]

the network is Nmap. Nmap scans the network to find hosts and determines the services running on that host. The information gathered from Nmap is stored in the database on the sensor machine using Perl scripts. The custom host monitoring software is a java based program sending updates to the database on the sensor machine. The host monitoring program gathers information about memory usage, CPU usage, bandwidth, and service information. Additionally, the host monitoring program controls the anti-virus software for each host. The anti-virus software used is AVG Anti-Virus 2013, because of the ease to send commands and receive information through the command line [23].

3.2.2 Database.

A MySQL database located on the sensor machine acts as the storage for the sensors on the network. Snort, Nmap, vulnerability scores, and host monitoring software are the four databases used to store the information. The database acts as the collection of

information about the network as shown in Figure 3.2. The Snort database is created during the Snort installation. The event table includes the key information for alerts with the signature identifier, and timestamp. The other tables in the Snort database store signature identifiers and specific information about events with foreign keys to the event table. The Nmap database includes two tables: machines and services. The machines table stores information about specific machines on the network such as Operating System (OS), and Internet Protocol (IP) address. The services table stores data about services running on each machine. The host monitoring database is where the host monitoring program running on each machines sends the updates of the node's status and anti-virus software results. The vulnerability scores database stores the requirements for the host to meet in order to be vulnerable to attacks found in the Snort alerts [23].

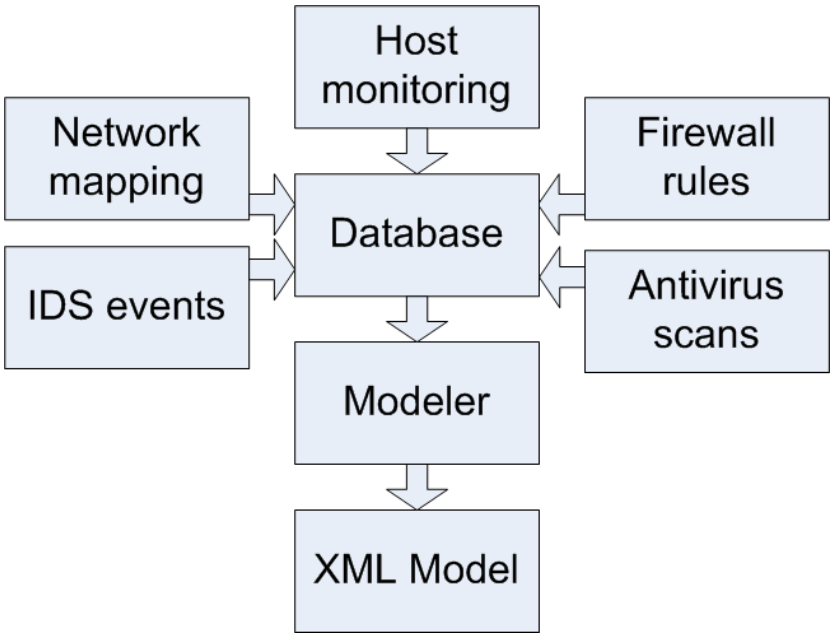

Figure 3.2: Flow of the IDS Information [23]

3.2.3 Modeler Algorithm.

The network modeler collects and classifies the information about the network in order to build a better understanding of network health. The classification and assessment aspect determines if there is a threat to the network based on the information gathered from all of the sensors on the network. The flow of the modeler is summarized in Figure 3.3. The designer of the IDS created his own scale and requirements to determine the level of vulnerability for the network and labeled events based off of the ranking system. For the purposes of the recommender system testing the vulnerability threat level assigned by the IDS will not be considered. Another valuable contribution to the network picture

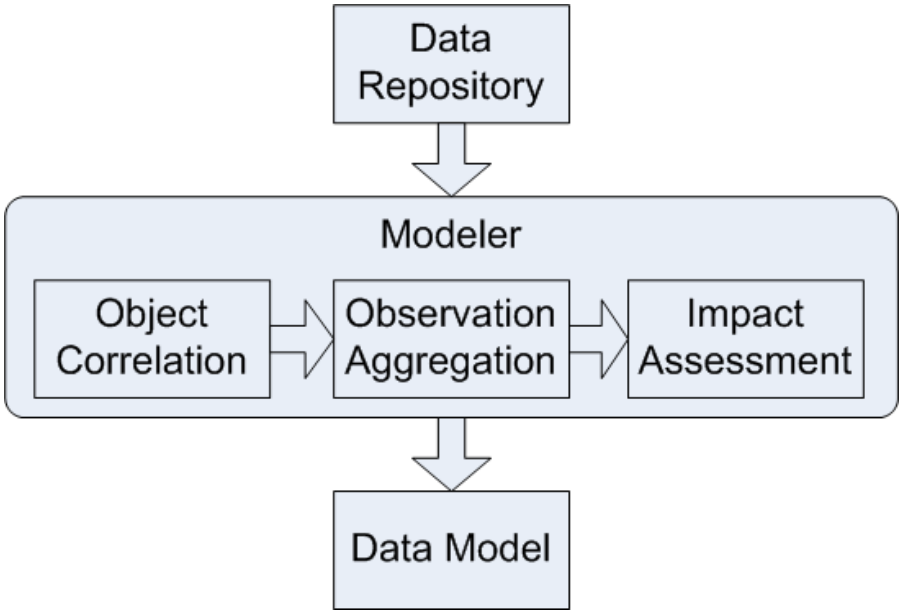

Figure 3.3: Data Fusion of the Modeler [23]

is the generation of possible actions that can be taken on each host using administrative privileges. The modeler does not include an exhaustive list of all possible actions, but functions as a demonstration of the capability of the modeler to perform such an action.

With such a functionally built into the system it is possible to expand to a more complete list of possible actions to be taken on the network. Actions are assigned to hosts based on known actions in the database and the host information gathered. The list of actions are shown in Table 3.1 and Table 3.2. Actions are included on this list with no regard to what is occurring in the network. Instead the modeler provides all actions that can be taken in order to provide the cyber defender the option to react as they deem appropriate [23].

Table 3.1: List of basic actions that can be taken on client machines [23]

Action
Update operating system
Update application
Create user account
Delete user account
Reset user password
Disable port/service
Enable port/service

Table 3.2: List of basic actions that can be taken on the firewall machine [23]

Action
Block source IP address
Allow source IP address
Block destination IP address
Allow destination IP address
Block ICMP traffic
Allow ICMP traffic

3.2.4 Output.

The output from the modeling algorithm is an XML file which summarizes the network health. Each host on the network has information about services running on the host, user accounts, infections, and actions that can be taken for that host. The XML file includes the information for all the hosts on the network. The tags used in the XML file are show in Figure 3.4. The host will only have tags for events associated with that individual host. For example only a host which is acting as a firewall will have firewall rules and only a host with an infection detected from the anti-virus will have the infection tag [23].

```
<network>
<host>
[attributes]
<service>[attributes]</service>
<user>[attributes]</user>
<event>[attributes]</event>
<av_scan>
[attributes]
<infection>[attributes]</infection>
</av_scan>
<action>[attributes]</action>
<firewall_rule>[attribute]</firewall_rule>
</host>
</network>
```

Figure 3.4: Format example of XML file output from network modeler [23]

3.3 Collaborative Recommender System

The collaborative recommender system functions as the attack predictor. The algorithm is the user-based nearest neighbor recommendation. First the similarity between the nodes is calculated with the nodes replacing user and the attributes for each node replacing the products. The Pearson's correlation coefficient assigns similarity values from -1, strong negative correlation, to $+1$, strong positive correlation. For each pair of nodes the similarity is calculated using Equation (3.1) [15].

$$similarity(a,b) = \frac{\sum_{i \in I}(r_{a,i} - \overline{r_a})(r_{b,i} - \overline{r_i})}{\sqrt{\sum_{i \in I}(r_{a,i} - \overline{r_a})^2}\sqrt{\sum_{i \in I}(r_{b,i} - \overline{r_b})^2}} \tag{3.1}$$

For example, the output from the modeler shown in Table 3.3 results in the similarity shown in Table 3.4. The prediction is calculated using the similarity values. Based on pilot studies, five nearest neighbors were selected for the algorithms. The predicted value is found using Equation (3.2) [15].

Table 3.3: Example Model Output

Machine IP	OS	Event	Service	User A Account
192.168.1.1	2	0	1	0
192.168.1.2	1	1	0	1
192.168.1.3	1	?	0	1

Table 3.4: Example Similarity Results for Nodes on the Network

Machine IP	192.168.1.1	192.168.1.2	192.168.1.3
192.168.1.1	1	-0.17408	0.301511
192.168.1.2	-0.17408	1	0.57735
192.168.1.3	0.301511	0.57735	1

$$prediction(a,i) = \overline{r_a} + \frac{\sum_{b \in N} similarity(a,b) * (r_{b,i} - \overline{r_b})}{\sum_{b \in N} similarity(a,b)} \tag{3.2}$$

Continuing with the earlier example the predicted values are shown in Table 3.5. The prediction for the event occurring for the similar node is higher. Related to the cyber defense domain the rational is that similar nodes will have similar vulnerabilities and therefore similar attacks expected against them. The predicted rating for nodes are the values used to determine vulnerabilities that the knowledge based recommender system needs to consider when generating defensive actions.

Table 3.5: Example Predicted Value for Nodes

Machine IP	OS	Event	Service	User A Account
192.168.1.1	2	0	1	0
192.168.1.2	1	1	0	1
192.168.1.3	1	0.40693	0	1

3.4 Knowledge-based Recommender System

The knowledge based recommender system is the core element to making defensive action recommendations. The knowledge-based recommender system paradigm chosen is commonly referred to as a constraint-based recommender. A constraint-based recommender system in the manner that it is implemented for the cyber defense recommender system is a subset of knowledge-based recommender system. The recommender system depends on pre-set knowledge about which actions mitigate or counter which cyber attacks. It is reasonable to assume that there is a good understanding of what actions counter certain attacks.

When the recommender system is first started it loads from an XML file all the actions that it knows. The actions supplied by the IDS are the only ones considered for the scope

of the experiment. The IDS and recommender system could both be expanded to include a significantly larger set of actions, but that is not the focus of this work. The XML file only needs to be loaded at the initial start up which leads to an expectation that the first computation time for the recommender system should be significantly larger than the other computation times. An example of the XML file format is shown in Figure 3.5. Each

```
<attack>
 <defact>
 <name></name>
 <exploit></exploit>
 </defact>
</attack>
```

Figure 3.5: Format example of XML file used by knowledge-base recommender system

action may counter multiple exploits. Therefore, the action suggested should mitigate as many of the predicted attacks as possible. The recommender system presents the top 10 actions in order of the most number of possible attacks mitigated. Currently no consideration is given for functionally of the network, but instead leaves the decision up to the cyber defender. If a vital node is at risk the cyber defender may chose a more dramatic reaction, than would be chosen if a non-vital node was under attack. The intention of suggesting 10 defensive actions is to give the cyber defender options while presenting what the recommender system has determined to be the best defensive action up front.

The knowledge-based recommender system considers all the possible actions that can be taken to counter the current threats. Equation (3.3) is formed where events A, B, and C are all events associated with the specific action.The t stands for threat level and o stands

for the occurrence of event. Once all the action values are calculated they are sorted in descending order and presented to the cyber defender as recommendations.

$$value = (t_A) * (o_A) + (t_B) * (o_B) + (t_C) * (o_C) \tag{3.3}$$

3.5 Summary

In this chapter the design of the system used in the experiment was described in detail. First, the inner workings of the IDS and network model were broken into parts. Second, the collaborative recommender system algorithm was shown using an example network model. Finally, the defensive action generation was described using the knowledge-based recommender system algorithm.

IV. Methodology

4.1 Problem Definition

4.1.1 Goals and Hypothesis.

The application of recommender systems has been constrained to recommending items of interest to users. The goal of this research is to determine the value of employing a recommender system as an attack predictor, and determine the a configuration of a recommender system for the cyber defense domain. Every implementation of a recommender system must be designed to function efficiently in a given domain. The research effort identifies the performance of five recommender system algorithms to predict cyber attacks. It is expected that the recommender system will be a more accurate predictor for cyber attacker actions than previously developed attack predictors. A hybrid recommender system considers multiple aspects of the system to make a prediction, while current attack predictors attempt to make decisions based on one dimension of the problem domain.

4.1.2 Approach.

To determine the superior recommendation algorithm multiple types must be observed. Hybrid recommender systems are comprised of collaborative, content-based, and knowledge-based with different combinatorial techniques. Because of the nature of cyber attacks, sparsity in collaborative and content-based recommender systems means that a knowledge-based recommender must be used in the hybrid recommender. The knowledge-based recommender is combined with a collaborative technique. Different configurations of the recommender systems are used as an attack predictor for the same set of cyber attacks. The accuracy of the different recommender systems will be compared. The accuracy results of the experiment are analytically contrasted with current attack

predictor systems to determine if the recommender system is an effective cyber defense tool.

4.2 System Boundaries

The testing environment includes a virtual network, a host server machine, an IDS, and the recommender system. A system overview is shown in Figure 4.1. The host server machine is the physical machine on which the virtual network exists. The virtual network contains multiple virtual machines running both Windows and Linux operating systems. Within the virtual network is a black hat machine which acts as the source of the cyber attacks. The IDS is configured to alert based on the cyber attacks used for testing [23]. The information about the network health feeds into the recommender system from the IDS. Therefore, the IDS must be able to identify the attack to give the recommender system correct information about the network. The recommender system includes a hybrid configuration with a knowledge-based recommender system in addition to collaborative techniques. The hybrid technique used is limited to cascade hybridization. Instead of running different recommendation techniques in parallel then combining the output, cascade uses the output of one recommendation as the input for a different technique.

4.3 System Services

The system provides the service of recommending actions for a defender to counter the current cyber attack. The virtual machines act as users and sensors for the network. The IDS takes in information from the virtual network and correlates alerts to identify and classify attacks on the network. The recommender system, using domain knowledge and information from the IDS, predicts the attacker's next action and gives the defender an action to counter the attacker. Possible outcomes of the network service include: (1) the cyber attack was not successful, (2) a loss of network traffic, and (3) the cyber attack was successful. The IDS outcomes include: (1) correctly identifying an attack, (2) incorrectly

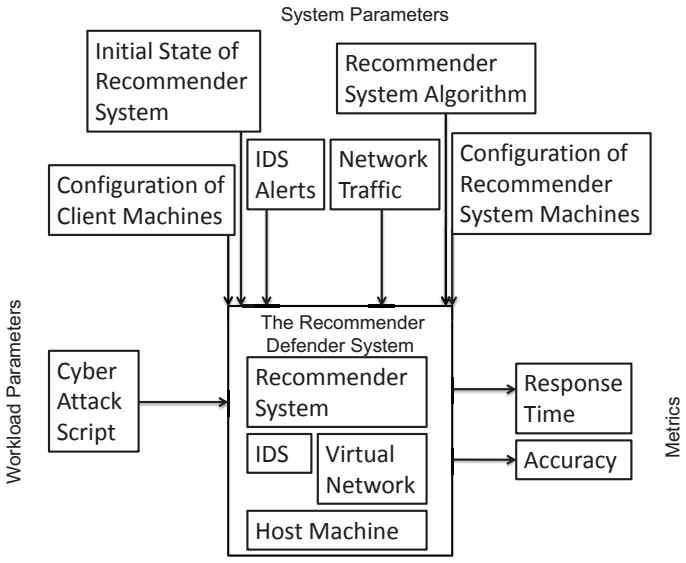

Figure 4.1: The Recommender Defender System

identifying attack, (3) no alert to attack, and (4) error. The recommender system outcomes include: (1) correctly predicting the defense counteraction, (2) incorrectly predicting the defense counteraction, and (3) error. A correct prediction of a defense attack is an action which is appropriate for mitigating or stopping the next attack to be performed by the cyber attacker. An error outcome includes a failure of the component or a null output from the component. The recommender system is the focus of the experiment, therefore metrics are defined to measure these three possible outcomes. A successful outcome for the system occurs when the recommender system presents a correct defender counteraction for the next attacker action.

4.4 Workload

The workload for the network includes the traffic from the attacker machine. The traffic from the attacker machine varies to include multiple attacks. For each attack the basic attacker principles are followed show in Figure 4.2 [26, 28]. Information gathering

1. Information Gathering - Collection of general information about the target.

2. Scanning and Vulnerability Assessment - Network scan performed to obtain information about users and potential vulnerabilities on the network.

   Nmap

3. Intrusion - Exploitation of vulnerabilities to gain access to the system.

   ms08_067_netapi exploit

4. Maintaining Access - Deployment of a backdoor or other malicious software.

   netcat

5. Clearing Tracks - Removal of the evidence of cyber attack.

   alter register information

Figure 4.2: General cyber attacker process

does not include an attack tool, because it is usually not performed on the network directly, but instead as a social networking attack. The attacks are from the suite of available tools on the Linux Backtracking OS, specifically metasploit modules. Using different tools from the metasploit framework attacks perform each step in the process. The order and types of attacks performed is considered the workload for this system. The first attack is Nmap for the scanning step, then ms08_067_netapi vulnerability from the metasploit modules for the intrusion step.

4.5 Performance Metrics

The accuracy of recommender systems reveals the effectiveness of using a specific recommender system in the cyber defense domain. The widely accepted metric used for

recommender systems is a calculation of Root Mean Square Error (RMSE) which requires comparison between the utility assigned by the recommender system and the true utility of an item [11, 14]. RMSE highlights the difference in the recommendation from the true value, which shows how well the recommender can determine the attacker's next action.

The accuracy of the system is not the only metric of interest. It is useful to know how long it takes for the recommender system to determine a recommendation. The speed of recommendation calculations determines if the recommender system can have practical applications on a real network. For a recommender system to be useful on a real network it must be able to compute an accurate recommendation within a reasonable amount of time. Some recommender systems perform computation offline, but this is not an acceptable solution for a cyber attack predictor [18]. The IDS performs the analysis of the system and creates an output file which is sent to the recommender system. The computation time for the recommender system begins when it receives the file from the IDS and ends when a recommendation has been displayed to the defender. The time measurement includes the processing of the IDS information into the matrix used by the recommender system. By measuring the time from when the recommender system receives a new alert, it focuses the metric on the recommender algorithm and is not affected by the performance of the IDS or the virtual network.

4.6 System Parameters

The system parameters are characteristics of the system which affect the performance of the system. The majority of these parameters are fixed.

- Configuration of client machines - The OS and the Random Access Memory (RAM) for the machines on the network determine the services available on each machine and the speed. The configuration influences the effectiveness of attacks. The effectiveness of attacks are also directly related to the vulnerabilities on the user machines which depend on the OS and service settings such as open ports.

- Configuration of recommender system machine - The recommender system machine configuration affects the performance of the computation speed of a suggestion for a defender's reaction.

- IDS Alerts - The appearance of the attacks impacts the response from the IDS. The IDS classifies and correlates information from the virtual network to generate updates for the recommender system. The output from the IDS is the only information the recommender system uses to create a suggestion of defense. Depending on what the IDS presents as output greatly affects the recommendations made by the recommender system.

- Initial state of recommender system before execution of algorithm - The accuracy of the recommender system depends heavily on information from the IDS, and the state of the user-item matrix. The current information from the IDS must be added to the user-item matrix before executing the recommender system algorithm. The user-item matrix relates attacker actions to a specific attacker with a utility value. The utility value ranges between 0 and 1, where 1 means the attacker used the attack and all other values represents the likelihood the attacker will use the attack. The results from the recommender system are limited to the potential actions given for each machine by the IDS output.

- Implementation of recommender system algorithm - The algorithm implemented for the recommender system affects the speed and the recommendation generated. The type of recommender system used would return different suggestions. The two main types of collaborative and knowledge-based recommender systems all use very different approaches to predicting the next action of the attacker. The output of the system depends on the implementation specifics of the hybrid recommender. The different types of recommender systems have different computational complexity

which depends significantly on implementation of the algorithm. Depending on which one is implemented in the system the speed of the defender's reaction is significantly impacted.

Cascade Technique - Cascaded hybrid recommenders group items and then prioritize within the groups to form an overall recommendation. The cascade recommender algorithm must complete the first method before applying the second method.

4.7 Factors

The factors for the system are the parameters and workload which are varied in the experiment. The parameter which is varied for the experiment is the algorithm implemented for the recommender system. All of the other parameters will remain constant to reveal the effects of different recommender system algorithms in the cyber defense domain. The factors are shown in Table 4.1.

Table 4.1: Factor Levels

Level	Hybridization	Algorithm 1	Algorithm 2	Variation
1	Cascade	Collaborative	Knowledge-based	Attributes no IP association
2	Cascade	Collaborative	Knowledge-based	Threshold 0
3	Cascade	Collaborative	Knowledge-based	Threshold 0.5
4	Cascade	Collaborative	Knowledge-based	Threshold 0.75
5	Cascade	Collaborative	Knowledge-based	Threshold 0.95

Different cyber attacks are executed to show how the recommender system works in two different situations. The objective of the attack will be to compromise a specific user machine on the network. The attacks will target a machine on the network. The recommender system should react to protect the target machine, but depending on the

information it has in the user-item matrix it may predict incorrectly. The cyber attacks are scripted with different attacks which follow the basic principles of cyber methodology. In order for the recommender system to be able to predict the attacker's next action the recommender system needs to understand attacker actions.

4.8 Evaluation Technique

Measurement of the virtual network is the main evaluation technique. The virtual network functions as both a realistic network and good testing environment. Between experiments, a virtual network snapshot can easily be taken at any stage and reverted back to previous states.

The virtual network is located on a server with two 3.46 GHz CPU, 192.0 GB memory, 1.8 TB disk capacity, and a VMWare ESXi hypervisor 5.0.0. The topology of the virtual network is show in Figure 4.3. For this experiment the machine with the IDS and the recommender is labeled "sensor". The recommender system code is written in java and compiled using java version $1.6.0_24$ OpenJDK (IcedTea61.11.5) ($6b24-1.11.5-0ubuntu1-12.04.1$). As stated in the previous section the system is affected by the computation speed of machines on the network. Table 4.2 shows the OS, Central Processing Unit (CPU), the RAM, version, and function for each of the different machines on the network. The OS corresponds to the label given in Figure 4.3 and the values in Table 4.2 apply to all of the machines with the same OS.

4.9 Experimental Design

A full-factorial design is implemented with five different factors. The algorithm factor uses five different algorithms. The number of experiments is 5 and with 30 replications that results in 150 total experiments. Considering the deterministic nature of the recommender system the variance is very low assuming that the IDS has valid output. With such a low variance a 95% confidence level is achieved with only 30 replications of each. A statistical

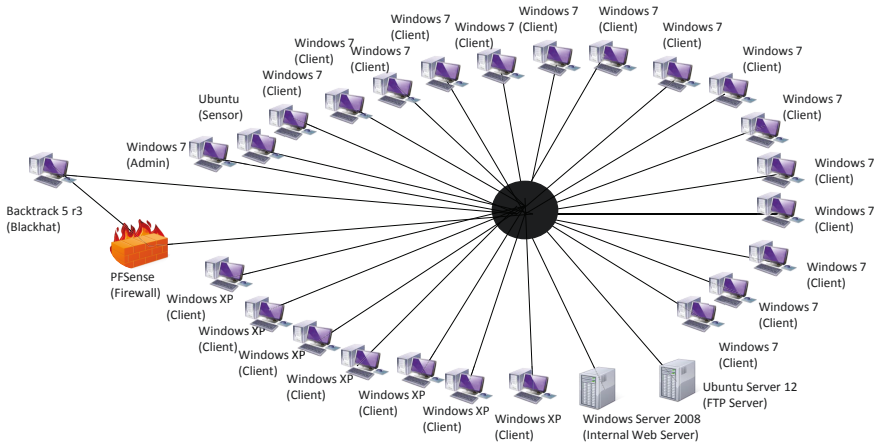

Figure 4.3: Virtual Network Topology

Table 4.2: Configuration of Virtual Machines

OS	CPU	RAM	Version	Function
Ubuntu Desktop 12	3.46 GHz	4 GB	3.2.0-27-generic-pae	Sensor
Windows 7 Enterprise 32-bit	3.47 GHz	1 GB	SP1	Client
Windows XP Professional	3.47 GHz	512 MB	SP2	Client
Ubuntu Server 12.04 LTS	3.46 GHz	1 GB	3.2.0-23-generic-pae	Sever
Windows Server 2008 64-bit Enterprise	3.47 GHz	4 GB	SP1	Server
PFSense	3.46GHz	1 GB	2.0.1	Firewall

difference in the algorithm's performance can be seen in the 95% confidence interval even at a low percentage of difference between the results. The levels chosen limit the factors to a reasonable number of possibilities making full-factorial experiments practical.

4.10 Summary

This chapter describes the methodology implemented to determine the effectiveness of recommender systems as an attack predictor in the cyber defense domain. The workload designed simulates a realistic attacker methodology following the five step process. The parameters include the configuration of the virtual network, IDS alerts, the state of the recommender system, and the implementation of the algorithms. The metrics of response time and accuracy are selected to observe the factors and levels of recommender system algorithm implementation. A virtual network is used because it enables multiple experiments to be conducted efficiently and still represent a realistic network. A full-factorial experimental design is shown for that measurement technique.

V. Results and Analysis

5.1 Overview

In this chapter the results of the previously outlined test are presented, beginning with the metrics from the experiments, followed by the statistical analysis of the results, and ending the chapter with the implications of the experiment's results.

5.2 Computation Time Results

The first experiment used the recommender system algorithm where each attribute was not directly related to the IP address. The computation time for the individual run had a max of 1.477 seconds and a minimum of 0.03843 seconds. The mean computation time was 0.1425 seconds with a median of 0.09689 seconds. Figure 5.2 shows the many large outliers from the computation time, which correspond to the peaks in Figure 5.1. The first large peak is the maximum value and can be explained by the initialization which occurs in the first run of the algorithm. The first time the recommender system runs it must load the attack knowledge base. After the first run of the algorithm the same knowledge base is used for the following runs. The other peaks in the computation time in Figure 5.1 occur when the network is under attack. The computation time for the recommender system spikes during an attack on the network because the knowledge based recommender section of the algorithm must determine the best response. When the network is not under attack there are no current attacks to react to, but depending on the threshold there maybe predicted attacks on vulnerable nodes on the network.

The next set of algorithms were run in parallel. Each was subjected to the same attacks with only the modification of different threshold values applied. The threshold value is the number calculated for the attack event for each node. If the predicted value is above the threshold value that node is viewed as being attacked for the purpose of determining the

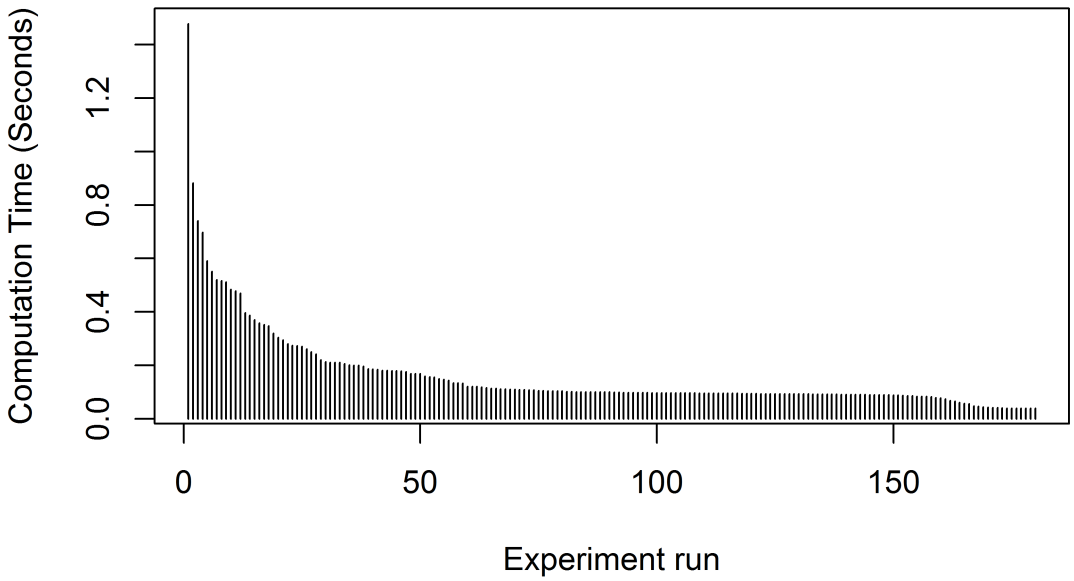

Figure 5.1: Computation Time for Recommender System Algorithm: Attributes with no IP association

counter action to be taken. The change in threshold value could affect the outcome of the computation time because with more possible attacks different response actions must be considered when the knowledge based recommender system determines the best defensive action. In Table 5.1, the summary statistics for the four different algorithms are shown.

An ANalysis Of VAriance (ANOVA) test is performed on the computation times for the five different algorithms to determine if there is a significant difference between the computation times. The null hypothesis for the ANOVA test is that the mean computation time is equal for all five of the algorithms. The results of the ANOVA test are shown in Table 5.2. The outliers for the data makes a linear model questionable for the data. The very

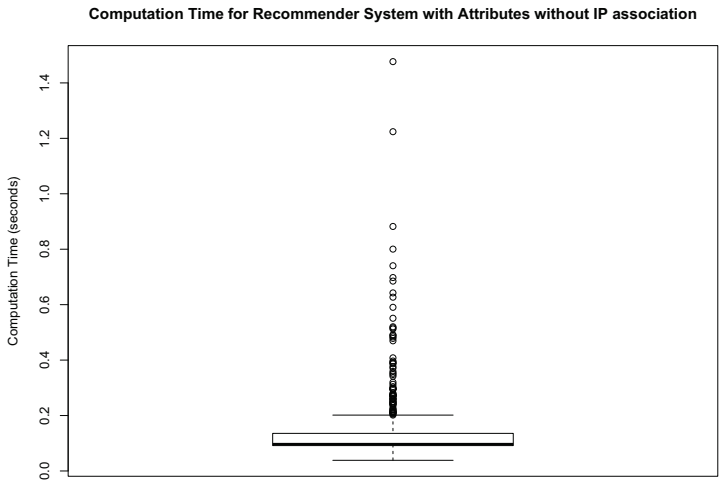

Figure 5.2: Computation Time for Recommender System

small p-value found for the ANOVA test shows that there is a significant difference in the mean values. The box plots in Figure 5.2 and Figure 5.3 support the conclusion of different mean values with the mean of the algorithm with attributes without IP association, with a threshold of 0, and a threshold of 0.95 mean values that do not fall within the quartiles of the algorithms with threshold values of 0.5 and 0.75.

5.3 Accuracy Results

The metric for determining the accuracy of the recommendation is RMSE. The RMSE compares the predicted value calculated by the recommender system to the known true value. Observing the specific attack performed on the Windows XP machine, the recommender system should predict higher values for that same event occuring for other Windows XP machines. For the purpose of these calculations the known true value for all XP machines with the same vulnerability as the attacked machine were given a value of 1 for that attack, while all other machines were given a value of 0. The RMSE is calculated

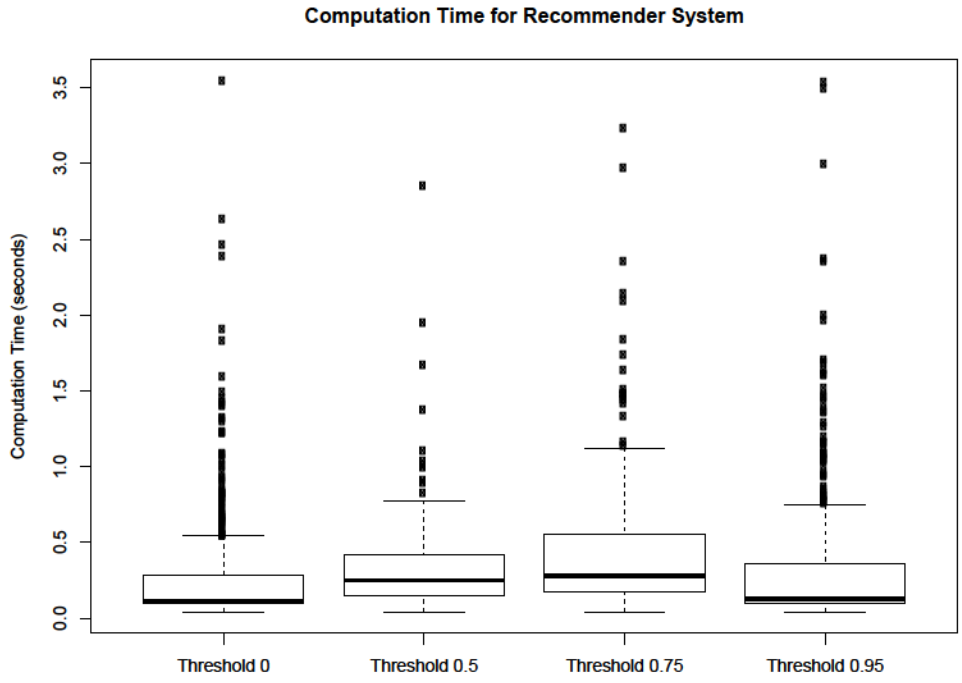

Figure 5.3: Computation Time for Recommender System

using Equation (5.1).

$$\sqrt{\frac{\sum(x - x_1)^2}{n}} \tag{5.1}$$

The RMSE for vulnerable machines was significantly higher than the RMSE results for non-vulnerable machines. Almost all of the predicted values were closer to 0 than 1 but the values for the vulnerable machines were statistically higher than the predicted values of the non-vulnerable machines. A Welch two sample t-test was performed comparing the mean predicted value for the attack on the set of vulnerable machines and the set of non-vulnerable machines. The Welch two sample t-test was performed on all of the predicted values for each of the five recommender system algorithms. The results of the Welch t-test for the recommender system without attribute association was a p-value $< 2.2x10^{-16}$. which is significantly less than the alpha value of 0.05. The 95% confidence

interval was between 0.06518022 and 0.09284223, which does not include 0 meaning that there is a difference in the mean values of the two sets of data. The mean of the vulnerable machines is at 0.12805207 and the mean of the non-vulnerable machines is at 0.04904085, therefore the mean of the predicted values of the event for vulnerable machines is higher than the mean for the non-vulnerable machines. The p-value is the same for all of the algorithms, therefore the conclusions hold true for all of the algorithms tested and are summarized in Table 5.3. Notched boxplots support these conclusions shown in Figure 5.8, and Figure 5.9. All of the notched boxplots do not have overlapping notches which is strong statistically evidence that the means are different for the vulnerable and non-vulnerable predicted values. The notches are calculated using Equation (5.2), which is based off of the same computations used for t-tests. With statistically higher values it shows that the recommender system did predict that those vulnerable machines were possible targets for a similar cyber attack.

$$\frac{1.58 * IQR}{\sqrt{n}} \tag{5.2}$$

Table 5.1: Computation Time of Algorithms with Varied Threshold Values

Threshold	Minimum	1st Quartile	Median	Mean	3rd Quartile	Maximum
0	0.04176 sec	0.1019 sec	0.1115 sec	0.2565 sec	0.2798 sec	3.548 sec
0.5	0.04153 sec	0.1480 sec	0.2497 sec	0.3392 sec	0.4181 sec	2.850 sec
0.75	0.04184 sec	0.1756 sec	0.2823 sec	0.4464 sec	0.5546 sec	3.236 sec
0.95	0.04121 sec	0.1020 sec	0.1301 sec	0.3146 sec	0.3623 sec	3.538 sec

Table 5.2: The results of the ANOVA test on the computation times for the five algorithms

	Degrees of Freedom	Sum Sq	Mean Sq	F-value	Pr(>F)
group	4	$5.7361x10^{18}$	$1.4340x10^{18}$	12.711	$4.579x10^{-10}$
Residuals	895	$1.0097x10^{20}$	$1.1282x10^{17}$		

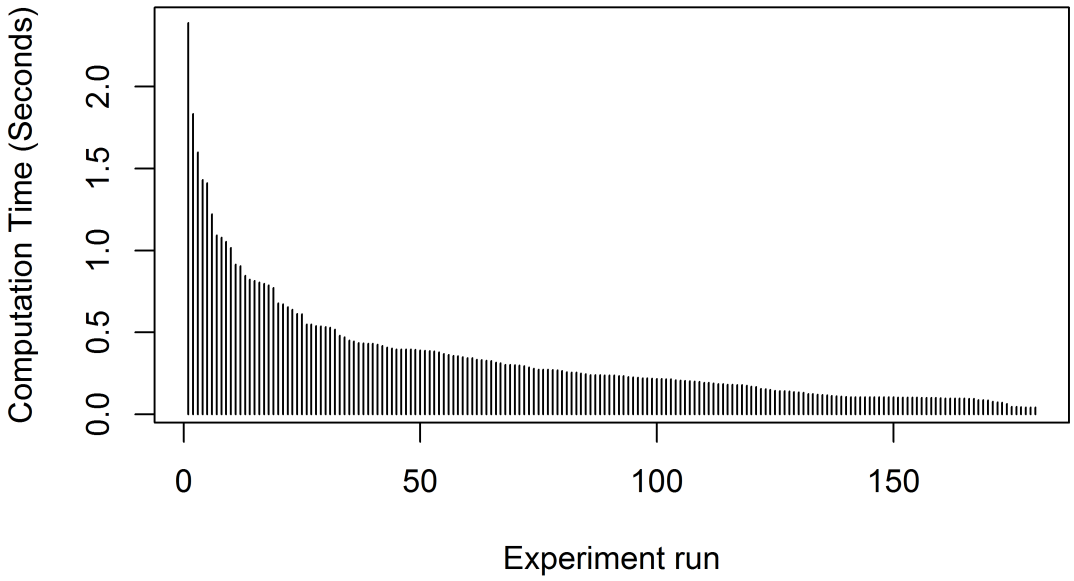

Figure 5.4: Computation Time for Recommender System Algorithm: Threshold Value 0

The recommender system algorithm gave the responses to the netbios attack shown in Table 5.7. There were slight deviations from these responses for each of the five algorithms. The first recommendation was made based on the fact that the knowledge-based recommender system determined that it covered the largest number of threats to the network. The number of threats could change based on the threshold value for each

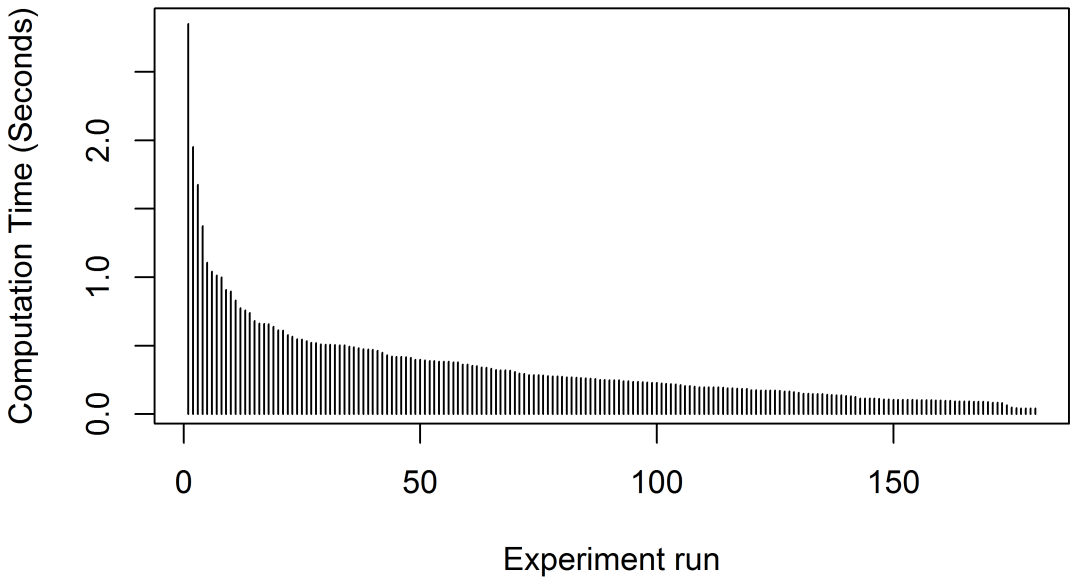

Figure 5.5: Computation Time for Recommender System Algorithm: Threshold Value 0.5

Table 5.3: Welch Two Sample T-test Comparing Vulnerable and Non-vulnerable Machines

Algorithm	Confidence Interval	Mean of Vulnerable	Mean of Non-Vulnerable
No Attribute Association	0.065-0.0927	0.127	0.049
Threshold 0	0.072-0.093	0.151	0.068
Threshold 0.5	0.140-0.160	0.151	0.001
Threshold 0.75	0.146-0.162	0.157	0.003
Threshold 0.95	0.145-0.160	0.156	0.003

algorithm. The predicted values for each other vulnerable node was so low they were only included in the algorithm with a threshold value of 0.

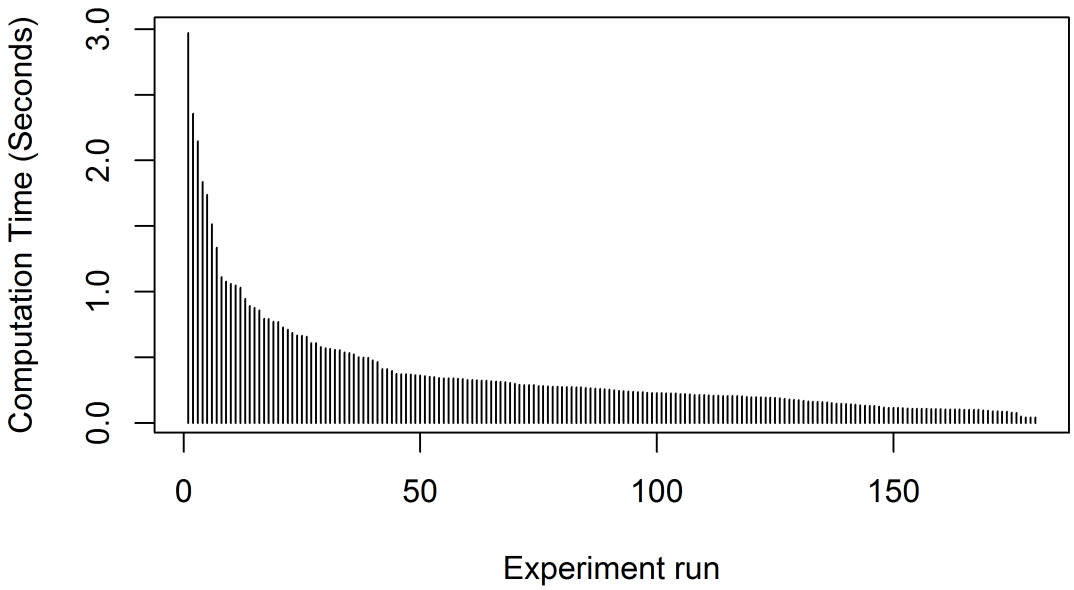

Figure 5.6: Computation Time for Recommender System Algorithm: Threshold Value 0.75

Comparing the recommendations given by each algorithm to the correct cyber response showed 100% correct selection of actions with varied calculated threat levels. For the purpose of these experiments the firewall rule is the correct cyber defense response from the knowledge base that was given to the recommender system. The correct recommendation was generated, but only of the attacks that were above the threshold.

The recommender system algorithm with attributes not associated with IP addresses resulted in the same 45 of 47 recommended actions with all 47 recommendations listed having the same first response of a firewall rule to block the source IP address. The defensive actions are shown in Table 5.8 which shows that the algorithm found vulnerable machines and recommended a defensive action to resolve the threat.

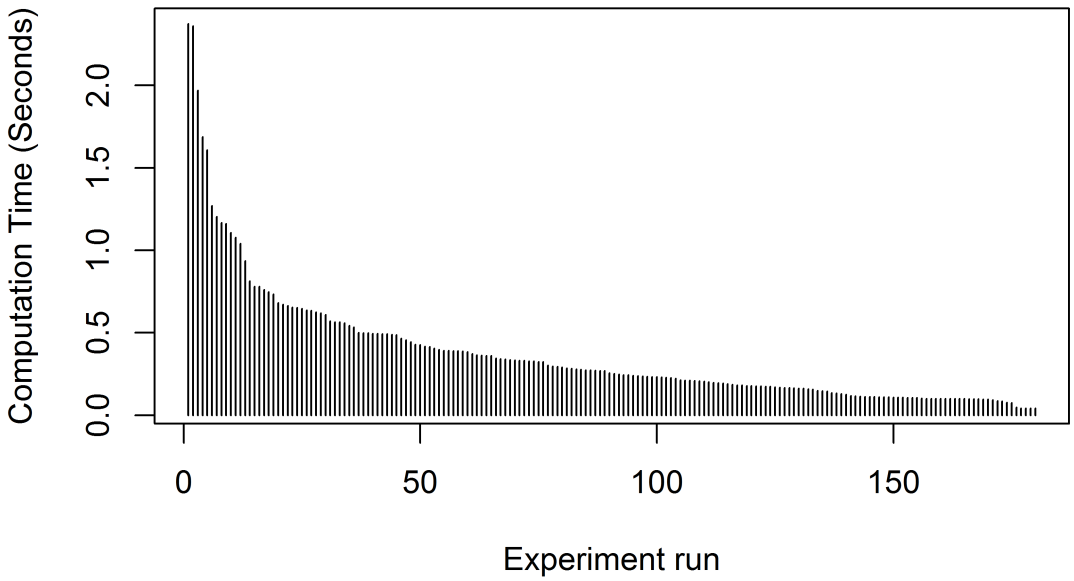

Figure 5.7: Computation Time for Recommender System Algorithm: Threshold Value 0.95

Based on predicted values for vulnerable machines centering around 0.2 the threshold values for the algorithms should be adjusted. Three more algorithms were tested with threshold values of 0.15, 0.18, and 0.2. The recommendations for the thresholds of 0.15, 0.18, and 0.2 identified more threats and addressed them appropriately. The recommendations are show in Table 5.9. The first recommendation of creating a firewall rule, addresses the threat for all vulnerable machines. The following recommendations suggests updating the OS for each vulnerable machine. For the threshold of 0.15, 37 of the 93 total recommended lists alerted to all 6 of the vulnerable machines and gave the correct recommendation. For the threshold of 0.18, 29 of the 88 total recommended lists alerted to all 6 of the vulnerable machines and gave the correct recommendation. For the threshold of 0.2, only 4 of the 93 total recommended list alerted to all 6 of the vulnerable machines and

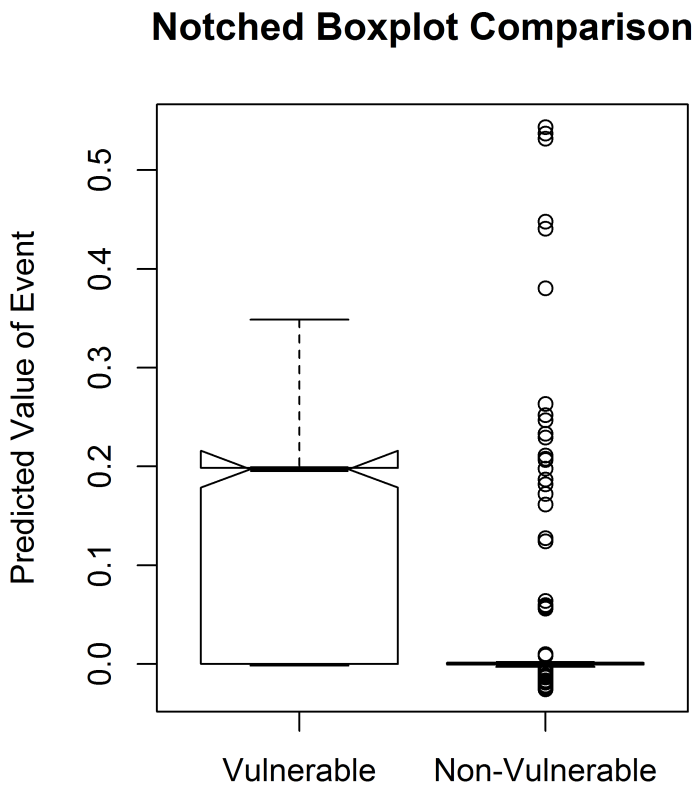

Figure 5.8: Algorithm Attributes with no IP association

gave the correct recommendation. With a lower threshold there were some false positives, meaning that non-vulnerable machines were identified as vulnerable machines. The break down of false positives, true positives, false negatives, and true negatives are shown in Figure 5.10, Figure 5.11, and Figure 5.12. Overall, a threshold of 0.15 appears to be the best choice for the algorithm.

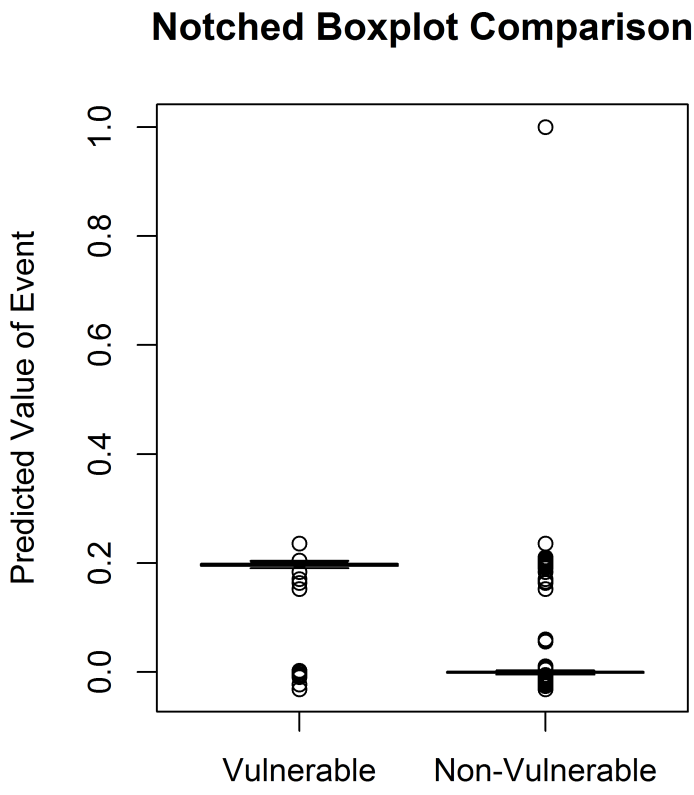

Figure 5.9: Algorithm All Threshold Values

5.4 Implications

The attack prediction algorithm was not as revealing as expected, but correct actions were generated from weak predictions. With the adjusted threshold values the recommendations improved significantly. At the same time the prediction results reveal that more work needs to be done in this area. While, the computation time for a reasonably sized network was on the order of millisecond with the longest time reaching only 3.5 seconds. The computation time shows the recommender system is a viable tool for network security. The computation time for all of the algorithms are impacted by running on a

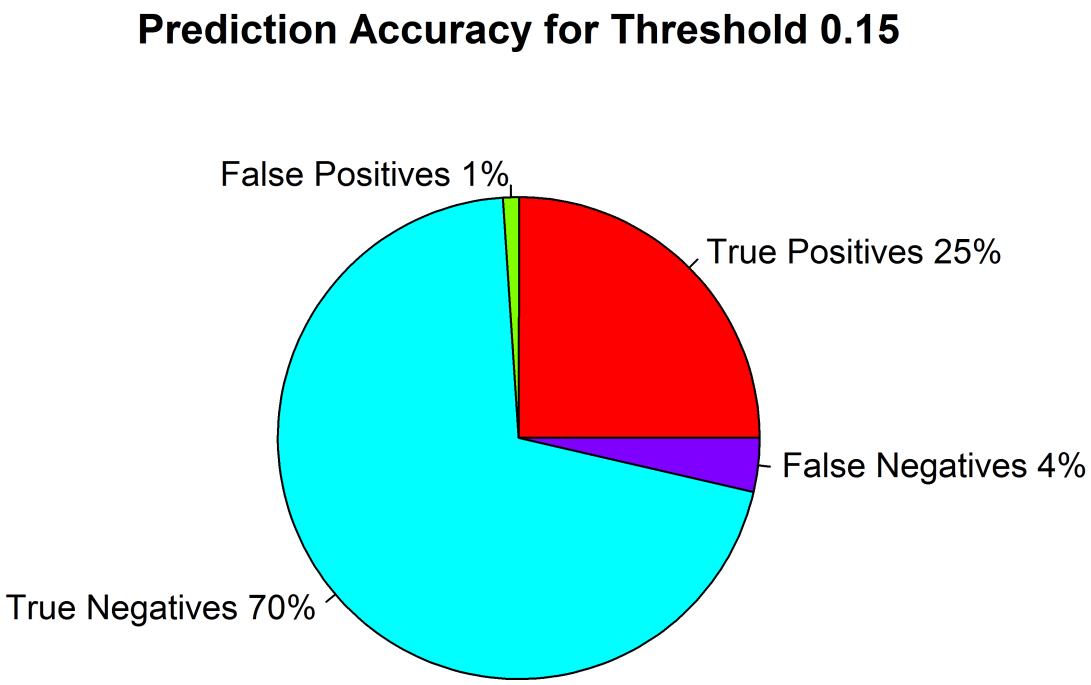

Figure 5.10: Algorithm Threshold Value 0.15

virtual machine sharing resources on the ESXi server. If the algorithm ran on a dedicated physical machine it would run within reasonable time constraints.

5.5 Summary

In this chapter the computation time results were presented. The metrics for accuracy were shown and analyzed. The implications for the data concluded that the recommender system is a valuable tool for cyber defense, but requires refinement.

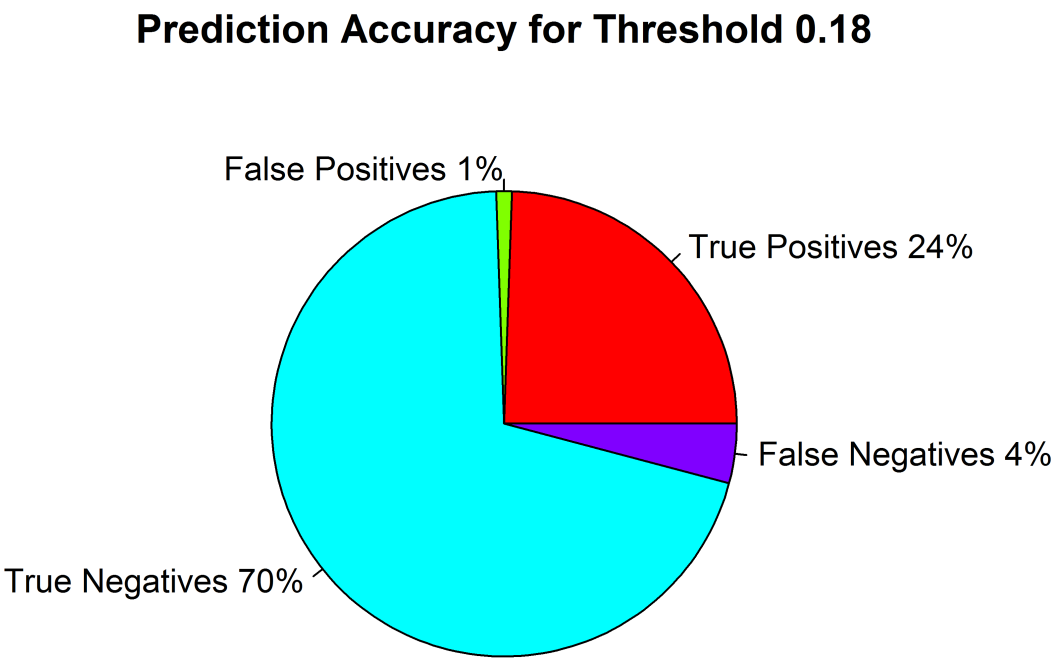

Figure 5.11: Algorithm Threshold Value 0.18

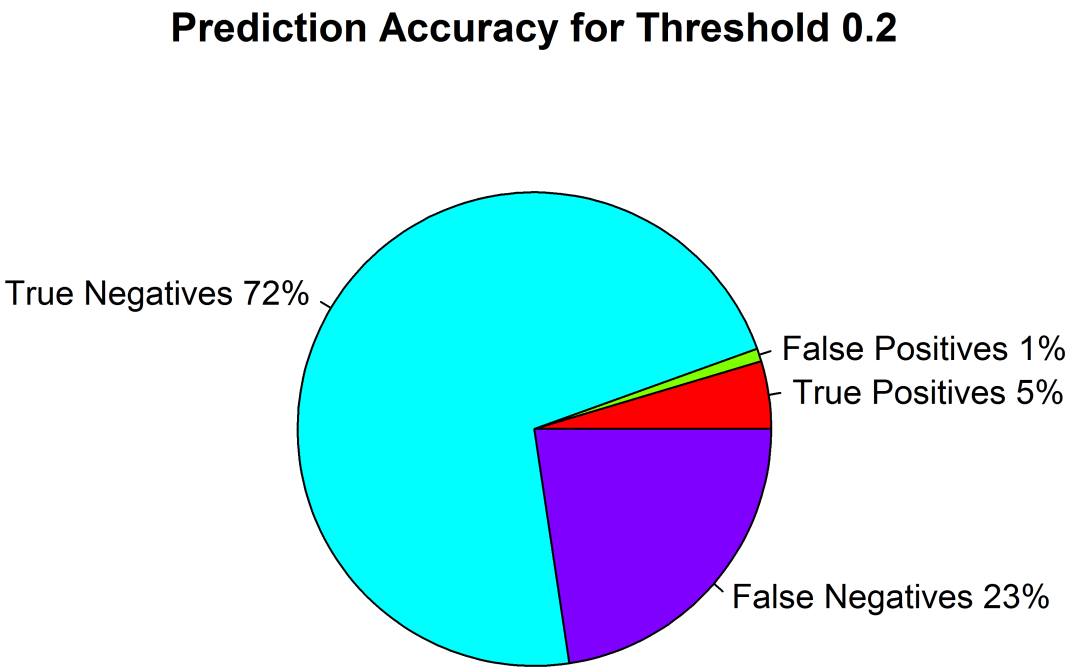

Figure 5.12: Algorithm Threshold Value 0.2

Table 5.4: Root Mean Squared Error

IP address	True Value	RMSE Attributes not associated by IP
192.168.1.6	0	0.00079
192.168.1.8	0	0.00052
192.168.1.9	0	0.00115
192.168.1.10	0	0.04348
192.168.1.11	0	0.00389
192.168.1.12	0	0.01872
192.168.1.13	0	0.00462
192.168.1.14	0	0.01872
192.168.1.15	0	0.00464
192.168.1.16	0	0.00156
192.168.1.17	0	0.02656
192.168.1.18	0	0.00118
192.168.1.19	0	0.00102
192.168.1.21	0	0.00340
192.168.1.23	1	0.84271
192.168.1.24	1	0.94085
192.168.1.25	1	0.88309
192.168.1.26	1	0.87494
192.168.1.27	1	0.86616
192.168.1.28	1	0.85244

Table 5.5: Root Mean Squared Error

IP address	True Value	RMSE Threshold 0	RMSE Threshold 0.5
192.168.1.6	0	0.00923	0.007414
192.168.1.8	0	0.00478	0.00438
192.168.1.9	0	0.00196	0.00227
192.168.1.10	0	0.10993	0.04345
192.168.1.11	0	0.00602	0.00324
192.168.1.12	0	0.00056	0.00065
192.168.1.13	0	0.00604	0.00424
192.168.1.14	0	0.00216	0.00279
192.168.1.15	0	0.00296	0.00380
192.168.1.16	0	0.00451	0.00049
192.168.1.17	0	0.00095	0.00110
192.168.1.18	0	0.00328	0.00500
192.168.1.19	0	0.00261	0.00057
192.168.1.21	0	0.00100	0.00122
192.168.1.23	1	0.85319	0.83878
192.168.1.24	1	0.86101	0.89111
192.168.1.25	1	0.86075	0.86134
192.168.1.26	1	0.85166	0.83917
192.168.1.27	1	0.84098	0.84757
192.168.1.28	1	0.85220	0.83877

Table 5.6: Root Mean Squared Error

IP address	True Value	RMSE Threshold 0.75	RMSE Threshold 0.95
192.168.1.6	0	0.00917	0.00944
192.168.1.8	0	0.00508	0.00525
192.168.1.9	0	0.00218	0.00525
192.168.1.10	0	0.10360	0.10585
192.168.1.11	0	0.00433	0.00484
192.168.1.12	0	0.00059	0.00059
192.168.1.13	0	0.00644	0.00642
192.168.1.14	0	0.00233	0.00223
192.168.1.15	0	0.00315	0.00290
192.168.1.16	0	0.00425	0.00465
192.168.1.17	0	0.00100	0.00116
192.168.1.18	0	0.00350	0.00338
192.168.1.19	0	0.00277	0.00268
192.168.1.21	0	0.00106	0.00131
192.168.1.23	1	0.84758	0.84751
192.168.1.24	1	0.86430	0.85682
192.168.1.25	1	0.84276	0.85681
192.168.1.26	1	0.84693	0.84678
192.168.1.27	1	0.83187	0.83537
192.168.1.28	1	0.84448	0.84736

Table 5.7: Recommended Cyber Defense Action

Rank	Action	Threat Level
1	Firewall Rule Block Source IP Address of Attack	1
2	Update Operating System for 192.168.1.22	1
3	Update Operating System for 192.168.1.23	0
4	Update Operating System for 192.168.1.24	0
5	Update Operating System for 192.168.1.25	0
6	Update Operating System for 192.168.1.26	0
7	Update Operating System for 192.168.1.27	0
8	Update Operating System for 192.168.1.28	0
9	Update Application	0
10	Disable Port	0

Table 5.8: Recommended Cyber Defense Action

Rank	Action	Threat Level
1	Firewall Rule Block Source IP Address of Attack	1
2	Update Operating System for 192.168.1.22	1
3	Update Operating System for 192.168.1.23	1
4	Update Operating System for 192.168.1.24	1
5	Update Operating System for 192.168.1.25	1
6	Update Operating System for 192.168.1.26	1
7	Update Operating System for 192.168.1.27	1
8	Update Operating System for 192.168.1.28	1
9	Update Application	0
10	Disable Port	0

Table 5.9: Recommended Cyber Defense Action

Rank	Action	Threat Level
1	Firewall Rule Block Source IP Address of Attack	6
2	Update Operating System for 192.168.1.22	1
3	Update Operating System for 192.168.1.23	1
4	Update Operating System for 192.168.1.24	1
5	Update Operating System for 192.168.1.25	1
6	Update Operating System for 192.168.1.26	1
7	Update Operating System for 192.168.1.27	1
8	Update Operating System for 192.168.1.28	1
9	Update Application	0
10	Disable Port	0

VI. Conclusion

6.1 Overview

This chapter highlights the results and the contributions of this research effort. Finally, suggested future work is presented.

6.2 Results

The results for the experiment looked at computation time and accuracy. The computation time for the algorithm fell on the order of seconds for the maximum and milliseconds for the average time. The recommendations of defensive actions gave the correct order of responses. The predicted values from the collaborative recommender system algorithm had large residuals when compared to the actual values. Even with weak prediction values the recommender system algorithm was able to present the defense actions that mitigated the cyber attack.

6.3 Contribution

The most important contribution of this research effort was the use of recommender systems to generate an ordered list of cyber defense actions. The test bed created for this research can be used for future work. The IDS has been fully incorporated for all the machines on the network. The size of the network makes any research performed on the test bed network applicable to most real networks. Even very large enterprise networks could be defended using the backbone of the network as nodes instead of all the machines.

6.4 Future Work

The recommender system did not appear to be significantly more insightful compared to other attack predictors. With so many different recommender system algorithms in existence, more algorithms need to be explored to determine their value as a cyber defense

tool. The value of using a recommender system as a cyber defense tool is the resulting list of recommended actions. Instead of building a purely recommender system attack predictor and action recommendation creator, a recommender system could be used to augment a current attack predictor to generate a list of recommended actions. At the same time recommender systems hold the potential to be a very useful attack predictor but needs more development.

Instead of using the collaborative technique for making recommendations a content-based approach could be taken. A recommender system to utilize information from different attackers with different styles viewing the attackers then comparing them to known archetypes of attackers in order to predict their future attacks should be built. The attacker would be the customer and the attack is an item. The recommender system would calculate a rating for other attacks, acting as an attack predictor. Based on the prediction a recommended list of actions can be generated.

Using some of the same recommender system algorithms from this research should be further explored by focusing on the knowledge-base. The knowledge-base could easily be expanded which may lead to more insightful recommendations. The next step would be to add a learning aspect to the algorithm where it builds on the knowledge-base by viewing how certain attacks are mitigated depending on the cyber defense action which is executed. The similarity calculation performed by the recommender system could be very valuable but more work should be done to focus on key attributes of each machine on the network. The IDS provided a wide range of attributes, some of which should be given more weight than others instead of the flat consideration given by the algorithms in this research effort. With more refinement, a better attack predictor can be developed.

6.5 Summary

This research has shown that a recommender system can be used as an attack predictor and cyber defense tool. The results show that a recommended list of cyber defense actions

can be quickly and accurately presented to the cyber warrior. The recommender system and tests designed act as a foundation for more exploration into using recommender systems in the cyber domain. There are many opportunities for continuing work in this area of study.

Bibliography

[1] Billsus, Daniel, Michael J. Pazzani, and James Chen. "A learning agent for wireless news access". *Proceedings of the 5th international conference on Intelligent user interfaces*, IUI '00, 33–36. ACM, New York, NY, USA, 2000. ISBN 1-58113-134-8. URL http://doi.acm.org/10.1145/325737.325768.

[2] Boyd, John. "A discourse on winning and losing", 1987.

[3] Burke, Robin. "Knowledge-Based Recommender Systems". *ENCYCLOPEDIA OF LIBRARY AND INFORMATION SYSTEMS*, 2000. Marcel Dekker, 2000.

[4] Burke, Robin. "Hybrid Recommender Systems: Survey and Experiments". *User Modeling and User-Adapted Interaction*, 12(4):331–370, nov 2002. URL http://dx.doi.org/10.1023/A:1021240730564.

[5] Cipriano, Casey, Ali Zand, Amir Houmansadr, Christopher Kruegel, and Giovanni Vigna. "Nexat: a history-based approach to predict attacker actions". *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, 383–392. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0672-0.

[6] Claypool, M., A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. "Combining Content-Based and Collaborative Filters in an Online Newspaper". *SIGIR 99 Workshop on Recommender Systems: Algorithms and Evaluation*. 1999. URL http://web.cs.wpi.edu/~claypool/papers/content-collab/content-collab.pdf.

[7] Daley, K., R. Larson, and J. Dawkins. "A structural framework for modeling multi-stage network attacks". *Parallel Processing Workshops, 2002. Proceedings. International Conference on*, 5–10. 2002. ISBN 1530-2016. ID: 1.

[8] Fava, D., J. Holsopple, S. J. Yang, and B. Argauer. "Terrain and behavior modeling for projecting multistage cyber attacks". *Information Fusion, 2007 10th International Conference on*, 1–7. 2007. ID: 1.

[9] Felfernig, A. and R. Burke. "Constraint-based recommender systems: technologies and research issues". *Proceedings of the 10th international conference on Electronic commerce*, ICEC '08, 3:1–3:10. ACM, New York, NY, USA, 2008. ISBN 978-1-60558-075-3. URL http://doi.acm.org/10.1145/1409540.1409544.

[10] Goldberg, David, David Nichols, Brian M. Oki, and Douglas Terry. "Using collaborative filtering to weave an information tapestry". *Commun.ACM*, 35(12):61–70, dec 1992. URL http://doi.acm.org/10.1145/138859.138867.

[11] Herlocker, Jonathan L., Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. "Evaluating collaborative filtering recommender systems". *ACM Trans.Inf.Syst.*, 22(1):5–53, Jan 2004. URL http://doi.acm.org/10.1145/963770.963772.

[12] Huang, Zan, Hsinchun Chen, and Daniel Zeng. "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering". *ACM Trans.Inf.Syst.*, 22(1):116–142, jan 2004. URL http://doi.acm.org/10.1145/963770.963775.

[13] Ingols, K., R. Lippmann, and K. Piwowarski. "Practical Attack Graph Generation for Network Defense". *Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual*, 121–130. 2006. ISBN 1063-9527. ID: 1.

[14] Jahrer, Michael, Andreas Toscher, and Robert Legenstein. "Combining predictions for accurate recommender systems". *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, 693–702. New York, NY, USA, 2010. ISBN 978-1-4503-0055-1. URL http://doi.acm.org/10.1145/1835804.1835893.

[15] Jannach, Dietmar, Markus Zanker, Alexander Felfernig, and Gerard Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, New York, NY, 2011.

[16] Junker, Ulrich. "QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems". 167–172, 2004. URL http://dl.acm.org/citation.cfm?id=1597148.1597177.

[17] LeMay, Elizabeth, Willard Unkenholz, Donald Parks, Carol Muehrcke, Ken Keefe, and William H. Sanders. "Adversary-driven state-based system security evaluation". *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, MetriSec '10, 5:1–5:9. New York, NY, USA, 2010. ISBN 978-1-4503-0340-8. URL http://doi.acm.org/10.1145/1853919.1853926.

[18] Linden, Greg, Brent Smith, and Jeremy York. "Amazon.com Recommendations: Item-to-Item Collaborative Filtering", 2003. URL http://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf.

[19] McSherry, David. "Similarity and compromise". *Proceedings of the 5th international conference on Case-based reasoning: Research and Development*, ICCBR'03, 291–305. Springer-Verlag, Berlin, Heidelberg, 2003. ISBN 3-540-40433-3. URL http://dl.acm.org/citation.cfm?id=1760422.1760448.

[20] Melville, Prem, Raymod J. Mooney, and Ramadass Nagarajan. "Content-boosted collaborative filtering for improved recommendations". *Eighteenth national conference on Artificial intelligence*, 187–192. American Association for Artificial Intelligence, Menlo Park, CA, USA, 2002. ISBN 0-262-51129-0. URL http://dl.acm.org/citation.cfm?id=777092.777124.

[21] Ning, Peng and Dingbang Xu. "Learning attack strategies from intrusion alerts". *Proceedings of the 10th ACM conference on Computer and communications security*,

CCS '03, 200–209. ACM, New York, NY, USA, 2003. ISBN 1-58113-738-9. URL http://doi.acm.org/10.1145/948109.948137.

[22] Qin, Xinzhou and Wenke Lee. "Attack Plan Recognition and Prediction Using Causal Networks". *Proceedings of the 20th Annual Computer Security Applications Conference*, ACSAC '04, 370–379. IEEE Computer Society, Washington, DC, USA, 2004. ISBN 0-7695-2252-1. URL http://dx.doi.org/10.1109/CSAC.2004.7.

[23] Raulerson, Evan. "Modeling Cyber Situational Awareness Through Data Fusion", 2013.

[24] Salton, G., A. Wong, and C. S. Yang. "A vector space model for automatic indexing". *Commun.ACM*, 18(11):613–620, nov 1975. URL http://doi.acm.org/10.1145/361219.361220.

[25] Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl. "Item-based collaborative filtering recommendation algorithms". *Proceedings of the 10th international conference on World Wide Web*, WWW '01, 285–295. ACM, New York, NY, USA, 2001. ISBN 1-58113-348-0. URL http://doi.acm.org/10.1145/371920.372071.

[26] SonicWALL. "Anatomy of a Cyber-Attack", 2012.

[27] Tran, Thomas and Robin Cohen. "Hybrid Recommender Systems for Electronic Commerce". *In Knowledge-Based Electronic Markets, Papers from the AAAI Workshop*, 78–83, 2000.

[28] Walker-Brown, Andrew. "The art of the cyber war in six steps", April 19, 2013 2013.

[29] Wang, Jun, Arjen P. de Vries, and Marcel J. T. Reinders. "Unifying user-based and item-based collaborative filtering approaches by similarity fusion". *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, 501–508. ACM, New York, NY, USA, 2006. ISBN 1-59593-369-7. URL http://doi.acm.org/10.1145/1148170.1148257.

[30] Yang, Shanchieh J., Adam Stotz, Jared Holsopple, Moises Sudit, and Michael Kuhl. "High level information fusion for tracking and projection of multistage cyber attacks". *Inf. Fusion*, 10(1):107–121, January 2009.

[31] Ziegler, Cai-Nicolas, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. "Improving recommendation lists through topic diversification". *Proceedings of the 14th international conference on World Wide Web*, WWW '05, 22–32. ACM, New York, NY, USA, 2005. ISBN 1-59593-046-9. URL http://doi.acm.org/10.1145/1060745.1060754.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
27–03–2014	Master's Thesis	Oct 2013–Mar 2014

4. TITLE AND SUBTITLE

A Recommender System in the Cyber Defense Domain

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Lyons, Katherine B., Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-14-M-49

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In the cyber domain, network defenders have traditionally been placed in a reactionary role. Before a defender can act they must wait for an attack to occur and identify the attack. This places the defender at a disadvantage in a cyber attack situation and it is certainly desirable that the defender out maneuver the attacker before the network has been compromised. The goal of this research is to determine the value of employing a recommender system as an attack predictor, and determine the best configuration of a recommender system for the cyber defense domain. The most important contribution of this research effort is the use of recommender systems to generate an ordered list of cyber defense actions.

15. SUBJECT TERMS

recommender system cyber defense

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Kenneth Hopkinson (ENG)
U	U	U	UU	81	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4579 Kenneth.Hopkinson@afit.edu