View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by DigitalCommons@Kennesaw State University

Kennesaw State University
DigitalCommons@Kennesaw State University

KSU Proceedings on Cybersecurity Education, Research and Practice

2017 KSU Conference on Cybersecurity Education, Research and Practice

Experiments with Applying Artificial Immune System in Network Attack Detection

Alexis Cooper
North Carolina A & T State University, adcoope2@aggies.ncat.edu

Follow this and additional works at: https://digitalcommons.kennesaw.edu/ccerp


Part of the Information Security Commons, Management Information Systems Commons, and the Technology and Innovation Commons

Cooper, Alexis, "Experiments with Applying Artificial Immune System in Network Attack Detection" (2017). *KSU Proceedings on Cybersecurity Education, Research and Practice*. 3.
https://digitalcommons.kennesaw.edu/ccerp/2017/research/3

This Event is brought to you for free and open access by the Conferences, Workshops, and Lectures at DigitalCommons@Kennesaw State University. It has been accepted for inclusion in KSU Proceedings on Cybersecurity Education, Research and Practice by an authorized administrator of DigitalCommons@Kennesaw State University. For more information, please contact digitalcommons@kennesaw.edu.

Abstract
The assurance of security within a network is difficult due to the variations of attacks. This research conducts various experiments to implement an Artificial Immune System based Intrusion Detection System to identify intrusions using the Negative Selection Algorithm. This research explores the implementation of an Artificial Immune System opposed to the industry standard of machine learning. Various experiments were conducted to identify a method to separate data to avoid false-positive results. The use of an Artificial Immune System requires a self and nonself classification to determine if an intrusion is present within the network. The results of an Artificial Immune System based Intrusion Detection System achieved high accuracy when the data records were separated by service. The Negative Selection Algorithm created a range and it provided detectors to determine if an intrusion was present based off of the threshold. The threshold is the number of detectors that must be triggered for the system to identify an intrusion. Many services were unusable as they did contain the requirement of both self and nonself data records, that did not overlap. The results were high accuracies in general for the remaining tested services.

Disciplines
Information Security | Management Information Systems | Technology and Innovation

This event is available at DigitalCommons@Kennesaw State University: https://digitalcommons.kennesaw.edu/ccerp/2017/research/3

Experiments with Applying Artificial Immune System in Network Attack Detection

INTRODUCTION

Securing information "covers all the processes and mechanisms by which computer based equipment, information and services are protected from unintended or unauthorized access, change, or destruction" (Yang, Li, Hu, Wang, & Zou, 2014). An Intrusion Detection System (IDS) helps to identify a possible malicious attack or odd behaviors within a network (Tavallaee, Bagheri, Lu, Ghorbani, 2009). An Intrusion Detection System applied to the principles of an Artificial Immune System can provide a resolution to the problems that can occur while securing information. This implementation of an Intrusion Detection System differs from the industry standard of implementing machine learning.

This research applies an Artificial Immune System (AIS) based Intrusion Detection System to the KDD CUP 1999 Corrected dataset. The KDD CUP 1999 full dataset is primarily used for machine learning. The full dataset's sub-datasets include training and test datasets. The KDD CUP 1999 Corrected dataset was used because it is a smaller dataset than the full data set and it is a testing dataset. The dataset is comprised of normal and abnormal data records. Each normal data record has different features that make it an example of normal traffic flow. The abnormal data records or intrusions have different characteristics that identify them differently. The problem that arises when implementing the Artificial Immune System to the dataset is the makeup of the dataset. The KDD CUP 1999 Corrected dataset is composed of a mixture of different types of intrusions and different normal data records. This research takes a new approach of defining self and nonself by isolating self and nonself of a service. The implementation of the AIS based IDS will be defined below.

ARTIFICIAL IMMUNE SYSTEM

An Artificial Immune System uses the model of a Human Immune System to implement a self and nonself identities (Yang et al., 2014). The AIS model identifies key components of the problem and sets parameters on how to identify that problem (Yang et al., 2014).

The Negative Selection Algorithm was the algorithm was used in this experiment to implement an AIS. This algorithm requires classification of self and

nonself. This research shows that strict self and nonself classification for intrusions renders higher accuracies to detect an intrusion.

Negative Selection Algorithm

The Negative Selection Algorithm is a generation algorithm used to create "accurate and efficient detectors" (Yang et al., 2014) that distinguish between self and nonself. The components needed in the Negative Selection Algorithm are the threshold, number of detectors, and the number of features. The number of detectors and threshold relationship resemble the antigen and antibody relationship within the Human Immune System. In humans, an antigen is any substance that causes an immune system to produce an antibody that acts in response of the antigen (Yang et al., 2014). An antibody is used by the immune system to neutralize pathogens such as bacteria and viruses. Similar to the Human Immune System, the AIS detects its version of an antigen, the intrusion, by the Negative Selection Algorithm. The threshold is what determines an intrusion. The artificial body uses the sensory attribute of the Negative Selection Algorithm whereby a specified number of detectors tripped will cause the data record to be classified as an intrusion.

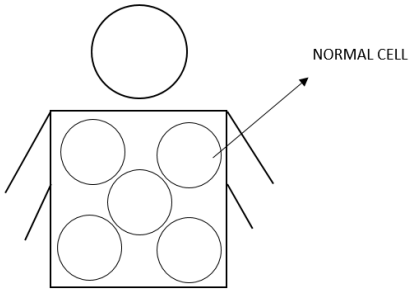

Figure 1 illustrates a normal Human Immune System.

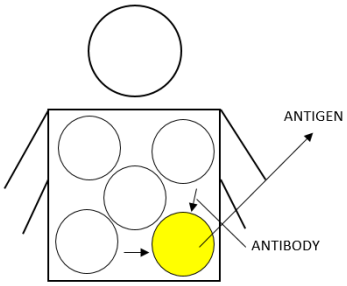

Figure 2 illustrates a Human Immune System with an antigen (yellow circle). The antibodies (arrows pointed to cell) are attacking the antigen to protect the immune system.

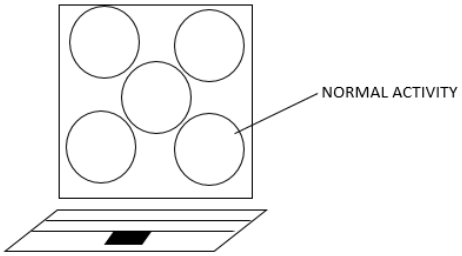

Figure 3 shows an Artificial Immune System with normal activity.

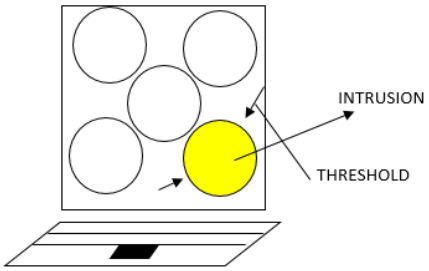

Figure 4 is an illustration of an Artificial Immune System with an abnormal cell (yellow circle). The circles in this figure represent detectors within the system. The arrow represents the threshold variable used to identify an intrusion.

EXPERIMENTS

Methods

The KDD CUP 1999 Corrected dataset was chosen to be used throughout this experience due to the inability to test the full data set due to the size of the file and the restrictive nature of the other datasets. The KDD CUP 1999 Corrected dataset is comprised 311,029 data records that are comprised of 41 features and a classification. Each data record is classified as "normal" or a type of intrusion.

The Negative Selection Algorithm was used in this experiment to identify if a data record is an intrusion. The Negative Selection Algorithm requires the testing dataset, the threshold, the number of self-vectors, the number of nonself vectors, the number of features, and the number of detectors. The testing dataset is a file that consists of a reorganized KDD CUP 1999 Corrected data. The KDD CUP 1999 Corrected dataset is reorganized to a smaller amount of data in the Initial Testing phase. The Corrected dataset is reorganized in the Service Testing phase to directly compare the normal data records within that service to the abnormal data

records of the same service. The threshold is a range of detectors to indicate if a data record is an intrusion. The number of self vectors is the number of normal data records present within the testing dataset. The number of nonself is the number of data records that are classified as a type of attack. The number of features is the amount of features that are included in the test dataset. When tested the number of features are counted by starting at 0. The number of detectors is the number of detectors that will be generated and applied to the Negative Selection Algorithm.

This experiment is split into 4 different parts. The parts are the Initial Testing phase, the Service Isolation phase, the Feature Selection phase, and the Service Testing phase. Each phase explains how its implication has impacted the conclusion of this experiment.

Initial Testing

In the initial testing phase, there are three different testing datasets that are tested. The testing datasets consist of a random selection of normal and abnormal network traffic. The network traffic that is normal is counted as a part of the number of self vectors. The network traffic that is classified as a type of intrusion is counted as a part of the number of nonself vectors. There were three tests that were conducted were randomly chosen as described below. The accuracy of each nonself data record is determined by comparing the self-identity to the data record to see if they match. If the two do not match then a higher the accuracy is given to the data record. The higher accuracies represent an intrusion is present for that nonself data record. The lower accuracies represent that an intrusion was not detected.

Test 1

Test 1 tested the Corrected KDD CUP 1999 dataset. The test dataset must be structured into a self and a nonself ratio. The data records tested were selected randomly. For this research project, the first 11 data records that were classified as normal and the first 179 data records of abnormal traffic were placed into an Excel spreadsheet to be tested. This test also includes 38 features out of the 41 features. The 3 features that were excluded from this test contained letters.

Test 1 used the Negative Selection Algorithm to examine the accuracy of identifying the 179 abnormal data records. An individual accuracy was calculated for each data record. The accuracy displayed in the chart below was measured by computing the average of abnormal data records accuracies divided by the total of nonself vectors.

Test 1 Results					
threshold	*numSelf*	*numNonSelf*	*features**	*numDetectors*	*Accuracy (%)*
1	11	179	2	25	1.98
11	11	179	12	25	1.78
13	11	179	15	25	2.04
18	11	179	22	45	27.86
30	11	179	37	25	52.70

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 1 above shows the results of the Test 1 testing data when applied to Negative Selection Algorithm. The accuracies are rounded to the nearest hundredth.

Test 2

Test 2 tested the following features: dst_host_svr_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, and dst_host_srv_rerror_rate. These five features were randomly selected. The Test 2 testing dataset was formatted to include 12 normal data records and 94 abnormal data records. The 12 normal data records were randomly chosen from the KDD Cup 1999 Corrected dataset. The 94 abnormal data records were an assortment of intrusions (smurf, back, pod, teardrop, ftp_write, and guess_passwd intrusion). The 18 smurf data records, 22 back data records, 21 pod data records, 12 teardrop data records, 3 ftp_write data records, 18 guess_passwd data records were included in the Test 2 testing dataset.

The Negative Selection Algorithm provides the accuracy of each data record. This testing dataset's accuracy is determined by using the Negative Selection Algorithm to identify an intrusion. The average accuracy was computed by taking the sum of each nonself vectors accuracy and dividing it by the number of nonself vectors.

Test 2 Results					
threshold	*numSelf*	*numNonSelf*	*features**	*numDetectors*	*Accuracy (%)*
2	12	94	4	25	0
4	12	94	4	100	0
5	12	94	4	25	0
6	12	94	4	25	0
17	12	94	4	25	0

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 2 shows the results of the Test 2 testing data when applied to Negative Selection Algorithm.

Test 3

Test 3 tested 38 out of the 41 features. The 3 features that were excluded are protocol_type, service, and flag. The test dataset included the first 30 self vectors and the first 130 nonself vectors. The accuracy indicates the ability to identify an intrusion for each nonself vector using the Negative Selection Algorithm. The average accuracy is computed by adding all 130 nonself vector accuracies and dividing the summation by 130.

Test 3 Results					
threshold	*numSelf*	*numNonSelf*	*features**	*numDetectors*	*Accuracy (%)*
20	30	130	37	1,000	90.24
2	30	130	37	100	0
25	30	130	37	1,000	89.04
20	30	130	37	2,000	91.34

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 3 shows results of various parameters applied to the Test 3 testing dataset.

Service Isolation

Categorizing each service is the first step of this method of AIS implementation. Within the KDD CUP 1999 dataset there are 65 different services. Once the services are identified the services are classified into Normal Only, Abnormal Only, and Testing Data. The services that are classified as Normal Only have data records that are solely normal traffic flow. The Abnormal Only services are services that contain intrusions. The Testing Data are the services that contain data records that are normal and abnormal traffic flow, but do not overlap respectively.

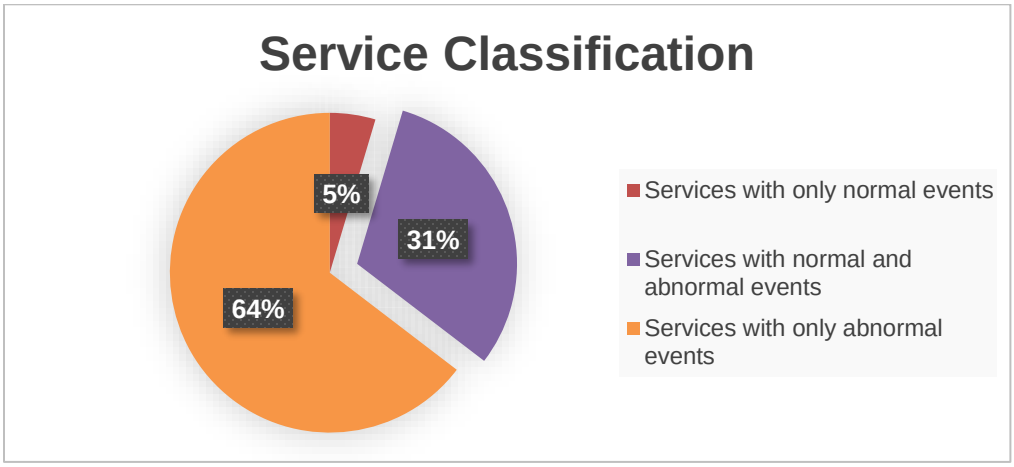

Figure 5 is a pie chart shows the variance of the types of traffic within the 65 services. The services with normal and abnormal data records were used in this testing phase of this experiment.

Feature Selection

Feature Selection is enacted on each service that has examples of normal and intrusion data records. The KDD CUP 1999 dataset only consist of 20 services that contain normal and abnormal traffic flow. Each service was separately evaluated to find the maximum and minimum of the normal data records and then abnormal data records. The ranges for the normal data records were notated to be compared to the abnormal range of the same feature. This form of feature selection examines ranges to determine the repetitiveness of numbers within the different types of data records.

The 41 features are individually classified to determine if they are substantial to use in the testing phase of the experiment to avoid false positive results. A feature can be classified as a consistent feature range, mixed feature range, and varied feature range. A consistent feature range occurs when a feature

is consistent and repetitive for both normal and abnormal data records. The mixed feature range occurs when the normal and abnormal ranges are observed to overlap. This overlap is defined as a single number that is present within both the normal and abnormal ranges. A varied feature range is a feature that has a normal range and an abnormal range that do not overlap. These features were selected to be used in this Selection based Artificial Immune System experiment. The consistent feature range and the mixed feature ranges were not included during testing due to a possible false positive result.

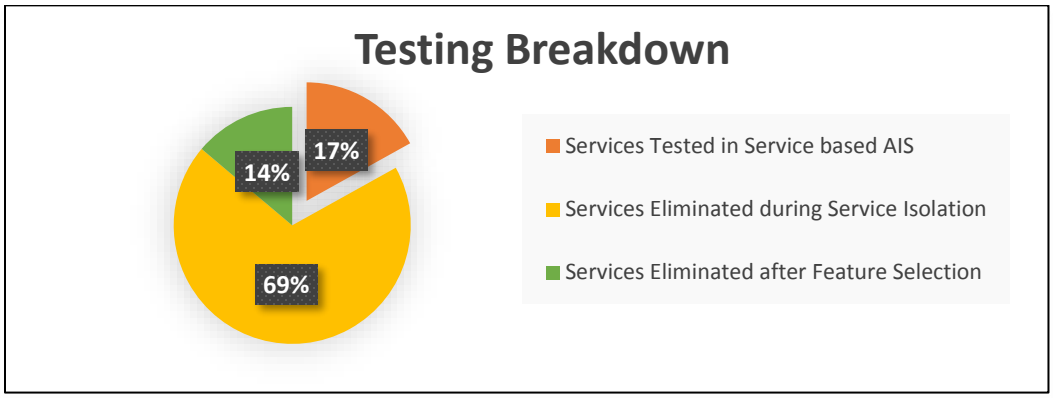

Figure 6 is a visual representation of the amount of services that were tested in the service testing phase of this experiment.

Service Testing

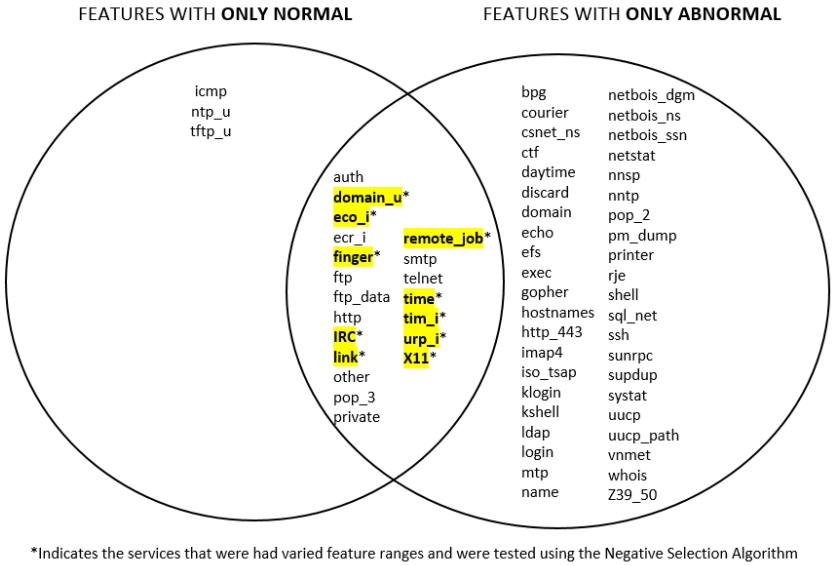

Figure 7 is a relationship figure that shows the amount of services that could be used in the testing phase of this experiment. The services that are highlighted, bolded, and have an asterisk beside them are the services that were tested.

RESULTS

There were 11 services that were eligible to be tested after the feature selection process. Due to the amount of ecr_i data records this service was classified as too complex to isolate. The 10 services that were tested include: domain_u, eco_i, finger, IRC, link, remote_job, time, tim_i, urp_i, X11.

The accuracy each nonself data record is determined by comparing the self-identity to the data record to see if they match. If the two do not match then a higher the accuracy is given to the data record. The higher accuracies represent an intrusion is present for that nonself data record. The lower accuracies represent that an intrusion was not detected. The accuracy displayed in the below tables for each service represents the sum of all the nonself accuracies of each services divided by the number of nonself vectors of that service.

Domain_u Service

The domain_u service test tested all of the domain_u data records. The numSelf of this test is represented by the number of normal data records within the domain_u service. The numNonSelf is the number of abnormal data records within the domain_u service. The diff_srv_rate feature and the dst_host_same_srv_rate feature were the only two features tested. The threshold and number of detectors were varied to find the best combination that warranted the best results.

Domain_u Service Results					
threshold	*numSelf*	*numNonSelf*	*features**	*numDetectors*	*Accuracy (%)*
1	3158	2	1	10,000	100
1	3158	2	1	100	100
1	3158	2	1	1,000	100
2	3158	2	1	10,000	0
25	3158	2	1	100	0
25	3158	2	1	1,000	0

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 4 displays the results for the domain_u service tests.

Eco_i Service

The eco_i service test consisted of 5 tests that only tested the following services: src_bytes, count, srv_count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_rerror_rate. The numSelf represents the number of normal data records within the eco_i service. The numNonSelf represents the number of abnormal data records within the eco_i service. The threshold and number of detectors were varied to find the best combination that warranted the best results.

Eco_i Service Results					
threshold	*numSelf*	*numNonSelf*	*Features*	*numDetectors*	*Accuracy (%)*
3	146	401	10	1,000	99.50
4	146	401	10	1,000	99.50
5	146	401	10	100	98.70
10	146	401	10	10,000	28.27
25	146	401	10	10,000	0

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 5 consist of the results of the eco_i service test. The accuracies are rounded to the nearest hundredth.

Finger Service

The finger service test consisted of 9 tests that only tested the following services: dst_bytes and land. The numSelf represents the number of normal data records within the finger service. The numNonSelf represents the number of abnormal data records within the finger service. The threshold and number of detectors were varied to find the best combination that warranted the best results.

Finger Service Results					
threshold	*numSelf*	*numNonSelf*	*features*	*numDetectors*	*Accuracy (%)*
1	174	117	1	10,000	6.84
1	174	117	1	1,000	6.84
1	174	117	1	1,000	6.84
1	174	117	1	100,000	6.84
1	174	117	0	100,000	0
2	174	117	1	1,000	0
5	174	117	1	1,000	0
25	174	117	1	1,000	0
25	174	117	1	1,000	0

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 6 shows the results from the finger service testing. The accuracies are rounded in the nearest hundredth.

IRC Service

The IRC service test consisted of 7 tests that only tested the following services: src_bytes, dst_bytes, dst_host_count, and dst_host_same_srv_rate. The numSelf represents the number of normal data records within the IRC service. The numNonSelf represents the number of abnormal data records within the IRC service. The threshold and number of detectors were varied to find the best combination that warranted the best results.

IRC Service Results					
threshold	*numSelf*	*numNonSelf*	*features*	*numDetectors*	*Accuracy (%)*
1	21	3	3	100	88.89
2	21	3	3	100	66.67
2	21	3	3	1,000	100
2	21	3	3	10,000	100
2	21	3	3	2,000	100
5	21	3	3	1,000	0
10	21	3	3	1,000	0

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 7 shows the results for the IRC service testing with varying parameters. The accuracies are rounded to the nearest hundredth.

Link Service

The link service test consisted of 4 tests that only tested the following services: duration, count, diff_srv_rate, dst_host_count, dst_host_diff_srv_rate, dst_host_same_src_port_rate, and dst_host_serror_rate. The numSelf represents the number of normal data records within the link service. The numNonSelf represents the number of abnormal data records within the link service. The threshold and number of detectors were varied to find the best combination that warranted the best results.

Link Service Results					
threshold	*numSelf*	*numNonSelf*	*features*	*numDetectors*	*Accuracy (%)*
2	1	65	6	2,000	100
2	1	65	6	100	100
2	1	65	6	1,000	100
3	1	65	6	100	100

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 8 shows the results for the link service testing with varying parameters.

Remote_job Service

The remote_job service test consisted of 4 tests that only tested the following services: duration, count, diff_srv_rate, dst_host_count, dst_host_diff_srv_rate, dst_host_same_src_port_rate, and dst_host_rerror_rate. The numSelf represents the number of normal data records within the remote_job service. The numNonSelf represents the number of abnormal data records within the remote_job service. The threshold and number of detectors were varied to find the best combination that warranted the best results.

Remote_job Service Results					
threshold	*numSelf*	*numNonSelf*	*features*	*numDetectors*	*Accuracy (%)*
2	1	64	6	100	100
3	1	64	6	100	100
4	1	64	6	100	99.95
5	1	64	6	100	92.50

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 9 shows the results for the remote_job service with varying parameters.

Time Service

The time service test consisted of 6 tests that only tested the following services: dst_bytes, and dst_host_count. The numSelf represents the number of normal data records within the IRC service. The numNonSelf represents the number of abnormal data records within the IRC service. The threshold and number of detectors were varied to find the best combination that warranted the best results.

Time Service Results					
threshold	*numSelf*	*numNonSelf*	*features*	*numDetectors*	*Accuracy (%)*
1	28	54	1	1,000	100
1	28	54	1	100	59.26
1	28	54	1	500	98.15
1	28	54	1	750	100
2	28	54	1	100	0
2	28	54	1	750	0

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 10 shows the results for the time service test. The accuracies are rounded to the nearest hundredth.

Tim_i Service

The tim_i service test consisted of 6 tests that only tested the following services: dst_host_count, dst_host_same_src_port_rate, and dst_host_rerror_rate. The numSelf represents the number of normal data records within the tim_i service. The numNonSelf represents the number of abnormal data records within the tim_i service. The threshold and number of detectors were varied to find the best combination that warranted the best results.

Tim_i Service Results					
threshold	*numSelf*	*numNonSelf*	*Features*	*numDetectors*	*Accuracy (%)*
1	1	6	2	100	100
2	1	6	2	100	100
3	1	6	2	100	100
3	1	6	2	1,000	100
4	1	6	2	100	100
5	1	6	2	100	0

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 11 shows the results for the tim_i service testing with varying parameters.

Urp_i Service

The urp_i service test consisted of 8 tests that only tested the following services: dst_host_count and dst_host_diff_srv_rate. The numSelf represents the number of normal data records within the urp_i service. The numNonSelf represents the number of abnormal data records within the urp_i service. The threshold and number of detectors were varied to find the best combination that warranted the best results.

Urp_i Service Results					
threshold	*numSelf*	*numNonSelf*	*features*	*numDetectors*	*Accuracy (%)*
1	58	5	1	1,000	60
1	58	5	1	100	60
1	58	5	1	10,000	0
1	58	5	1	50	60
1	58	5	1	100	60
2	58	5	1	100	0
2	58	5	1	1000	0
2	58	5	1	10,000	0

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 12 shows the test results for the urp_i service with varying parameters.

X11 Service

The X11 service test consisted of 8 tests that only tested the following services: src_bytes, dst_bytes, dst_host_count, dst_host_same_srv_rate, and dst_host_diff_srv_rate. The numSelf represents the number of normal data records within the X11 service. The numNonSelf represents the number of abnormal data records within the X11 service. The threshold and number of detectors were varied to find the best combination that warranted the best results.

X11 Service Results					
threshold	*numSelf*	*numNonSelf*	*features*	*numDetectors*	*Accuracy (%)*
1	4	11	4	200	94.21
1	4	11	4	1,000	99.17
1	4	11	4	2,000	100
1	4	11	4	10,000	100
1	4	11	4	500	99.17
1	4	11	4	1,500	100
2	4	11	4	200	91.74
2	4	11	4	1,000	91.52

**The number features represented in the table is run information. The run information for features start at 0. For the actual number of features add one.*

Table 13 shows the results for the X11 service testing with varying parameters.

DISCUSSION

This experiment uses a feature selection to isolate constancies within KDD CUP 1999 Corrected dataset. Test 1 and Test 2 of in the Initial Testing phase of the experiment had low accuracies because of variance in the testing data parameters (i.e. threshold, numSelf, numNonSelf, features, and the numDetectors). In Test 3 of the Initial Testing phase the testing data parameters (numSelf, numNonSelf, and features) remained constant throughout the test resulting in high accuracies. Isolating the services that have normal and abnormal data records and completing a feature selection is vital to creating a consistent form of testing. Future work to extend this research should include testing another form of feature selection to implement on the Negative Selection Algorithm.

CONCLUSION

The experiments conducted to implement an Artificial Immune System to detect intrusions within a network provided insight on necessary isolation. The isolation used in this research is service based. A service is one of the 41 features that is used to classify a data record as normal traffic or abnormal traffic (intrusion) within the KDD CUP 1999 Corrected dataset. The other 41 features were then used to find consistencies within each type of service. The features with consistencies were eliminated. The features that were not eliminated were

separated into self and nonself classification by separating normal data records and abnormal data records. The normal data records became self and the abnormal data records became nonself for each of the remaining services. A range of normal data records of a service is compared to the range abnormal data records of the same service are compared to observe if there is a form of overlap. Overlap is defined as any data record that is included within the normal data range of a service is included in the abnormal data range of the same service. The services that had no overlap were tested in the Service Testing phase.

The results of this experiment conclude that each individual service has different features that indicate when a data record is an intrusion or a normality. Some of the services require a low threshold that will serve as a strict indicator of when an intrusion is detected. Other services may need a high number of detectors with a low threshold which provides a strict indicator.

ACKNOWLEDGMENTS

This work is partially supported by NSF under the grant CNS-1460864. Any opinions, findings, and conclusions or recommendations expressed in this material are those author(s) and do not necessarily reflect the views of NSF.

I would like to thank Dr. Yuan for her guidance support throughout my research experience. Thank you to Nawaf Aljohani for his contribution of his code.

REFERENCES

University of California, Irvine. (1999). KDD Cup 1999 Data. *The UCI KDD Archive Information and Computer Science.* http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Yang, H., Li, T., Hu, X., Wang, F., & Zou Y. (2014). A Survey of Artificial Immune System Based Intrusion Detection. *The Scientific World Journal, vol. 2014* (Article ID 156790), 11 pages. http://dx.doi.org/10.1155/2014/156790

Tavallaee, M., Ebrahim, B., Lu, W., Ghorbani, A. A. (2009). "A Detailed Analysis of the KDD CUP 99 Dataset". *2009 IEEE Symposium on Computational Intelligence in Security and Defense Applications, ISBN: 978-1-4244-3764-1*