# Deep Learning for Prioritizing and Responding to Intrusion Detection Alerts

Steven McElwee[1], Jeffrey Heaton, Ph.D.[1], James Fraley, Ph.D.[1], James Cannady, Ph.D.[2]
[1]College of Engineering and Computing Engineering
Nova Southeastern University
Fort Lauderdale, FL 33314
sm2752,jh2435,jf1280@mynsu.nova.edu
[2]Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, GA 30332
cannady@gatech.edu

***Abstract*—Network intrusion detection systems are widely deployed to detect cyberattacks against computer networks. These systems generate large numbers of security alerts that require manual review by security analysts to determine the appropriate courses of action required. The review of these security alerts is time consuming and can cause fatigue for security analysts, especially during long work shifts. This paper reviews a case study of the application of machine learning to the initial triage of security alerts to help reduce the manual burden placed on Department of Defense (DOD) cyber defense security analysts. This study implemented a Federated Analysis Security Triage Tool prototype. The FASTT prototype attempted to highlight, summarize and categorize tens of thousands of daily alerts/events. The prototype integrated a number of tools including TensorFlow deep neural network classifier, Elasticsearch and Kibana to provide an alternative approach for cyber defense analysts. FASTT provides a quick way to perform security alert reviews, speed up analyst queries/response, and to semi-automate threat intelligence reporting. The results of this study were evaluated for accuracy and usability. Results demonstrated that the accuracy of a deep neural network classifier was very high, as it was able to determine the heuristics that the cyber defense security analysts used in their review. Demonstrations and interviews with the security analysts showed that the prototype was able to quickly categorize security alerts into meaningful categories, provide fast query of the alerts, and save time in generating reports. Lastly, FASTT was able to pull the necessary information to generate Indications of Compromise (IOC) in a STIX/TAXII format that could be shared across DOD and US Government.**

***Index Terms*—Intrusion defense, security analysis, deep learning, neural network**

## I. Introduction

ONE of the significant problems with network intrusion detection that remains for both practitioners and researchers is the volume of alerts that requires human analyst review. With the continually growing volume of attacks and changing tactics of adversaries, the volume of security events places a burden on individuals who must review the events to determine if any are significant and require incident response activity. Further, this problem places a burden on organizations to provide adequate staffing levels to keep up with the large volumes of alerts on a daily basis.

In addition to the human problem of manually reviewing intrusion alerts, many traditional intrusion detection systems (IDS) and security incident and event management (SIEM) systems are inadequate for analysts to quickly identify incidents, pivot through indicators, and retrieve appropriate data. This is because these systems are heterogenous by design and were often not designed to work together and share information for fast retrieval of data. SIEM technology also does not have automated out-of-the-box or tailored workflows and processes to support security analysts.

This problem is especially important in military and government networks, where time and information is critical. This is the result of the large size, complexity and distributed nature of these networks as well as the broad attack surface these networks provide to adversaries.

This paper reports on the initial results of a Federated Analysis Security Triage Tool (FASTT) prototype that addresses the problem of the volume of IDS alerts requiring manual review by security analysts. FASTT was designed to address the challenges of dealing with large volumes of IDS alerts on military and government networks. FASTT uses a TensorFlow Deep Neural Network (DNN) classifier to automatically categorize IDS alerts and determine which type of security analyst should review the alerts, which can be automatically reported to interested parties, and which can be ignored. In addition, FASTT uses an Elasticsearch indexed data store to speed the retrieval of IDS alerts and a Kibana user interface to allow flexible display of visualizations and dashboards that can be tailored to meet the security analysts' workflow.

This remainder of this paper is structured as follows. First, it reviews the background and previous work related to intrusion detection and triage of IDS alerts. Next, it reviews the technical implementation of FASTT. Subsequently, this paper discusses the results of the initial prototype, and finally, concludes with opportunities for future research.

978-1-5386-0595-0/17/$31.00 ©2017 IEEE

## II. Background and Previous Work

FASTT was built upon previous research in intrusion detection, machine learning, and deep neural networks. This section introduces the concepts of intrusion detection and related previous research. Next, it reviews the background of deep neural networks and supporting literature.

### A. Intrusion Detection

Intrusion detection was pioneered by Denning and Neumann and focused on detecting abnormal patterns of system use for detecting security violations [1], [2]. Research in intrusion detection continued with the development of network-based intrusion detection [3]. Since then, network intrusion detection systems have extended beyond research and have been widely deployed in operational networks.

Intrusion detection has generally been categorized as either signature-based or anomaly-based. Signature-based network IDS relies upon matching patterns in network traffic that represent previously identified attack traffic. [4] As an example, SNORT is a commonly-used signature-based IDS [5]. An anomaly-based IDS determines what is normal network traffic and alerts when abnormal traffic is identified. Bro is an IDS that combines features of both signature-based and anomaly-based intrusion detection. Commercial IDS products such as McAfee Network Security Platform (NSP) combine various aspects of intrusion detection including special detection of policy violations and malware.

Regardless of the type of IDS utilized, these systems detect a wide variety of legitimate security events, but are prone to generating a high number of informational events and false positives that require security analyst review as well as alert triage to determine what initial actions to take. Previous experimental research found that analysts initially detect security events with a high degree of accuracy, but near the end of their watch, their accuracy erodes, especially if the analysis is routine and lacks complexity and diversity [6]. The proposed solution to addressing analyst fatigue was to focus on work scheduling and monitoring analysts' need of rest.

To deal with the volume of alerts, previous research has focused on clustering IDS alerts [7]. By identifying representative clusters that have commonly associated root causes, security analysts could label clusters of alerts and thereby account for approximately 90% of alerts with only a few root causes, thus reducing the workload of analysis. This study builds upon this approach by applying a deep neural network classifier for supervised categorization of alerts. By categorizing alerts, they can be assigned to the appropriate level of analyst, automatically reported, or appropriately disregarded, before requiring a security analyst's manual review effort.

Figure 1 shows where the FASTT prototype fits in the life cycle of a security alert. The alert is generated by the IDS, FASTT determines the appropriate triage category, and the alert is provided to the appropriate security analyst for action. Other triage results include automatic reporting or disregarding the alert, if the alert is informational and requires no action.

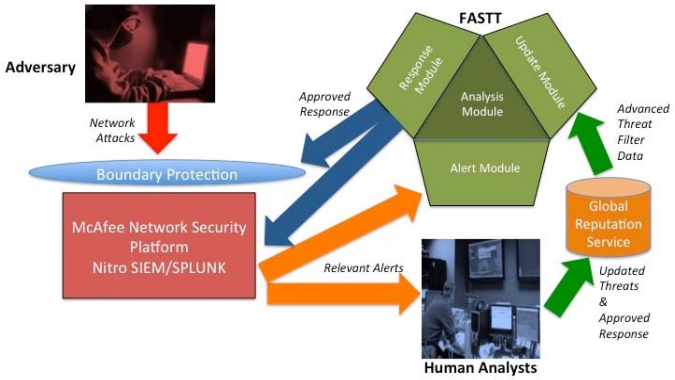


Fig. 1. FASTT

### B. Deep Learning

Artificial neural networks have been an area of active research since 1943 [8]. There was rapid progress over the next four decades, but starting in the early 1990's research slowed as the limitations of relatively simple designs prevented the algorithms from effectively being applied to more complex problems.

However, research conducted by Hinton [9] and Bengio [10] renewed interest in the potential for artificial neural networks. Unlike early simple networks a deep neural network utilizes multiple layers of hidden neurons, with each layer using the output from the preceding layer to provide increased abstraction of the problem space. As shown in Figure 2, the shallow network extracts the first level concepts and feeds those concepts into another shallow network for additional feature extraction. As an example, if an image is used as input for a deep learning network, the first layers of the network may extract features such as edges. The next layer may start extracting primitive shapes (circles, squares, etc.). Ascending the layers, more complex and abstract concepts and features are extracted from the input data.

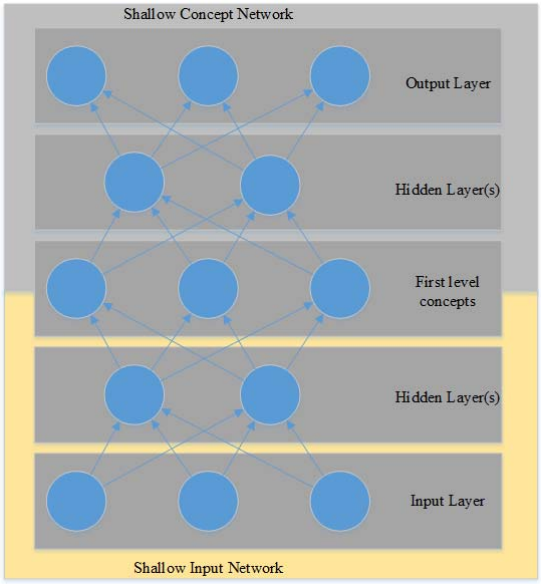


Fig. 2. Deep Neural Network

## III. Technical Implementation

This study implemented the FASTT prototype and was tested in a lab environment. A diagram of the FASTT prototype is shown in Figure 3. It consists of an IDS and a data pipeline that retrieves data from the IDS and classifies it using the DNN classifier. The results of the classification are stored in an indexed data store and are presented to security analysts using a visualization and dashboard engine. The remainder of this section elaborates on these components as well as on the data set used for evaluation.

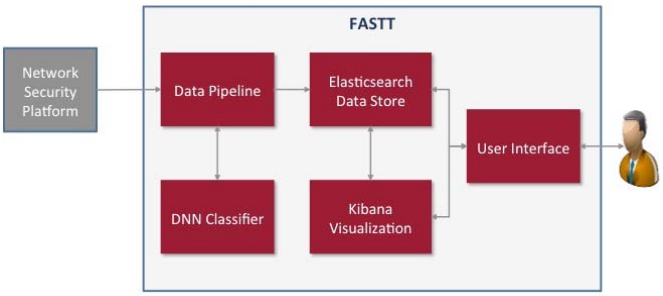


Fig. 3. FASTT Prototype

### A. Intrusion Detection System

FASTT was developed using the McAfee Network Security Platform (NSP). This system is primarily a network-based IDS, but it also integrates security alerts from other systems. Integration with NSP allows access to events in the event database and allows access to events using SQL queries, thus simplifying data retrieval for the FASTT prototype.

### B. Data Pipeline

The data pipeline component was designed to retrieve data on an hourly basis from the NSP IDS. Because of the volume of data in the NSP alert/event database, the data pipeline used several optimization strategies that worked best with the existing database indexes and integrity keys. Using caching of commonly used data, the data pipeline created a flat, single-record structure for each alert and stored it in a Python Pandas DataFrame object.

After retrieval of the data from NSP and construction of the DataFrame object, the data pipeline component called the DNN classifier to obtain the correct label for the record, which it assigned to the record in the DataFrame. Next, it used the JSON interface of the data store to load the DataFrame for subsequent review by security analysts using the user interface.

### C. Training and Evaluation Data

This study was based on an NSP IDS that was deployed in a lab environment and was used to monitor red-team/blue-team training exercises. As a result, there were a variety of IDS alerts that were available for training and evaluation of FASTT.

Training data was created by retrieving alert data from representative categories and subcategories of alerts from NSP. Security analysts provided a general heuristic approach for labeling the training data, which was then automated to create a large training data set.

Evaluation data was created using the same approach, but without the labels, and was applied directly to the DNN classifier to evaluate the accuracy. Additionally, data was pulled in hourly batches from the full NSP IDS and classified using the DNN. The results of the data pipeline classification were reviewed by human analysts to determine if the labeling was appropriate and consistent with security analysis practices.

### D. Deep Neural Network

The DNN was implemented using TensorFlow 1.0. The DNN was presented with a Pandas DataFrame, which it then classified and applied the appropriate label. The DNN returned the labeled DataFrame to the data pipeline for subsequent storage.

The neural networks that were investigated by this research had hidden layer counts between one and five. Neuron counts for the first hidden layer were three times the number of input neurons needed for the feature vector. The neuron counts for subsequent hidden layers were half of the preceding layer. Using this methodology a 50 input neural network would contain hidden counts of [150, 75, 37, 18] for a four hidden layer network. This is a similar methodology as demonstrated by [11]. The output layer made use of a softmax function and six output neurons, which corresponded to the six labels in the training data.

All hidden layers made use of the rectified linear unit (ReLU) transfer function [12]. The adaptive moment estimation (Adam) was used to train the neural network [13]. Weight initialization was accomplished using the Xavier algorithm [14]. The feature vector was optimized using a feature ranking algorithm that was developed specifically for TensorFlow [15].

### E. Data Store

The FASTT prototype implemented the data store using Elasticsearch, which met two main design criteria. First, it provided a means to store denormalized data using index structures, making it very fast for data retrieval. Second, it was the underlying data store used by the Kibana visualization engine, which will be described in the next section. Elasticsearch provides a JSON interface that was used by the data pipeline to store data and by the Kibana visualization engine to retrieve it and present it to the user.

### F. User Interface and Visualizations

Figure 4 shows a sample screen shot of the FASTT user interface. The user interface was implemented using a variety of components. The main user interface was the Kibana visualization engine. It provided flexible, customizable visualizations and dashboards that were tailored to meet the needs of security analysts. Additional functionality was needed to clearly delineate between the classification results, so tab navigation was implemented with one tab per label to


Fig. 4. FASTT User Interface

allow analysts to quickly find the events of interest to them in their role. To tie the tab navigation and links for automated report generation, Flask was used to integrate the components and provide an iframe element for displaying the Kibana dashboard.

### G. Automated Report Generation

One of the requirements for FASTT was to reduce the amount of time security analysts spent on manually generating reports. After they reviewed events and decided about the initial triage action, many of the events required that the analyst create reports to send to other stakeholders. Using the Elasticsearch data store, this study implemented a reporting package that allows security analysts to click on a link in the user interface to automatically create a STIX/TAXII or MS Word report that could be emailed based on the rule for the particular security alert.

## IV. Results

FASTT was evaluated in two ways. First, it was evaluated in terms of classification accuracy to determine if the DNN classifier was accurately identifying security alerts for each triage category. Second, it was evaluated by security analysts to evaluate the usability and the value provided in the automation.

### A. Classification Accuracy

The neural network was trained to minimize classification multiple log loss, as measured by scikit-learn. Multiple log loss is an extension of ordinary log loss for multi-class classification. A log loss of zero is perfect, whereas a bad log loss scores can approach infinity. Because the data used for FASTT included 6 labels, a log loss of –log(1/6), or 1.8 would indicate no predictive power, under a theoretical uniform distribution of labels.

The neural network achieved a log loss below 0.0001. This resulted in a very robust classification accuracy of 98%, which was calculated using out of sample data obtained from a 5-fold cross validation. The final neural network, created by this research, was the neural network that corresponded to the fold that achieved the best accuracy.

Further research will measure the neural networks accuracy with wider, more diverse data sources. This may require some modification to the feature vector of the neural network as it is optimized to handle the most diverse situations possible. Multiple neural networks may also be employed to handle more diverse situations.

### B. Usability

To evaluate usability, FASTT was demonstrated for security analysts who were accustomed to working with the NSP IDS and who had experience with the data that was utilized from the lab environment. Three iterations of configurations of the user interface were developed. The user interface was able to quickly lookup data as a result of the indexing.

## V. Conclusions and Future Work

This study made three contributions to the field of intrusion detection. First, it found that there is a significant time-saving value in applying machine learning to the initial triage of security alerts received from an IDS – some results suggest a 70% increase in productivity for elimination of previously known alerts/events. The initial triage of alerts is routine and is well suited to automation using a DNN classifier. Based on the results, it is reasonable to generalize this approach to a variety of security event monitoring systems that produce large volumes of alerts.

Second, the FASTT prototype shows that transferring security alerts to an intermediate system for indexing and visualization is valuable to security analysts. This approach allows the security analysts to quickly narrow the scope of their review and prioritize alerts. Analysts have provided positive feedback that packaging the security information in bundles saves hours of research on daily basis.

Third, the FASTT prototype demonstrates the value of semi-automated report generation for threat information sharing. This approach allows the security analyst to determine which alerts require reporting and, with a click of a single link, automates the collection of data.

Based on the results of the prototype, two areas of future research should be considered. First, the DNN classifier was trained using initial labeled data from security analysts. Over time, security threats change and new alerts are created. As a result, the DNN classifier results will be improved by retraining based on analyst feedback. For the FASTT prototype, user feedback can be incorporated into the user interface to allow analysts to change the triage category for security alerts and allow the DNN classifier to use this feedback during the next model training cycle. Second, clustering following the initial triage classification will allow security analysts to see groupings of events that they may not have detected and that are not based on heuristic groupings. This will speed up the human analyst review and will complement the automated triage of events.

## REFERENCES


[1] D.E. Denning and P.G. Neumann, "Requirements and model for IDES—a real-time intrusion detection expert system," *Document A005*, SRI International, 333, 1985.

[2] D.E. Denning, "An intrusion-detection model," *IEEE Trans. Software Eng*., vol. 2, pp. 222–232, 1987.

[3] L.T. Heberlein, G.V. Dias, K.N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A network security monitor," *IEEE Comput. Soc. Symp. on Research in Security and Privacy*, pp. 296–304, 1990.

[4] B. Mukherjee, L.T. Heberlein, and K.N. Levitt, K.N. "Network intrusion detection," *IEEE Network*, vol. 8, no. 3, pp. 26–41, 1994.

[5] *SNORT Users Manual*, 2.9.9, The SNORT Project, Nov. 2016. Available: https://www.snort.org/documents/1

[6] B.D. Sawyer, V.S. Finomore, G.J. Funke, V.F. Mancuso, M.E. Funke, G. Matthews, and J.S. Warm, "Cyber vigilance: Effects of signal probability and event rate," *Proc. of the Human Factors and Ergonomics Soc. Annual Meeting*, vol. 58, no. 1, pp. 1771–1775, 2014.

[7] K. Julisch, "Clustering intrusion detection alarms to support root cause analysis," *ACM Trans. Inf. Syst. Secur.* vol. 6, no. 4, pp. 443–471, Nov. 2003.

[8] W.S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics,* vol. 5, no. 4, pp. 115–133, 1943.

[9] G. Hinton, S. Osindero, and Y-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computing,* vol. 18, no. 7, pp. 1527–1554, 2006.

[10] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, 2009.

[11] S. Ioffe and C. Szegedy, C. "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *Int. Conf. on Machine Learning*, pp. 448–456, Jun. 2015.

[12] V. Nair and G.E. Hinton, "Rectified linear units improve restricted boltzmann machines," *Proc. 27th Int. Conf. Machine Learning,* pp. 807–814, 2010.

[13] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*. 2014.

[14] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Aistats,* vol. 9, pp. 249–256, May 2010.

[15] J. Heaton, S. McElwee, J. Cannady, and J. Fraley, "Early stabilizing feature importance for TensorFlow deep neural networks," *Int. Joint Conf. on Neural Networks,* pp. 4618–4624, May 2017.