Bio-Inspired, host-based firewall

James Ballard, Kevin Hamilton, and Jay Chow

{khamil36, jchow6, jballa12}@jhu.edu

Advisor: Dr Lanier Watkins, JHUISI/APL

Abstract - This research explores the viability of using biologically-inspired design decisions for experimental firewalls. Most firewall designs in use today are signature-based which tend to struggle to protect against never seen before attacks and against encrypted traffic. By applying cellular biology concepts such as endocytosis, ligand-gated ion channels and active sites, we created a firewall prototype capable of giving a binary classification of benign or malicious to ingested network traffic. Using PCAPS (packet captures) generated by Stratosphere Lab, we extracted data about the streams into CSVs. With those CSVs we tested and evaluated both supervised and unsupervised machine learning algorithms to determine which may be best situated for the bio-inspired firewall.

1. Introduction

A firewall is an important security technology that is widely deployed in networks, restricting network traffic from one side of the network to the other side. Firewalls can be deployed anywhere, and are most commonly seen separating trusted and untrusted components of a network. Firewalls are also useful in differentiating networks within a trusted network. It can create a clear distinction between various divisions in an organization. The main functionalities include filtering data, redirecting traffic, and protecting against network attacks.

For quite some time firewalls have represented the first line of defense for most client-side consumer networks. As network-based attacks have become more intricate, firewalls are also required to evolve to meet these new challenges. In an attempt to find a working analogy to better guide this development, we can look no further than our own cell biology. Because it is pivotal for cell membranes to identify beneficial and benign substances and allow them to pass through, into the cell; we thought that attempting to mimic that behavior would be a potential solution for firewall deployment. After all cells being able to successfully do this for tens of millennia is what allowed for human evolution to come to be in the first place.

Our team sees many applications of biological-inspired cybersecurity in the world today. When cells face an invading virus or hostile bacteria, they protect themselves by inserting the wrong amino acid into new proteins to defend themselves against damage. These regulated errors are a novel non-genetic mechanism by which cells can make proteins more resistant to attack when stressed.

Biological-inspired cybersecurity is still in its nascent phase and there is still so much research and development to do in this field. We see a natural mapping between firewalls and cell membranes in terms of both allowing and blocking network packets and molecules respectively. There are natural biological mechanisms that can be applied to cybersecurity to increase the cybersecurity defense of a host-based system. Also, our team saw how solving this problem could scale massively up in a distributed computing system that mimics a human body.

In an attempt to push firewall design in a new direction we looked into endocytosis, active sites, and ligand-gated ion channels for design inspiration. We applied and evaluated both supervised and unsupervised machine learning algorithms to be the brains of the classification engine. We created a prototype that can be used with various different models even ones outside the scope of the research posed in this paper. To conclude, we tested and evaluated our prototype on test data.

1.1. Look into Modern Firewalls

Cisco's Next-Generation Firewall(NGFW) is an application-layer firewall that utilizes access control lists and snort to inspect network traffic up to the application layer. According to Cisco's Talos that authors the official snort subscriber rule set, snort is capable of real-time traffic analysis and packet logging on networks. It can perform protocol analysis, do content searching and matching, and can be used to detect attacks and probes such as port scans, buffer overflows, CGI attacks, SMB probes and OS fingerprinting attempts. Also, snort can be used as a packet sniffer like tcpdump, a packet logger that is used for network traffic debugging. In this mode, snort can serve as a network file logging device capturing files in realtime from network traffic or a network intrusion prevention system. Cisco's NGFW could be managed on the same premise by the Cisco Firepower Management Center or remotely via Cisco Cloud Defense Orchestrator. In the context of modern firewall technology, Cisco's main competitors are Palo Alto and Fortinet. [1]

From our team's research, findings and technical insights from Kevin Klous, Technical Leader, CX

Americas in Security at Cisco, we learned the major challenges of NGFW out in the field include performance, encryption and zero day threats, and the lack of machine learning due to high false positives. In terms of performance, NGFWs have more work inspecting traffic compared to stateless routers that forward traffic in hardware. With cryptography involved, there is a challenge to keep up. For encryption, TLS 1.3 is making internet traffic more difficult to decrypt for inspection. Lastly, since most modern Intrusion Detection Systems and Intrusion Prevention Systems are signature based, zero-day threats are very difficult to detect. More importantly, anomaly detection via machine learning is in its nascent phase because most machine learning algorithms today still have high false positive rates that make the algorithms unreliable. In current computer networks, there is a need to detect anomalies with a high effective rate with lower false positive rates, combined with signatures to increase the cyber defense of modern application layer firewalls.

2. Problem Definition

Cisco's next-gen firewall is not much different from the rest of the popular signature based firewalls. These firewalls certainly have their advantages such as the ability to map a specific threat to a malware signature. That is particularly useful for threats that are well known and have been researched by the security community at large. Those firewalls struggle against attacks that have not been seen before. Signature based firewalls also struggle against encrypted traffic as the only thing that can be inspected is packet headers. With these limitations in mind we decided to make a system using biomimicry, the process of modeling a system off of biological entities.

2.1. Project Scope

Our objective was to create a prototype firewall capable of giving a binary determination of benign or malicious network traffic after ingesting a certain number of tcp packets. Since creating a user interface was out of this project's scope, the firewall must be started on the command line. We also only perform analysis on TCP packets with nothing being done with UDP. The firewall is only able to block IPs on linux machines.

2.2. Limitations

We would be negligent if we did not cover in detail the limitations we found in our approach through the course of this research. We tried to cast a wide net in terms of classification paradigms so we decided to test supervised learning and unsupervised learning but with the time we had we only protyped one algorithm from each. This is a start but for more comprehensive testing it would strengthen the research to prototype more to allow for a better recommendation of the algorithm best suited for this work. We did explore trying to add a third paradigm, being fuzzy logic, but were unable to get it in a testable state at the time of writing this paper. Again, had we been able to test and evaluate fuzzy logic it would strengthen the conclusion of this paper. A limitation of the firewall is it's only as effective as the training data. The way we sourced our dataset for the test and evaluation has its own flaws that are discussed later in this paper. When you use machine learning algorithms to classify the algorithms themselves, they often act like black boxes with little information being relayed back about why a certain classification was labeled. This is a limitation when trying to understand network traffic more than just the classification. It doesn't differentiate between a botnet or a malicious pdf. This is a major limitation for security analysts who would like to implement this type of firewall. A binary classification does no favors to the start of an investigation by an analyst whereas signature firewalls would be an advantage for post-classification investigations. One of the major challenges we had with this project was balancing the biomimicry versus traditional methods. The blocking mechanism also has no means of being undone so the IP blocks are more or less permanent unless manually changed in the system's IP tables.

2.3. Research Contribution

The first research contribution is the dataset we collected and compiled into csv files that are in our project github (see Appendix below). We also have our dataset in arff file format so it can be used with weka, a machine learning tool. Our second contribution is the firewall prototype itself which is also available on the project github. The firewall allows for the model it implements to be restrained so we encourage the continued use of the prototype.

3. Literature Review

The available literature on bio-inspired firewalls is quite sparse although we did find a few papers published which had similar goals to ours. The first paper we found on the subject was by Sharma [2] who explored the use of fuzzy logic after being inspired by plant cell walls. Saleous [3] evaluated the use of neural networks in firewall packet filtering. The disadvantages swayed us away from using this approach because the performance hit was too great. Neural networks also suffer from a similar problem as our algorithms, however, as both act as a black box. [4] Prasse showed that over a large training set neural networks would outperform random forests. It was compelling to try neural networks given this result but we felt within the scope of our project it would be best to use random forest for speed. Muehlstein [5] did work on extracting features on TCP headers that was largely influential on the features that we decided to use on this project. E. B. Beigi [6] also used timing type features but also broke their dataset up by the size of data which was out of our scope.

4. Biology

In the next three sections we define biology concepts we were inspired by and in the final section

discuss how we combined all of the concepts in our solution.

4.1. Endocytosis

Endocytosis is the process of actively transporting molecules into the cell by engulfing it with its membrane into a vacuole. [7] The immune system's purpose is to rid the body of any pathogens or foreign particulates that can cause disease. These molecules can be detrimental to the body so it is imperative that the invading pathogen is cleared out quickly. [7] The cell will then transport the bacteria to the lysosome, where they fuse together forming a phagolysosome. Within the phagolysosome are various factors that destroy and break apart the microbe.[7]

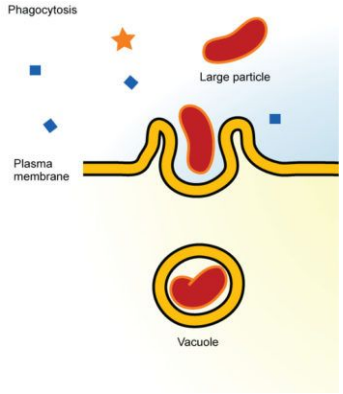

Fig. 1. Example of Endocytosis [8]

4.2. Ligand-Gated Ion Channels

Ligand-gated ion channels are large, multisubunit (4 or 5 subunits) receptors that form a membrane ion channel that, when open, allows the passage of Na+, K+, Ca++, or Cl−. Once the receptor/channel complex is activated, the membrane potential may become depolarized or hyperpolarized, depending on the direction of the ion flow and the ion involved. [9]

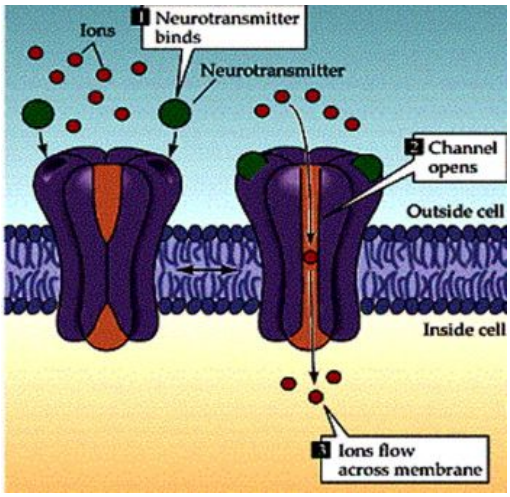

Fig. 2. Example of Ligand-Gated Ion Channels [9]

4.3. Active Sites

Enzymes are proteins which drastically increase the speed of chemical reactions by lowering their activation energy. Enzymes do this by interacting with the chemical reactants – called substrates – in ways that make them more likely to react. [10]. Specifically, we saw a lock and key theory equivalent to our machine learning implementation. The lock-and-key model of enzyme active sites postulates that enzyme active sites are perfectly shaped to receive substrates and "pop" them into their new forms. [10].

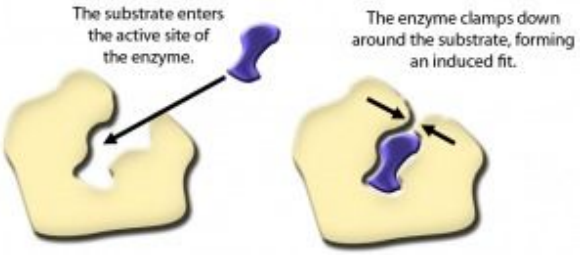

Fig. 3. Example of Substrate entering active site [10]

4.4. Application Of Biology

We viewed the vacuole mechanism used in endocytosis as our guide for creating our packet threshold. Just like the cell behaves based on the characteristics of the content contained inside this vacuole, our machine learning can determine if the content will be allowed in or be expelled. In the case of our firewall the IP will be blocked permanently. Our determination mechanism is analogous to the active sites lock and key theory that substrates only react if they fit perfectly into the enzyme. If our packet stream does not fit our model then it gets the classification of malicious. The channel is controlled similarly to the ligand-gated ion channels. The channel will stay open as long as the packet stream from the specific IP is found to be normal traffic. If that ever changes classification the channel will be closed and the connection will cease.

5. Technical Solutions

To fully integrate the description of the biology above with our technical design we will discuss the parallel analogies here. The firewall that we have designed is meant to represent the cell wall as has been described above. The feature selection and analysis, which will be described in detail later, represents the protein folding that is used to identify and classify materials that are external to the cell. The training of machine learning models based on the training data that we've acquired can be likened to the external protein markers that exist on the exterior of the cell. These protein sites identify what types of proteins that will be allowed to pass into a shell. All of this, albeit to a limited extent, occurs in the prototype we designed to simulate this. To cast as wide a net as possible, we build a detection engine

using both a supervised and unsupervised machine learning algorithm. This approach will be significant more useful than the more traditional signature and rules frameworks of traditional firewalls and intrusion detection systems. While it is almost trivial to change a program to bypass signature detection, we believe that the 'protein marker' of a malicious program is significantly more difficult[11]. This generalization of what attacks could look like also give the system the ability to detect and respond to zero-day threats.

5.1. Supervised Learning (Random Forest)

The supervised learning algorithm we chose is random forest, which is a descendant of the simpler decision tree algorithm. A decision tree is a classification algorithm that trains using provided features to make branching decisions that parse the data into various subgroups. The leaf nodes on these trees represent the classification that is made after following that particular set of branches. The goal is to reduce the entropy at every branch which is defined as:

$$entropy = - p_1 \log(p_1) - p_2 \log(p_2) - \cdots$$

With p being the proportion of each class in the data. [12] The intermediate step between a decision tree and the random forest algorithm is the random tree. The random tree is a decision tree as described above that has been augmented by the addition of a random vector. This leaves the definition of a random forest which Breiman defined as: "A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(x, \Theta k), k = 1,...\}$ where the $\{\Theta k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x."[13] The classification output of this random forest is the class that gets the most votes from the trees in the forest. The added flexibility of bagging these trees together strengthens the classification capabilities. In our design, we trained the classifier on the dataset to have 100 trees in the forest. Being that this is supervised learning, the algorithm was trained by being given feature vectors and their corresponding labels and then placed inside the prototype.

5.2. Unsupervised Learning (K-means Clustering)

The unsupervised technique that we selected was K-means clustering. Clustering is traditionally attempting to find trends in data where there is no known ground truth. The K-means algorithm is defined as: "Let $X = \{x_i\}$, $i = 1, \ldots, n$ be the set of n d-dimensional points to be clustered into a set of K clusters, $C = \{c_k; k = 1,\ldots,K\}$. K-means algorithm finds a partition such that the squared error between the empirical mean of a cluster and the points in the cluster is minimized. Let μ_k be the mean of cluster c_k. The squared error between μ_k and the points in cluster c_k is defined as"[14]

$$J(c_k) = \sum_{x_i \in c_k} ||x_i - \mu_k||^2.$$

The clusters that are produced by the algorithm are meant to represent meaningful partitions in the data. In our research, we used this algorithm to see what if any meaningful partitions could be made between normal traffic and malware. Using a k of 2 to cluster the data with the labels removed the clusters almost exclusively contained members of a single class. This very positive result led us to take these clusters and attempt to use them for classification. This was done by inserting data into the trained K-means algorithm and getting the cluster assignment value and using that as an additional feature to train a supervised learning method, in this case a Logistic Regression. Ideally, the addition of this cluster number will make it a much simpler classification and improve accuracy, however, this may possibly lead to an overfitting problem if the cluster assignment feature is given too much weight.

5.3. Dataset

5.3.1 Data Gathering

The first thing that needs to be done when deciding to utilize machine learning to approach a research problem is what data are you using and how are you going to source it. The gold standard for datasets in network security research has been the KDD99 data, but we identified a few issues with using this data.[15] The primary reason this dataset was not used in this research was because of its age, computer networks have changed quite significantly in the past 20 years. Another issue was that it didn't offer the source packet capture for the dataset to elicit extraction of additional features. The final major issue was that attacks that exist today don't neatly fall into the four attack categories that are used in that dataset. The source that we used for the normal training data was Stratosphere Lab's public data set, which had two-hour packet captures of normal network behavior (i.e. browsing to sites in the Alexa top 1000).[16] The data for the malware traffic was sourced from malware-traffic-analysis.net, a blog hosted by a security researcher that does network analysis on various malware. He posts packet captures of various malware executions, but the issue for our research is that the flows within these captures are unlabeled. This means we are certain that some amount of these captures is malicious, but without analyzing each connection in these lengthy captures. The type of malware traces that were selected for our training data were different flavors of banking trojans. These were selected because banking trojans

represent a class of malware that have cost consumers millions of dollars and are notorious for being a type of malware that permutes often.[17]

To better support this avenue of research it would be beneficial if more authors made the datasets that they produce for their research publicly available, so other researchers aren't left needing to reinvent the wheel every time machine learning is being applied to a problem.

5.3.2 Malware Overview

Because modern malware is often rather quite complicated a complete malware is often not completely contained inside an initial payload and must dial out to fetch addition files to escalate the malware through the systems. Also in the case of the banking trojans there are often command and control servers that also regulate the data exfiltration being done by the malware. The three pieces of malware that were analyzed were hancitor, emotet, and trickshot. Because these malware operate quite similarly for the sake of simplicity the infection vector of only emotet will be explained. If we look at the diagram in figure 4 you can see there are two areas of interest for the network based analysis that we will be conducting. First is the initial download of the malware. This is facilitated by the execution of a malicious document macro, and is where the network trace first starts. After compromising the host the malware must then contact and receive directives from a command and control server. Those exposures to the network are what will be analyzed and by blocking either from occurring would essentially render the virus inert giving a human time to intervene and remove the compromised files from the system. Before this removal the trojan will continue executing and data gathering on the host system, but stopping the malware at a process level is beyond the scope of this investigation.

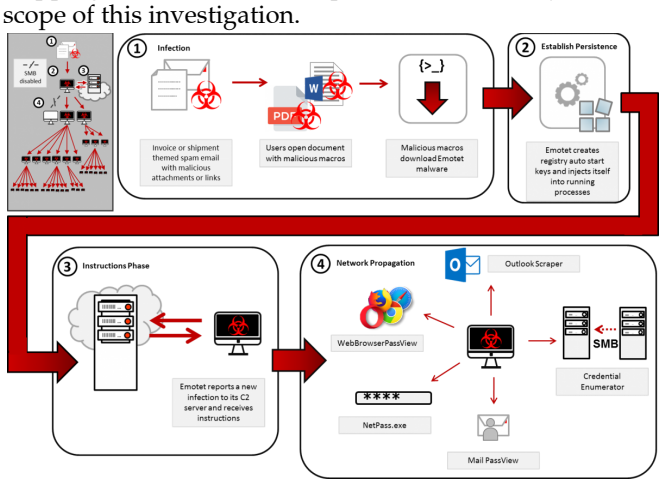

Figure 4. A trace of the execution of the emotet virus[18]

5.3.2 Feature Selection and Extraction

When dealing with network data it is tempting to do analysis of cleartext HTTP to add to the feature space. Approaches using features derived from these features have been able to form quite well. However, as a larger and larger amount of Internet traffic is being done through TLS encrypted channels this will no longer be a useful set of features. That is why this research used a feature set that was payload agnostic and will always be communicated in the clear. That feature set is 29 features that are broken down into three categories:

Size Features

- Number of Forward Packets- packets in a flow from the host machine outwards
- Number of Backward Packets
- Total Number of Packets
- Forward Total Bytes
- Backward Total Bytes
- Minimum Packet Size
- Maximum Packet Size
- Mean Packet Size
- Forward Minimum Packet Size
- Forward Maximum Packet Size
- Forward Mean Packet Size
- Standard Deviation Forward Packet Size
- Backward Minimum Packet Size
- Backward Maximum Packet Size
- Backward Mean Packet Size
- Standard Deviation Backward Packet Size
- First Packet Size- The size of the first packet after the completion of the TCP handshake

Timing Features

- Mean forward inter arrival time difference- Average value of the time in between two forward packets being sent
- Max forward inter arrival time difference
- Min forward inter arrival time difference
- Standard Deviation forward inter arrival time difference

o Mean backward inter arrival time difference

o Max backward inter arrival time difference

o Min backward inter arrival time difference

o Standard Deviation backward inter arrival time difference

TCP/IP Features

o Source Port

o Destination Port

o Average Backward TTL value- average TTL of packets when they arrive at the monitored host

o TCP window size

A reason that IP addresses were omitted from the feature list is because much of the malware that will be used in the training and testing of the models will be running in simulated environments, so it is likely that a significant amount of malicious addresses would be in private address spaces. This is not reflective of a live environment and we did not want to bias the models in this fashion. This is not the same for ports which on the whole are much more application specific.

The actual feature extraction was facilitated by filtering out all of the non-TCP traffic and splitting each flow into its own file and calculating these statistics on them. The total training dataset consists of 4247 connections with 400 of those being malicious. We used an unbalanced training set because this better reflected the actual proportion of normal network traffic to malicious traffic. The issue of labeling for the malicious flows was done by making the assumption that all flows in the malware captures were malicious. The argument being that the overwhelming majority of the flows are truly malicious, so any statistical noise caused by the inclusion of a few normal flows should not cause much, if any, change.

5.4 Prototype

This will be a walkthrough of how the prototype functions with the biological justification for its inclusion. When the firewall receives a packet, it will assign an ID based on the connection characteristics and is placed into a list with other packets of that flow. The fact that after a set number of packets the group is ingested is rooted in the functionality of endocytosis where there are a limited number of receptors on the cell surface that can facilitate the transport[17]. When these sites are full, the packets are then passed to the extractor.

The extractor is given the list of packets and extracts the feature statistics from those ostensibly measuring the protein folding pattern of this set of packets and sends it to the detector. The detector takes the feature vector and runs the classifier and returns a prediction. If this is a normal classification the packets are ingested and then nothing further happens when the packets are determined to be malicious there is a slight divergence from how the biology of the cell operates. Ideally, these packets would be sandboxed in a DMZ before being allowed in and purged when denied this is infeasible. It is infeasible because it is impossible to continue a connection long enough to get the data required for classification with applications receiving messages and being able to respond to them. Instead the surrogate process is to insert a blocking rule into an iptables firewall that prevents any further packets from coming in. The prototype will take a detection model as input so that it can run with either the supervised model or unsupervised.

6. Experimentation

There are two parameters that we will manipulate during the testing to examine what effect they have on the accuracy of the detection engine; those are the size of each ingestion load and the model being tested. There are also four testing sets that will be used to give varying scenarios to the prototype, these are: a normal packet capture generated from Stratosphere Labs, different from the one used for train but created the same way, a capture of a different banking trojan execution source also sourced from Malware Bytes, a network capture created using Cobalt Strike, a threat emulation software that is used to simulate different kinds of malware attacking a network, and a network capture generated from monitoring the activity of one of our researchers.

To conduct the experiments the test captures are reinjected into the network using tcpreplay. This software allows for a network capture to be replayed in real time, effectively simulating a live operation A difficulty in doing rigorous testing on this framework is that it is difficult to know the ground-truth of a single connection among thousands. This is especially the case for Cobalt Strike, which generates lots of normal traffic in addition to its attacks. That is why for the judgement of the models performance we will be focused on the three test sets that we are at least decently certain of the ground truth of all of the contained connections.

Model	Receptors	Packet Capture	TPR	FPR	TNR	FNR	Malicious Connections

Supervised	10	Normal (4073 flows)		0.01 %	99.99 %		4
Supervised	20	Normal		0.002 %	99.998 %		1
Supervised	30	Normal		0.002 %	99.998 %		1
Supervised	20	Home made (183 Flows)	N/A		100.00 %		0
Supervised	10	Attack (43 Flows)	50%	N/A	N/A	50 %	23
Supervised	20	Attack	30.40%	N/A	N/A	69.60 %	14
Unsupervised	20	Normal		0.002 %	99.998 %		1
Unsupervised	20	Cobalt Strike	N/A	N/A	N/A	N/A	221
Supervised	10	Cobalt Strike	N/A	N/A	N/A	N/A	922
Supervised	20	Cobalt Strike	N/A	N/A	N/A	N/A	221
Supervised	30	Cobalt Strike	N/A	N/A	N/A	N/A	220

But as far as the known test cases are concerned our prototype performed rather admirably. Because malicious connections account for a staggeringly small percentage of network traffic it is imperative to have very low false positive rates. This is clearly true in our case, as in the case of the normal test sets, the FPR comes in at .01%. The main issue with our prototype is that the true positive rate is quite low. The reason for this might be because of the problem explained earlier. The attack capture sourced from Malware Bytes might contain an amount of normal traffic that is actually being correctly identified as such by the prototype. Another result worth pointing out is that both of the models tended to have extremely similar performances, so in the future the supervised approach should be pursued since that method is much more straightforward. The amount of receptor sites had a more drastic impact on the detection rate than we would have originally thought. Though the lower amount of receptor sites increases the false positive rate, the increase is extremely small compared to the large increase in detection rate. The Cobalt Strike packet tests weren't scored because the ground truth value for those connections is unknown. But we conducted a rough estimation based on the knowledge that all of the traffic conducted over HTTPS was malicious and that other protocols were unknown. This yielded a detection rate of 77% which given the fact that Cobalt Strike malware was completely unrelated to that of the banking trojans the detector was trains on is quite good. This also lends some credence to the thought that there might be something that is inherently differentiable about malicious network traffic.

7. Conclusion

Although the experimentation itself was quite limited in scope, we were able to create a successful prototype that displayed there is something to be gained by taking a biologically guided approach. We were able to design a system that is able to detect and prevent the network execution of novel banking trojans. It is our belief that by expanding the profile of malicious applications the detection engine is trained for, the ability to prevent an even wider range of malicious activity is possible. This method however, is not a universal solution. In order for this solution to be useful, security researchers are required to constantly monitor malware development to see if a new class of malware needs to be added to the training data. Another point that needs to be considered is although this biological method provides defenders with another tool to make systems more secure it does present the same tool to adversaries. This will incentivise bad actors to take the same actions as synthetic biologists and design their malware such that they can effectively mimic the "protein folding" of normal web traffic rendering this method of detection moot. Even

though the FPR of the prototype is astronomically low, a real implementation would need to reduce this even further in order to be feasible. Assuming the Stratosphere Lab data represents an average amount of connections over 4 hours we can estimate that an average user would have around 8000 network flows over the course of an eight hour workday. This means that anywhere from 2-8 benign hosts could be blocked, potentially costing productivity. To get this solution to a point where it can be deployed on a live system, it would be ideal to further reduce the FPR by an order of magnitude to decrease disruption for users.

7.1. Future Work

An interesting expansion of this project would be to expand beyond cellular biology. For example, consider a tissue based model where the host-based firewalls act in coordination with each other. While we have not explored many details of this type of approach, it would be a natural expansion for this project moving forward. As mentioned in the limitations section, future work should include more machine learning algorithms, an expanded feature set, and an implementation of fuzzy logic.

We learned that fuzzy logic deals with situations that have a degree of certainty rather than a fixed truth value. For example, the truth value that a network packet is malicious can be 0.8 and the truth value that a network packet is non-malicious of good can be 0.5. Also, we learned that fuzzy logic can be best understood with set membership. With the same example of a malicious network packet x in a set of network packets passing through a firewall, it could be written as (network packet x, 0.8), with the first value in the 2-tuple indicating a candidate of inclusion and the second value in the 2-tuple indicating the degree of membership. Fuzzy sets contain ordered pairs and there are operations that can act on fuzzy sets include union, intersection and complement.

Fuzzy logic uses the fuzzy control which is used in a variety of machines and processes. For example, a fuzzy logic controller has application in a hot water heater by adjusting the heating element power according to both the amount of water used and the temperature. For future work, we could design a fuzzy logic controller by first defining the inputs and outputs for the fuzzy logic controller, second fuzzifying the inputs, third setting up fuzzy membership functions for the outputs, forth creating a fuzzy rule base and lastly defuzzifying the outputs. In the context of our research, we could utilize fuzzy logic to mimic the artificial intelligence or brains of the host-based firewall so it could make a better and smarter decision to block or allow a packet. The fuzzy logic also provides a more accurate representation of the softer kinds of decision making that is made at the cellular level.

The firewall would be more practical if we could attribute some confidence grade to the classification. In theory with a confidence threshold, you could create some third classification that would be a grayware classification. This would reduce the false positive rate. Other places of focus that improve the quality of life of users would be the ability to easily unblock an IP after it had been blocked and to create a user interface. It would also be useful to investigate ports and protocols being used by malicious applications and allowing the firewall to block these if they exhibit malicious behavior.

8. References

[1] "*Cisco Snort*". Accessed on: Nov. 28, 2019. [Online]. Available: https://talosintelligence.com/snort

[2] R. K. Sharma, B. Issac, and H. K. Kalita, "Intrusion detection and response system inspired by the defense mechanism of plants," vol. 7, pp. 52427–52439, 2019.

[3] "H. Saleous and Z. Trabelsi, "Enhancing Firewall Filter Performance Using Neural Networks," *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, Tangier, Morocco, 2019, pp. 1853-1859. doi: 10.1109/IWCMC.2019.8766576

[4] P. Prasse, L. Machlica, T. Pevny, J. Havelka, and T. Scheffer, "Malware Detection by Analysing Network Traffic with Neural Networks," 2017, pp. 205–210.

[5] J. Muehlstein *et al.*, "Analyzing HTTPS Traffic for a Robust Identification of Operating System, Browser and Application," Mar. 2016.

[6] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches." pp. 247–255, 2014.

[7] "*Endocytosis*". Accessed on: Nov. 24, 2019. [Online]. Available: https://biologydictionary.net/endocytosis/

[8] "*Cell Membranes*". Accessed on: Nov. 24, 2019. [Online]. Available :https://courses.lumenlearning.com/suny-wmopen-biology1/chapter/endocytosis-and-exocytosis/

[9] Pacheco, Mary A. "Receptor Types and Subtypes." pp. 1-5. 2017.

[10] "*Active Site*". Accessed on: Nov. 24, 2019. [Online]. Available: https://biologydictionary.net/active-site/

[11] B. B. Rad, M. Masrom, and S. Ibrahim, "Evolution of Computer Virus Concealment and Anti-Virus Techniques: A Short Survey," vol. 8, no. 1, p. 113, Jan. 2011.

[12] Ding, Lei "Data Science Basics and Decision Tree Models" Lecture. Johns Hopkins University September 16, 2019

[13] L. Breiman, "Random forests," vol. 45, no. 1, pp. 5–32, 2001.

[14 Jain, Anil K. "Data clustering: 50 years beyond K-means." Pattern recognition letters 31, no. 8 (2010): 651-666.

[15] Özgür, Atilla & Erdem, Hamit. (2016). A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. 10.7287/PEERJ.PREPRINTS.1954.

[16] Garcia, Sebastian. Malware Capture Facility Project. Retrieved from https://stratosphereips.org

[17] Marsh, Mark (2001). Endocytosis. Oxford University Press. p. vii

[18]"Emotet Malware: CISA." Emotet Malware | CISA. Accessed December 15, 2019. https://www.us-cert.gov/ncas/alerts/TA18-201A.

9. Appendix

https://github.com/jaychowjingjie/JHUISI-capstone-research-bio-inspired-cybersecurity