A Self-Organizing Map and its Modeling for Discovering Malignant Network Traffic

Chet Langin
Southern Illinois University Carbondale

Hongbo Zhou
Southern Illinois University Carbondale

Shahram Rahimi
Southern Illinois University Carbondale, rahimi@cs.siu.edu

Bidyut Gupta
Southern Illinois University Carbondale

Mehdi Zargham
Southern Illinois University Carbondale

See next page for additional authors

A Self-Organizing Map and its Modeling for Discovering Malignant Network Traffic

Chet Langin, Hongbo Zhou, Shahram Rahimi,
Bidyut Gupta, and Mehdi Zargham
Dept. of Computer Science
Southern Illinois University
Carbondale, Illinois USA

Mohammad R. Sayeh
Dept. of Electrical and Computer Engineering
Southern Illinois University
Carbondale, Illinois USA

***Abstract*—Model-based intrusion detection and knowledge discovery are combined to cluster and classify P2P botnet traffic and other malignant network activity by using a Self-Organizing Map (SOM) self-trained on denied Internet firewall log entries. The SOM analyzed new firewall log entries in a case study to classify similar network activity, and discovered previously unknown local P2P bot traffic and other security issues.**

I. Introduction

Armies of infected computers in homes, schools, and workplaces are being used in warfare and criminal activities unbeknownst to their owners. One such army, for example, was used in a coordinated Denial of Service (DoS) attack in cyber warfare with the strength to disable the network of Estonia in 2007[1], [2]. These armies are also widely used to mass produce spam e-mail in fraudulent schemes to defraud their victims.

These armies are called *botnets* with each infected computer called a *bot* (short for *robot*.) Bots take their orders from bot masters via command and control (C&C) centers using various protocols, such as Internet Relay Chat (IRC) and Peer-to-Peer (P2P). Estimates of the numbers and sizes of botnets vary, but one study, for example[3], discovered 3,290 unique IRC botnets with 700,700 distinct IP addresses. P2P botnets are more problematic because they encrypt their traffic, and their distributed system makes it difficult to trace and find the C&C. Botnets are particularly insidious because they can accomplish whatever code their malicious master is capable and imaginative enough to deliver to them. *It is clear that botnets have become the most serious security threat on the Internet*[4].

P2P botnets are a recent phenomenon. Cooke, et al, [5] mentioned them in 2005 under the heading *Botnets of Tomorrow*. Grizzard, et al,[6] presented a case study in 2007 of a P2P botnet. Their work is good for historical and descriptive information. Dagon, et al,[7] presented a taxonomy of botnet structures in 2008 which is a good resource for further study in this area.

Cyber defense needs new Computational Intelligence (CI) techniques because traditional methods of intrusion detection are being foiled by P2P botnets. Denning[8] provided the seminal intrusion detection system (IDS) model in 1986, which led to the labeling of two main types of intrusion detection, apparently first mentioned together as such by Kumar in 1994[9], and referred together by many papers since then: misuse and anomaly detection. Misuse detection uses rules written by security experts to detect known characteristics of malicious network traffic by typically looking at protocols, flags, ports, strings in packets and other available network information. Anomaly detection involves examining normal use and then reporting what is abnormal.

Misuse detection is insufficient for P2P botnets because it requires advance knowledge of specific characteristics of malicious software in order to create rules that look for traffic containing these characteristics. These rules were helpful when infected computers used Internet Relay Chat (IRC) for communications. However, the use of encrypted Peer-to-Peer (P2P) networks for communications has thwarted misuse detection because of the encrypted packets and also because analysis of distributed P2P communications needs to be more than packet analysis between two IP addresses at a time.

Anomaly detection is insufficient for P2P botnets because it requires training on a pure network, but there is no way to guarantee that a live network is pure at any given time, so anomaly detection will classify existing malicious traffic as being normal. Also, much of the P2P malicious network traffic cannot be distinguished from normal network traffic. A constantly changing network such as at a university generates more anomalies than can be investigated, so many of which are false positives that they can jeopardize the integrity of network security staff if too many of them are taken seriously.

See Lazarevic, et al,[10] for an extensive survey of intrusion detection in general as of 2005. However, they do not mention bots or botnets. The 2008 doctoral dissertation of Zanero[11] provides an excellent discussion of Self-Organizing Maps (SOM) and other unsupervised learning algorithms for intrusion detection. However, it does not mention bots or botnets.

This paper proposes using model-based intrusion detection combined with Soft Computing (SC) knowledge discovery. Model-based intrusion detection is a third method (in addition to misuse and anomaly detection) which has been mentioned less often in the literature. Previous models have used transition analysis to look for intrusions. Knowledge discovery by SC should become a new method of intrusion detection because it can look for currently unknown characteristics of

TABLE I
SOM TRENDS IN INTRUSION DETECTION

Consistently Used	Solid Trend	Tried Recently
Anomaly detection	Hierarchical maps	Labeling
Network data		Temporal methods
Hexagonal maps		Dynamic subset selection
U-Matrix		Growing maps
		Hybrid/SC

P2P botnets and other types of malicious network traffic.

Soft Computing (SC) is an appropriate form of Computational Intelligence (CI) for knowledge discovery of malicious traffic because the data is voluminous and incomplete, and SC has tolerance for imprecision and uncertainty, employing methods of reasoning that are approximate rather than exact. SC was formalized as a discipline by Zadeh with the concept and formation of the Berkeley Initiative in Soft Computing (BISC) beginning in 1990[12].

This paper reports that Self-Organizing Maps (SOM), a type of SC, clustered network traffic of feral P2P botnets, then successfully classified new P2P botnet traffic, discovering previously unknown bot infected computers. SOM is a type of Artificial Neural Networks (ANN) described by McCullough in 1943[13]. Over 4,000 scientific papers have been written on SOM, which is a mature method of clustering, visualization, and producing abstractions. Examples of SOM use include preprocessing of optic patterns, acoustic preprocessing, process and machine monitoring, diagnosis of speech voicing, transcription of continuous speech, texture analysis, contextual maps, organization of large document files, robot-arm control, telecommunications, and estimating. This paper provides the SOM parameters used, but please see [14] for how these parameters are applied in the SOM, and for further information on uses of SOM.

Next, in Section II, is related work. Then, Section III describes the model. Section IV relates the methodology in using this model. Section V gives a case study, and Section VI is the conclusion.

II. RELATED WORK

Much of SOM research has been conducted on standard data and these papers have been grouped together in this report. The remainder of this section is grouped into four parts: 1) SOM which used independent data; 2) SOM which used standardized data; 3) botnets; and 4) model-based intrusion detection. Table I summarizes the main SOM methods mentioned in the more recent reports below.

A. SOM Research with Independent Data

Fox,et al,[16] proposed using SOM for anomaly detection in 1990. They trained a prototype SOM with model parameter data consisting of CPU utilization, paging activity, mailer activity, disk accesses, memory utilization, average session time, number of users, absentee jobs, reads of help files, failed logins, and multiple logins. Then they tested the SOM with a simulated virus attack, and it worked as expected.

Another prototype anomaly detection SOM was proposed in 1998 by Hoglund and Hatonen[17]. They used a hexagonal map with the nodes darkened to indicate the distances between them, a method called a *U-matrix*, making it easier to visualize user behavior.

Girardin and Brodbeck[18] used SOM for visualization of firewall logs in 1998. They used a rectangular map and divided each node into four parts which were color and pattern coded to depict characteristics of each node. They also displayed a map which showed the dominant characteristic of each cell. He compared SOM visulazations with parallel coordinates and spring layouts.

Rhodes, et al,[19] used multiple two-dimensional SOMs for anomaly detection of tcpdump data in 2000. The network data was collected locally and successfully tested with two different exploits the authors perpetrated against their own server.

Copeland and Garcia[20] used SOM in 2001 to break the 64 possible values for the TCP flag field into four groups as a part of a larger Soft Computing (SC) intrusion detection system. The four groups were SYNs, SYN/ACKs, ACKs, and all other flag combinations.

Lee[21] tested a SOM 2001 and reported that the visualization was attractive, but that the clustering was poor, and that *the presentation of novel inputs after training produces unpredictable results.*

Techniques including Hidden Markov Modeling (HMM) and a hexagonal SOM were used by Cho in 2002[22] on audit data produced by Sun's Basic Security Model (BSM). The SOM was used to preprocess and reduce the data size for the HMM. Cho also used the SOM for visualization with the darkness indicating distances between the nodes. Fuzzy Logic (FL) fused the HMM results in this SC setup.

Hoglund, et al,[23] used SOM for host-based anomaly detection in 2002. They noted that the anomalies *are interesting for the system administrator*, but did not demonstrate that the anomalies produced conclusive results of abuse or intrusions.

Lichodzijewski, et al,[24] introduced a two-level hierarchical SOM for host-based intrusion detection using session information in 2002. The first level of the hierarchy had three maps with each one summarizing one of these features: location, user, and connection type. The second level of the hierarchy combines the results from the first level. They represented time in the data either explicitly or with a moving window. They tested this SOM with a data set with patterns that exhibited potentially suspicious behavior. They concluded that the moving window approach was better than the explicit method at detecting the anomalies.

Ramadas, et al,[25] used SOM for network anomaly detection in 2003 with a vector consisting of these six fields for DNS and HTTP traffic: interactivity, average size of questions, average size of answers, question-answer idle time, answer-question idle time, and duration. The authors generated a BIND exploit which was successfully detected as an anomaly by the SOM. They generated anomalous HTTP traffic which was also detected by the SOM.

Tauriainen[26] classified P2P software with SOM in 2005. The vector included the number of packets sent, the frequency of the transmissions, and the total amount of data transmitted.

A hexagonal SOM was used by Vokoros, et al,[27] in 2006 for user anomaly detection. A six-field vector was used with these fields: user activity times, user login hosts, user foreign hosts, command set, CPU usage, and memory usage. The system was simulated on a server with an average of 32 users, and a feature map was created, but details were not given on how intrusions were tested.

B. SOM Research with Standardized Data

A standardized data set for intrusion detection was developed in 1998 by the United States Department of Defense Advanced Research Project Agency (DARPA) consisting of approximately 5 million connections of training data and 2 million connections of test data[28]. The KDD-99 (apparently for 1999) dataset is based on the 1998 DARPA initiative[29]. Numerous papers have reported research on this data, and the ones related to SOM are mentioned below.

Lichodzijewski, et al,[28] tested the data with an incremental hierarchical SOM in 2002. They used a six-field vector labeled as being *basic TCP information*: duration, protocol, service, flag, destination bytes, and source bytes. The SOM had two layers each of two dimensions: The first layer had a map for each of the six basic TCP features; and, the second layer combined the outputs from the first layer. Time was handled with a fixed temporal horizon. Shading was used to indicate distances between nodes in the second-layer map. They found that attacks were synonymous with Nodes 32 and 33 being the Best Matching Units.

Kayacik, et al,[29] in 2003 used a six-field vector consisting of protocol, service, status, duration, destination bytes, and source bytes. A three-layer hexadecimal SOM was used. Layer 1 had one map for each of the nine features. These nodes were labeled and combined in Layer 2. Layer 3 contained only nodes which won for both attack and normal behaviors. The conclusion was that this system provided competitive performance.

Depren, et al,[30] combined a SOM for anomaly detection with a decision tree for misuse detection for a hybrid system. A Decision Support System (DSS) was used to interpret the anomaly and misuse detection results. The SOM used these six fields in the vector: Duration, protocol, service, connection status, bytes sent to destination, and bytes sent to host. They concluded that the hybrid system was better than the individual misuse and anomaly detection systems.

Sarasamma, et al,[31] used a three-layer SOM with a neighberhood parameter of zero (*winner take all*). It was a flexible system where the types of vectors and numbers of nodes were tunable. Layer 1 detected certain kinds of intrusions. If there were no positive matches, then the data was sent to Layer 2, which detected other types of intrusions, and so on to Layer 3. The detection rate was high for some types of intrusions, but there were also some false positives.

Wetmore, et al,[32] used Dynamic Subset Selection (DSS) in a SOM to reduce the training time. They reported a significant speedup without loss of accuray.

Kayacik and Zincir-Heywood[33] used SOM to create a hexagonal U-matrix attack map for forensic analysis. They proposed a simple method for labeling nodes which takes into account the top five Best Matching Units (BMUs) for each feature and the number of times each node is a BMU.

Yeloglu, et al,[34] used a growing recurrent hierarchical SOM on the data in 2007. Training of this model starts with only one neuron and new neurons are added until the desired accuracy is reached. Neurons can also be pruned. The recurrent aspect of this SOM is to represent temporal qualities of the data. The SOM had three layers. The first layer had a map for each of six basic TCP features. The second layer combined these features, and the third layer was built from second-layer nodes which were BMUs for multiple classes of vectors. They concluded that this SOM was competitive with other methods.

C. Bots and Botnets

Wang, et al,[15] proposed a hybrid P2P botnet in 2007 with two classes of bots: client and servent. A client bot is one which is not accessible from the Internet, typically because it has a dynamic IP address, a private IP address, or because it is behind a firewall. The label *servent* is a P2P term derived from the words *server* and *client*. A servent bot is both a server and a client and has a static IP addresss which is accessible from the Internet.

Wang, et al, proposed a honeypot as a way to detect P2P botnets. However, honeypots are not feasible at all organizations because of issues concerning risk, ethics, legalities, and resources. They also proposed attempting to detect outgoing contacts to the botnet sensor. However, these outgoing contacts can be accomplished in any number of insipid ways which are indistinguishable from normal Internet traffic.

D. Model-Based Intrusion Detection

Only a few papers have been written on model-based intrusion detection.

Kemmerer [35] proposed a network intrusion detection model in 1997 which he called Network State Transition Analysis Tool (NSTAT) and which was based on earlier host-based models. He noted that IDS did not take into consideration two or more users working together to execute a penetration!

Cho and Park[36] proposed a Hidden Markov Model (HMM) model in 2003 to improve intrusion detection performance by only considering the privilege transition flows based on the domain knowledge of attacks.

More recently, in 2007, Zhou, et al,[37] noted the *massive number of simple alerts of low-level security-related events* for signature-based (rule-based) IDS, and proposed a formal model utilizing the concept of *capability* to implement an alert correlator for complex multistage intrusions, expanding an earlier *requires/provides* model.

None of these previously published network models are appropriate for P2P botnets because there are no observable state transitions, privilege transitions, or alert correlations to consider.

III. Model

A full explanation of the model is given in this section, but the model is similar to the P2P botnet description by Wang, et al,[15], mentioned in the previous section, with these three distinctions: the model adds denied inbound Internet traffic as a method to discover malignant network traffic; descriptive details are left out which pertain to a P2P botnet but which are unnecessary to understanding the model; and, what is referred to as a *sensor* is labeled differently, as described below.

Two variations of the model are explained in this section: a specfic version of the model for P2P botnets, and a general version of the model that also includes other network security problems. The specific version of the model is described first, followed immediately by the general version of the model.

A. P2P Botnet Version of the Model

The P2P botnet version of the model is that a computer on the local network, behind a border firewall, gets infected with a P2P bot, notifies the bot master's command and control (C&C), and receives commands on what to do next. The computer can get infected many ways, including by the user responding to spam e-mail, clicking on a malicious web site, and downloading a trojan horse in the guise of music, a movie, or a program, such as a game.

Fig. 1 illustrates the local computer contacting the C&C, which can be done any number of insipid ways such as accessing an unpublicized web page on a benign-appearing web site provided by the bot master for this purpose. Step 1 shows this initial contact going through the firewall without logging because it appears to be normal Internet traffic. Even if it were logged, it could not be distinguished from millions of other packets of routine traffic. The recipient of Step 1 forwards the IP address of the newly infected computer through the Internet cloud in Step 2 to the C&C, further disguising the path to the bot master. In Step 3, the C&C provides reports to the bot master. This initial contact is labeled as occuring at $time = t$.

Wang, et al,[15] refer to the recipient in Step 1 as a *sensor*, however, it does more in the P2P botnet version of the model than sense—it is the designated receiver of the initial contact information of a newly infected computer. The general version of the model, however, does use the term *sensor* as described in Section III.B.

The local computer is not a honeypot set up to entrap the bot master, but, rather, is a computer which is infected unbeknownst to anyone associated with the local network, and which makes the initial contact to the bot master's C&C in secret. An IDS does not detect this because there is no information in the network traffic to match given rules. Anomaly detection does not catch this because it appears to be normal Internet traffic.

The situation remains dormant until the bot master decides to use the local computer, as illustrated in Fig. 2. The bot master's response is labeled as occuring at $time = t + y$ in order to emphasize that this happens when the bot master is ready, and that y could be days later. The encrypted command from the bot master could be, for example, to download and execute certain programming code. This code could be to send spam, DoS an IP address or network, or to do anything else which the bot master is imaginative and capable enough to program.

Step 1 is when the bot master communicates with his or her C&C, which could be from 1 to x computers for redundancy and coverage. Step 2 shows the C&C sending orders through the P2P network. This P2P network can include other computers infected with bots if these computers are not protected with properly configured firewalls.

Step 3 illustrates the P2P botnet attempting to contact the local computer. Experience shows that these contacts can come from over 40,000 unique source IP addresses on the Internet. However, since the local firewall is properly configured, and these sessions originate outside of the local network, this traffic is denied and logged. Step 4 shows the logs going to a log server for analysis. The local computer may or may not even be connected to the network at this time.

Without the P2P botnet version of the model, virtually all of the incoming denied Internet traffic appears to be noisy and random, especially if the local network has many thousands of computers, all behind a firewall, and none of which are known to be bot infected. However, the model warns that some of the denied Internet traffic may point to a previously unknown locally infected computer, even though the P2P botnet traffic does not even reach this computer. The general version of the model is described next.

B. General Version of the Model

The general version of the model can be used to discover other types of network security problems, in addition to P2P botnets, as illustrated in Fig. 3. Part a illustrates a malignant IP address on the local network giving off a putrid network *scent* to the the Internet. This could be something as simple as transmitting a software banner indicating a vulnerable version of a program, or more complex such as backlash to a spoofed IP address that is not even assigned to a computer.

The putrid scent is picked up by a sensor. Compare this to the *recipient* in the P2P botnet version of the model. In the first case, a newly infected bot is notifying a botnet's designated receiver; in the present case, a sensor notices a malignant IP address, which may not even be assigned to a computer.

The outgoing putrid scent imitates normal network traffic to anomaly detectors and does not fit any misuse detection rules, so it either is not logged, or else the log entry cannot be distiguished from normal firewall logs.

Information about the putrid scent could make its way into a P2P botnet as a computer ripe for attack, it might be posted on a cracker's web site as a good target, or it might be spread and labeled by other means.

Fig 3.b illustrates probes and/or attacks directed back towards the malignant IP address. This network traffic is blocked

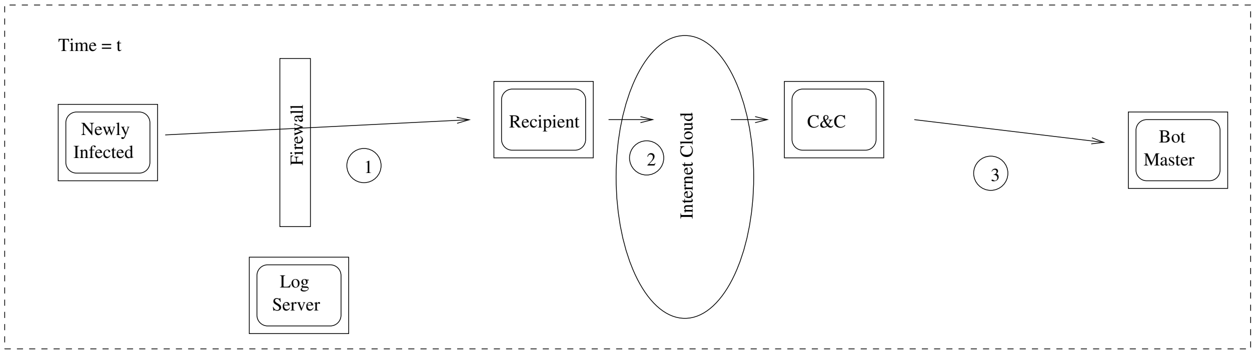

Fig. 1. This figure illustrates a newly infected computer contacting the botnet in the P2P botnet version of the model. See Section III.A for a detailed description.

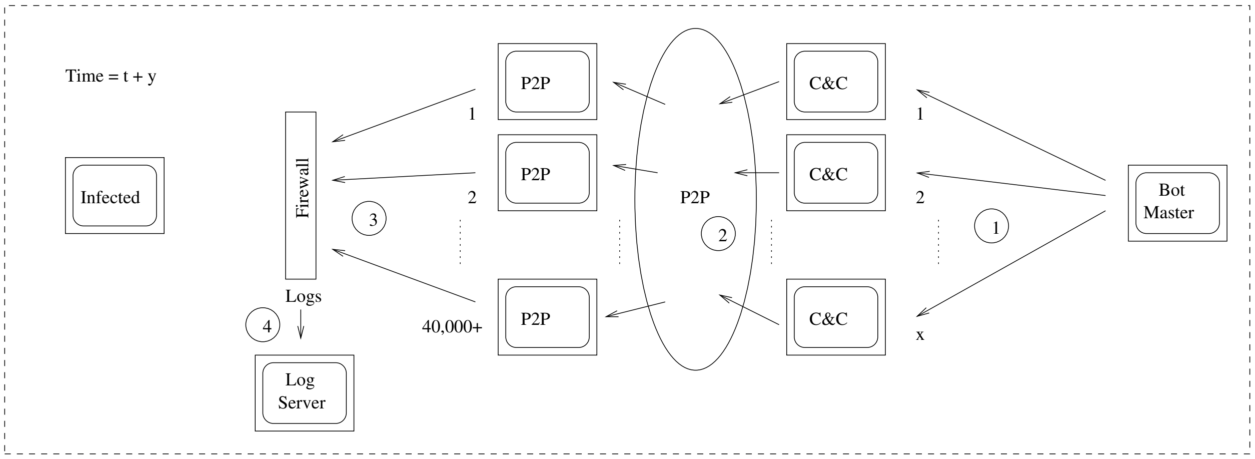

Fig. 2. This figure illustrates the bot master issuing a command in the P2P botnet version of the model. See Section III.A for a detailed description.

and logged because the sessions originated outside of the local network. Once again, the local IP address with a security problem can be discovered by network traffic which is not even on the local network. In this case, there might not even be a computer assigned to the local IP address. The next section provides a methodology on how to take advantage of this model.

IV. Methodology

The methodology for the P2P botnet version of the model is the same as for the general version of the model: The goal for both is to cluster and classify the firewall log data in order to discover new unknown bots and other network security issues. The model provides privacy by using non-local network data and by abstracting this data in a SOM. The clustering steps will be shown first, followed immediately by the classifying steps. Here are the clustering steps for both versions of the model.

1) Begin with the firewall log entries in a text file, such as in a file created from syslog, for a 24-hour time period.
2) Create an *entries* table in a database, such as in MySQL, with these columns to store and manipulate the data: keyID, date, time, protocol, source_ip, source_port, dest_ip, and dest_port.

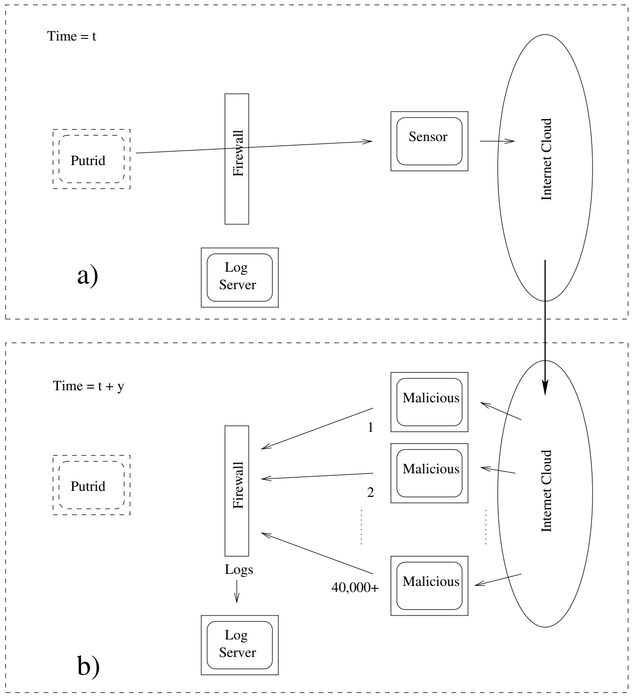

Fig. 3. This figure illustrates the general version of the Model. See Section III.B for detailed information.

   a) The keyID is a unique number to identify each entry to assist in administering the database.
   b) The date is the date of each firewall log entry.
   c) The time is the time of each firewall log entry.
   d) The protocol is the protocol used for each entry of denied network traffic.
   e) The source_ip is the external IP address attempting to access the local network for each entry.
   f) The source_port is the external source port for each entry.
   g) The dest_ip is the destination IP address on the local network for each entry.
   h) The dest_port is the destination port on the local network for each entry.
3) Write and run a script to read and parse the external denied firewall log entries from the text file and insert them into the database.
4) Create a *matrix* table in the database with these columns: keyID, dest_ip, total_count, source_count, port_count, low_port, high_port, icmp_count, tcp_count, and udp_count.
   a) The keyID is a unique identifier for each table row to assist in administering the database.
   b) The dest_ip is the destination IP on the local network to which the remaining data in this table row applies.
   c) The total_count is the total number of firewall log entries for the destination IP address in this table row.
   d) The source_count is the total number of unique external source IP addresses in the log entries for the destination IP address in this table row.
   e) The port_count is the total number of unique destination ports in the log entries for the destination IP address in this table row.
   f) The low_port is the lowest destination port in the entries for the destination IP address in this table row.
   g) The high_port is the highest destination port in the entries for the destination IP address in this table row.
   h) The icmp_count is the total number of ICMP protocol log entries for the destination IP address in this table row.
   i) The tcp_count is the total number of TCP protocol log entries for the destination IP address in this table row.
   j) The UDP_count is the total number of UDP protocol log entries for the destination IP address in this table row.
5) Preprocess the data by writing database queries to summarize information for each destination IP address from the entries table to the matrix table.
6) Write a query to save the data from the matrix table to a matrix text file, one table row per line, with each line being a vector, using these fields: total_count, source_count, port_count, low_port, high_port, icmp_count, tcp_count, and udp_count. This matrix text file is the training data.
7) Note which vectors in the matrix text file represent IP addresses with computers infected with bots. (However, the SOM trains itself without this information.)
8) Cluster the vectors in the the matrix text file with an algorithm such as SOM.
9) Note which cluster or clusters contain the vectors for the bots. Label these the *bot clusters*.

The bot clusters from above can now be used in classifying new firewall log entries in order to discover new bots and other network security issues. Write an automatated script to run daily in order to determine which local IP addresses match a bot cluster as determined above. However, not every destination IP address needs to be fully processed for classification—only the ones with a threshold of log entries. Therefore, the previous steps may be significantly modified to process much of the data directly from the text file, bypassing some of the database queries. Here are the classifying steps.

1) Begin with the firewall log entries in a text file, such as in a file created from syslog, for a 24-hour time period.
2) Scan the log file and obtain the total_count values of external denied entries for each IP address on the local network. (The local IP addresses will be destinations in the log entries.) Since a match with a bot cluster will require a high total_count value, keep track of the IP addresses whose total_count values exceed a high threshold.
3) Rescan the log file and save the entries of IP addresses which meet the threshold value determined in the previous step into a new working file.
4) Scan the working file for each appropriate IP address to determine the summary information and save this into a vector text file.
5) Find the Best Matching Unit (BMU) for each vector in the vector text file. If the BMU is a bot cluster, then label the IP address for this vector as being a suspect.
6) Rescan the working file and load the entries for each suspect into a database for further query analysis to obtain detailed information such as port frequencies and times.
7) Manually investigate further and follow up as appropriate.

Although the new data is matched to bot clusters, other types of network security issues may be discovered using this method, as well. Since the training data is specific to the local network environment and the exact firewall configuration producing the logs, the SOM is not transferable to other locations.

V. Case Study

The above methodology was tested on data from the Southern Illinois University Class B network when it contained two known P2P bots, later labeled Bot1 and Bot2. The testing involved two stages: the clustering stage when the SOM trains itself; and, the classifying stage when the SOM looks for new bots. These stages are covered in the next two subsections. A visual map was not pertinent to the results and so none is provided.

A. The SOM Trains Itself (Clustering Stage)

Over 20 million firewall log entries were in the test data, the bulk of which were logs of incoming traffic which was denied by the border firewall. Approximately 60,000 of the local IP addresses had entries. Since the data vector for each IP address had eight fields, this gave an input matrix of virtually 8 x 60,000.

A one-dimensional 1,000-node ANN was used for the SOM as a foundation for the research and this provided positive results as explained below. Each node had a randomly created eight-field vector to correspond to the vectors representing the firewall log file entries. The node vectors were sorted by total_count in order to speed up the training with Node 0 having the lowest value and Node 999 having the highest value. One thousand nodes gave an average of approximately 60 IP addresses per node, thus each node represented a cluster of local IP addresses.

Due to the long anticipated self-training time of the SOM, the code was able to save state and stop after each epoch. The code also logged the average and greatest changes of the nodes for each epoch in order to provide ongoing evidence that the nodes were converging on the data.

Euclidean distance was used to measure multidimensional distances and the SOM began with a neighborhood of all 1,000 nodes, which was systematically reduced down to a neighborhood distance of 0. The initial learning rate factor was set to approximately $1/60,000$, and this was gradually changed to approximately $1/60$. The training period was 180 Epochs, an unusually small amount for SOM training, but monitoring indicated that the data was converging in an acceptable manner. The vectors for the IP addresses representing Bot1 and Bot2 both had Node 996 as the BMU, and none of the other IP address vectors had Node 996 as the BMU. The SOM had thus isolated the IP addresses of the bots from the other IP addresses into one bot cluster. This isolation occurred early in the training and was maintained to the end. Therefore, Node 996 would be the only cluster used to discover suspects.

B. The SOM Analyzes New Data (Classifying Stage)

An automated daily script examined new firewall log entries and determined suspects as described earlier: Vectors were created for IP addresses with enough log entries to surpass the threshold, and the Best Matching Unit (BMU) was found for each. If Node 996 was the BMU then the IP address for that vector was labeled as being a suspect.

The SOM produced 18 suspects in 37 alerts in 96 days. Not all of the suspects were available to be investigated. However, some further information was obtained on the other seven suspects.

Suspect01 probed other computers in a manner indicating a malware infection after being discovered by the SOM. Suspect01 was found to have both P2P software and multiple infections, but the exact types of P2P software and infections were not available.

The users of Suspect02 and Suspect03 stated that they were using non-malicious P2P software.

There was no evidence that the IP address for Suspect04 was assigned to any computer. One possible explanation of this could be that the traffic was backlash from a spoofed IP address.

Suspect10 was an appliance which was using outdated firmware and had a web interface which was vulnerable to a cross-site scripting attack.

Suspect13 was corroborated a couple of hours later by another detection method indicating it was infected with a P2P bot. The data from Suspect13, existing before it could be disabled, was detected by both the SOM and the other detection method again the next day.

Suspect17 was a DHCP IP address used at a station to assist incoming students in finding and cleaning malware off of their computers and in setting up Windows firewalls, automated patching, and anti-virus software. No computer was logged in as using this IP address when the pertinent log entries were recorded by the firewall.

No other P2P bots of the same type were detected on the network by other means on days in which the SOM ran an analysis. The SOM discoveries do not fall clearly into true positive and false positive categories, but it is clear by observation that the model and the SOM are validated in discovering feral bots and other network security issues.

VI. Conclusion

This paper provides a model describing how external denied firewall log entries can be used with clustering and classifying knowledge discovery tools to discover internal P2P bots and other local security problems. As far as we know, the case study reports the first times that a self-trained Soft Computing (SC) Computational Intelligence has discovered previously unknown feral malicious software.

The model and SOM in this report have two advantages: 1) the knowledge discovery of intrusions and other malignant network problems that is not available by other methods; and, 2) the privacy afforded by the use of non-local network data and the abstraction of the data in the SOM. It has two disadvantages: 1) the resources required to self-train the SOM and interpret the results; and, 2) the trained SOM is not transferable to other locations.

SOM is viable as an intrusion detection procedure and other researchers have provided many excellent SOM variations as reported in Section II. The difference between their work and the work in this paper is the combination of using the model and SOM on external denied firewall log entries.

Direct comparison of the different methods used by other researchers on the DARPA/KDD-99 standard data is not feasible because too many variations in the research are involved.

McHugh[38] wrote a critical analysis of research involving the DARPA dataset in 2000 saying that it was flawed in many respects, but was the only large-scale attempt at an objective evaluation. Zanero[11] was more straightforward in 2008, saying, *the 1999 dataset is hopelessly outdated, both because the protocols, applications and operating systems used are not representative any more of network usage; and also because the attack types are not representative of the modern threat scenario*.

Cyber defense needs new Computational Intelligence (CI) techniques because traditional methods are being thwarted by P2P botnets. A hexagonal SOM has been trained on the same data as in this paper, is being analyzed, and will be reported for comparison in the future. This area of research should also continue by testing other types and variations of CI on the model. Table 1 provides suggestions for future directions using SOM.

References

[1] J. Davis, "Hackers take down the most wired country in europe," *Wired*, vol. 15, no. 9, September 2007.

[2] J. Robb, "When bots attack," *Wired*, vol. 15, no. 9, September, 2007 2007.

[3] J. Zhuge, T. Holz, S. Y. Han, J. Guo, and W. Zou, "Characterizing the irc-based botnet phenomenon," 2007. [Online]. Available: http://honeyblog.org/junkyard/reports/botnet-china-TR.pdf

[4] W. Lee, C. Wang, and D. Dagon, *Botnet Detection: Countering the Largest Security Threat*, ser. Advances in Information Security. Springer, 2008.

[5] E. Cooke, F. Jahanian, and D. McPherson, "The zombie roundup: Understanding, detecting, and disrupting botnets," 2005 2005, steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI '05).

[6] J. B. Grizzard, V. Sharma, C. Nunnery, and B. B. Kang, "Peer-to-peer botnets: Overview and case study," 2007 2007, uSENIX Workshop on Hot Topics in Understanding Botnets (HotBots '07).

[7] D. Dagon, G. Gu, and C. Lee, "A taxonomy of botnet structures," in *Botnet Detection: Countering the Largest Security Threat*, ser. Advances in Information Security, W. Lee, C. Wang, and D. Dagon, Eds. Springer, 2008, pp. 143–164.

[8] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. 13, no. 2, pp. 118–131, 1986.

[9] S. Kumar and E. Spafford, "An application of pattern matching in intrusion detection," Purdue University, Tech. Rep., 1994.

[10] A. Lazarevic, V. Kumar, and J. Srivastava, "Intrusion detection: A survey," in *Managing Cyber Threats*, V. Kumar, J. Srivastava, and A. Lazarevic, Eds. Springer, 2005, pp. 19–78.

[11] S. Zanero, "Unsupervised learning algorithms for intrusion detection," Ph.D. dissertation, Politecnico di Milano, 2008.

[12] L. A. Zadeh, "History; bisc during 90's," 1994.

[13] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[14] T. Kohonen, *Self-Organizing Maps*, 3rd ed., ser. Springer Series in Information Sciences. Berlin Heidelberg New York: Springer-Verlag, 2001, vol. 30.

[15] P. Wang, S. Sparks, and C. C. Zou, "An advanced peer-to-peer botnet," 2007 2007, uSENIX Workshop on Hot Topics in Understanding Botnets (HotBots '07).

[16] K. L. Fox, R. R. Henning, and J. H. Reed, "A neural network approach towards intrusion detection," in *13th National Computer Security Conference*, 1990.

[17] A. Hoglund and K. Hatonen, "Computer network user behavior visualization using self-organizing maps," in *ICANN*, 1998, pp. 899–904.

[18] L. Girardin and D. Brodbeck, "A visual approach for monitoring logs," in *12th Systems Administration Conference (LISA '98)*, Boston, 1998.

[19] B. C. Rhodes, J. A. Mahaffey, and J. D. Cannady, "Multiple self-organizing maps for intrusion detection," 2000 2000.

[20] J. A. Copeland and R. C. Garcia, "Real-time anomaly detection using soft computing techniques," in *IEEE SoutheastCon 2001*, 2001.

[21] H. Lee, "Training a neural-network based intrusion detector to recognize novel attacks," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 31, pp. 294–299, 2001.

[22] S.-B. Cho, "Incorporating soft computing techniques into a probabilistic intrusion detection system," *IEEE Trans. Systems Man Cybernet*, vol. 32, no. 2, p. 154, 2002.

[23] A. Hoglund, K. Hatonen, and A. Sorvari, "A computer host based user anomaly detection system using the self organizing map," in *International Joint Conference on Neural Networks, IEEE IJCNN*, vol. 5, 2002, pp. 411–416.

[24] P. Lichodzijewski, A. N. Zincir-Heywood, and M. Heywood, "Host-based intrusion detection using self-organizing maps," 2002.

[25] M. Ramadas, S. Ostermann, and B. Tjaden, "Detecting anomalous network traffic with self-organizing maps," in *Recent Advances in Intrusion Detection, 6th International Symposium, RAID 2003*, ser. Lecture Notes in Computer Science, G. Vigna, E. Jonsson, and C. Kruegel, Eds., vol. 2820. Pittsburgh, PA, USA: Springer-Verlag, 2003, pp. 36–54.

[26] A. Tauriainen, *A Self-learning System for P2P Traffic Classification*. Helsinki: Helsinki University of Technology, 2005.

[27] L. Vokorokos, A. Balaz, and M. Chovanec, "Intrusion detection system using self organizing map," *Acta Electrotechnica et Informatica*, vol. 6, no. 1, p. 6, 2006.

[28] P. Lichodzijewski, A. N. Zincir-Heywood, and M. Heywood, "Dynamic intrusion detection using self-organizing maps," 2002.

[29] G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "On the capability of an som based intrusion detection system," in *IEEE International Joint Conference on Neural Networks,*, 2003, pp. 1808–1813.

[30] O. Depren, M. Topallar, E. Anarim, and M. Ciliz, "An intelligent intrusion detection system (ids) for anomaly and misuse detection in computer networks," *Expert Systems with Applications*, vol. 29, no. 4, pp. 713–722, 2005.

[31] S. T. Sarasamma, Q. Zhu, and J. Huff, "Hierarchical kohonenen net for anomaly detection in network security," *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, vol. 35, no. 2, pp. 302–312, 2005.

[32] L. Wetmore, A. N. Zincir-Heywood, and M. Heywood, "Training the sofm efficiently: An example from intrusion detection," in *IEEE Internation Joing Conference on Neural Networks, IJCNN 2005*, 2005, pp. 1575–1580.

[33] H. G. Kayacik and A. N. Zincir-Heywood, "Using self-organizing maps to build and attack map for forensic analysis," in *ACM International Conference on Privacy, Security, and Trust (PST 2006)*, 2006, pp. 285–293.

[34] O. Yeloglu, A. N. Zincir-Heywood, and M. Heywood, "Growing recurrent self organizing map," in *IEEE International Conference on System, Man and Cybernetics (SMC 2007)*, 2007.

[35] R. A. Kemmerer, "Nstat: A model-based real-time network intrusion detection system," University of California-Santa Barbara, Tech. Rep., November 1997.

[36] S.-B. Cho and H.-J. Park, "Efficient anomaly detection by modeling privilege flows with hidden markov model," *Computers and Security*, vol. 22, no. 1, pp. 45–55, 2003.

[37] J. Zhou, M. Heckman, B. Reynolds, A. Carlson, and M. Bishop, "Modeling network intrusion detection alerts for correlation," *ACM Transactions on Information and System Security (TISSEC)*, vol. 10, no. 1, 2007.

[38] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262–294, 2000.