Network Anomaly Detection Using Artificial Neural Networks

Sergey Andropov, Alexei Guirik
ITMO University
Saint-Petersburg, Russia
Andropov.Sergey1@gmai.com, avg@corp.ifmo.ru

Mikhail Budko, Marina Budko
ITMO University
Saint-Petersburg, Russia
mbudko@corp.ifmo.ru, mbbudko@corp.ifmo.ru

***Abstract*—This paper presents a method of identifying and classifying network anomalies using an artificial neural network for analyzing data gathered via Netflow protocol. Potential anomalies and their properties are described. We propose using a multilayer perceptron, trained with the backpropagation algorithm. We experiment both with datasets acquired from a real ISP monitoring system and with datasets modified to simulate the presence of anomalies; some Netflow records are modified to contain known patterns of several network attacks. We evaluate the viability of the approach by practical experimentation with various anomalies and iteration sizes.**

I. Introduction

Information security of modern computer networks is a growing source of concerns. The need for tools capable of detecting the ever-increasing number of network attacks, viruses and other incidents continues to grow. There are methods of anomaly detection based on known signatures and metrics; while giving acceptable results and having a low false alarm rates, they are normally unable to detect a previously unknown attack or vectors of a virus spreading, which means such programs have to rely on databases being updated on a regular basis.

Anomaly can be defined as a deviation from the normal behavior. An anomaly can be defined as "an event (or object) that differs from some standard or reference event, in excess of some threshold, in accordance with some similarity or distance metric on the event" [1]. Major sources of network anomalies are network attacks, hardware or software malfunctions and malware. Anomalies should be considered dangerous, as breaking or disabling one component of the network can lead to the entire network being compromised. Detecting – and classifying – an anomaly is an important step in preventing, or at least reducing, the potential damage caused by its occurrence.

This article presents a method of detecting and classifying an anomaly using an artificial neural network that analyses data received with the Netflow protocol.

The topic of using machine learning for intrusion and anomaly detection is a well researched one [2], [3]. As stated in [2], using neural networks and other means of adaptable systems for network monitoring presents a set of challenges, such as:

- high cost of errors;
- difficulties of obtaining labeled data for specific kinds of anomalies/attacks;
- semantic gap between results and their operational interpretation;
- high variance in network's behavior;
- difficulties in evaluating and verifying results.

Dealing with these problems is necessary to construct an effective anomaly detection system. One part of the solution is setting a narrower focus for the analysis program, such as limiting the scope of the network that is being monitored.

Any network will change over time, as the new hardware gets installed, the new programs with different behavior patterns emerge, etc. Usage of a neural network allows the system to adapt to the gradual changes as it can adjust its weights to accommodate for the shifts in the behavior.

Another problem of automated anomaly detection is the difficulty of obtaining proper training data and filtering the network noise. Usage of Netflow protocol allows countering some of these difficulties, as it provides vital information without overloading the system with too much data and thus has relatively small storage requirements [10].

The rest of the article is structured as follows: section II contains the overall design of the detection method; section III describes network anomalies and their typical properties; sections IV and V contain Netflow packet structure and the aggregation criteria; section VI shows the design and the training process of the neural network; sections VII presents the dataset acquisition methods and evaluation of the detection algorithm; finally, conclusions are drawn in section VIII.

II. Design

The detection system has the following capacities:

1) Offline traffic analysis. With this type of analysis a model of normal behavior for the network can be created. It also provides information about different anomalies. Implementation of offline analysis requires a dataset of normal network behavior for a certain period.

2) Online traffic analysis. This is the primary mode for the system. Data from the Netflow protocol is received, processed and stored by the system; different aspects of it are analyzed by an artificial neural network in real time. If an anomaly is detected, a warning will be issued along with a report containing information about the incident.

Data aggregation allows the system to see patterns in the otherwise highly variable traffic properties. Netflow protocol

ISSN 2305-7254

is used to receive information from the network devices, which is then filtered and aggregated based on a number of criteria such as number of packets per hour, average packet size, port usage. This data serves as an input to a neural network, which consists of three layers: input layer, hidden layer and output layer.

The outputs of the neural network show if an anomaly is present. For every known anomaly the system is designed to detect, there is a neuron that shows the probability of that anomaly occurring. There is an additional neuron for an "unknown" anomaly, as well as one for a "normal" behavior.

We discuss further details on the internal design of the neural network in section VI.

III. Network anomalies

Network anomaly is a situation when the network behaves differently from its established pattern. There are numerous reasons for anomalies to appear:

- hardware malfunctions
- unauthorized actions of the staff
- intentional security breaches involving the staff
- network attacks
- virus infection process
- "flash-crowd" behavior

Anomalies will have different features depending on their source. For example, hardware malfunctions are likely to have distinct drops in incoming/outgoing traffic, as presented in Fig. 1.

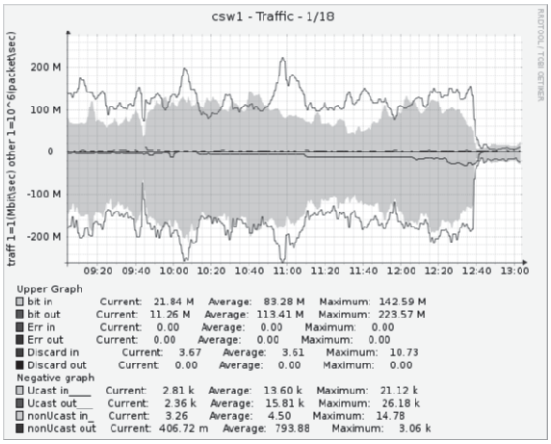

Fig. 1. An example of network anomaly: uplink hardware failure

Another examples are the rising number of discarded packets, as shown in Fig. 2, and the result of GRE protocol being active in Fig. 3.

While unauthorized actions and intentional security breaches might be detectable through traffic analysis, in this work we primarily focus on monitoring anomalies caused by network attacks, failures of hardware and changes in network behavior.

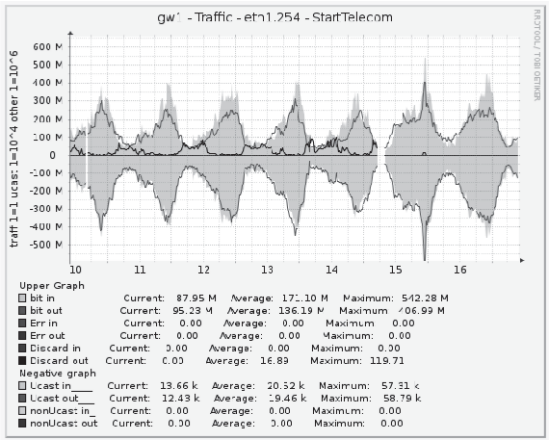

Fig. 2. An example of network anomaly: abnormally high number of discarded packets

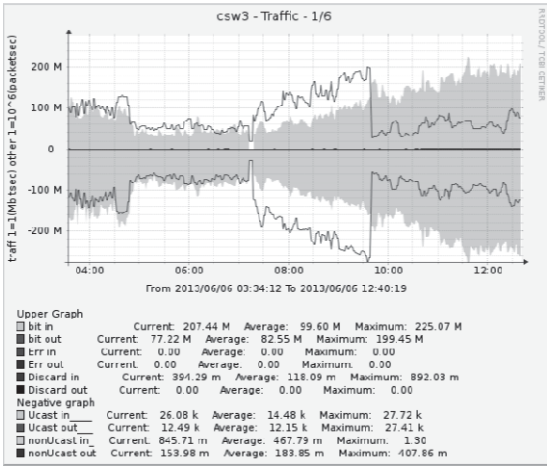

Fig. 3. An example of network anomaly: GRE active

The topic of network attacks has been researched extensively [4], [5]. We will give short descriptions along with identifiable properties of each anomaly we plan to detect.

- DoS and DDoS attacks

Denial of service attack aims to make an online service unavailable. It is usually carried out by overwhelming the target with superfluous requests, using multiple sources in case of a Distributed DoS. This type of attack is regarded as one of the most common. (D)DoS attacks often use minimal packet sizes, set an incorrect protocol number and use the same source/destination addresses. Targeted service experiences a noticeable increase of traffic flow and specific port number usage. There are multiple subtypes of this attack: UDP Flood, ICMP Ping Flood, SYN Flood, NTP Amplification etc.

- Network scans

Scanning attacks are performed by making multiple attempts to connect with different hosts in order to find ports

and addresses that are open for connections. This procedure can be performed by network administrators to check security levels, as well as by potential adversaries to find vulnerabilities. Network scans can be spotted by observing an increase in connections from a certain host, or from multiple hosts to one. Traffic generated by this type of attack is comparatively small.

- Flash-crowd

This anomaly can be mistaken for a DDoS attack. It happens when public's interest towards a particular resource is significantly increased; an example of this is a release of long-awaited program that a group of people rushes to download or access. One of the ways to distinguish this from DDoS is to look at the packet size, which should be noticeably higher compared to the actual attack.

- ARP spoofing

ARP spoofing involves sending falsified ARP (Address Resolution Protocol) messages over a local area network. In their most basic form, ARP spoofing attacks are used to steal sensitive information. Beyond this, ARP spoofing attacks are often used to facilitate other attacks like DDoS attacks, session hijacking (granting attackers access to private systems and data) and man-in-the-middle attacks (intercepting and modifying the traffic between victims). Monitoring the network data should reveal an increase in number of packets with conflicting source address information.

- Idle scan

Idle scan is sometimes called a "stealth scan". The main difference from the network scan type of attack is the fact that the attacker does not need to send packets from his own IP address. Attack itself is performed from a number of "zombie" machines. Even though the target's intrusion detection systems might raise the alarm, the true direction is much harder to determine. The danger of this attack lies in the fact that it can produce a list of open ports from the "zombie" point of view; a list that basically exposes the relationship between hosts within the targeted network.

Each of the mentioned anomalies has their own set of traits by which they can be recognized. In reality, however, it is often difficult to spot these among all the information noise within a network, not to mention that attackers constantly find new ways of performing malicious acts. As such, a detection system needs to be able to learn new patterns and identify them.

IV. NETFLOW PROTOCOL

Information volume that gets transferred in even a middle-sized network is immense, and analyzing it all would be very resource consuming for a real-time intrusion detection system. Netflow protocol provides data about that information without actually storing the information itself, which is perfectly suited for the task.

This protocol was initially developed by Cisco for packet switching and nowadays "efficiently provides a key set of services for IP applications, including network traffic accounting, usage-based network billing, network planning, security, Denial of Service monitoring capabilities, and network monitoring" [6]. It became a de-facto standard and is used widely on Cisco devices. There are similar protocols, most notably sFlow and IPFIX, however comparison between them beyond the scope of this article.

Netflow uses UDP/SCTP protocols to transfer data from routers to special collectors. According to protocol description, all packets with the same source/destination IP address, source/destination ports, protocol interface and class of service are grouped into a flow and then packets and bytes are tallied [6]. Then these flows are bundled together and transported to the Netflow collector server.

Most common version of Netflow is 5 [6]. The following information is used in analysis:

- Source and destination addresses
- Source and destination ports
- Protocol type
- Number of packets
- Size of packets

This data is used to:

- aggregate the traffic by different criteria
- determine the number of connections and transfers
- spot the types of attacks that use specific ports
- determine direction of the flows and usage of protocols

V. AGGREGATION CRITERIA

The following criteria were chosen based on the information provided by the Netflow protocol:

1) Incoming/outgoing traffic volume for specific protocols.

The traffic intensity changes throughout the day, week and month, as different types of users connect and disconnect from the network and perform various number of activities. A perfect detection system would therefore create the network's model for all listed periods of time, however it is important to limit the scope of the system to a reasonable level [3]. As such, for the purposes of this work normal behavior model is assumed to be created for a period of one day.

2) Usage of ports.

Certain viruses are known to use a specific port to perform malicious actions (e.g. AckCmd – port 1054, WinHole – 1081 [7]). A DDoS attack can also target a specific port.

There is normally an increase in number of connections during an attack or spreading of a virus. Analyzing the port usage data might reveal these situations. One of the challenges here is distinguishing port scanning from other anomalies, since the former is not necessarily performed with malicious intents.

3) Packet size.

Viruses and attacks sometimes use specific packet sizes. As mentioned above, DDoS attack differs from "flash-crowd" behavior in packet size. It is common to use small packet size to increase the number of packets sent. A good example of

specific packet size is NTP Amplification attack, which requires a command of 234 bytes to be sent.

4) Number of open connections.

A virus tries to infect as many machines as it possibly can, and it does so by opening a large number of connections to different hosts. While Netflow protocol does not provide information on currently opened sessions, it is still possible to determine the relation between the number of connections and port numbers.

6) Average traffic-to-host value.

While this value will differ widely for various clients and hosts, certain patterns are still present. Analysis on this criteria can be done for a specific set of hosts in order to avoid a selection that is too difficult to establish a pattern on.

VI. Neural network

A. General design

An artificial neural network consists of a number of computational units called "neurons". Neurons receive inputs and process them, obtaining output. Different functions, known as "activation functions", can be used for this calculation, ranging from linear to sigmoid and hyperbolic tangent functions.

A typical artificial neural network has three layers – input, "hidden" and output. Second, hidden, layer makes network's behavior non-linear. The amount of neurons within that layer and the amount of hidden layers itself can be chosen with different techniques in mind and depends on the conditions of the task, however, for tasks without especially complex computations one layer is usually enough [8]. The amount of neurons can be an average between input and output neurons, but may be increased.

The output layer has several neurons, one for each anomaly the system is going to be able to detect, plus one for an "unknown" anomaly and one for "normal" behavior. These neurons output values between 0 and 1, which are considered to be the probabilities of a relevant anomaly occurring.

Each connection has a weight; neurons in the hidden and output layers have biases. Weights and biases are adjusted in order to reduce the overall output error in both the training and deployment.

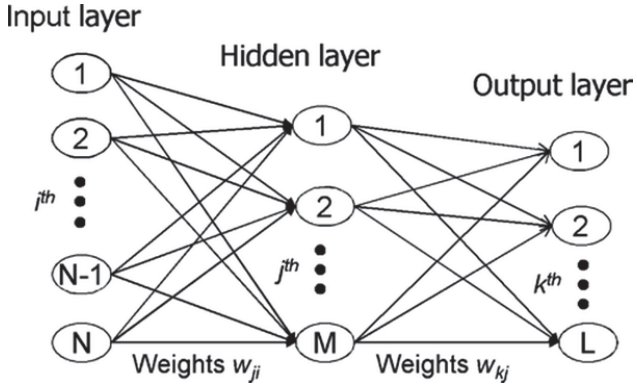

Fig. 4. A basic neural network structure

Each neuron in the hidden and output layers has an activation function for producing non-linear output and propagating it forwards through the network. In this work we used a sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

where x is a sum of neuron's input xi with weights ωi and bias b:

$$x = \sum \omega_{ij} x_i - b$$

Sigmoid function is bounded, easily differentiable, monotonic, and produces a smooth output. Small changes in input coefficients (weights and bias) produce small changes in the output of the function, thus making precise tuning possible. Its output range is [0;1], which makes it useful in classification cases.

For purposes of this work a neural network with 6 inputs, 10 neurons in the hidden layer and 7 output neurons is chosen. Both output and hidden layer neurons use a sigmoid activation function.

B. Training algorithm

In order to train the network, we use the backpropagation algorithm [8]. The backpropagation algorithm uses the gradient descent method to look for the minimum of the error function. A solution is thus the combination of weights that minimizes the error.

First, the input information is presented to the network and propagated forward until it reaches the output layer. Then the desired and actual outputs are compared and the error for each output neuron is calculated. This error is propagated backward through the network, thus giving the error for each neuron in all hidden layers. Using these values, a backpropagation algorithm can update weights and biases.

Initial weights of the network are selected at random. When input x_i is presented to the network, it is propagated through the network, producing an output o_i. The goal of the training algorithm is to make the output o_i close or identical to the desired output t_i for each input. This is done by minimizing the error function:

$$E = \frac{1}{2}\sum_{i=1}^{n}(o_i - t_i)^2$$

First, the error signal in the output layer k is calculated:

$$\Delta_k = t_k - o_k$$

$$\delta_k = \Delta_k \alpha'_k \quad (1)$$

where α'_k is a derivative of the activation function. For the output layer this derivative equals 1.

The weights of the output layer are adjusted according to:

$$\Delta\omega_{jk} = x_k \delta_k \gamma \quad (2)$$

where x_k is the input from a neuron in the previous layer (i.e. the output of the relative neuron in the hidden layer), γ is the learning rate. This learning rate is typically a small number

(eg. 0.004), regulating the speed at which the weights are adjusted. Big learning rate values may cause the network's outputs to oscillate around the target, thus never converging on a solution; small values might cause the learning process to be very slow.

It is worth noting that the gradient descend method has a downside in that it might get "stuck" at the local minimum of the error function. In order to get over the "small hill" and continue moving towards a global minimum, we can modify the equation (2) as follows:

$$\Delta\omega_{jk}{}^{n} = x_k\delta_k\gamma + \Delta\omega_{jk}{}^{n-1}\varphi$$

where φ is a *momentum factor*. The introduction of the momentum accelerates the learning process by keeping track of the previous changes, thus allowing the algorithm to move in larger steps. The faster movement prevents the network from settling in a local minimum by helping it move past the "hill".

The error signal for the nodes in hidden layer is calculated in a similar way to the output layer.

$$\delta_j = \alpha'_j \sum \omega_{jk}\delta_j$$

where $\sum \omega_{jk}\delta_j$ is the weighted error signal. A derivative of the activation function for the hidden layer is:

$$\alpha'_j = o_j(1 - o_j)$$

therefore

$$\delta_j = o_j(1 - o_j) \sum \omega_{jk}\delta_j \qquad (3)$$

Weights of the hidden layer are updated in the same way as the output's:

$$\Delta\omega_{ij}{}^{n} = x_j\delta_j\gamma + \Delta\omega_{ij}{}^{n-1}\varphi$$

VII. Dataset and evaluation

Section V presented a list of aggregation criteria. Data acquired through aggregation is used as an input to the artificial neural network. However, acquiring a good labeled dataset is quite a challenging task. In order to correctly train the network and not have it "overfit", the dataset should to be sufficiently large and diverse; it should include enough outliers for the network to be able to detect the patterns between the inputs and the desired outputs.

For this work the data was collected from the local ISP network with several hundred L2 nodes for a period of one month. Dataset collected this way is considered to show "normal" behavior before any modifications are made. For testing purposes, several small-scale anomalies were created: DoS and DDoS attacks, port scans, email spamming and routers turning off. Data was then modified with the Flame tool, as suggested in [9].

Flows in the dataset were edited by adding and modifying Netflow records in order to simulate suspicious activity. Patterns of known anomalies were injected into the gathered data. The full list of anomalies added to the dataset is as follows:

- DDoS UDP flood
- DDoS TCP flood
- Port scan
- Idle scan
- ARP spoofing
- "custom" anomaly

The "custom" anomaly was created to test the neural network's ability to spot an unknown class of anomalies. It consists of several random port/packet usage combinations, simulating the way some viruses work.

Each anomaly in the dataset was labeled and fed into the neural network. Backpropagation algorithm adjusted the weights and biases each time the outputs classified an anomaly. This process was done for 150000 and 300000 iterations with the learning factor γ of 0.004.

The outcomes for different anomalies can be seen in Table I for 150000 iterations and Table II for 300000 iterations.

TABLE I. CLASSIFICATION RESULTS AFTER 150000 ITERATIONS

Type	Accuracy	Quantity in dataset
DDoS UDP flood	0.87	10
DDoS TCP flood	0.79	10
Port scan	0.84	10
Idle scan	0.76	8
ARP spoofing	0.71	8
custom	0.63	5
normal	0.95	-

TABLE II. CLASSIFICATION RESULTS AFTER 300000 ITERATIONS

Type	Accuracy	Quantity in dataset
DDoS UDP flood	0.91	10
DDoS TCP flood	0.85	10
Port scan	0.96	10
Idle scan	0.78	8
ARP spoofing	0.83	8
custom	0.85	5
normal	0.96	-

As we can see, the neural network shows promising results in detecting both known types of anomalies and new ones. It is also apparent that detection accuracy greatly depends on the quality of the training data and the number of learning iterations.

Given the current configuration, a possible solution to improve the detection rates is to obtain a better represented dataset, as well as optimize the aggregation criteria.

VIII. Conclusions

This paper presented a way of using an artificial neural network as an anomaly detection and classification tool. A Netflow protocol was used in order to obtain and process the

data, which was then aggregated based on several properties. The results show high percentages of successful identifications after a number of iterations.

Further work will be aimed at reducing the false positive rates and optimizing the aggregation algorithms.

REFERENCES

[1] C. A. Carver, J. M. D. Hill and U. W. Pooch, "Limiting Uncertainty in Intrusion Response", 2001 IEEE Man Systems and Cybernetics Information Assurance Workshop, pp. 142-147, New York, June 2001.

[2] Robin Sommer, "Outside the Closed World: On Using Machine Learning For Network Intrusion Detection, SP '10 Proceedings of the 2010 IEEE Symposium on Security and Privacy, pp. 305-316, 2010.

[3] Salima Omar, Asri Ngadi, Hamid H. Jebur, "Machine Learning Techniques for Anomaly Detection: An Overview", International Journal of Computer Applications (0975 – 8887), Volume 79 – No.2, October 2013.

[4] Bhuyan, Monowar H., Dhruba Kumar Bhattacharyya, and Jugal K. Kalita, "Network anomaly detection: methods, systems and tools." Ieee communications surveys & tutorials 16.1, pp. 303-336, 2014.

[5] Hansman, Simon, and Ray Hunt. "A taxonomy of network and computer attacks." Computers & Security 24.1, pp. 31-43, 2005.

[6] Cisco official site, Cisco IOS NetFlow http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html

[7] Ahmad, Muhammad Aminu, Steve Woodhead, and Diane Gan, "Early containment of fast network worm malware." Information and Computer Science (NICS), 2016 3rd National Foundation for Science and Technology Development Conference on. IEEE, 2016.

[8] Demuth, Howard B., et al. *Neural network design*. Martin Hagan, 2014.

[9] Cynthia Wagner, Jerome Francois, Radu State, Thomas Engel, "Machine Learning Approach for IP-Flow Record Anomaly Detection", IFIP Networking 2011, Valencia, Spain, May 2011.

[10] Hofstede, Rick, et al. "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix.", IEEE Communications Surveys & Tutorials 16.4, 2014, pp. 2037-2064.