A Frequency-Based Approach to Intrusion Detection

Mian Zhou and Sheau-Dong Lang

School of Electrical Engineering & Computer Science
and National Center for Forensic Science, University of Central Florida,
Orlando, FL 32816, USA

Abstract

Research on network security and intrusion detection strategies presents many challenging issues to both theoreticians and practitioners. Hackers apply an array of intrusion and exploit techniques to cause disruption of normal system operations, but on the defense, firewalls and intrusion detection systems (IDS) are typically only effective in defending known intrusion types using their signatures, and are far less than mature when faced with novel attacks. In this paper, we adapt the frequency analysis techniques such as the Discrete Fourier Transform (DFT) used in signal processing to the design of intrusion detection algorithms. We demonstrate the effectiveness of the frequency-based detection strategy by running synthetic network intrusion data in simulated networks using the OPNET software. The simulation results indicate that the proposed intrusion detection strategy is effective in detecting anomalous traffic data that exhibit patterns over time, which include several types of DOS and probe attacks. The significance of this new strategy is that it does not depend on the prior knowledge of attack signatures, thus it has the potential to be a useful supplement to existing signature-based IDS and firewalls.

Keywords: network intrusion detection, intrusion simulation, time series, Fourier transform.

1. Introduction

Network intrusions and explorations are occurring almost routinely, and have become a major concern of our networked society. A recent example is the DOS attack exploiting a buffer-overflow flaw of Microsoft SQL servers which caused severe disruption to networks around the world. How to filter and detect intrusions presents many challenges to an organization, including selecting the IDS software, assessing the tradeoffs between the risks and cost factors, personnel and staffing, etc. With the help of an IDS, system administrators can more easily handle the monitoring, audit, and assessment of their systems and networks. This ongoing assessment and audit activity is a necessary part of sound security management practice.

Most security measures employed at an organization are not 100% secure. Intrusion detection systems can help the security analysts in monitoring the network and identifying intrusions, either in real time or immediately after the attack takes place. One of the challenges faced by the security managers is how to identify network intrusions and how to evaluate the effectiveness of IDS. Studying and testing a new intrusion detection algorithm against a variety of (perhaps simulated) intrusive activities under realistic background traffic is an interesting and difficult problem.

Many network attacks are executed by running pre-written scripts which automate the process of attempting connections to various ports, sending packets with fabricated payloads, etc. Based on this observation, we propose a new intrusion detection strategy which is based on the periodicity (frequency) or other statistical patterns within the traffic data. Listed below are some statistical measures that may be of interest to intrusion detection:

- The inter-arrival time of packets
- The size of packet payloads
- The time interval of the initial TCP connection attempt
- The number of the distinct IP addresses reached during a time period, etc.

The first three measures are particularly relevant to the DOS, probe, and password guessing attacks. The last measure may be observed when a worm or virus is spreading itself from an infected computer to other computers. All these statistical measures could exhibit certain patterns over time that distinguish the attack traffic from legitimate traffic. Our intrusion detection strategy is to search for frequency patterns of the connection history maintained in the firewall for protected computers.

In this paper, we adapt the frequency analysis techniques such as the Discrete Fourier Transform (DFT) used in signal processing to the design of intrusion detection algorithms. We demonstrate the effectiveness of the frequency-based detection strategy by running synthetic network intrusion data in simulated networks using OPNET software. The remainder of the paper is organized as follows. Section 2 describes our frequency-based approach to intrusion detection. Section 3 reports experimental results evaluating the effectiveness of the detection strategy using simulated network traffic data that contain DOS attack packets. Finally, Section 4 concludes the paper and points out directions for further research.

2. Frequency Based Intrusion Detection

We adapt the techniques of signal processing and data mining on time-series data to designing intrusion detection strategies. A time series consists of a sequence of numbers each representing a data value at some point in time. Examples of time-series data include financial data, stock price, weather data, network traffic throughput, etc [9]. In our research, we focus on the time series of network traffic in terms of the

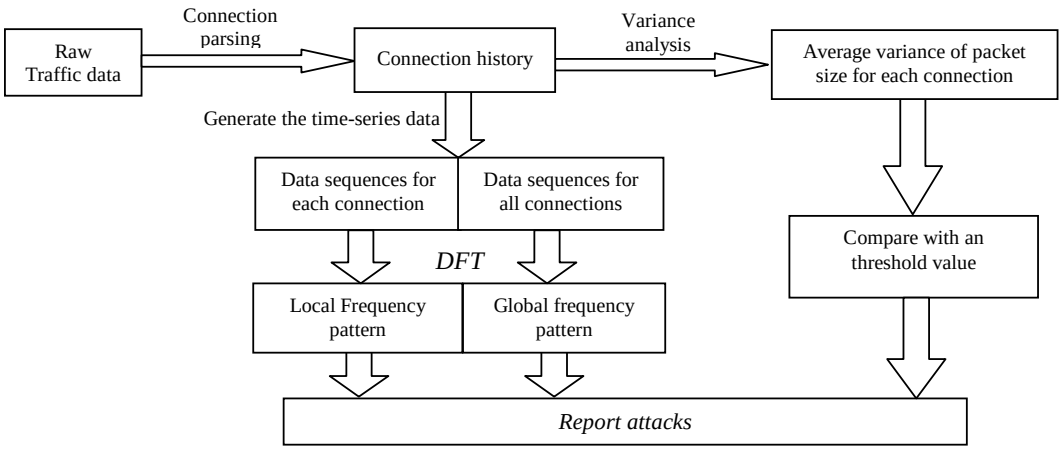


Figure 1: Strategy of frequency-based intrusion detection

number of packets received per unit time. Our observation is that attacks generated by a brute-force approach such as DOS, probes, and password guessing, will create a large number of network packets by the coded scripts, thus causing some regular patterns in the traffic data. In addition, such attacks often use fabricated payloads of a constant size. Based on these patterns, our algorithm is capable of detecting anomalous traffic behaviors and is not dependent on the specific intrusion or attack signatures.

Our intrusion detection strategy consists of three parts. First, we construct the connection history for each protected machine and convert the captured traffic data into a time series. Next, we apply the Discrete Fourier transform (DFT) to the time series data and collect the frequency information for each connection history. Both the global frequency and local frequency patterns are computed using DFT. We also compute the average variances of the packet sizes for each connection. Finally, the variance and the frequency patterns are correlated in order to identify those packets and the connections suspicious of attacks, which are then marked for further analysis. Figure 1 depicts the overall structure of the intrusion detection strategy. The details are presented in the following sections.

2.1 History of IP connections

A network connection includes all the network traffic (packets) sent between two connected IP addresses in a certain time period, which has the following properties:

- A pair of source and destination IP addresses
- One or more protocols used during the connection
- A set of packets captured during a connection which are not necessarily sent or received in a consecutive order

We assume each protected computer uses a firewall (or a filtering device) to maintain the history of recent connections and compute the frequency patterns using the DFT. For each connection within a connection history, the time series provides the number of packets per unit time, where the unit time is an adaptive value dependent on the density of the network traffic.

2.2 Frequency extraction

For a given data sequence $s(n)$ where $n \geq 0$ is a discrete value representing the time, we apply the Discrete Fourier Transform (DFT) to compute the frequency information. Fourier Transform is a well-known tool used in signal processing. The DFT takes the original time series in the time domain, and transforms them into the associated frequency data in the frequency domain. The DFT coefficients are defined as follows [3]:

$$F(k) = \sum_{n=0}^{N-1} s(n) e^{-i 2\pi k n / N}$$

Expanding the right-hand side yields the following:

$$F(k) = \sum_{n=0}^{N-1} s(n) \cos(2\pi k n / N) - i \sum_{n=0}^{N-1} s(n) \sin(2\pi k n / N),$$

where N is the length of $s(n)$. Using the Fast Fourier Transformation (FFT) procedure, the frequency data $F(k)$ can be computed in $O(N \log N)$ time.

2.3 Global frequency vs. local frequency

Both the local frequency patterns within each connection and the global frequency patterns over all connections are analyzed in our method. Basically, the local frequency patterns can be used in detecting attacks originated from a single source IP targeting a single victim IP. However, when an attack sends packets using multiple spoofed IPs as their source addresses, aimed at misleading the detection systems, we will see many incoming connections to the target, but in fact, the attack traffic is not independent and is possibly generated from one source computer. The global frequency analysis is effective for this type of attacks.

In real-life environments when connections are established between different computers, the connection traffic from one connection is statistically unrelated to the connection traffic from others. Therefore, there should be no frequency patterns existing over the different connections, and we can use the local frequency patterns to detect the uncoordinated attacks occurring in different connections.

2.4 Average variance of packet sizes

The average variance ζ of the packet size $p(i)$, $1 \leq i \leq n$, for a connection is defined as follows:

$$\zeta = \frac{1}{n} \sum_{i=1}^{n} \sqrt{(p(i) - \mu)^2},$$

where n is the number of packets within a connection., and μ is the mean of packet sizes defined as follows:

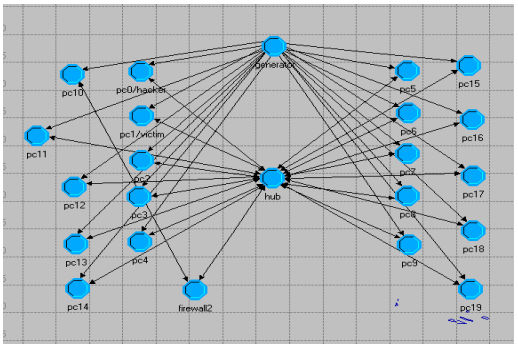


Figure 2: The network model used in simulating ProcessTable and sshProcessTable attacks

$$\mu = \frac{1}{n} \sum_{i=1}^{n} p(i)$$

When the goal of an attack is to consume the resources of the victim machine or to probe the information, the payload of each packet normally would be fabricated and is of a similar size. As a result, the average packet size variance ζ would be relatively small for attack traffic compared to that of the normal traffic. Thus, this factor is used in our approach for detecting intrusions.

2.5 Testing the Detection Strategy

To implement our intrusion detection strategy, a firewall is placed on each of the target machines to capture the network traffic data. The firewall maintains a table of the recent connections within a given time window. Each table entry contains an IP address that has been connected to the protected computer at least once within the time window. The traffic data of each connection is converted to a time series; the average variance of the packet sizes is computed and recorded by the firewall. In addition, information on connected port is recorded which will be useful in detecting probe attacks. Finally, the firewall will report the suspicious IPs based on the observed frequency patterns and the variance of packet sizes. In the next section, we will describe our intrusion simulation framework using OPNET [7], and report the simulation results evaluating the frequency-based intrusion detection strategies.

3. Simulation Studies of Network Intrusion Detection

Intrusion detection algorithms are typically effective in detecting intrusions of known signatures but poor in detecting new attacks. Studying and testing a new intrusion detection algorithm can be performed either in a real environment or in a simulated environment [8,10]. We first briefly summarize our approach to intrusion detection simulation.

3.1 Network Intrusion Simulation Using OPNET

OPNET and NS2 are two popular tools used for network simulation. Our studies use OPNET [6] as the simulation platform for testing the intrusion detection strategies.

We use two data sources for the network traffic: one comes from the MIT/Lincoln Lab TCPDUMP files, which contain intrusion traffic simulating various network attacks [2]. The second data source is the synthetic data generated by the network sniffer tool Ethereal [1], which is installed on our local network capturing both the intrusion traffic and normal traffic. The intrusion traffic was generated by NMAP, a network exploration and auditing tool [5].

For both the TCPDUMP files and Ethereal files, we use our own pre-processing tools to extract the traffic information that is necessary for OPNET simulation. After parsing the TCPDUMP files and Ethereal files, our tool extracts the data packet headers, the flag information, and the time distribution. Figure 2 presents a typical network model used in OPNET simulation. The top node in the center column is a "generator," which generates the packets extracted from the traffic source and distributes them to the simulated network. Once a packet is ready, it is given to its source PC node, and from there it will be sent to the destination PC node through the hub (located at the bottom of the center column).

The number of virtual PCs is determined by the outcome of pre-processing the source traffic file. Since there are twenty distinct IP addresses in the original traffic data, this model uses twenty PC nodes connected to each other through a hub to simulate a network environment. We put a firewall node between node zero (the attack victim) and the hub to monitor suspicious data traffic. The intrusion detection algorithm is implemented inside the firewall node, where we could test our intrusion detection strategies. More details about simulating intrusions using OPNET can be found in [7].

In our studies, we simulated DOS attacks and remote-to-root attacks using the MIT/Lincoln Lab TCPDUMP files, and we tested our detection algorithm with these attacks. The following sections report the simulation results of these attacks, and illustrate how to detect them by extracting the frequency patterns.

3.2 The ProcessTable Attack

The ProcessTable attack is TCP connection based attack, which may attack various network services by launching a huge number of TCP/IP connections to a particular port in a short time period. For each incoming TCP/IP connection, the underlying Unix system allocates a new process to handle the connection. Therefore, it is possible to completely fill a target machine's process table with multiple instantiations of network services, eventually rendering the system lifeless until either the attack terminates or it is killed by the system administrator [4]. The SshProcesstable attack is similar, except that it uses *ssh* to establish connections.

We set up a network model in OPNET using 20 PC nodes since there are up to 20 distinct IP addresses involved in the traffic sources for Processtable and sshProcessTable attacks. The network model is showed in Figure 2. The MIT/Lincoln Lab TCPDUMP file used in ProcessTable attack simulation contains less than 2 minutes of data with a total of 5526 data packets. We collected two statistical measures in the firewall node attempting to detect and identify the ProcessTable attack.

Figure 3 depicts the number of distinct port connections to the victim PC during simulation. It can be seen very clearly that there are 3 jumps in the graph, indicting rapid increases of port connections to the victim during 3 distinct time intervals.

The ProcessTable attack can also be directed at a particular port of the victim. Figure 4 depicts the network traffic directed to Port 25 of the victim during simulation. The graph displays two peaks: the first occurred around the one-minute mark; the second started after one minute 20 seconds and lasted to the

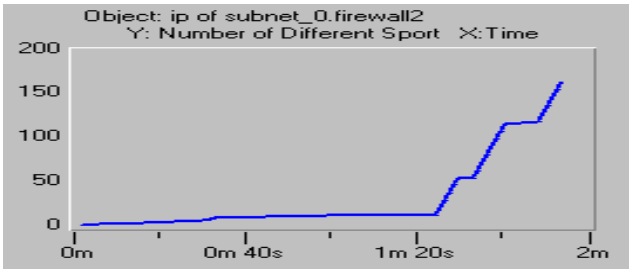


Figure 3: Number of distinct port connections to the victim.

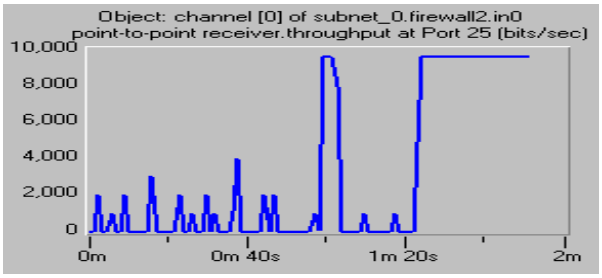


Figure 4: Data traffic to Port 25 of the victim PC

end. Thus, the two figures 3 and 4 clearly demonstrated data packets that are suspicious of the ProcessTable (or similar) attacks.

To further identify the sources of the intrusion traffic, our detection algorithm builds the connection history table for the target computer. Each record of a connection contains five fields, including the IP address, duration, and number of packets, average variance of the packet size, and the DFT coefficients. By analyzing the last three fields, the firewall marks the suspicious IPs. For example, a record of the connection history for the target computer "zeno.eyrie.af.mil" is given below showing the five fields:

IP address: pc7.eyrie.af.mil (172.16.117.52)
Duration: Starts at Apr 7, 1999 15:01:05.301579000 ends at Apr 7, 1999 15:31:50.040396000
Number of packets: 4724
Average variance (packet size): 1.172802
The DFT coefficients: 6.6765, 4.2667, 3.3488, 3.2531, 2.0636, 2.6532, 3.8849…

Based on the DFT coefficients (the last column) in the table, the frequency patterns of each connection are plotted in Figure 5. There are a total of 12 connections in the connection history table. Five of them contain less than 20 packets and two of them less than 40 packets. Figure 5 plots the frequency information of six connections that are long enough to bear meaningful DFT

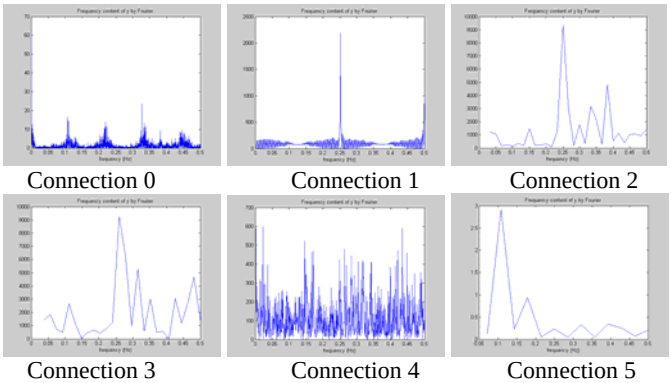


Figure 5: Frequency patterns of each connection in detecting the ProccessTable attack

analysis.

It can be seen that the frequency plots for connections 0 and 1 show distinctive peaks, while there are no clear peaks for the rest. Since the average variances of packet sizes for connections 0 and 1 are, respectively, 1.172802 and 0.482538, which are below the threshold value 3.0, both connecting IPs are marked as suspicious. Further analysis revealed that the first connection is from a ProcessTable attack, and surprisingly, the second one is actually a Probe attack, which probes the target PC's ports ranging from 1794 to 2631. We identified the attacker's IP address for the probe attack as "hobbes.eyrie.af.mil (172.16.112.20)".

3.3 The sshProcessTable Attack

The source traffic data for the sshProcessTable attack is from MIT/Lincoln Lab, Thursday of week 4 inside data. There are 10 connections recorded in the connection history table for the target computer "pascal.eyrie.af.mil" (IP 172.16.112.50); information about the connections is listed in Table 1:

	IP address	Number of Packets	Average variance(ps)
0	196.37.75.158	686	0.000333
1	197.182.91.233	3991	1.490531
2	linux2.eyrie.af.mil	1869	0.060086
3	196.227.33.189	181	0.516071
4	www.a-be.org	232	0.459419
5	172.16.114.50	7	252.441605
6	hume.eyrie.af.mil	1	0.000000
7	172.16.113.105	174	0.668880
8	zeno.eyrie.af.mil	32	12.755207
9	191.216.13.135.in-ddr.arpa	11	6.808864

Table 1: The connection history from the sshProcessTable attack

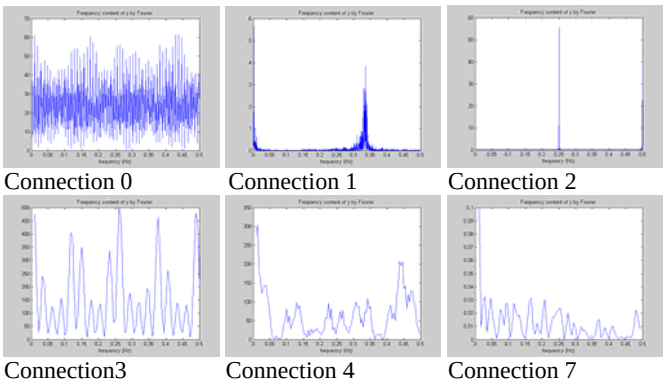
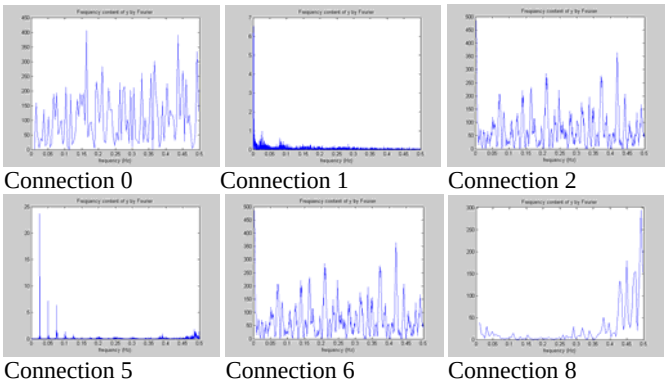


Figure 6: Frequency patterns of each connection in detecting the sshProccessTable attack.

The frequency patterns for connections 0, 1, 2, 3, 6, and 7 are depicted in Figure 6 (we chose connections that have sufficiently long packet sequences for the DFT analysis). We observed that the plots for connections 1 and 2 have distinctive frequency patterns, and that the average variances of the packet size are 1.490531 and 0.060086, respectively, small enough to be marked as suspicious. Thus connections 1 and 2 are likely to



Connection 0 Connection 1 Connection 2

Connection 5 Connection 6 Connection 8

Figure 7: Frequency patterns of each connection in detecting the Dictionary attack.

be automated attack traffic. For both of these two DOS attacks, there was no global frequency patterns observed.

In addition to the DOS attacks, other types of attack such as the Dictionary attack, which belongs to the type of remote-to-root attack, also falls into the scripted attack category. Figure 7 plots the frequency patterns for the dictionary attack, where the source data comes from MIT/Lincoln Lab's 1999/Monday of week 5 TCPDUMP files. There are a total of 11 connections to the victim computer "marx.eyrie.af.mil". The frequency patterns are plotted in Figure 7. It can be seen that there is a clear frequency pattern for connection 5. After checking the packet signature, we confirmed that this connection contains a password guessing attack, which tried different username/password combinations to the telnet service. Our algorithm successfully identified the presence of an attack without the prior knowledge of the attack signature.

4. Conclusions and Future Work

In this paper, we proposed a novel frequency-based intrusion detection strategy to detecting the automated, scripted attacks, which typically exhibit frequency patterns over time. We integrated this strategy into a simulation framework using OPNET. With the aid of our pre-processing tools, we extracted pertinent information from the previously captured TCPDUMP data sources, and instantiated the OPNET simulation model. We used the datasets from the MIT/Lincoln lab in our simulation studies; the experimental results demonstrated that the frequency-based intrusion detection approach is effective in, but not limited to, detecting the DOS and probe attacks that typically run from pre-written scripts and have relatively long duration.

For future work, we plan to pursue an overall assessment of our frequency-based detection algorithm by testing the complete attack database from MIT/Lincoln Lab. We also plan to integrate our intrusion detection strategy into a firewall device of a live system, and evaluate the detection effectiveness of our approach in real network environments.

References

[1] Gerald Combs, Ethereal – Network Protocol Analyzer, http://www.ethereal.com/

[2] DARPA Intrusion Detection Evaluation project, at http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html.

[3] Kyoji Kawagoe, Tomohiro Ueda, "A similarity search method of Time series data with combination of Fourier and wavelet Transforms," in **Proceedings of 9th International Symposium on Temporal Representation and Reasoning**, pp. 86-93, 2002.

[4] Kristopher Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems," **Master's Thesis, Massachusetts Institute of Technology**, 1998.

[5] NMAP, at http://www.insecure.org.

[6] OPNET online documentation 8.0.C, OPNET Technologies, Inc., Washington, DC.

[7] S. Razak, M. Zhou, S.D. Lang, "Network Intrusion Simulation Using OPNET", in **Proceedings of 2002 OPNETWORKS Conference**.

[8] Tao Wan, Xue Dong Yang, "IntruDetector: A Software Platform for Testing Network Intrusion Detection Algorithms", in **Proceedings of 17th Annual Computer Security Applications Conference**, 2001.

[9] Yi-Leh Wu, D. Agrawal, A.E.Abbadi, "A Comparison of DFT and DWT Based Similarity Search in Time-Series Databases," **Proceedings of the ninth International Conference on Information and Knowledge Management**, pp. 488-495, 2000.

[10] Zheng Zhang, Jun Li, C.N. Manikopoulos, Jay Jorgenson, Jose Ucles, "HIDE: A Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification," in **Proceedings of 2001 IEEE Man Systems and Cybernetics Information Assurance Workshop**, pp. 85-90, 2001.