Network Intrusion Detection System – A Novel Approach

Krish Pillai, Ph.D
Department of Computer Science
Lock Haven University of Pennsylvania
Lock Haven, PA 17745, U.S.A
kpillai@lhup.edu

ABSTRACT

Network intrusion starts off with a series of unsuccessful break-in attempts and results eventually with the permanent or transient failure of an authentication or authorization system. Due to the current complexity of authentication systems, clandestine attempts at intrusion generally take considerable time before the system gets compromised or damaging change is affected to the system giving administrators a window of opportunity to proactively detect and prevent intrusion. Therefore maintaining a high level of sensitivity to abnormal access patterns is a very effective way of preventing possible break-ins. Under normal circumstances, gross errors on the part of the user can cause authentication and authorization failures on all systems. A normal distribution of failed attempts should be tolerated while abnormal attempts should be recognized as such and flagged. But one cannot manage what one cannot measure. This paper proposes a method that can efficiently quantify the behaviour of users on a network so that transient changes in usage can be detected, categorized based on severity, and closely investigated for possible intrusion. The author proposes the identification of patterns in protocol usage within a network to categorize it for surveillance. Statistical anomaly detection, under which category this approach falls, generally uses simple statistical tests such as mean and standard deviation to detect behavioural changes. The author proposes a novel approach using spectral density as opposed to using time domain data, allowing a clear separation or access patterns based on periodicity. Once a spectral profile has been identified for network, deviations from this profile can be used as an indication of a destabilized or compromised network. Spectral analysis of access patterns is done using the Fast Fourier Transform (FFT), which can be computed in Θ(N log N) operations. The paper justifies the use of this approach and presents preliminary results of studies the author has conducted on a restricted campus network. The paper also discusses how profile deviations of the network can be used to trigger a more exhaustive diagnostic setup that can be a very effective first-line of defense for any network.

Keywords: Fast Fourier Transforms, statistical anomaly detection, network security, signal processing, surveillance, traffic analysis, intrusion detection

1. INTRODUCTION

Case studies have long indicated that legitimate users of network resources have a specific behavioural profile. It follows that changes in usage profiles can be used to detect and differentiate a masquerader from a regular user [1] , on the basis of reference signature profiles constructed from logged information. In the simplest approach, audit trails can be used to build a signature profile for normal users. More sophisticated Intrusion Detection Systems (IDS) may generate metrics from audit records to build profiles. The use of interval timing between accesses, session counters, and resource utilization tracking has been used successfully to generate reference profiles.

Such Intrusion Detection Systems generally collect and analyze time domain volume data such as number of logins per hour, program usage, and group resource or file system usage statistics. These are compared to thresholds which if exceeded could indicate a possible intrusion. Denning's classic paper classifies Statistical Models as follows [2] :

- Operational Model
- Mean and Standard Deviation Model
- Multivariate Model
- Markov Process Model
- Time Series Model

The Operational Model tracks abnormality by comparing metrics generated by event counters to an operational reference. On the other hand, the Time Series Model maintains event counters that are used to keep track of periodicity, and thereby the frequency of events. These two models, though cognizant of the periodicity of various events that occur in the system, fall short of providing a global view of the operational model of the system.

To illustrate this exposure, consider a simple counter that maintains a count of Secure Shell (ssh) logins over time (C_{ssh}), with a view to comparing it with a reference as in an Operational Model. Establishing a tolerance limit for C_{ssh} will not be easy since the aggregate logins being measured is the sum of multiple login cycles across various users. Clients log onto the network at individual times of the day based on cycles that are different for each one of them. As a result these cycles of differing periodicity may interfere constructively or get evened out causing the standard deviation of the C_{ssh} value to fluctuate over a wide range when viewed entirely in the time domain. In other words, the metric C_{ssh} being measured tends to have cycles within cycles as a result of the convolution of various usage patterns. Setting tight tolerance limits on instantaneous C_{ssh} may cause false alarms while setting wide tolerance limits to accommodate occasional spikes may result in a weaker detection system.

On the other hand, generating a spectrum of C_{ssh} will yield a frequency domain view of the metric. The samples contributing to each cycle would be available for individual inspection since spectral content is represented along the frequency axis. As a result, interference between high frequency and low frequency cycles would be easily discernable. A reference spectral profile can now be established for the spectrum of the metric C_{ssh}, where tolerance limits for each cycle could be set independent of the other. For example, a network may have a regular metric that cycles on a daily basis from nine-to-five biased by office hours.

For the sake of illustration, let us assume that superimposed on this regular traffic, there is a sharp increase in the measured metric twice every week on Tuesdays and Thursdays, due to the development team working on code compilation related issues. The traffic from the development team adds considerably to the regular traffic on these two days, but not on other days. Setting a threshold that changes dynamically on specific calendar dates is not feasible. Additionally, setting a single limit to represent the highly variant peak is not safe since there is the same likelihood of an intrusion on a busy day as on other days. Representing the metric in frequency domain and maintaining limits on the spectral profile allows the IDS to keep track of references to specific cycles that occur for the metric being monitored.

There are several algorithms for generating discrete frequency domain data from raw time domain data, of which the algorithm most widely used in signal processing is the Fast Fourier Transform (FFT). Spectral analysis is also done using Wavelet Transforms (DWT), especially in the area of pattern recognition or extraction. DWT though computationally less complex, is unsuitable for this specific purpose since we are not looking for time domain patterns but rather its stability. Furthermore, the advantage of FFT over DWT is that in addition to providing a Dirac comb (series of Dirac-Delta impulses) that is modulated by a series of discrete-valued coefficients that are also complex-valued, the basis functions are simple sinusoids and cosines. This is particularly useful since we only have knowledge of the periodicity of the pattern we are attempting to identify and not its nature. DWT on the other hand presumes knowledge for this pattern in time domain, so that a basis function can be generated. Moreover, FFT can be easily computed in real time using the Cooley-Tukey algorithm with very low computational complexity, $\Theta(N \log N)$, equivalent to that of a simple sorting algorithm.

The preliminary step to being able to detect variations in network usage is to establish a reference frequency-domain reference profile. Once a reference pattern has been established, the near-real-time profile can be checked against this reference for deviations from the normal and tolerance limits can be established. A method for defining and capturing a reference profile is detailed in the following section.

2. PROFILE DEFINITION AND METRICS

A behavioural profile can be based on various metrics that may be defined at any layer of the TCP/IP protocol stack. A metric such as SSH logins would count shell sessions and would be an *application* layer metric. Counting the number of TCP or UDP sessions would be a *transport* layer metric. Generating audit information for Internet Control and Messaging Protocol (ICMP) [4] messages would constitute a *network* layer metric. Metrics collected at lower layers of the TCP/IP stack would encapsulate metrics collected at upper layers. For example, a count of TCP sessions would include Email, SSH, as well as any other application layer protocol that is transported on TCP.

The choice of a metric is biased by the nature of the network under surveillance. For example, in a network that deploys an email server and a secure shell server, stronger cycles may be observable at the transport layer than at the application layer if most users use SSH redirection to get to their email service. Metrics for monitoring have to be chosen on the basis of the most vulnerable protocols that needs to be protected.

2.1 Defining a profile for a network

On a typical campus LAN (Local Area Network), there are servers and there are clients. Typically the services deployed on a network, such as Network File Service (NFS) or the Light Weight Directory Access Protocol (LDAP) would run periodically, driven by client access. Using LDAP access or NFS mount requests from clients as metrics would generate profiles that correlate very strongly with user access. For example repeated attempts to gain authentication would translate to perturbations in LDAP profiles, and attempts to use a compromised LAN port to gain access to distributed file systems would translate to failed mount requests, which in turn would show up in NFS profiles.

The simplest profile may consist of aggregate data or packet count collected periodically from the network for specific metrics. But metrics can be combined to create a palette of metrics for which spectral data can be arrived at independently. A palette may be limited to key protocols that are of special interest to the network administrator.

A typical metrics palette would be-

- SSH sessions and email sessions
- Java and PHP scripts
- NFS/LDAP

The key protocols in the aforementioned palette are SSH, SMTP (Simple Mail Transfer Protocol), Hyper Text Transfer protocol (HTTP), NFS and LDAP. The process of generating the palette spectrum involves collecting raw time domain date for each key protocol and then running the FFT algorithm on each time record. The following section explains the steps involved and the pre-processing that needs to be done before FFT can be applied.

2.2 Generating time domain statistics

Pegging metrics on the network can be done with minimal engineering impact through port duplication on the default gateway or ingress point into the network. All traffic ingressing and egressing the subnet should ideally be duplicated off to a port that connects to a machine dedicated to metrics collection, so that processing overheads do not impact network bandwidth. Alternately the data can be dumped to a file for near-real time or off-line analysis. Processing a file with time-domain metrics information involves the following steps-

- Filtering out instantaneous packet counts for each metric
- Aggregating them into counts over clearly defined time intervals (packets per minute is recommended) to extract time records for the metrics palette
- Preparing each time record for spectral analysis by averaging several time records and assigning weights them
- Application of Fast Fourier Transforms to the time records

A time record represents the window of time that is of interest to the observer. The FFT algorithm assumes that the time record repeats endlessly. Therefore, it is important that the time record captures the operational behaviour of the network and can be used as a reference for comparisons. The time record is dependent on the periodicity of the metric being measured, and should last long enough to capture at least one cycle of the metric of interest. Each time record is composed of samples of the metric that are taken periodically. The sampling rate of any metric should be at least twice the maximum cycling frequency one expects to observe for a metric, as stipulated by the Nyquist-Shannon sampling theorem[3] . The more samples in a time record, the higher the resolution of the analysis would be. Choosing the time record for FFT analysis is critical and should be done on the basis of prior knowledge of the metric being sampled. As an example, a calendar week is a good candidate for a facility that is sparingly used on weekends, but has a usage pattern that repeats every week.

Once a time record is captured, the next step in processing the profile is the translation of these records to frequency domain using Fast Fourier Transforms (FFT). If ‘f’ is the frequency of a metric cycle detected by the transform, and ‘$2\pi f$’ is denoted by ‘ω’, then a value $K(\omega)$ can be generated for each $K(t)$ acquired. Viewing data in frequency domain has the effect of highlighting cycles in metric variations and separating them for each frequency. In other words, FFT takes a time record containing samples along the time axis and transforms them to amplitudes along the frequency axis.

If ‘N’ denotes the number of samples in a time record, then FFT generates ‘N’ points of the spectrum of which only $N/2$ represent positive frequencies since the transform yields complex valued coefficients. As a result, the spectrum generated by FFT is basically reflected about the Y-axis. Each frequency point generated by the FFT algorithm is referred to as a ‘*bin*’, and for this purpose only the positive bins are of importance. Since phase information is of little value the negative valued bins can be discarded. The resolution of the spectrum generated by the FFT algorithm depends on how well the time record lines up with variation in the metric. Since this is not controllable easily, additional processing should be done on the time record before FFT can be applied. A time record may start capturing a metric at any point within its cycle and may end at any arbitrary point, as data acquisition is not synchronized with metric variations. Since FFT assumes that time records repeat, truncated cycles can leads to spurious results. Large variations in metric values at the beginning and end of time records can cause frequency values in one bin to leak into adjacent bins, the effect being a blurring of the frequency profile. To avoid this, the beginning and end of each time record should be smoothed out with a process called *windowing*. Windowing is the multiplication of each time-record of data collected with a function that is zero at the beginning and end of the time record. This has the affect of weighing down the values of samples taken at the beginning and end of each record so that their contribution to the algorithm is diluted, reducing frequency domain blurring. The literature describes many such windowing functions. The Hanning, Rectangular, Blackman-Harris, Kaiser, and the Tukey window are extensively used in conventional signal processing.

The Tukey window [5] gives fairly good amplitude accuracy and is simple enough for this application domain. This window gave consistent results when applied to weekly time records in this case study. The windowing function should always be chosen on the basis on what needs to be observed. For example, if transient load variations are of importance, then no window or a Uniform Window Function (UWF) should be used (Figure 1.), where each value measured in the time domain is equally weighted within the time record.

To summarize, the steps involved in capturing a metric profile are as follows:

1. Definition of the metric palette
2. Identification of time record for each metric
3. Data acquisition and generation of time domain data
4. Choice of windowing function and multiplication of each time record with the windowing function
5. FFT analysis of the time records using the Cooley-Tukey algorithm

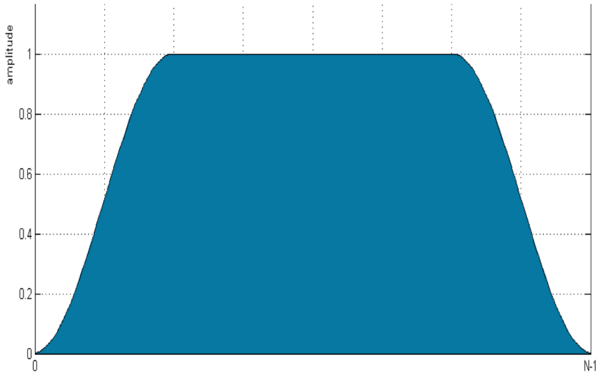

Figure 1. Tukey window to highlight data collected over week days

3. TIME DOMAIN VERSUS FREQUENCY DOMAIN

Time domain metrics collection gives a simple two-dimensional view of the behaviour of the network. The presence of cycles in the metric under observation is largely undetectable, especially if cycles of different periodicity coexist within the time record. For example, consider a Master-Slave file server system where the slave machine transfers files from the master every ten minutes. Assuming a secure setup, this would involve the slave using a protocol such as ssh to remotely copy files over 144 times a day from the master as super user. If ssh is the metric being measured, then a stable system should consistently show this base value, over which other attempts to login as super user onto the master machine would be superimposed. A system administrator may login regularly during office hours to monitor the system, and this would translate to a slower cycle with office-hours periodicity, super imposed on the base profile. An attempt to break into the master system on a daily basis by a potential intruder would affect the profile and would introduce more cycles or perturbations. Though these attempts might not affect the aggregate value of ssh login attempts substantially, it will definitely have a different periodicity from the normal behaviour of the ssh metric being collected.

To obtain a theoretical view of the application of FFT, one may visualize the network as a “black box”. The network system takes an input and responds by producing an output. In the most simplistic model, the network may be categorized as a Single Input Single Output or a SISO system [6] . The input could be in reality an aggregation of several metrics or a metrics palette. Likewise, the output may be a composite attribute built up of several metrics. The discussion in this paper is limited to a single metric acquisition system for the sake of brevity.

There are certain interesting relationships between time domain and frequency domain metric variations. An impulse spike in the time domain for a metric will affect all bins in the frequency domain. A perfect cycle in the time domain on the other hand will show up as a spike in the frequency domain.

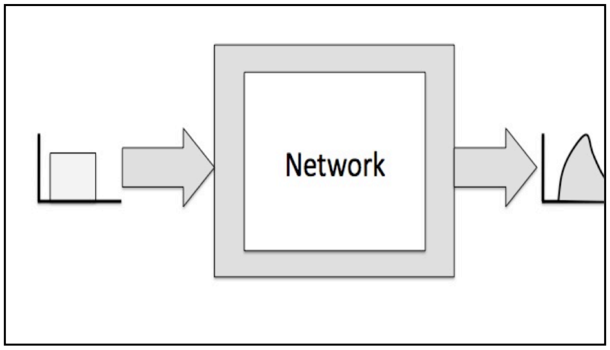

Figure 2. The network as a SISO block

Observation of metric patterns in frequency domain yields information on how the network responds to service requests. Standard control theory suggests that there is no non-determinism in how the network affects various traffic flows within the constraints imposed by the model. The objective of this paper is to use FFT to identify a signature pattern in the metric of interest, so that a reference can be established.

3.1 Capturing Metrics in Time Domain

In its simplest form the metric being observed may simply be the total number of packets flowing into the system. In the following sections, a simple traffic metric is used to illustrate the process of identifying reference patterns. Aggregate traffic can easily be captured using software such as MRTG to poll counters on a gateway or ingress point into the network. If specialized metrics are to be pegged, then more advanced software would have to be used. IPtables is an ideal user space application that allows for traffic accounting. IPtables is configured on the basis of rules that match a wide range of criteria. Rules can be implemented to match the criterion for a metric and a count of the number of times the rule gets matched can be generated. The IPtables device can then be periodically polled to generate a fairly accurate count of the metric under consideration. In the following section, we take a look at a simple traffic metric that establishes a reference profile and monitors for deviations from the reference.

3.2 A Case Study – Traffic volume as a Metric

A conventional Simple Network Management Protocol (SNMP) Management Information Base (MIB) was used to gather time domain information for the traffic volume metric. This metric is simply a bit-rate count in its simplest sense. An open source application (Multi Router Traffic Graphing Application - MRTG) was used to query the MIB [7] on the router to generate the necessary information. The average of aggregate packet flow with a five-minute sampling rate collected with MRTG is shown below. Each hour yields twelve samples and this translates to 288 samples a day. The data shown in Figure 3. shows a clear periodicity for every 288 samples or on a daily basis. This profile includes user activity superimposed over the no-load profile of the network under observation.

The network being observed has a Master-Slave file service and a centralized Network Information System (NIS) for authentication and authorization. Client machines have their file systems supplied from the file server using the "*automount*" protocol. Automount is a protocol that periodically mounts files on demand and dismounts folders from clients during moments of inactivity to optimize network traffic.

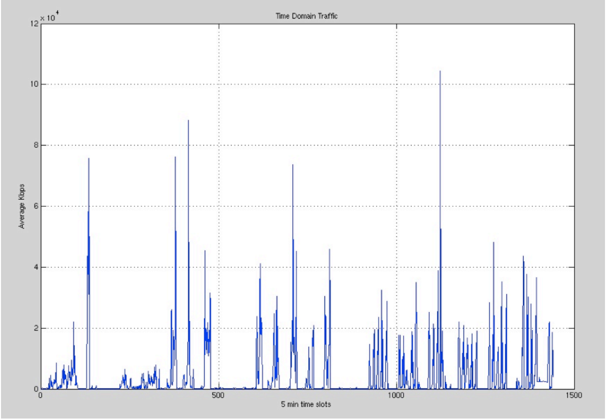

Figure 3. Five minute sample of traffic showing repetitive pattern

Distinct patterns are observable during periods of inactivity or no load, when the only traffic on the network is that generated from applications that run periodically to maintain network connectivity and the functionality of the distributed system. The figure shows traffic profile captured over a Sunday at the university UNIX Lab. The cyclic traffic from control software is evident in this graph.

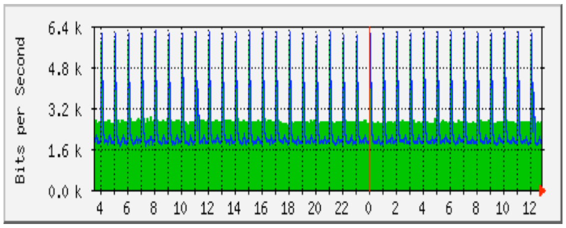

Figure 4. No-Load traffic pattern on a Sunday

A frequency spectrum would show a clear cycle associated with this record. A weekly profile, magnified here to accommodate the peaks, indicates a distinct deviation from the daily patterns, particularly on Tuesdays and Thursdays. This deviation from the pattern can be correlated to the fact that the database design class is scheduled on Tuesdays and Thursdays when client-server traffic is considerably elevated.

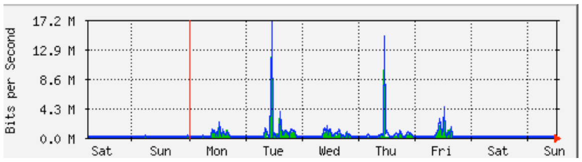

Figure 5. Weekly profile showing high activity on Tue and Thu

Converting from time to frequency domain using transient data yields unreliable results. The resulting spectrum can be stabilized considerably through *ensemble averaging*. This is basically the process of creating multiple spectra and combining them to provide a stable spectrum. As data is gathered over five minute intervals, a running average using a sliding window of a fixed number of profiles can used to track the near real-time spectrum and compare it with the signature for deviations.

For the empirical study conducted, the signature itself was built up of data captured on a typical day. FFT computation gives more stable profiles if more points are used. Since this data represents a typical day, larger calendar records were artificially generated through extrapolation and transformed to get a clearly defined signature profile.

3.3 Identifying a Signature Profile for the Metric

A unique signature would have peaks that are invariant in the profile as multiple ensembles are acquired. During periods of abnormal activity the cycles generated by usage traffic will be superimposed on the reference profile. Permanent changes in the reference profile could indicate the no-load behaviour of the network has changed, or the network steady state has been affected in some way. This could be indicative of misconfiguration or that some permanent alteration to the system has taken place. Selection of tolerance limits would require knowledge of load variations and deviations on a network. A few weeks of historical data would therefore be necessary to define a profile, and is not something that can be arrived at apriori. Once tolerance limits are established around a reference profile for the metric, deviations beyond tolerance limits can be used to trigger a Network Management Service (NMS) alarm. Multiple such profiles may be generated for a palette of metrics so that deviations along a range of values could be monitored for instability.

```
% Load file containing typical day run
load dayrecord.dat;

% Number of points to use for FFT computation
NPOINT = 256;

% Spectrum is scaled for easy graphing
% We are not interested in Power Spectral density
% Only the resonances matter
SPEC_D = abs(fft(dayrecord,NPOINT))/1000;

%
% Now generate a frequency scale that runs
% from 0 - (1 - 1/N)
% FFT yields real and imaginary components
% The whole spectrum is a reflection about the
centre
% Graph only half the screen
FREQ = [0:NPOINT-1]/NPOINT;
FREQ = FREQ *2;

plot(FREQ,SPEC_D),grid on, title('Daily Profile'),
axis([0 1 0 1000]);
```

Figure 6. Matlab code for generating FFT

A signature derived for the network under study is shown in Figure 7. The peaks circled identify the invariant part of the profile. FFT on the data set was computed using Matlab® [8] as a proof of concept, but will be implemented eventually using functionality that is part of an in-house application to facilitate this analysis.

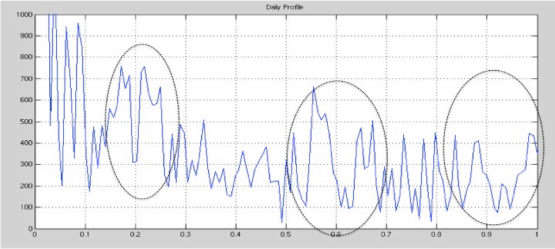

Figure 7. Signature for the lab under observation

The Matlab code snippet is fairly straightforward and can easily be implemented in C or Java using the Cooley-Tukey algorithm [9] . Packet capture can be implemented using libpcap [10] and supplemented with scripting and extraction from IPTables audits.

4. FUTURE WORK AND CONCLUSIONS

More data is being collected and various metrics are being investigated as part of this research. The author is in the processes of categorizing various use-case scenarios and mapping each to a metrics palette that would react to intrusion attempts in a timely manner. Validation would require extensive testing across various scenarios and will be done as more audit records and logs become available from various network service administrators. The functionality of the proposed detection system can be further enhanced using Artificial Intelligence (AI) [11] . The spectral analysis method described in this paper would add considerable value to a detection system that is driven by an AI Inference Engine, making it cognizant of variations in usage patterns. Identifying a signature profile for a metric and being able to represent this important attribute in quantitative terms makes it possible to use an automated scheme to track and monitor the network, and react in more sophisticated ways than just generate alarms.

As part of future work, the author plans to use this technique of monitoring networks as a backend to an expert system. The advantage of using Artificial Intelligence in network surveillance opens up various possibilities in the field of automated Intrusion Detection Systems.

References:

[1] James P. Anderson, “Computer Security Threat Monitoring and Surveillance”, **Contract 79F296400, National Institute of Standards and Technology**, April 15, 1980.

[2] Dorothy E. Denning, “An Intrusion Detection Model,” **IEEE Transactions on Software Engineering**, Vol. SE-13, No. 2, pp. 222-232, February 1987.

[3] C. E. Shannon, “Communication In The Presence of Noise”, reprint as classic paper, **Proc. IEEE**, vol.86, no.2, (Feb. 1998)

[4] J. Postel, “Internet Control Message Protocol,” **Internet Engineering Task Force (IETF) Request for Comments (RFC) 792***,* Standard, September 1981.

[5] V. Oppenheim, R.W. Schafer and J.R. Buck, **Discrete-Time Signal Processing**, Prentice Hall, ISBN: 0-13-754920-2

[6] B.C.Kuo, and F. Golnaraghi. **Automatic Control Systems**, 8th Edition, Wiley, ISBN-13: 978-0471134763, September 2002

[7] D. Harrington, R. Presuhn, and B. Wijnen, “An Architecture for Describing Simple Network Management Protocol management Framework,” **IETF, RFC 1157**, Standard, December 2002.

[8] David M. Smith, **Engineering Computation with MATLAB®,** ISBN-13: 978-0-321-48108-5, Addison-Wesley Computing, pp.369–373.

[9] Cooley, James W., and Tukey, John W., “An Algorithm for the Machine Calculation of Complex Fourier Series” in **Magnetism, Math. Comput**. vol: 19, pp.297-301, 1965.

[10] S. McCanne, and V. Jacobson, “The BSD Packet Filter: A New Architecture for User-level Packet Capture,” **USENIX Conference**, January 25-29, 1993, San Diego, CA, U.S.A.

[11] S. Russell, and P. Norvig, **Artificial Intelligence: A Modern Approach,** Prentice Hall, 3rd Edition, ISBN-13: 978-0136042594, December 11, 2009.