# Investigating False Positive Reduction in HTTP via Procedure Analysis

A.A Abimbola, J.M Munoz and W.J Buchanan
a.abimbola@napier.ac.uk , School of Computing, Napier University, EH10 5DT, Scotland, UK

## Abstract

*This paper focuses on high false-positive rate of attacks. First, the merits and demerits of research work in curbing false positive rate of attacks in Intrusion Detection Systems (IDSs) are discussed. Then we present our research efforts in the form of an IDS called NetHost-Sensor, recap on past NetHost-Sensor research contributions and discuss in detail its novel procedure analysis technique in curbing false-positive. We discuss in detail, the NetHost-Sensor methodology, its procedure analysis technique and report on our experimental investigation that shows the reduction of false-positives, using HTTP network communication as a medium for analysis. Finally, we validate our research work by comparing false-positives with Snort IDS.*

## 1. Introduction

The popularity of web servers and web-based applications has increased, owing to the increase in network connectivity [1]. To detect web-based attacks, Intrusion Detection Systems (IDSs) are configured with a number of intrusive signatures that detect known attacks. For instance, Snort 2.3 IDS [2] devotes more than 90% of its total intrusive signatures to detecting web related attacks. Unfortunately, it is difficult to update IDS intrusive signatures in respect to the rate of new vulnerabilities discovered. Also, these intrusive signatures are normally specific to pattern matching variant of similar attacks [3]. To tackle these issues, other subsets of signature-based techniques [4] and a completely different approach called anomaly-based detection technique, are employed [5]. This paper presents the results of research aimed to reduce false-positives. In section 2 it discusses the concept of false-positives, and describes several intrusion detection techniques employed, to date, to reduce it. Section 3 presents our contribution to this effort in the form of the IDS NetHost-Sensor [6], which uses an intrusion detection technique called procedure analysis; this technique is discussed in detail in this section. In section 4 the results of experiments conducted using procedure analysis are presented, and we validate our methodology in reducing false-positives by comparing them with results obtained using Snort IDS.

## 2. Background and Related Work

The term false-positive can be defined as alerting of an intrusive event by an IDS, while in reality, the event is non-intrusive. They are problematic, because they trigger unjustified alerts and result in diminishing value and urgency of real attacks. Other likely terms use to define false-positives are false alarms and benign triggers.

Several attempts have been made by the research community to reduce false-positives [7], including:

***Operating System Fingerprinting:*** This involves false-positives that occur in a non-vulnerable scenario. This scenario exists because most network IDS do not take the host vulnerability profile into account when detecting attacks. A possible key to reducing false-positives in this scenario is to produce a context-based alerting in which target host information is incorporated into the detection framework. As a result, monitored network packet details can be compared with the target host operating system profile information stored in an expert system database. If the result of the comparison is positive, then an intrusive detection analysis could be done, otherwise, the network packet is dropped [7].

***Alert-Flooding Suppression:*** This involves an alert on the same intrusion that continually propagates throughout the monitored network. An example includes the MS Blaster or SQL Slammer outbreak [8], resulting in a network IDS repeatedly alerting the same intrusion, causing a flood of alerts. A possible solution will be to pre-process "potential alerts" prior to notification on the basis of rules using parameter that take into account alert type, source and destination IP addresses, just to name a few. These parameters will enable the IDS to suppress identical alerts and log them for statistical analysis [7].

***Meta-Alert Correlation:*** They are generated by the correlation of two or more alerts, possibly from different detection sensors. Meta-Alert Correlation enables the generation of a higher priority alert whenever certain conditions related to lower-level alerts are fulfilled. The absence of meta-alert correlation will result in an IDS viewing intrusive activities as isolated and discrete events, hence possibly dismissing isolated events that, when correlated, could result in an intrusive event. Meta-Alert Correlation parameters will probably include time window, event count and type, IP address and port number [7].

A possible solution to address high false-positives will be to design smarter intrusive detection engines, which may include enterprise context for detection analysis, alerting on parameterised rules and correlation and aggregation of rules. Another alternative in curbing false-positives could model the communication protocol and then use the syntax and/or semantics associated with that protocol to design intrusive signatures. The latter is our main contribution in this paper. Most Network Based Intrusion Detection System (NBIDS) are typical signature based. Examples are Snort and Bro [9], which offer a straightforward way to write signatures to restrict services to a range of trusted addresses. However, it is a little more challenging to relieve the network administrator of the task of keeping the signatures updated by monitoring the traffic to determine normal usage patterns. Systems such as ADAM [10], NIDES [11], SPADE [12], and Emerald [13] do just that. These mentioned IDS systems use an expert system database consisting of intrusive signature, encoded with knowledge gleaned from security experts to test files or network traffic for patterns known to occur in attacks. Therefore, minor variations in attack method can often defeat such systems. None of these IDS's aim to enhance their detection technique to reduce false-positives, but simply implement established signature pattern matching techniques.

The research work carried out by Estevez-Tiapador et al. [14], can be expressed in two steps: first, some statistical analysis of both normal and hostile traffic is presented. The experimental results of this analysis reveal that certain features extracted from HTTP requests can be used to distinguish anomalous (and therefore, suspicious) traffic that corresponds to correct normal connections. The second part of their research presents a new anomaly-based technique to detect attacks carried out over HTTP traffic. The technique introduced is statistical and makes use of Markov Chains [15] to model HTTP network traffic. The incoming HTTP traffic is parameterized for evaluation on a network packet payload basis. Thus, the network packet payload of each HTTP connection is segmented into a certain number of contiguous blocks, which are subsequently quantized according to a previously trained scalar codebook. Finally, the temporal sequence of the symbols obtained is evaluated by means of a Markov model derived during their training phase. A simple visual inspection is carried out on payload length to compute payload histogram and standard deviation, via grouping network traffic into protocol and services. Unlike our NetHost–Sensor procedure analysis presented below, knowledge of the associated network traffic is not used to determine intrusions and reduce false positive rate of attacks.

NETAD [16], describes a two stage anomaly detection system for identifying suspicious traffic. This was done by filtering traffic to pass only packets of interest e.g. the first few packets of incoming server request. In addition, they model the most common protocols (IP, TCP, Telnet, FTP and others) at the packet byte level to flag events (byte value) that have not been observed for a long time. Reported results showed that the system detected 132 of 185 attacks in the 1999 DARPA IDS evaluation data set [17] with 100 false alarms, after training on one week of attack-free traffic. NETAD makes an effort to reduce false-positives by only analysing the first few network packets of an incoming web server request. As a result, it ignores non-IP, SYN-ACK, and all outgoing network packets. It models 48 attributes, consisting of the first 48 bytes of the network packet, starting with IP and TCP headers, then TCP payload. An anomaly threshold score is computed from these attributes and if exceeded in a training phase, the weighting of the associated intrusive signature that triggered this anomaly is lowered. Thus, the overall false-positive rate is reduced. This research differs from ours, as it does not enhance the analysis of the network packet payload with context aware knowledge of the monitored network traffic, but reduces false-positives via lowering its detection threshold.

## 3. NetHost-Sensor An Overview

In this section we present our research: - NetHost-Sensor. This involves investigative experiments of scenarios where End-To-End (ETE) encrypted communication like IPSec [6] technology between participating peers will elude the scrutiny of a network based IDS, as network packet's payload will be encrypted. In addition, since the network based IDS is between both hosts, it is susceptible to network fragmentation, evasion and insertion attacks [6].

### 3.1. NetHost-Sensor's Procedure Analysis Methodology

Our current research uses the data captured by NetHost-Sensor to perform a procedure analysis technique that models HTTP network data to detect and reduce false-positives. The work carried out is expressed in two main steps: first, data modelling of HTTP request information is performed, second, based on this HTTP data model a procedure protocol that uses formal syntax and semantics is designed to create intrusive signature to reduce false-positives. The following subsection below describes the first step, and the next, describes the second.

### HTTP Data Model

Our procedure analysis approach focuses on the GET request in an HTTP protocol that uses parameters to pass values to server programs.

We expressed formally HTTP request as the set $U = \{U_1, U_2 .. U_m\}$ of URLs extracted from a successful GET request. A URL $U_i$ can be expressed as the composition of path to the desired resource $(path_i)$, an optional path information component $(pathfo_i)$, and an optional query string (q). The query string is used to pass parameters to the referenced resource and it is identified by a leading '?' character. A query string consists of an ordered list of 'n' pairs of parameters (or attributes) with their corresponding values. That is $q = (a_1, v_1), (a_2, v_2), ..., (a_n, v_m)$ where $a_1 \in A$, is the set of all attributes, and $v_1$ is a string. The set $S_q$ is defined as the subset $\{a_j, ....., a_k\}$ of attributes of query q. Figure 1 shows an example of an entry from a web server log and the corresponding elements that are used in the analysis. For this example of query q, $S_q$ = $(a_j, ....., a_k)$ and /lame.cgi or any string coming after the cgi-bin could be the optional path (*pathfo*).

The analysis process focuses on the association between program, parameters, and their values. Therefore, each referred program 'r' is assigned a set of corresponding $U_r$. Our procedure analysis model will apply to each $U_r$, independently, and are not implemented on current IDSs like Snort. Snort's detection rule options uses context keyword that allows a user to set rules that search for specific content in a packet payload using six modifiers:- depth, offset, distance, within, nocase and rawbytes. Using Snort modifiers, it is not possible to create an intrusive signature that will include $(path_i)$ and a non-adjacent query string $(a_2 = v_2)$. It is only possible to create intrusive signatures that will include $(path_i)$ alone or $(path_i)$ plus all query string $q = (a_1, v_1), (a_2, v_2), ..., (a_n, v_m)$ or $(path_i)$ plus adjacent query string values $(a_1 = v_1)$ or just a query string $q = (a_1, v_1), (a_2, v_2), ..., (a_n, v_m)$. Our HTTP data model cannot be implemented using current IDSs like Snort because of the limitation of their detection rule options; as a result, these IDSs are more prone to false positive rate of attacks.

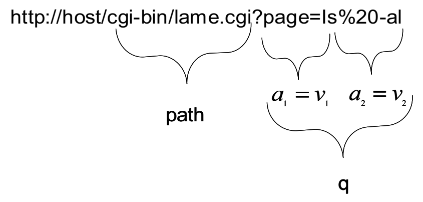
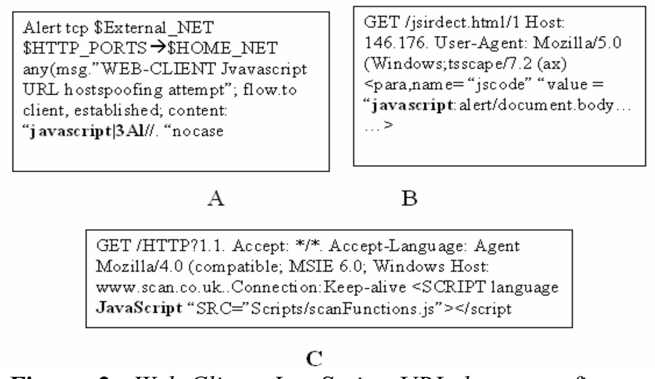


***Figure 1:*** *Practical example of HTTP data model used.*

### Procedure Analysis Attack Scenarios

We describe a possible HTTP false positives attack scenarios below, and with the aid of Snort signatures, malicious and false positive attacks network packet payload, design new intrusive signature pattern via our HTTP data model for NetHost-Sensor's procedure analysis.

***Web-Client JavaScript URL Host Spoofing Attempt (CVE 2002-0815):*** this attack occurs when a client on the protected network visits a website containing malicious JavaScript code, that will access sensitive information. For example, certain version of Mozilla and Netscape may allow script code to access local cookie data. To curb this exploit, Snort IDS's community have introduced associated intrusive signature in their expert system database, as shown in Figure 2(a). Although, efforts have been made to accurately design this signature, reports that peer-to-peer applications may cause associated alerts are known (http://www.snort.org/pub-bin/sigs.cgi?sid= 1841). An example of an actual web-client JavaScript URL host spoofing exploit network packet payload is presented in Figure 2(b), while normal network packet' payload that may trigger this intrusive signature is shown in Figure 2(c).



```
Alert tcp $External_NET
$HTTP_PORTS→$HOME_NET
any(msg."WEB-CLIENT Jvavascript
URL hostspoofing attempt"; flow.to
client, established; content:
"javascript|3Al//. "nocase
```

A

```
GET /jsirdect.html/1 Host:
146.176. User-Agent: Mozilla/5.0
(Windows;tsscape/7.2 (ax)
<para,name= "jscode" "value =
"javascript:alert/document.body...
...>
```

B

```
GET /HTTP?1.1. Accept: */*. Accept-Language: Agent
Mozilla/4.0 (compatible; MSIE 6.0; Windows Host:
www.scan.co.uk..Connection:Keep-alive <SCRIPT language
JavaScript "SRC="Scripts/scanFunctions.js"></script
```

C

***Figure 2:*** *Web-Client JavaScript URL host spoofing attempt network packet payload scenarios. a:- Snort IDS's signature, b:-actual attack exploit and c:-false-positives.*

In designing our intrusive signature for NetHost-Sensor's procedure analysis using our HTTP data model, $(path_i)$ was set to GET/cgi-bin/xxx.html, while $q = (a_1, v_1), (a_2, v_2), ..., (a_n, v_m)$ to (javascript, window.open). The overall intrusive signature was a combination of the two. Other attacks scenarios used include: web-misc invalid http version string (Bugtraq 9809) and web-client Microsoft emf metafile access (CVE 2003 0906) just to name a couple.

## 4. NetHost-Sensor's Procedure Analysis Experiment

This section describes our experimental investigation in evaluating NetHost-Sensor's procedure analysis technique in reducing false-positives. Our experiments had the following main objectives:-

1. Validate the performance of NetHost-Sensor procedure analysis technique to reduce false-positive in comparison with Snort. Snort's architecture does not facilitate the usage of procedure analysis, only strict signature pattern matching technique.
2. Measure the computational overhead introduced by intrusive signature derived from our HTTP data model used in our NetHost Sensor IDS and those used in Snort IDS.

**Procedure Analysis Experiment Test Data and Equipment**

A 2 months, 6 hours a day "in the wild", trace of HTTP request network traffic was collected in one of the School of Computing laboratories at Napier University, Edinburgh, Scotland, and used as our test data. The laboratory has several internal hosts, and the test data was collected on its Gbps access link to the Internet. The laboratory trace total 20Gb, 15.million packets, and 220k connections. The machines used where Microsoft Windows 2003, IIS 6, and several web application and their associated cgi scripts. We performed all measurements on a 550MHz Pentium 3 system containing ample memory.

**Procedure Analysis Experimental Details**

Each section of an HTTP GET Request has its own set of allowed values according to its purpose and semantic, hence the probability of certain strings occurring within each section of the payload will not be uniform throughout the request. As a result, we will be able to differentiate between strings that constitute innocuous and harmful HTTP request. With this in mind, false-positives can be reduced by an IDS like Snort, that implement signature-based detection technique for HTTP network data by adjusting signature attack strings patterns to include more harmful and less innocuous string patterns.

This is performed experimentally using a controlled environment that includes windows IIS web-server and Common Gateway Interface (CGI) programs that are prone to generate high false-positives, as described in [18]. These CGI programs were installed in the IIS web-server and used to generate HTTP innocuous request.

In addition, we also tried to attack or hack into these CGI programs and associated web sites, as a result generated harmful HTTP request to do just that. In our controlled lab, we use these innocuous and harmful HTTP request to generate corresponding false-positives via Snort IDS. Then model this request using our NetHost-Sensor HTTP data model technique to isolate strings consistent to the harmful but not to innocuous HTTP request generated during our initial training phase. While in our evaluation phase, NetHost-Sensor was exposed to "in the wild" normal HTTP network GET request via Napier University local network and false-positives were measured for both Snort and NetHost-Sensor IDS. Figure 3, presents our experimental set-up.

In our **modelling phase**, we use these innocuous HTTP requests to generate corresponding false-positives via Snort IDS. Then model these requests manually using our NetHost-Sensor HTTP data model technique to isolate strings consistent to the real attacks the intrusive signature was designed to detect and not our innocuous HTTP request. The intrusive signatures that resulted from our HTTP data model technique were stored in NetHost-Sensor expert system database awaiting evaluation and were over 50 in number.

In the **evaluation phase**, two main issues were tackled: -

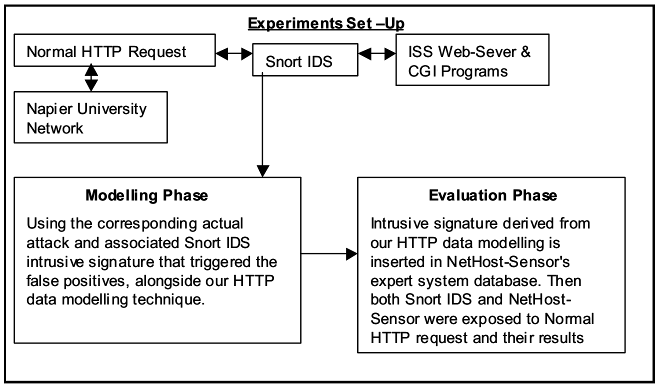
1. Performance comparison of Snort IDS to NetHost-Sensor procedure analysis technique in reducing false-positive,
2. The computational overhead introduced by our NetHost-Sensor procedure analysis technique in comparison with Snort IDS.

In tackling issue "1", we expose both Snort IDS and NetHost-Sensor to normal HTTP network GET request via Napier University local network and false-positives were measured for both IDSs. Figure 4 presents the results of this experiment.

While in tackling issue "2", we measured the computational overhead on the Central Processing Unit (CPU) introduced both IDS intrusive signatures. Figure 5 presents the result of this experiment.

**Evaluation of Experiments Analysis**

In modelling NetHost-Sensor intrusive signature we used our HTTP data modeling technique, described earlier in this paper. We expressed a HTTP Get request as the set $U = \{U_1, U_2 .. U_m\}$ of URLs extracted from a successful GET request. Also, we express each $U_i$ as a path to the desired resources $(path_i)$, and an optional query string (q). Further still, we express the query string "q" as a list of parameters and their corresponding value. That is $q = (a_1, v_1), (a_2, v_2), \ldots, (a_n, v_m)$ where $a_1 \in A$, the set of all attributes, and $v_1$ is a string or value.



**Modelling Phase**

Using the corresponding actual attack and associated Snort IDS intrusive signature that triggered the false positives, alongside our HTTP data modelling technique.

**Evaluation Phase**

Intrusive signature derived from our HTTP data modelling is inserted in NetHost-Sensor's expert system database. Then both Snort IDS and NetHost-Sensor were exposed to Normal HTTP request and their results

***Figure 3:*** *NetHost-Sensor's Procedure Analysis Experiment Set-Up. The above Figure illustrates the experimental set-up and procedure we carried out in deriving our intrusive signatures via novel HTTP data modeling technique.*

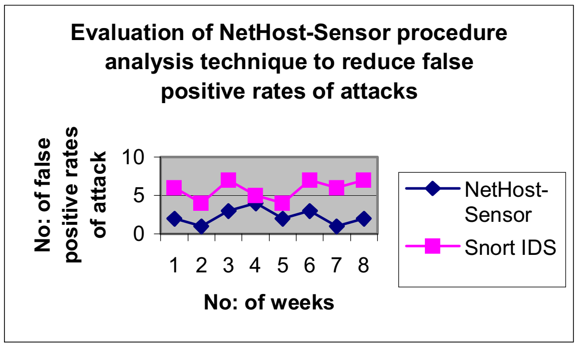
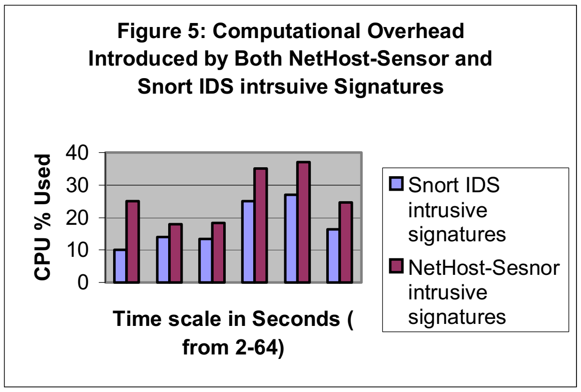


***Figure 4:*** *Compares NetHost-Sensor and Snort IDS in terms of false-positives triggered within an 8-week period. Our experimental results show that NetHost-Sensor's procedure analysis performed better than Snort IDS's pattern matching detection algorithm within this period. A difference of 28-triggered false-positives was observed between both IDS. All efforts were made to update Snort's expert system database and use all relevant pre-processors in detecting true positives.*



As an illustration, when Snort IDS was exposed to normal HTTP Matt Kruse Calendar (www. mattkruse.com /scripts) Get request, it alerted on a false-positive. The intrusive signature that was triggered was "calendar" and the normal HTTP that caused the trigger was http*://host/cgi-bin/calender/ calender.pl* and an actual attack could be *http://host/cgi-bin/calender/calender.pl?config=xx* , or *http://host/cgi-bin/calender/calender.pl?password=xx* or *http://host/cgibin/calender/calender.pl? username =xx.* To apply our procedure analysis data model to the above, we take into account both Snort IDS intrusive signature that was triggered and the actual http attack strings. For this illustration, we manually use as $(path_i)$ "/calendar.pl? and $q$ either config=xx or password=xx or username=xx. A combination of $(path_i)$ alongside each $q$ will be our NetHost-Sensor intrusive signature that is stored in its expert system database. By including the intrusive signature derived from our procedure analysis data modeling in NetHost-Sensor expert system database, we will be able to reduce false-positive. In addition, since Snort IDS intrusive signatures are in most cases a rough subset of the derived NetHost-Sensor intrusive signatures, a possible inference from this and also proven during our experiments is that the true-positive detected for both NetHost-Sensor and Snort IDS were on the average the same.

A likely side effect of increasing the strings of any intrusive signature could be an increase in computational overhead. The difficulties that may arise in measuring the overhead of two distinct IDSs are their architectural differences, such as pattern matching, methodology, computation of algorithm and size of expert system database, amongst others. Any architectural differences could be the main culprit of a high percentage of CPU usage. Hence, we measured the CPU usage of NetHost-Sensor exposed to normal HTTP traffic then replaced all the intrusive signatures that were stored NetHost-Sensor's expert system database with corresponding Snort IDS signatures and then perform the same experiment. In evaluating our research methodology in this paper, we choose to use only Snort IDS as it is the de factor standard in intrusion prevent and detection [19].

Inferences from our experiments show that our procedure analysis does detect less false-positives in comparison with Snort IDS, but with a side effect of a higher computational overhead. Therefore, our procedure analysis technique could be used in enterprises with large processing power and a high demand for valid true positive rate of attacks.

## 5. Conclusion

In this paper we focused our research efforts on suppressing false-positives using NetHost-Sensor's procedure analysis technique. We initially described possible types of detection techniques implemented in the IDS community and narrow our efforts to specific IDSs. Alongside this, we discussed about the causes and research efforts in thwarting false-positives. We then recapped on our previous research project the "NetHost-Sensor" and described its novel features and its contributions. In addition, we presented our latest novel contribution: - NetHost-Sensor's procedure analysis and its HTTP data model technique. This is based on HTTP GET requests that it expresses in terms of paths, resources, optional paths information components, and optional query string. Our rational behind our HTTP data model techniques was to express the syntax of a HTTP GET request. As a practical example, we introduced possible attack scenarios, in relation with their Snort intrusive signature, true and false-positives network packet payloads, to aid in the design of intrusive signatures using our HTTP data modelling technique. To validate our research efforts in reducing false-positives, investigative experiments were carried out to compare NetHost-Sensor's procedure analysis technique with Snort's pattern matching algorithm and our results showed that NetHost-Sensor showed more accuracy than Snort, by alerting 28 less false-positives over an 8-week evaluation period.

Future research experiments should investigate the detection rate of varied IDS detection algorithm, when implementing different detection techniques like procedure analysis and stringent string pattern matching; the results generated from these experiments would provide an alternative means of testing the effectiveness these detection techniques. Also our novel procedure analysis approach could be used on other communication protocols like Simple Mail Transfer Protocol (SMTP). A key difficulty that may hinder these experiments, is that many IDSs, either free or commercial, do not provide access to their detection algorithm source code.

## 6. References

[1]Computer Crime Research, http://www.crime-research.org/news/11.06.2004/423/ , 2005

[2]M.Roesch, "Snort-lightweight Intrusion Detection for Network", Proceeding of USENIX LISA 99, pp: 229-238

[3]C.Kruegel, G.Vigna and W.Robertson, "A Multi-Model Approach to the Detection of Web-Based Attacks", Computer Networks, V(48), Iss(5), pp:717-735, 2005

[4]P.Porras, "STAT-A State Transition Analysis Tool for Intrusion Detection", Technical Report TRCS93-25, Computer Science Department, University of California at Santa Barbara, 1993

[5]G.H.Kim, E.H.Spafford, "Experiences with Tripewire using Integrity Checker for Intrusion Detection", SANS Conference III, USENIX, 1994

[6]A.A.Abimbola, J.M Munoz and W.J Buchanan, "Investigating the capture of End-To-End Encrypted Intrusive Data", accepted to Journal of Computer and Security, 2006

[7]A.Yee, "Marking False Positives Go Away", ComputerWorld, www.computerworld.com, 2006

[8]E. Schultz, J.Mellander and D.Peterson "The MS-SQL Slammer Worm", Network Security, V(2003), ISS(3) , pp: 10-14

[9]V.Paxson, "Bro: A System for Detecting Network Intruders in Real-Time", Computer Networks, V(31) Iss (23-24), pp:2435-2463, 14 December 1999

[10]D.Barbora and W.S.Jajodia, "Detecting Novel Network Intrusion Using Bays Estimators", First SIAM International Conference on Data Mining, 2001

[11] Anderson et al, "Detecting Unusual Program Behaviour Using the Statistical Component of the Next Generation Intrusion Detection Expert System (NIDES)," Computer Science Laboratory SRI-CS2 95-06 , May 1995

[12]SPADE,SiliconDefence, http://www.silcondefence.com /software/spice, 2005

[13] P.Neumann and P.Porras, "Experiment with EMERALD to Date", Proceeding 1st USENIX Workshop and Intrusion Detection and Network Monitoring , pp:73-80, 1999

[14]J.M.Estevez-Tapiador, P.Garcia-Teodoro, J.E.Diaz-Verdejo, "Measuring Normality in HTTP Traffic for Anomaly-Based Intrusion Detection", Computer Networks V(45), 175-193, (2004)

[15]G.Benrit, "Application of Markov Chains in an Interactive Information Retrieval System" Information Processing & Management, Vol(41), Iss(4), pp: 843-857, July 2005

[16]M.V.Mahoney, "Network Traffic Anomaly Detection Based on Packets Bytes", SAC 2003

[17]R.Lippmam et al, "The 1999 DARPA Off-Line Intrusion Detection Evaluation", Computer Network V(34), Iss(4), pp: 579-595, 2000

[18] S.Patton,W.Yurcik and D.Doss, "An Achilles's Heel in Signature-Based IDS: False Positives in Snort" RAID 2001

[19] "NDIS", www.ndis.org. June 2000