

available at www.sciencedirect.comjournal homepage: www.elsevier.com/locate/cose

**Computers
&
Security**



NetHost-Sensor: Investigating the capture of end-to-end encrypted intrusive data

A.A. Abimbola*, J.M. Munoz, W.J. Buchanan

School of Computing, Napier University, EH10 5DT, Scotland, UK

ARTICLE INFO

Article history:

Received 5 April 2005

Revised 25 August 2005

Accepted 4 April 2006

Keywords:

Host-based

Intrusion

Detection

Kernel mode

End-to-end

Encryption

Network communication driver

ABSTRACT

Intrusion Detection Systems (IDSs) are systems that protect against violation of data integrity, confidentiality and availability of resources. In the past 20 years, these systems have evolved with the technology and have become more sophisticated. Despite these advances, IDS is still an immature field, and the benefits obtained from detecting end-to-end encrypted attacks justify the need for more research.

This paper presents possible advantages of an IDS that uses a target host's kernel as its audit source for intrusion analysis against specific attacks. In addition, we describe our research experience in determining what layer, within a protocol stack of a target host, where decrypted data can be captured for intrusion detection. Then, it examines how to capture decrypted data, while communicating via an End-to-End (ETE) encryption channel. The paper proceeds further to discuss our methodology using network communication driver interfaces, investigative experimental procedures and present our experimental results. Finally, discussions on the methodology of our future research, modelling HTTP network data via procedure analysis technique to reduce false positive rate of attacks are presented.

© 2006 Elsevier Ltd. All rights reserved.

1. Introduction

The increasing complexity and speed of operation of current distributed systems have contributed to the difficulty of monitoring network functions for intrusions. This challenging scene is aggravated by the immeasurable growth of security incidents in the last years. Reports from CERT Coordination Centre on security incidents (CERT Coordinated Centre, 1988), vulnerabilities security alerts and so on corroborates this fact. Thus, the number of mail message handled relating to security incidents increased from 56,365 in 2000, to 542,754 in 2003 and 717,863 in 2004. Furthermore, there is a general agreement that these statistics do not include all the incidents that occurred, but merely included those reported. Companies have their own reasons to keep secret the security incidences suffered.

To reduce the level of security incidences, network/host-based Intrusion Detection Systems (IDSs) (Jung et al., 2004; Porras and Neumann, 1997; Patil et al., 2004) – systems that protect against violation of data integrity and confidentiality, and availability of resources – are employed. While Network Based Intrusion Detection Systems (NBIDS), use as their audit sources for intrusions network packet's payload, Host-based Intrusion Detection System (HBIDS) use, as their audit source for intrusion, operating system log files.

This paper, discusses the concept of an IDS, and identifies their key vulnerabilities. Then it describes in detail a NetHost-Sensor (Abimbola et al.) system, which addresses these vulnerabilities using target host's audit sources in the kernel. The main attacks identified include end-to-end encrypted attacks and network fragmented attacks. To detect these

* Corresponding author.

E-mail address: a.abimbola@napier.ac.uk (A.A. Abimbola).

0167-4048/\$ – see front matter © 2006 Elsevier Ltd. All rights reserved.

doi:10.1016/j.cose.2006.04.001

main attacks, we compare possible experimental methodologies using Microsoft Network Communication Drivers. Our experimental aim was to capture decrypted target host data has an audit source for intrusion analysis, while communicating in an End-to-End (ETE) encrypted channel, e.g. IPSec. In addition, we present our investigative experimental results. All NBIDS we surveyed (Jung et al., 2004; Porras and Neumann, 1997; Patil et al., 2004) are susceptible to these key vulnerabilities. Finally, we discuss our future research efforts that model HTTP network data using procedure analysis technique to reduce false positive rate of attacks.

2. Background

2.1. Intrusion detection system

There are two general approaches to the problem of intrusion detection systems: signature-detection (also known as misuse detection) where we look for patterns signalling well-known attacks, and anomaly-detection, where we look for deviations from normal behaviour.

Signature-detection works reliably on known attacks, but has the obvious disadvantage of not being able to detect novel attacks. On the other hand, anomaly-detection can signal that an unusual event has occurred, which may not be by a real attack, and as a result false positive rates of attacks are high.

Signature-detection methods are better understood and widely applied. They are used in both NBIDS (Caswell et al., 2002; Luo et al., 2001) and HBIDS (Price, 1997). These systems use a set of rules encoded knowledge gleaned from security experts to test file or network traffic for patterns known to occur in attacks. A limitation of these systems is that these rules require update at intervals and may fail to detect variations in attack pattern.

Anomaly-detection is a harder problem than signature-detection because, while a signature can be precisely defined, what is considered a normal pattern is more abstract and ambiguous. Rather than finding rules that characterise attacks, it attempts to find abnormal behaviour (Forrest et al., 1996). Since what is considered normal can vary across different environments, a distinct model of normality may not be valid in all networks. Much of the research in anomaly-detection uses the approach of modelling normal behaviour from attack free data set.

Forrest et al. (1997), making the connection between anomaly-detection systems and biological immunology, found that when a vulnerable Unix system program or server is attacked, the program makes sequences of system calls that differ from those sequences found in normal operations. Other models of normal system call sequences have been used, such as finite state automata (Sekar et al., 2001) and neural networks (Jing-xin et al., 2004).

3. Motivation

IDSs have evolved over the past decades, but still have vulnerabilities in their methodology, which motivates this research work. These vulnerabilities and other works investigating

using kernel mode as an audit source for intrusion detection are described below:

- *End-to-End (ETE) encryption*, with security improvement in communication protocols, encrypted traffic on an ETE basis is increasing. This thwarts an eavesdropper, but does not allow an IDS to monitor and analyse a network packet's payload for intrusion. If the IDS cannot analyse a packet's payload, this is likely to result in a high false positive rate of attacks (Goregaoker, 2001).
- *TCP port scanners*, are specialized programs used to determine what TCP ports of a host have processes listening on them for possible exploitation. Scanning techniques like TCP SYN that uses TCP 3 way hand shake mechanism are detectable by TCP wrapper (Atkins et al., 1997); stealth scans like (Maimon, 1996) Full, Xmas Tree and Null, are more clandestine. To detect stealthy scans an HBIDS will have to monitor TCP segment's headers and not only packet's payload.
- *Network-speed and infrastructure*, fast network communication directly hinders an IDS from monitoring and analysing network packets. This results in many dropped network packets. In addition, the trend towards switched communication also increases the difficulty of an NBIDS to monitor multiple communication streams. Also, owing to multiple routing paths to a target host, a knowledgeable attacker can avoid an IDS device and still attack a target host.
- *Network packet fragmentation*, the problems with network packet's fragmentation in relation to a network IDS (Ptacek and Newsham, 2003) are as follows:
 - The target host and network IDS may reassemble out-of-order IP datagram fragments differently, as a result an attacker can intentionally scramble the IP datagram fragments to elude the IDS.
 - A network IDS can be attacked by flooding the network with partially fragmented IP datagram that will never be completed. A naïve network IDS runs out of memory as it attempts to cache each fragments for reassembly.
 - Fragmentation overlap may occur when fragments of different sizes arrive out-of-order and in overlapping positions in the target host's network layer. An attacker that understands the specific inconsistencies between a target and a network IDS can obscure the network IDS by couching malicious data inside overlapping fragment streams.
- *Client TCP/IP puzzles*, propose one mechanism for protecting protocols against Denial of Service (DoS) attacks. A server or network being protected generates a cryptographic puzzle that a client must answer correctly before it is given service. Such a mechanism gives servers and the network the ability to selectively push back load to the source of attack. The article by Wu-chang (2003) argues that such puzzles must be placed within the slim waistline of the TCP/IP protocol stack in order to truly provide protection and reduce overhead on the server.

Other IDS vulnerabilities that exist include evasion and insertion attacks (Ptacek and Newsham, 2003), and scope of attacks (Koba).

Fig. 1 below outlines our research work, where ETE encrypted communication like IPSec (Seitz, 2002) or Cisco Encryption

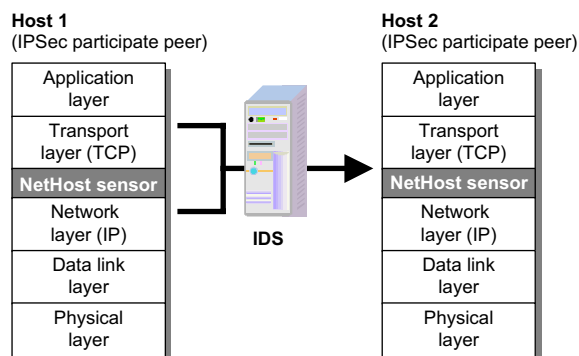


Fig. 1 – Two hosts participating in IPSec communication.

Technology between participating peers (Host 1/2) 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. The NetHost-Sensor overcomes these vulnerabilities by using audit data from either the target host's network or transport layer, in order to detect intrusion.

4. Related work

Before we present our solution it is convenient to compare it with relevant existing network based IDSs. Daniels and Spaford detect low-level network attacks by auditing the kernel of a Linux target host. This is done by collecting data from several points in the protocol stack of the target host. Although this research involves experiments in the kernel mode, investigation to determine the performances of the above research against attacks identified in this paper was not carried out. Hence, no conclusion can be inferred from this paper in comparison with our research work.

Shankar and Paxson (2003) have contributed to the attempted thwarting of evasion attacks using the following:

- *Bifurcation analysis*: in which the monitor handles ambiguous network traffic stream by instantiating separate analyses for each possible interpretation of the ambiguous network traffic.
- *Traffic normalization*: in which a network-forwarding element attempts to eliminate ambiguous network traffic and reduces the amount of connection state that the IDS must analyse.
- *Active mapping*: this effectively builds profiles of the network and TCP/IP policies of hosts. An IDS may then use the host profiles to establish a single semantic network traffic on a per-host basis.

These techniques are limited in scope, as the bifurcating analysis can be subject to a DoS attack by overloading it with infinite threads. Traffic normalising and active mapping only allow known network traffic (thus limiting their usage) and have to be updated with new network knowledge. NetHost-Sensor IDS avoids any mis-interpretation of network

traffic like evasion attacks between itself and the target host by using as it audit source the associated target host protocol stack.

There has recently been a fair amount of research activity in Network Interface Cards (NIC) based computing. More closely related to our research, embedding firewall functionality, such as packet filtering, and packet auditing have been proposed (Otely et al., 2003). While most of these ideas are in their infancy, a couple have been commercialised (e.g. 3Com's embedded firewall). 3Com embedded firewall consists of a 10/100 PCI Network Interface Card (NIC) with 3XP processor that monitors individual host network traffic via a policy server. While conventional firewall protects the parameter of the network, 3Com protects the host and it is independent of the operating system. 3Com embedded firewall only permits or denies network traffic to associated host and does not perform any packet payload analysis for intrusion. As a result, it is susceptible to the main attacks identified in this paper.

Snort (Caswell et al., 2002), the chosen network IDS used in our experiment, is a recent open-source, public-domain effort to build a lightweight efficient IDS tool that can be deployed on a wide variety of platforms. Snort features rule-based logging and can perform content searching/matching. It can be used to detect a variety of attacks and probes, such as buffer overflow, port scans, and Common Gateway Interface (CGI) attacks. It is currently undergoing rapid development. Snort does not attempt to tackle ETE encryption, as our proposed NetHost-Sensor, but makes the following efforts in thwarting evasion and network fragmented attacks:

- *Rule optimiser*: this is a major component of the Snort detection engine. It optimises active Snort rules by sorting them into smaller, unique rule sets. This allows Snort to quickly inspect a packet against any applicable rule set, while providing Snort with the opportunity of using faster and more efficient set inspection technologies.
- *Protocol flow analyser*: this classifies network application protocols into client and server data flows, and allows it to make in-depth analysis for intrusions.

Other IDSs like Real Secure Network Sensor (RNS), Internet Security System's (ISS) Micro Agent, Network Flight Recorder (NFR), Cisco's IDS products and NetRanger are currently not capable of thwarting ETE encryption attacks, and network fragmented attacks (Price, 1997).

5. NetHost-Sensor

This section describes the methodology of the NetHost-Sensor and its unique features in thwarting ETE encryption attacks and network fragmented attacks. In order for the NetHost-Sensor to thwart these attacks, we collect network data within the protocol stack of the target host. The reason being, at these layers all network packets will have been decrypted and all fragmented network packets reassembled to enable more efficient intrusion detection. In this research, we do not consider application-to-application based encryption. Fig. 2 below illustrates possible implementation of the NetHost-Sensor on a protocol stack of a target host.

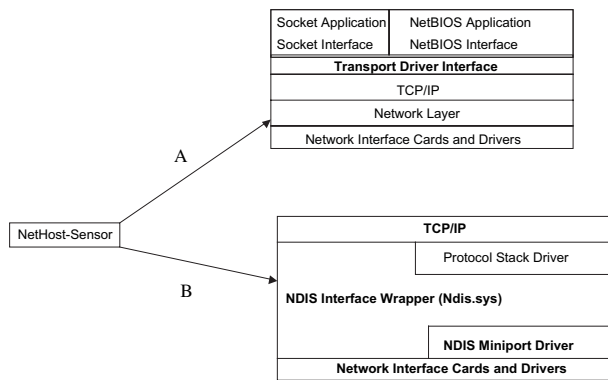


Fig. 2 – NetHost-Sensor analyses data from the protocol stack of a target host by two possible approaches, ‘A’ being TDI and ‘B’ NDIS, for further references (www.microsoft.com/windows2000).

An obvious weakness in this approach is that it may not be portable as it is embedded within the protocol stack of the target host. Our argument is that for a network IDS to avoid the pitfalls described by Ptacek and Newsham (2003), it must be customised to the target host. Two implementations that have been investigated using Windows architecture are as follows:

- *Transport Driver Interface (TDI)*: an interface for a driver that communicates with various network transport protocols. This allows services to remain independent of transport protocols. Microsoft developed TDI to provide greater flexibility and functionality than is provided by existing interfaces (e.g. Winsock and NetBIOS). This flexibility allows a TDI to provide a consistent interface for the transport protocols. TDI specification describes the set of functions and call mechanism by which transport drivers communicate. This research presents the implementation of a TDI filter – a device that sits between a protocol driver and a TDI client like TCP/IP.sys. In this logical position a TDI filter is able to monitor and influence network data passing between the TDI and its client by altering dispatch tables.
- *Network Driver Interface Specification (NDIS)*: this is a specification for a network driver architecture that allows transport protocols such as TCP/IP to communicate with an underlying Network Interface Card (NIC). The NIC can then send or receive data over the network. NDIS permits the high-level protocol components to be independent of the NIC by providing a standard interface to the network protocol. The NDIS is implemented by a file called Ndis.sys, referred to as the “NDIS wrapper”. The NDIS wrapper is a code that surrounds all NDIS device driver and contains supporting routines that make it easier to develop a driver. This paper outlines the implementation of an NDIS protocol and miniport drivers, also called NDIS intermediate drivers, that sit within the network layer to capture network packets. The reader is referred to Windows DDK 2003 for sample drivers like Passthru, for further reading.

Our approach involves no modification to the kernel of Windows and does not duplicate efforts already done in the

kernel. An Application Protocol Interface (API) can be designed for each platform, making the NetHost-Sensor platform independent.

A key advantage to using network communication drivers instead of socket technology is that only one application can bind to an associated socket under the Windows system. As a result, a buffer would be required to capture decrypted data in an ETE encrypted communication channel, with the necessary API to communicate with the vendor application. We justify our research by working below the Windows socket interface and in the kernel mode, to capture all data before its fed to any application, without using vendor APIs.

6. Experimental details

In this section, we describe our experimental methodology in order to justify our research efforts. We initially present our objectives, then experimental procedure and finally concluding results.

The experiments have the following objectives:

- To investigate the capturing of network packet data using NDIS intermediate drivers – a network device interface specification driver that overlaps the data-link and network layers of our target host protocol stack,
- To investigate the capturing of network data packets using transport filters – a Transport Driver Interface (TDI) filter that intercedes between TDI and TDI clients,
- To investigate the results of audit data captured by NDIS intermediate and TDI filter after implementing ETE encryption channel using IPsec.

6.1. Experimental procedure methodology

- We modified Microsoft NDIS intermediate driver, originally designed to receive/send network packets between the data-link and network layers, adding to the NDIS_Status PtReceive and int PtReceivePacket functions, ReadonPacket routines that enable it to read all NDIS network packets. We captured network packet data using structures fields found in IP and TCP header files, designed by the University of California (<http://www.ucsd.edu/>). These structures were then stored in a kernel buffer before being passed to using mode via Deviceiocontroler.
- We modified TDI filter samples found at www.pcausa.com to read incoming network packet data to a text file before passing it to the application layer.

In both procedure methodologies, we are able to trap network packet before passing them to the application layer.

- We implemented an ETE encryption channel using IPsec between the attacking and target host system. The target host system had both the intermediate driver and TDI filter installed. In addition, we installed a sniffer between both the target and attacking host and then generated network packets, from the attack to the target host to

simulate attacks. We then proceeded to capture packet payload with the sniffer, NDIS intermediate driver and TDI filter.

Table 1 below shows the configuration of tools used in our experiments. Fig. 3 below shows the network topology used during the experiments. Each experiment used Windows 2000 and 2003 systems with 128 MB of memory and 8 GB of disk space. Appendix A, shows the data captured by both the sniffer and our TDI filter, when ETE encryption was assigned to all IP traffic via IPSec security policies.

In configuring local IPSec policies for the target and attack hosts, the following summarised procedure was used. Microsoft Management Console (MMC) was run on Windows 2000 and 2003, followed by the IP security policies to modify the settings. Next, a PRE-SHARED key was created for both target and attack hosts, and this was applied to all IP traffic.

After configuring IPSec policies on both target and attack host, and installing our modified NDIS intermediate driver and TDI filter on the target host. We then simulated attacks from the attack to target host traffic and observed the data captured by our NDIS intermediate driver and TDI filter. From our observations, TDI filter proved more effective by capturing decrypted data or plain text data, while NDIS captured encrypted data, as did Snort IDS.

7. Future research work: NetHost-Sensor's procedure analysis

In this section, we describe our future research efforts in using NetHost-Sensor to curb false positive rate of attacks, but first we recap the contribution made by previous section of this paper.

NetHost-Sensor using Network Transport Driver Interface (TDI) to capture end-to-end encrypted network data for intrusion analysis before the payload is executed in the application layer of the target host protocol stack.

Our future research uses the data captured by NetHost-Sensor to perform a procedure analysis technique that models HTTP network data to reduce false positive. The work carried out is expressed in two steps: first, data modelling of HTTP request information is performed, second, based on model developed a procedure protocol that uses formal syntax and/or semantics is designed to analyse for false positive rates of attacks. The following subsections below describe briefly both steps.

7.1. HTTP data model

Our procedure analysis approach focuses on Get request in an HTTP protocol (Fielding et al., 1998) that uses parameters to pass values to server-side programs or active documents.

Table 1 – Configuration of tools used in experiments

Hosts	Purpose
Target	Target host
Sniffer	IRIS – a network analyser
Attack host	Generated network packets, simulated an attack

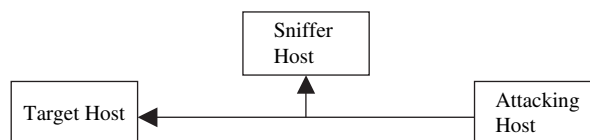


Fig. 3 – Diagrammatic representation of the experimental set-up.

We expressed formally, HTTP request as $U = \{U_1, U_2, \dots, U_m\}$ of URLs extracted from a successful GET request. A URL U_i can be expressed as the composition of the 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 $q = (a_1, v_1), (a_2, v_2), \dots, (a_n, v_n)$ where $a_1 \in A$, the set of all attributes, and V_1 is a string. The set S_q is defined as the subset $\{a_j, \dots, a_k\}$ of attributes of query q . Fig. 4 below shows an example of an entry from a web-server log and the corresponding elements that are used in the analysis. For this example query q , $S_q = (a_j, \dots, a_k)$.

The analysis process focuses on the association between program, parameters, and their values. Therefore, each referred program 'r' is assigned a set of corresponding queries U_r . The procedure analysis algorithm will run on each set of queries U_r , independently.

7.2. Design of NetHost-Sensor's procedure analysis

The HTTP specification defines a common structure for its network data, which composes of several sections described above. Since each section has its own set of allowed values according to its purpose and semantics, it is natural to suppose that the probability of occurrence of certain strings within each section of the payload is not uniform throughout the request.

With this rational in mind, false positive rates of attacks can be reduced for IDS like Snort, that implement signature-based detection technique for HTTP network data by adjusting signature attack string request.

This is performed experimentally using a controlled environment that includes ISS web-server and Common Gateway Interface (CGI) programs that are prone to generate false positive rates of attacks as described in ARACHNIDS (2005). In our controlled environment, we will simulate HTTP request

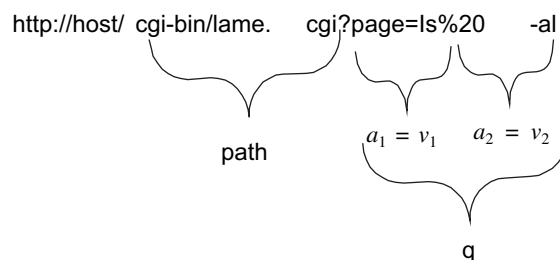


Fig. 4 – Practical example of HTTP data model used.

attacks and generate corresponding false positive via an IDS, then model NetHost-Sensor's procedure analysis technique to isolate query strings related to these attacks only during our training phase. While in the evolution phase, NetHost-Sensor will be exposed to 'in the wild' HTTP network data and false positive rate of attacks will be measured. Our design is depicted in Fig. 5.

8. Conclusion

Despite the research carried out over the past two decades, existing IDSs have not met the expectation that motivated initial work. In this paper, we have described IDSs and classified them using various criteria. We then proceeded to identify current vulnerabilities affecting IDSs and single out a few as the key motivations to our research work, the NetHost-Sensor. We justify our research by comparing it with current research work in thwarting ETE encryption and network fragmented attacks. Whereas, current NBIDS fail to detect and prevent these attacks, a NetHost-Sensor does otherwise. We proceed further in this paper by investigating, experimentally, possible location in the protocol stack of a target host in detecting the above attacks and reported our findings. Our experiments involved the use of network communication driver, both NDIS intermediate and TDI filter attempting to capture decrypted ETE network packet data. The reader is referred to Appendix A, which shows results of network packet payload captured by both sniffer and our TDI filter. Presently, NetHost-Sensor can be used as a plug-in to other IDS, feeding their detection analysers with decrypted ETE encrypted and fragmented data.

The NetHost-Sensor features two novel concepts:

- it sits between the TDI and TDI clients of the target host, and
- uses these layers as its audit source to collect data for possible intrusion detection analyses.

Currently we are designing a procedure analysis for NetHost-Sensor that will reduce false positives in HTTP network communication stream. Future work will include the use of a larger data set of attacks to test our methodology. Alongside this, an investigative comparison of our research efforts with other IDS mentioned in this paper will be done.

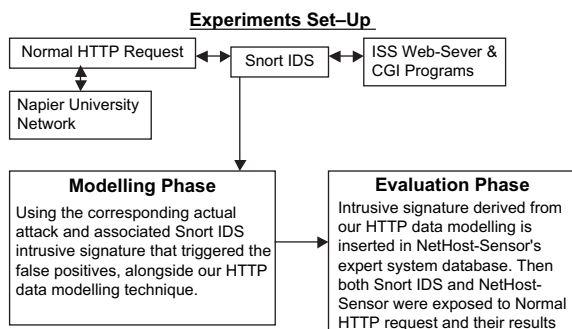


Fig. 5 – NetHost's procedure analysis experiment set-up.

Appendix A. The data captured by both the sniffer and TDI filter, when ETE encryption was assigned to all P traffic via IPsec security policies

```

NetHost-Sensor TCP Network Packet Payload Enter Remote Port Number: 21
Monitoring TCP Data on Remote Port: 21
Press Any Key to quit
Logging First 100 Send/Receive Operations ...
Entry ID = 57
LocalAddress = 0.0.0.1382
RemoteAddress = xxx.xxx.xxx.xxx
Protocol = TCP
Flags = RECEIVED
Data Offset/Length = 0/37
ProcessID = 1604
000000: 32 32 30 20 47 6F 6C 64 : 65 6E 20 46 54 50 20 53 220 Golden FTP
S
000010: 65 72 76 65 72 20 76 31 : 2E 33 32 62 20 72 65 61 erver v1.32b rea
000020: 64 79 2E 0D 0A : dy.....
    
```

Above Decrypted Data from our NetHost-Sensor TDI filter

```

No. Time Source Destination Protocol Info
55 4.338072 ESP ESP(SPI=0x01F587703)

Frame 55 (102 bytes on wire, 102 bytes captured)
Ethernet II, Src: xxxxxx, Dst: xxxxxx
Internet Protocol, Src Addr: xxxxxx, Dst Addr: xxxxxx
Encapsulating Security Payload

0000 00 08 7c 6e 90 00 00 04 75 e8 42 79 08 00 45 00 .jn...n.By.E.
0010 00 38 43 4b 40 00 80 32 4c 70 92 b0 a2 15 92 b0 XCK@_Lp.....
0020 a3 42 df 58 77 03 00 00 00 06 39 cc 94 57 14 df B.Xw....9.W.
0030 5e 3b 29 01 fa 09 95 5f 6e 51 1d b4 5e 01 df 1b ^D...nQ.^...
0040 b9 98 43 52 00 d8 7c f1 51 09 21 2d 67 6f 94 3b .CR.l.Q.!go.;
0050 d3 e0 df e1 1a 36 40 1b 59 89 5d 6c 53 72 2b c3 ....6@.Y.JSrt.
0060 38 52 b5 43 07 b0 8Rc..
    
```

Above Encrypted Data from our Sniffer: Ethereal

REFERENCES

ARACHNIDS: advanced reference archive of current heuristics for network intrusion detection systems. Available from: www.whitehats.com/ids; 2005.

Abimbola A, Shi Q, Merabti M. NetHost-Sensor: a novel concept in intrusion detection systems. In: Eight IEEE international symposiums on computers and communications; June 30–July 03, 2003. p. 232–40.

Atkins D, Buis P, Hare C, Kelley, Nachenberg C, Nelson AB, et al. Internet security. 2nd ed. New Riders; 1997. p. 413.

Caswell B, Beale J, Foster J, Faircloth J. Snort 2.0 intrusion detection. Syngress, ISBN 1931836744; 2002.

CERT Coordinated Centre. Coordinated Centre Statistics 1988–2004, <www.cert.org/stats/cert_stats.html>.

Daniels T, Spafford E. A network audit system for host-based intrusion detection (NASHID) in Linux. In: 16th Annual computer security applications conference (ACSAC 00). Cerias Purdue University.

Fielding R, et al. Hypertext Transfer Protocol – HTTP/1.1. RFC 2068; 1998.

Forrest S, Hofmeyer S, Somayji A, Longstaff T. A sense of self for Unix processes. In: Proceedings of 1996 IEEE symposium on computer security and privacy; 1996.

Forrest S, Hofmeyer S, Somayji A. Computer immunology. Communications of the ACM 1997;4(10):88–96.

Goregaoker S. A method for detecting intrusion on encrypted traffic. Florida State University, Department of Computer Science, Master of Science Degree. TR-010703; 2001.

Jing-xin W, Zhi-ying W, Kui D. IDS, content filtering, Java, etc.: a network intrusion detection system based on the

- artificial neural networks. In: Proceedings of the third international conference on information security InfoSecu 2004.
- Jung J, Paxson V, Berger AB, Balakrishnan H. Fast Portscan detection using sequential hypothesis testing. In: Proceedings of IEEE symposium on security and privacy; May 9-12, 2004.
- Koba J. Windows NT attacks for the evaluation of intrusion detection systems. Master of Engineering in Electrical Engineering and Computer Science Thesis. Massachusetts Institute of Technology June 2000.
- Luo W, Cao X, Wang X. NIDS research based on artificial immunology. In: LNCS, vol. 222. Springer Link; October 24, 2001.
- Maimon U. Port scanning without the SYN flag, TCP port stealth scanning. Phrack Magazine 1996;7(49).
- Otely M, Parthasarathy S, Ghoting A, Li G, et al. Towards NIC-based intrusion detection. In: SIGKDD; 2003. p. 723-28.
- Patil S, Kashyap A, Sivathank G, Zadok E. I³FS: an in-Kernel integrity checker and intrusion detection file. In: Proceedings of 18th large installation system administration; 2004. p. 229-38.
- Porras PA, Neumann PG. Emerald: event monitoring enabling responses to anomalous live disturbances. In: 20th NIS security conference; October 1997.
- Price KE. Host-based misuse detection and conventional operating systems' audit data collection. Master's thesis, Purdue University. <<http://www.purdue.edu/>>; 1997.
- Ptacek T, Newsham T. Insertion, evasion, and denial of service: eluding network intrusion detection. Secure Networks, <www.aciri.org>; November 9, 2003.
- Seitz J. Demystifying the IPsec puzzle. In: Frankel Sheila, editor. Boston, London: Artech House, ISBN 1-58053-079-6; 2001. p. 273. Computer Standards & Interfaces 2001;24(1):87.
- Sekar R, Bendre M, Dhurjati D, Bollinen P. A fast automation based method for detection anomalous behaviours. In: Proceedings of IEEE symposium on security and privacy; 2001. p. 144-55.
- Shankar U, Paxson V. Active mapping: resisting NIDS evasion without altering traffic. In: Proceedings of the IEEE symposium on security and privacy; May 2003.
- Wu-Chang F. The case for TCP/IP puzzles. In: Proceedings of ACM SIGCOMM workshop; 2003.

Abiola Abimbola holds a first degree in Electrical and Electronics Engineering and a Master of Philosophy (MPhil) in network security. He is currently a candidate for a Ph.D. in network security at Napier University. He has practiced Network security for over 7 years.