NetHost-Sensor: Enhancing Intrusion Detection via An Active Target Host

Abiola Abimbola, Jose Munoz and William Buchanan
School of Computing, Napier University, EH10 5DT, Scotland, UK

ABSTRACT

Over the past decade, there have been increases in network attacks. These attacks are typically attempts to compromise the integrity, confidentiality or availability of networked resources. In other to reduce these attacks, Intrusion Detection Systems (IDS) were introduced. These systems monitor and analyse network traffic, and try to detect network attacks, and, in response, execute countermeasures, which overcome current security weaknesses. In this paper we present a quick review of IDS and their vulnerabilities, discuss, in detail, the performance unreliability of IDS's against end-to-end encrypted attacks, network fragmented attacks and denial of service exploitation of programming flaws. These vulnerabilies are illustrated in order to verify and validate the discussion. The experiments measure the performance of Snort, which is a network IDS which detecting the stated network attacks. Our experimental findings show that Snort could only detect 50% denial of service exploitation of programming flaws and 0% end-to-end encrypted attacks. In addition, we describe the *NetHost-Sensor* which is a methodology in thwarting these attacks.

Keywords: network attacks, integrity, confidentiality, intrusion detection system, countermeasure, performance, unreliability, validation, verification, and computer-based diagrams

1. Introduction

Computer intrusion can be defined as an action which attempts to compromise the integrity, confidentiality or availability of a resource [1]. The rapid acceptance of computer networks and the Internet have also brought about new security threats and challenges. In response, the security community have tried to overcome these threats with preventative techniques, such as access control and cryptography.

Access control attempts to associate a system user, identified by a request or login session, with a known profile. The privileges associated with this profile can then be retrieved and compared with current actions. Many technologies have been applied to this such as reusable passwords, cryptographic tokens (comprising time-based and challenge-response scheme), and biometrics (where a user is identified via some physical attribute such as the user's fingerprint). Unfortunately, the most prominent technique, reusable password, is fundamentally flawed, as most passwords that are sufficiently complex and short–lived to be safe from brute-force attacks, are also impossible to remember. Alternative techniques all suffer from technical or social problems. For example reliable biometrics are expensive and have not gained universal acceptance, while a token-based scheme requires the user to have the token available, when authenticating. In all cases, the management of users profiles can be complex and error prone.

A more fundamental problem with the use of access control is that most applications authenticate during the initialisation of a session or transaction. Thus, hijacking or corruption on existing session allows an attacker to impersonate the victim [2, 3]. Authenticating every part of a session is otherwise impractical. In addition, access control does not prevent insider attack [4, 5]. Other research efforts include:

- **Cryptography**. This is a series of mathematical-based techniques for preventing unauthorisation observation or modification of a communication stream, but does not guarantee that the original network packet's payload was initially innocuous.
- **Firewall**. This filters network traffic which allows the network to allow, block or modify connections according to security policies. These, though, are vulnerable to application level attacks, such as the HTTP –based exploit used in Nimda and Code-red [6], and more fully defined by Nacht [7].

Current computer security architecture offer no viable solution to some network, host and application level attacks. Bridging these gaps currently relies on other security techniques like Intrusion Detection Systems (IDS).

In this paper we initially define the concept an IDS, and identify key vulnerabilities of these. These vulnesabilities are then overcome using the NetHost-Sensor [8] system. The attacks identified are end-to-end encrypted attacks, network fragmented attacks and denial of service exploitation of programming flaw attacks. For this we compare and contrast the methodology with current IDS features, investigate experimentally the feasibility of executing these attacks on a network IDS and finally the results show a detection rate of 50% denial of service exploitation of programming flaws and 0% end-to-end encrypted attacks.

2. Background

The introduction of IDS, in other to monitor/analyse and detect network packets that compromise the confidentiality, integrity and reduce the quality of service, has helped secure most enterprise networks. In other to secure an enterprise network most IDS utilise anomaly or signature detection techniques. *Anomaly detection* relies on identifying behaviour that is abnormal for an entity. Whereas, *signature detection* technique identifies behaviour that is close to some previously defined pattern signature of a known intrusion. The problem with the anomaly detection technique is that it does not necessarily detect undesirable behaviour, and that the false alarm rates can be high. With signature detection there is a reliance on a well defined security policy that may be absent and its inability to detect intrusions that have not yet been made known to the IDS [9].

It is apparent that they consist of more than just a means of detecting intrusions [10, 11]. A closer study revels the following dichotomies in other system characteristics:

- **Time of detection**. Two main groups can be identified as those that attempt to detect intrusions in real-time or near real-time, and those that process audit data with some delay, postponing detection (non-real-time), which in turn delays the time of detection.
- **Granularity of data-processing**. This is a category s contrasts systems that process data continuously with those that process data in batches at a regular interval.
- **Source of audit data**. The two major sources of audit in the surveyed systems are network data (typically data read directly off a multicast network, such as in Ethernet) and host-based security logs. The host-based logs can include operating system kernel logs and application program logs.
- **Response to detected intrusion**. There are either passive or active. Passive systems respond by notifying the proper authority and they do not try to mitigate the damage done or seek to harm or hamper the attacker. Active systems attempt to exercise control over the attacked system in order to mitigate the effect of the attack or seek out the attacking system to prevent addition attacks.
- **Locus of data-processing**. The audit data can either be processed in a central location, irresponsible of whether the data originates from one-possible source, or collected and collated from many different sources in a distributed fashion.
- **Security**. This is the ability to withstand hostile attack against the IDS itself. A basic classification would use a high-low scale. Most systems, with the exception of [10], all fall into the low scale.

In fairness it should be said that not all of these categories are dichotomies in the true sense. However, the authors believe that many of the surveyed systems display sufficient differences for it to be meaningful.

3. Motivation

Despite the fact that research in IDS has existed for more than two decades, there are currently vulnerabilities in its methodology that have motivate us to proposed this research work. These include;

- **End-to-end (ETE) encryption**. With security improvement in communication protocols, the ability to encrypt traffic on an ETE basis is increasing. Besides thwarting an eavesdropper, encrypted content does not allow an IDS to monitoring and analysing network packet's payload for intrusion. If the IDS cannot analyse a packet's payload, this is likely to result in high false positive rates [11].
- **Vulnerability to direct attacks**. Most IDS really on a hierarchical structure for their elements, as a result, they are susceptible to attacks. For example an attack can cut off a central element by using DoS exploitation programming flaw attack [12] to cripple the element. Even worse, decapitate an internal or central node by taking out the root or command elements can cause serious problems.
- **Network-speed and infrastructure**. Fast network communication directly hinders an IDS from monitoring and analysing network packets, which results in many dropped network packets. In addition, the trend towards switched communication also increases the difficulty for a network IDS to monitor multiple communication streams. Finally, owing to multiple routing paths to a target host, a knowledgeable attacker can avoid the IDS monitoring and still attack a target host.
- **Network packet fragmentation**. The problems with network packets fragmentation in relation to network IDS [13] are:
    - o The target host and network IDS may reassemble out-of-order IP datagram fragments differently, as a result an attacker can intentionally scramble their IP datagram fragments to elude the network IDS.
    - o A network IDS can be attacked by flooding the network with partially fragmented IP datagram that will never be completed. A naïve network IDS will run out of memory as it attempts to cache each fragments for reassembly.
    - o 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 be-

tween a target and a network IDS can obscure the network IDS by couching malicious data inside overlapping fragments streams.

Other IDS vulnerabilities that exist include evasion and insertion attacks [13], and breath of attacks [14].

Figure 1 outlines the motivation of our novel proposed research work. Any ETE encrypted communication like IPSec [15] or Cisco Encryption Technology between participating peers (Host 1&2) will elude the scrutiny of the network IDS, since the network packet's payload will be encrypted. In addition, since the network IDS is between both hosts, it is susceptible to network fragmented, evasion and insertion attacks. How proposed research work overcomes these vulnerabilities by being hosted by the target host and using data from the network and transport layer of the target host as it audit sources for detecting intrusion. The NetHost-Sensor uses heuristic technique to determine/detect/prevent system calls of DoS exploitation programming flaw attacks that could shutdown a target host's application server. And finally, we include in the NetHost-Sensor a novel detection technique coined protocol analysis that has pre-stored knowledge of the communication protocol in use and analyses for intrusion using these pre-stored knowledge instead of using a rigid signature pattern-matching detection technique.

In Figure 1, there are two hosts participating in IPSec communication, using end-to-end encryption and authentication. The intermediate devices such as a networked IDS between this communication part will not be able to analyse network traffic for malicious payload or identify participating host. In addition, a network IDS will be susceptible to evasion, insertion and network fragmented attacks. The NetHost-Sensor overcomes these vulnerabilities in NBIDS by analysing audit data from the network and transport layer of participating host.

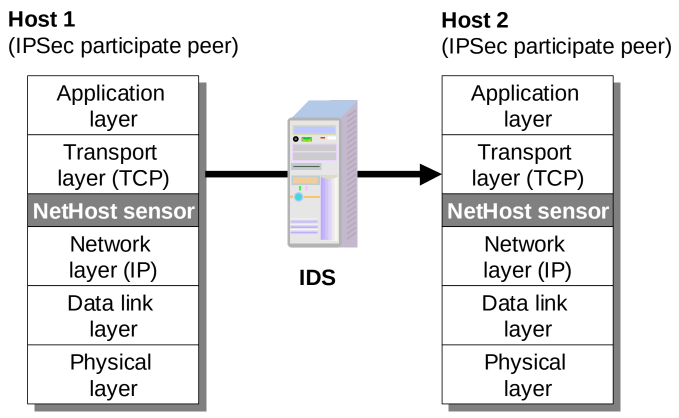

Figure 1: Two hosts participating in IPSec communication

4. Related Work

In this section we justify the novelty of the proposed research work by comparing it with relevant existing network IDS. Research work by Daniels and Spafford [16] tries to detect low-level network attacks by auditing the kernel of a Linux target host. This is done be collecting data from several points in the protocol stack of the target host. It would prevent ETE encryption and evasion attacks, if it addressed a filtering protocol against intrusion; as the researchers left the design of the protocol to users. The NetHost-Sensor with its protocol analysis technique will thwart ETE encryption and evasion attacks, and ensure the survivability of a target host's application server by preventing DoS exploitation programming flaw attacks.

Research work on a DoS resistant intrusion detection done by Mell et al [17] describes an architecture that makes IDS components invisible to attackers and allows IDS components to relocate from an attacked host to a more secure host through mobile agent technology. This is achieved by:

- Separate communication channels for IDS component and the rest of the network.
- A decentralised, non-hierarchical IDS system, without any interdependency.
- Mobile recoverable IDS components that move around the network and are able to be taken over by other mobile agents, if destroyed.

A vulnerability to this design is the number of available backups if an attacker should flood the entire system, using a DoS attack. However, NetHost-Sensor will not only be able detect threats through system calls, but also determine the direct parent process causing these threats and terminate all connections.

Paxson [18] has contributed to the attempted thwarting of evasion attacks, more so than any other researcher. These contributions include the following techniques in preventing evasion attacks:

- **Bifurcation analyses**. 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 that attempts to eliminate ambiguous network traffic and reducing the amount of connection state that the IDS's monitor must analyse.
- **Active mapping.** Which efficiently builds profiles of the network and TCP/IP policies of host on the network. An IDS may then use the host profiles to disambiguate the interpretation of the network traffic on a per-host basis.

These techniques are limited in scope as the bifurcating analyses can be subject to a DoS attack by overloading it with infinite threads, where as the traffic normalizer and active mapping only allow known network traffic (thus limiting their usage), and they both have to be updated regularly with new network knowledge. However, the NetHost-Sensor uses the target host's network and transport layer as its audit sources, where any network ambiguities will have been eliminated.

Research proposed by University of Idaho's Computer Science Department [19] uses a firewall mobile agent to handle a virtual private network through a firewall mobile custom agent (FMCA) and static agents. If Client B wants to communicate securely with a Host A, who sits behind a firewall, the firewall will send FMCA to Client B to inspect every network packets, and encrypt the legitimate ones, sign them and send them to Host A's firewall. Static agents in Host A's firewall verifies the signature and allow passage if the are valid. The research proposed assumes the agents to be *black boxes* and un-tampered, where all internal states and data are hidden from the users. If the proposed research is implemented *in the wild* without any mobile agent security [20], this assumption will serve as a vulnerability encouraging attacks.

Chari and Cheng [21] describes their experience in building *BlueBox*, which is a host-based IDS. Their approach can be viewed as creating an infrastructure for defining and enforcing very fine-grained process capabilities in the kernel. These capabilities are specified as a set of rules (*policies*) for regulating access to system resources on a per-execution basis. The expressed rules are intuitive and sufficiently expressive to effectively capture security boundaries. The BlueBox implements a *sandbox* around the kernel of its host, and as a result would probably detect DoS exploitation programming flaw attacks. The main problem with BlueBox is that it uses system call analysis as the only means of detecting intrusions, hence low-level attacks at the network layer may elude the BlueBox. In addition, any variation in monitored application will require a new set of rules (system calls). Finally, memory attacks will elude the BlueBox IDS, since checks on process behaviour are made only when the process makes a system call. However, the NetHost-Sensor makes use of system call analysis for thwarting DoS exploitation programming flaw attacks and network protocol analysis for detecting malicious data in the network and transport layer of the target host. As a result of the varied detection technique of the NetHost-Sensor, a higher false positive rate is expected.

Snort [22], the chosen network IDS used in our experiment described in the next section. It 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 and can be used to detect a verity of attacks and probes, such as buffer overflow, stealth port scans, common gateway interface (CGI) attacks. Snort is currently undergoing rapid development and addition security features will soon by introduced. The creators of Snort do not attempt to tackle ETE encryption, DoS exploitation programming flaw attacks as our novel proposed research work, but made the following efforts in thwarting evasion and network fragmented attacks:

- **Rule Optimiser**. This is a major component of Snort detection engine. It optimisers the 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 IDS like Real Secure Network Sensor (RNS), Internet Security System's (ISS) Micro Agent, Network Flight Recorder (NFR) and NetRanger are not capable of thwarting ETE encryption attacks, DoS exploitation programming flaw and network fragmented attacks [8]. The NetHost-Sensor system is hosted by the target host and uses the target host's network and transport layer data to analyse for network intrusions and also forms a system call wrapper around an application server of the target host, as a result mitigating these attacks.

5. Experimental Details

This section describes the experiments carried out with the following objectives:

- Investigating the feasibility of creating an ETE encryption channel using IPSec,
- Investigating the severity of the attacks by measuring the performance of Snort– a network-based IDS in detecting encrypted network packet attacks, network fragmented attacks and DoS exploitation programming flaw attacks.

The experiment initially involved launching various DoS exploitation programming flaw attacks from an attack host to a target host. Snort then detected and recorded intrusive attacks between both hosts, as illustrated in Figure 2. Then both the attack and target hosts were configured to communicate using an IPSec encryption channel. While using IPSec encryption channel, intrusive attacks were launched from the attack host to the target host. Snort analyses network packets, for intrusive attacks in the encrypted channel. Table 1 defines the configuration, where the network topology is given in Figure 2. Each experiment used Windows 2000, with 128MB of memory and 8GB of disk space.

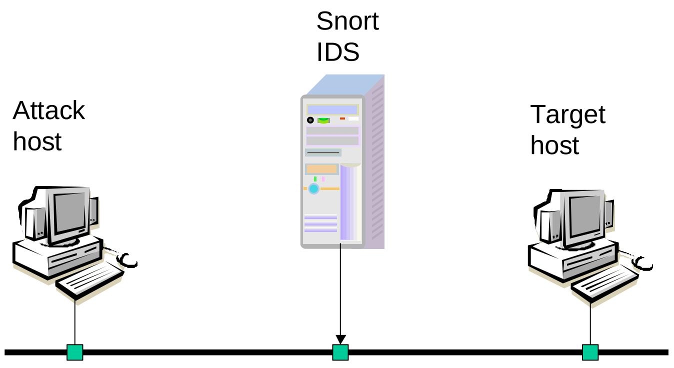

Figure 2: Diagrammatic representation of the experimental set-up

The attacking tools[1] used were many and varied (25 different tools in total) including DoS tools, network fragmented attacks tools, and port vulnerability attack tools.

In configuring local IPSec policies for the target and attack hosts, the following summarised procedure was used. Each ran Microsoft Management Console (MMC) on Windows 2000, following by the IP Security policies to modify settings. Next PRE-SHAERD key was created (such as 123456789) for both target and attack hosts, and this was applied to all IP traffic before clicking on assign secure traffic.

Table 1: Configuration of target, network IDS and attack hosts

Hosts	**IP Address**	**Purpose**
Target	169.254.118.102	Target Host
IDS	169.254.64.108	Installed Snort-a network IDS and IRIS- a network analyser[2] ()
Attack host	169.254.219.28	Installed various Windows attacking tools

6. Experiments Results

To test the configuration of IPSec ETE encryption, at the command prompt, the IP address of the target host from the attack host we pinged before and after assigning secure traffic. In the case of non-secure traffic, the packet payload was not encrypted, while for secure traffic it was. Figure 3 shows the network payload capture of both cases.

During the experiment, several attacks were launched from the attack host to the target host. In the attempt to determine Snort's performance, a measurement of detecting these attacks is give in Table 2.

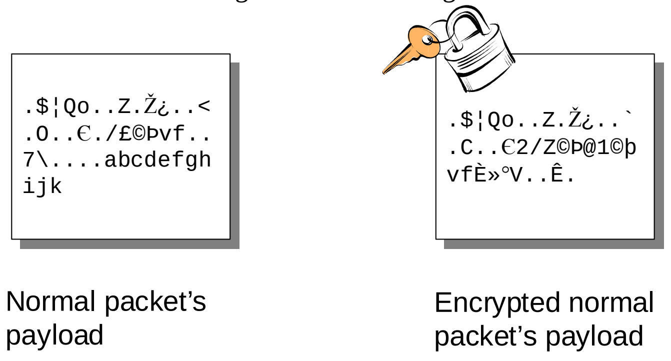

Figure 3: Normal and encrypted packet's payload

Table 2: Experimental Results

Type of attack launched	Number of attacks launched	Number of attacks detected by Snort
DoS Attacks	20	10
Encrypted Attacks	20	0

6. NetHost-Sensor

This section describes the methodology of the NetHost-Sensor and its unique features in thwarting ETE encryption attacks, DoS exploitation programming flaw and network fragmented attacks. In order for the NetHost-Sensor to thwart these attacks, we collect only valid network data and implement our data collection within the protocol stack of the target host. We attempt to use as our audit source the network and transport layers of the target host, since at these layers all network packets will have been decrypted and all fragmented network packets reassembled to enable more efficient intrusion detection. Figure 4 illustrates of NetHost–Sensor position on a protocol stack of a target host.

An obvious weakness of our approach is that it may not be portable since it is embedded within the protocol stack of the target host. Our assertion is that for a network IDS to avoid the pitfalls describes by Ptacek and Newsham [13], it must be customised to the target host, anyway. One approach would be to emulate the protocol stack based on the low-level interface provided by the operating system. This is difficult as it requires deep understand of the behaviour of a particular implementation. Our approach involves no modification to the kernel and does not duplicate efforts already done in the kernel, although an Application Protocol Interface (API) can be designed for each platform, making the NetHost-Sensor platform independent.

[1] www.networder.bos.sk
[2] www.eeye.com

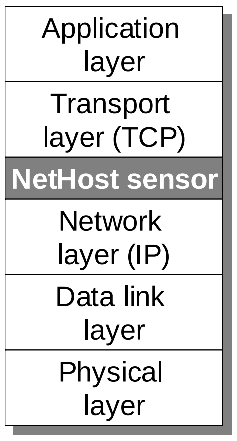

Figure 4: NetHost-Sensor in a protocol stack of a target host

6.1 NetHost-Sensor's Detection Technique

The basic technology to implement *signatures* is to simply record a unique pattern of an attack and search for it within network packet's payload. There are a number of problems with signature detection technique as it does not truly understand the nature of a network packet. For example, the pattern "`/cgi-bin/phf`" is flagged as an intrusion by Snort IDS may appear in a packet's payload for a reason not related to an intrusion, thereby causing Snort to trigger a false positive.

Rather than processing just the surface of packets, the NetHost-Sensor will dig deeper into network packet's payload, reconstructing the original meaning of the data. This requires that a lot more code be written. One way to understand this is using a full HTTP request involving "`/cgi-bin/phf`" such as:

```
GET/index.htmlHTTP/1.0Host:www.bellepress.comRefere
   rhttp://www.bellepress.com/cgi-bin/phfUser-
   Agent:Mozilla/2.0
```

In this HTTP request, NetHost-Sensor's protocol analysis applies more intelligence, by pulling apart each of the fields within the header and assigning *meaning* to them. This will give:

```
Method=GET
URL=/index.html
Version=HTTP/1.0
Fieldname=Host
HTTP_HOST=www.bellepress.com
Fieldname=Referrer
HTTP_REFERRER=http://www.bellepress.com/cgi-bin/phf
Fieldname=User-Agent
HTTP8_USERAGENT=Mozilla/2.0
```

Another vulnerabilityof signature detection technique is evasion attacks that can be described using Simple Network Management Protocol (SNMP) network packets. A raw Snort signature for detecting an example of SNMP attack is:

```
 Alert udp!$HOME_NET any→$HOME_NET 161 (msg:
   "NETBIOS_SNMP-NT USERList";content: "|2b 06
   01 04 01 4d 01 02 19|";)
```

However, SNMP allows padding within the data. With extra padding, the data on the network packet's payload would actually be sent as: `2b 80 06 80 0180 04 80 0180 4d 80 01 80 02 80 19`, as the original pattern has been *smudged*, Snort IDS will no longer trigger on this attack. In contrast, the NetHost-Sensor will automatically *unsmudge* the data back into a canonical form and correctly trigger on the intrusion, no matter how much extra padding is added to the data. Most protocols allow similar sorts of encoding or smudging that will hide the true signature of the intrusion. The NetHost-Sensor will automatically handle this through its protocol analysis technique, but most IDS that implement signature detection technique will fail to detect the intrusion.

6.2 Thwarting DoS exploitation of programming flaws

Using DoS exploitation of programming flaws as an example of a threat to survivability of a system, which we define as the capability of a system to fulfil its mission in a timely manner even in the presence of attacks or failures. We develop a novel concept within the NetHost-Sensor, using heuristic evaluation of several DoS exploitations of programming flaw attacks to determine the penultimate system call or a series of system calls that leads to an application crash or halt. Described more formally:

Let:
DoS exploitation of a program flaw be = EXEDoS(Appll)

Where:
EXEDoS = execution of Dos exploit
Appll = exploited application e.g telnet

Assuming that:
System calls pertaining to Appll = H
Since the execution of an application is a series of system calls

Therefore:
Execution of Appll= $H_0 \ldots\ldots H_{end}$

Where:
H_0 is the initial system call
H_{end} is the end of execution system call

We can then say:

$$\text{ExeDoS(Appll)} = H_{0\ldots\ldots}H_{DoS/a\ldots\ldots}H_{DoS/b\ldots\ldots}H_{DoS/crash-1\ldots\ldots}H_{DoS/crash}$$

Where;
$H_{DoS/a\ldots\ldots}H_{DoS/a\ldots}H_{DoS/b}$ = systems calls leading to abnormal apllication crash
$H_{DoS/crash-1}$ = penultimate system call that crashes the application
$H_{DoS/crash}$ = system call that crashes the application
From the above statements we can conclude that a resistance to DoS Exploitation of a program flaw monitoring the summation of H and stopping any series of $H_{DoS/a\ldots\ldots}H_{DoS/b\ldots\ldots}H_{DoS/crash-1}$ from executing as shown mathematically below.

$$\sum^{H_0 < H < H_{Dos/a-1}} EXEDoS(Appll)$$

To fully elimiante the DoS exploitation threat,
we determine the parent process identification of any system call within $H_{DoS/a....}H_{DoS/crash-1}$

from executing Appll. In knowing the Appll from any H series, we can terminate the Appll and any further exploitation.

By having the NetHost-Sensor stop DoS exploitation, threats leading to crash attacks on monitored target host's applications, we have introduced a survivability factor into our novel research proposal "the NetHost-Sensor".

7. Conclusion

Despite the research carried out by researchers over the past two decades, existing IDS have not met the expectation that motivated initial work. In this paper we have described IDS's and classified them using various criteria. We then proceeded to the current vulnerabilities affecting IDS and single out a few of the key motivations to our proposed research work, the *NetHost-Sensor*. We justify our proposed research work by comparing it with current IDS in thwarting DoS exploitation of programming flaws, ETE encryption and network fragmented attacks. We investigated, experimentally, the severity and feasibility of Snort-a network IDS in detecting DoS exploitation of programming flaws, ETE encryption and network fragmented attacks and reported the findings. We have finally proposed the design of a NetHost-Sensor, a network IDS that thwarts DoS exploitation of programming flaws, ETE encryption and network fragmented attacks. The NetHost-Sensor features two novel concepts:

- It sits between the network and transport layers of the target host and uses these layers has its audit source in detecting intrusions.
- A wrapper round a target host application server using system calls to determinate DoS exploitation of programming flaws and finally employs a protocol analysis detection technique to thwart network fragmentation attacks.

We are current experimenting on a network and transport layer of a Windows NT machine to determine the adequate points in the network and transport layer of the protocol stack to use as an audit source and then progress to implement the protocol analysis detection technique.

8. References

[1] R.Heady, G.Luger, A.Maccabe and M.Servilla, "The Architecture of a Network Level Intrusion Detection System", Technical Report CS90-20, University of New Mexico, Department of Computer Science, August 1990

[2] S.Bellovin, "Security Problems in the TCP/IP Protocol Suite", Computer Communications Rev, 19 (2): 32-48, 1989

[3] D.Dean, M.Franklin and A.Stubblefield, "An Algebraic Approach to IP Traceback". Information and System Security, 5(2), 119-137, 2002

[4] E.Schultz," A Framework for Understanding and Predicting Insider Attacks", Computer & Security, 21(6), 526-531, October 2002

[5] CERT Coordination Center Advisory CA-1995-01, "IP Spoofing Attacks and Hijacked Terminal Connections", Available online www.cert.org/ advisory/ CA-1995-01

[6] W.Madsen, "FBI At Centre Stage Of Code Red", Network Security, 2001(8), 14-15, 1 August 2001

[7] M.Nacht, "The Spectrum of Modern Firewalls", Computers & Security, 17(1), 54-56, 1998

[8] A.Abimbola, Q.Shi and M.Merabti, "NetHost-Sensor: A Novel Concept In Intrusion Detection Systems", Eight IEEE International Symposiums on Computers and Communications, 232-240, June 30-July 03

[9] T.Verwoerd and R.Hunt, "Intrusion Detection Techniques and Approaches", Computer Communications, 25(15) 1356-1365,15 September 2002

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

[11] S. Goregaoker, "A Method for Detecting Intrusion on Encrypted Traffic", Florida State University, Department of Computer Science, Master of Science Degree.TR-010703, 2001

[12] A.Abimbola, "Denial of Service Attack: What is Going on ?", ISSA Journal, November Issue, 2003

[13] T.Ptacek and T.Newsham, "Insertion, Evasion, and Denail of Service: Eluding Network Intrusion Detection," Secure Networks, www.aciri.org, November 9, 2003

[14] J.Koba, "Windows NT Attacks for the Evaluation of Intrusion Detection Systems", Master of Engineering in Electrical Engineering and Computer Science Thesis, Massachusetts Institute of Technology

[15] J.Seitz, "Demystifying the IPSec Puzzle:" Sheila Frankel, 273 pages, Boston, London: Artech House 2001, ISBN 1-58053-079-6, Computer Standards & Interfaces, 24(1), Page 87, March 2002

[16] T.Daniels, E.Spafford, "A Network Audit System for Host-Based Intrusion Detection (NASHID) in Linux", Cerias Purdue University, 16th Annual Computer Security Applications Conference (ACSAC 00)

[17] P.Mell, D.Marks, M.Mclarnon. "A Denial of Service Resistance Intrusion Detection Architecture, Computer Network, 34(4), 641-658, 2000

[18] U.Shankar and V. Paxson. "Active Mapping: Resisting NIDS Evasion Without Altering Traffic", Proc. IEEE Symposium on Security and Privacy, May 2003

[19] University of Idaho, Department of Computer System, Firewall Mobile Customs Agent Project, www.cs.uidaho.edu, November, 2003

[20] W.Jansen, "Countermeasures for Mobile Agent Security", Computer Communications, 23(17), 1667-1676 1 November 2000

[21] S.Chari and P.Cheng, “BlueBox: A Policy-Driven, Host-Based Intrusion Detection System”, ACM Transactions on Information and System Security (TISSEC), 6(2), 173-200, 2003

[22] J.Foster, B.Caswell, and J.Beale (Editor) J.Faircloth, “Snort 2.0 Intrusion Detection”, Publisher (Syngress), ISBN-1931836744, 2002