The Nucleus, **45** (1-2) 2008: 11-31

The Nucleus

A Quarterly Scientific Journal of Pakistan Atomic

NETWORK SECURITY: A SURVEY OF MODERN APPROACHES

*M.F. ZAFAR, F. NAHEED, Z. AHMAD and M.M. ANWAR

NITS Division, ICCC, P.O. Box 2191, Islamabad, Pakistan

(Received October 4, 2007 and accepted in revised form May 27, 2008)

Security is an essential element of information technology (IT) infrastructure and applications. Concerns about security of networks and information systems have been growing alongwith the rapid increase in the number of network users and the value of their transactions. The hasty security threats have driven the development of security products known as Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) to detect and protect the network, server and desktop infrastructure ahead of the threat. Authentication and signing techniques are used to prevent integrity threats. Users, devices, and applications should always be authenticated and authorized before they are allowed to access networking resources. Though a lot of information is available on the internet about IDS and IPS but it all is spread on so many sites and one has to spend a considerable part of his precious time to search it. In this regard a thorough survey has been conducted to facilitate and assist the researchers. The issues and defend challenges in fighting with cyber attacks have been discussed. A comparison of the categories of network security technologies has been presented. In this paper an effort has been made to gather the scattered information and present it at one place. This survey will provide best available uptodate advancement in the area. A brief description of open source IPS has also been presented.

Keywords: Network security, Intrusion detection, Intrusion prevention, Security threats, Cyber attacks, Information security.

1. Introduction

Network security is likely to become a key factor in the development of the information society as networking plays an important role in economic and social life. Attacks against networks and computers, threatening the operation of businesses and the privacy of corporate and personal data, continue to make headlines. Security in network design can no longer be an afterthought, but rather a pervasive system characteristic. Security must be at the forefront of concerns of IT managers [1]. They are responsible for ensuring that authorized users are accessing only the information they should and preventing infiltration of their corporate networks by unauthorized individuals [2]. Computers connected to networks are exposed to potentially damaging access by unauthorized "hackers". Protecting sensitive data and providing a stable computing environment must be a priority task [3]. In the recent years, however, this task has grown increasingly more difficult due to a variety of factors [4]:

- The number of users and the ways in which they access the network continues to expand, making it harder to tightly control and opening up many avenues for inappropriate use of resources.
- The quantity and complexity of attacks continues to grow, often exploiting vulnerabilities in the application-layer that require sophisticated attack detection and analysis to identify and mitigate.
- Hacking/attacking tools are widely available on the Internet and have become significantly less complicated, making it possible for almost any Internet user to download and run an exploit against an organization.
- Attacks increasingly target Windows components, rather than server software, which translate into more potentially vulnerable systems.

* Corresponding author : *hmfzafar@iccc.org.pk

- The number of vulnerabilities continues to increase, with the average time from vulnerability announcement to actual exploit release decreasing, further compounding the difficulties in ensuring effective security patching to protect the network.

Table 1. The list of abbreviations used through out this paper.

IDS	Intrusion Detection System
DNS	Domain Name System
ICMP	Internet Control Message Protocol
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol /Internet Protocol
OS	Operating System
IP	Internet Protocol
DoS	Denial of Service
POP	Post Office Protocol
SMTP	Simple Mail Transfer Protocol
DDoS	Distributed Denial of Service
IPS	Intrusion Prevention Systems
NIPS	Network IPS
HIPS	Host IPS
SMTP	Simple Mail Transfer Protocol
HTTP	Hyper Text Transfer Protocol
MAC	Mandatory Access Control

Potential hackers can break into system by exploiting security holes or bugs in the software that provides the network service. Hackers generally search for these bugs by "scanning" the network. V. Yegneswaran *et al.* [5] aggregated and analyzed firewall logs from over 1600 networks and reported that about 3 million scans happened everyday and 20% to 60% of these scans are Web server vulnerability scans and are linked to worm propagation attempts. Attackers use various kinds of scanning strategies to choose addresses of potentially vulnerable machines to scan [6].

A seamless security infrastructure protects against attacks on network devices as well as on applications. In the traditional environment, networks are often implemented using unique security mechanisms to detect and respond to attacks. Intrusion detection systems (IDS) and intrusion prevention systems (IPS) can detect and prevent attacks on the data network. Each threat and threat category highlight the potential for the loss of significant tangible and intangible business as well as private value. Solutions to these areas of concern should be factored into any network security design.

The rest of the paper is organised as follows. Section 2 describes the types of cyber attacks. Section 3 is about the firewall. Section 4 presents a detailed discussion of intrusion detection. In section 5, a brief introduction of intrusion prevention system has been presented. Section 6 of this paper is about implementation challenges of IPS. The requirements for effective intrusion prevention are highlighted in section 7. Available open source prevention solutions are discussed in section 8. At the end, section 9 concludes the whole study.

2. Cyber Attacks

A cyber attack is defined as a failed attempt to enter the system (no violation committed) [7]. Generally, attacks can be categorized in two areas:

- Passive (aimed at gaining access to penetrate the system without compromising IT resources),
- Active (results in an unauthorized state change of IT resources).

In terms of the relation intruder-victim, attacks are categorized as:

- Internal, coming from own enterprise's employees or their business partners or customers,
- External, coming from outside, frequently via the Internet.

Attacks are also identified by the source category, namely those performed from internal systems (local network), the Internet or from remote dial-in sources. The following types of attacks are detectable by IDS tools and can be in the ad-hoc categorization:

- Those related to unauthorized access to the resources:
  - Stealing information, for example disclosure of proprietary information,
  - Password cracking and access violation,

- Trojan horses,
- Spoofing (deliberately misleading by impersonating or masquerading the host identity by placing forged data in the cache of the named server i.e. DNS spoofing,
- Scanning ports and services, including Internet ICMP scanning (Ping), UDP, TCP Stealth Scanning (TCP that takes advantage of a partial TCP connection establishment protocol.) etc.
- Interceptions; most frequently associated with TCP/IP stealing and interceptions that often employ additional mechanisms to compromise operation of attacked systems (for example by flooding),
- Remote OS Fingerprinting, for example by testing typical responses on specific packets, addresses of open ports, standard application responses (banner checks), IP stack parameters etc.,
- Network packet listening (a passive attack that is difficult to detect but sometimes possible),
- Unauthorized network connections,
- Taking advantage of system weaknesses to gain access to resources or privileges,

- Unauthorized alteration of resources (after gaining unauthorized access):
  - Falsification of identity, for example to get system administrator rights,
  - Information altering and deletion,
  - Unauthorized transmission and creation of data (sets), for example arranging a database of stolen credit card numbers on a government computer,
  - Unauthorized configuration changes to systems and network services (servers).
- Denial of Service (DoS):
  - Flooding – compromising a system by sending huge amounts of useless information to lock out legitimate traffic and deny services:
- Ping flood (Smurf) – a large number of ICMP packets sent to a broadcast address,
- Send mail flood - flooding with hundreds of thousands of messages in a short period of time; also POP (Post Office Protocol) and SMTP (Simple Mail Transfer Protocol) relaying,
- SYN flood – initiating huge amounts of TCP requests and not completing handshakes as required by the protocol,
- DDoS; coming from a multiple source,
- Compromising the systems by taking advantage of their vulnerabilities:
- Buffer Overflow, for example Ping of Death — sending a very large ICMP (exceeding 64 KB),
- Remote System Shutdown,
- Web Application attacks; attacks that take advantage of application bugs may cause the same problems as described above.

It is important to remember, that most attacks are not a single action, rather a series of individual events developed in a coordinated manner. In the following paragraphs some of the threats have been discussed in brief alongwith protection measures and approaches which have been or could be taken against them.

Threats of disclosure include eavesdropping. Eavesdropping involves sniffing network packets for data that can be interpreted in real-time or saved for later analysis or playback. The probability of being vulnerable to eavesdropping increases as shared IP networks are directly accessible with wider user access and thus are easier to sniff for traffic [1].

A fully switched network limits the ability to "eavesdrop" on network traffic. This means that data packets exchanged by two computers are not broadcasted to any other computers on the network. But connections into computers from home or other daughter institutions may be vulnerable to eavesdropping [3].

Encryption can prevent disclosure threats. Encrypting stored files is a technique to prevent loss of sensitive data. Access to data for the purpose of decryption must be controlled using strong authentication and authorization techniques such as challenge-response techniques, one-time passwords, and role based access control. Thus

encryption is valuable but not sufficient to protect against disclosure.

Integrity threats are threats based on the insertion of bogus content into files or communication streams. Attackers may insert malicious or misleading data into unprotected files. When read or executed with the assumption that the files have integrity, the corrupt files may disrupt system operation. Attackers may also change the contents of data as they are transferred resulting in the improper interpretation of the data. Another integrity threat involves an attacker spoofing the identity of a valid user. When successful, the imposter may gain access to proprietary information or systems and operate with the full privileges of the impersonated user [1].

DoS attacks typically flood the network, with traffic in an attempt to render the entire network, to be unusable by authorized users. DoS attacks take many forms that include ICMP Floods, synchronised packet in transmission control protocol (TCP SYN) Floods, and UDP Floods. One common DoS attack technique, buffer overflows, may not only crash the targeted device but also be used as a means to gain control of the target and permit the attacker complete, privileged access. The library routines prevent a major source of buffer overflow attacks that may occur in applications, especially those that provide remote services and execute with root privileges [1].

Approaches to detect buffer overflow attacks can be divided into two groups: static techniques that detect potential buffer overruns by examining program's source code and dynamic techniques that protect programs at runtime. Wilander *et al.* [8, 9] presented a comprehensive overview of tools of both types. Greiner [10] gives an overview of manual code auditing techniques that help detect potential vulnerabilities. The real cause of buffer overflows is unchecked pointer or array access. Jones and Kelly [11] and Austin *et al.* [12] proposed to check each pointer access at run time to solve this problem. Purify [13] is a similar tool that instruments program's object code and therefore does not require access to its source code. CRED [14] is a project that aims to provide a comprehensive memory access bounds checking at a reasonable cost.

The return address is the most common target of buffer overflow attacks. Stackguard [15] is a system that protects the return address by placing a *canary word* on the stack before the return address. If the canary word is found modified upon the function return then an attack has taken place. RAD [16] takes a different approach. It compares the return address on the stack with the saved value and raises the red flag if the two values are different. StackShield [17], ProPolice [18] and StackGhost [19] are similar systems that protect other code pointers such as function pointers and stack frame register in addition to the return address. FormatGuard [20] provides a set of wrapper functions that protect a program from format string attacks.

For spoofing-based attacks, we need to identify the sources of attack traffic. This kind of approaches [21,22] try to figure out which machines attacks come from. Then appropriate measurement will be taking on those machines (or near them) and eliminate the attacks. In the case where attacker has a vast supply of machines, the trace approaches become not too helpful. A good example of the trace back technique is Traceback. Traceback [21] which locates the agent machines making the DDoS attacks.

DDoS attacks occur when an attacker gains control of multiple computers and directs them to simultaneously attack a single target. This type of DoS attack is more difficult to thwart because the perpetrators are more numerous. There have been a number of proposals and solutions to the DDoS attacks. However, there is still no comprehensive solution which can protect against all known forms of DDoS attacks [23].

3. Firewalls

A firewall is a hardware or software device which is configured to permit, deny, or proxy data through a computer network which has different levels of trust. There are several classifications of firewalls depending on where the communication is taking place or is intercepted and the state that is being traced.

While firewalls are certainly the first-line of defense and an absolute requirement for any company connecting to the Internet, so organizations have realized they cannot be the only line of defense. As a result, most

organizations have adopted a layered approach to network security to try to minimize the risks to their critical assets [4].

It is apparent that firewalls are not always effective against many intrusion attempts [24, 25]. The average firewall is designed to deny clearly suspicious traffic - such as an attempt to telnet to a device when corporate security policy forbids telnet access completely - but is also designed to allow some traffic through - Web traffic to an internal Web server, for example [26]. It's also generally accepted that a firewall is not enough to completely secure a network, for the following reasons:

- Firewalls are not always perfectly administered. Human error accounts for a significant number of security breaches.
- Trojan software can be downloaded disguised as something else, which the firewall doesn't block. Such software can then use “trusted” protocols (such as HTTP) to tunnel through the firewall, and allow remote exploitation of PC's within the private network.
- Back doors may exist, such as ISDN modems, or unauthorized wireless LAN links that are connected to a PC on the internal network. These can be used to bypass the firewall.

The problem is that many programmes called “Exploits” attempt to exploit the weaknesses in the protocols those are admissible through perimeter firewalls, and once the web server has been compromised, this can often be used as a spring board to launch additional attacks on other internal servers. Once a “root kit” and “backdoor” has been installed on a server, the hacker has ensured that he / she will have unfettered access to that machine at any point in the future [26].

Firewalls are also typically employed only at the network perimeter. However, many attacks, intentional or otherwise, are launched from within an organisation. Virtual private networks, laptops, and wireless networks all provide access to the internal network that often bypasses the firewall [26].

4. Intrusion Detection Systems (IDS)

An Intrusion Detection System (abbreviated as IDS) is a defense system, which detects hostile activities in a network. One key feature of intrusion detection systems is their ability to provide a view of unusual activity and issue alerts notifying administrators and/or block a suspected connection. In addition, IDS tools are capable of distinguishing between insider attacks and external ones.

Intrusion detection is an essential module in all network security products. It is the art of detecting inappropriate, incorrect, or anomalous activity [27]. Originally, system administrators used to perform intrusion detection by sitting in front of a console and monitoring user activities. Although effective enough at the time, this early form of intrusion detection was ad hoc and not scalable [28, 29].

In the late ’70s and early ’80s, searching through a stack of printed audit logs, four- to five-feet high, was obviously very time consuming. With this overabundance of information and only manual analysis, administrators mainly used audit logs as a forensic tool to determine the cause of a particular security incident after the fact. There was little hope of catching an attack in progress [30].

In the early ’90s, researchers developed real-time intrusion detection systems that reviewed audit data as it was produced. This enabled the detection of attacks and attempted attacks as they occurred, which in turn allowed for real-time response, and, in some cases, attack pre-emption. More recent intrusion detection efforts have centered on developing products that users can effectively deploy in large networks. This is no easy task, given increasing security concerns, countless new attack techniques, and continuous changes in the surrounding computing environment [31, 32].

Due to growing number of intrusions and use of the Internet and local networks, organizations are increasingly implementing IDSs which monitor IT security breaches.

4.1. Symptoms of intrusion

Let us take a closer look at the types of symptoms that are helpful in tracing intruders [7].

Utilizing known vulnerabilities: In most cases, any attempt to take advantage of faults in organization security systems may be considered as an attack and this is the most common symptom of an

intrusion. However, the organization itself may "facilitate" the task of attackers, using tools which aid in the process of securing its network – so called security and file integrity scanners. They operate either locally (assisting system administrators in vulnerability assessment) or remotely but may also be deliberately used by intruders. Since these tools are often a double-edged sword and are available for both the users and hackers, accurate monitoring of the usage of file integrity scanners and known vulnerability scanners is needed, to detect attacks in progress or trace damages from successful attacks. Hence, the following technical issue arises:

- Detection of file integrity scanners - The available file integrity testing tools operate in a systematic manner so that it is possible to use modeling techniques and specialized tools for detection purposes.
- A good correlation between scanning and usage is required – scanning for flaws may further use a service featuring such flaws, this may be a precursor of an attack to come.

Recurrent abnormal network activity: An intruder actually trying to compromise a system often uses a large number of exploits and makes many unsuccessful attempts. His activities differ from those of the user working with the system [33]. Any penetration-testing tool should be able to identify suspicious activities after a certain threshold has been exceeded. Then, an alert may be produced and diffused. This passive technique allows detection of intruders without discovering a clear picture of the event (exploits involved, tools, services, software configuration, etc.), by only quantitatively examining network activities.

Directional inconsistancies in traffic: Any directional inconsistency in packets or sessions is one of the symptoms of a potential attack. Considering the source address and location (outbound or inbound) can identify the direction of a packet. Session flow is identified by the direction of the first packet of that session. Therefore, a request for service on a local network is an incoming session and a process of activating a Web based service from a local network is an outgoing session. The following directional inconsistencies may be considered as attack evidence indicators:

- Packets originating in the Internet (incoming) and identified by their local network source address – request for service incoming from outside, for which the packets have their internal source address. This situation may indicate a possible outside IP spoofing attack. Such problems can be routinely solved at routers that can compare the source address with the destination location. In practice, few routers support this security option since this is the domain of firewalls.
- Packets originating in a local network (outgoing) and sent to an external network with an external destination address – this is a reverse case. An intrusion attempt is accomplished from outside and targeted at an external system.
- Packets with unexpected source or destination ports – if the source port of an incoming or outgoing request is not consistent with the type of service, this may indicate an intrusion attempt (or system scanning). Directional inconsistencies are most likely to be detected by firewalls that simply drop illegal packets. However, firewalls are not always merged with intrusion detection systems, therefore it is expected that the latter will also remedy the above problem.

Unexpected attributes as an intrusion symptom: The most frequent cases are the ones where one is expected to deal with a set of attributes of packets or specific requests for services. It is possible to define the expected attribute pattern. If encountered attributes do not match this pattern, this may indicate a successful intrusion or intrusive attempt.

Unexplained problems as intrusion indicators: A potential intruder may design its malicious activity with side effects that will cause odd behavior of the system. Monitoring such side effects is difficult since their location is hardly detectable [7]. Below there are some examples of:

- Unexplained problems with system hardware or software, for example server down, particularly daemons not running, unexplained system restart attempts, changes to system clock settings.

- Unexplained system resource problems: file system overflow; abnormal consumption of CPU usage.
- Odd messages from system daemons, system daemons not running or disturbed (particularly superuser daemons and those designed to monitor the system state, for example Syslog). Such symptoms are always suspicious.
- Unexplained system performance problems (routers or system services, for example long server response times).
- Unexplained user process behavior, for example unexpected access to system resources.
- Unexplained audit log behavior. Audit logs that shrink in size (unless intentionally reduced by the system administrator).

4.2 Detection techniques

In computer networks, intrusion detection is usually achieved in two ways – at the network level or at the host level. Network Intrusion Detection (NID) means scanning packets in network segments, looking for evidence of known attack signatures, or other suspicious activity. This is highly dependent upon the existing policies of the network and firewall administration [26]. The other form is Host-based Intrusion Detection (HID). This works on the principle that if a computer is successfully compromised by an intruder (who may be working locally and therefore won't be seen by the network), he will tend to make changes to certain files, loading exploitation programs or other unauthorized software. This can be detected by regularly running scans, and informing the central authority if there are any changes [34].

There are two basic categories of intrusion detection techniques: anomaly detection and misuse detection.

Anomaly-based detection uses models of the intended behavior of users and applications, interpreting deviations from this “normal” behavior as a problem [35, 36]. A basic assumption of anomaly detection is that attacks differ from normal behavior. The main advantage of anomaly detection systems is that they can detect previously unknown attacks. In actual systems, however, the advantage of detecting previously unknown attacks is paid for in terms of high false-positive rates. Anomaly detection systems are also difficult to train in highly dynamic environments. Another noteworthy problem with the use of anomaly-based detection techniques is that it is often difficult for analysts to determine why a particular alert was generated and to validate that an alert is accurate and not a false positive, because of the complexity of events and number of events that may have caused the alert to be generated [34].

Misuse (Signature-based) detection systems essentially define what’s wrong. They contain attack descriptions (or “signatures”) and match them against the audit data stream, looking for evidence of known attacks [37, 38]. The main advantage of misuse detection systems is that they focus analysis on the audit data and typically produce few false positives. The main disadvantage of misuse detection systems is that they can detect only known attacks for which they have a defined signature. As new attacks are discovered, developers must model and add them to the signature database.

A signature is a pattern that corresponds to a known threat. *Signature-based detection* is the process of comparing signatures against observed events to identify possible incidents. Signature-based detection is very effective at detecting known threats but largely ineffective at detecting previously unknown threats, threats disguised by the use of evasion techniques, and many variants of known threats.

Signature-based detection is the simplest detection method because it just compares the current unit of activity, such as a packet or a log entry, to a list of signatures using string comparison operations. Signature-based detection technologies have little understanding of many network or application protocols and cannot track and understand the state of complex communications. They also lack the ability to remember previous requests when processing the current request. This limitation prevents signature-based detection methods from detecting attacks that comprise multiple events if none of the events contains a clear indication of an attack [34].

Pattern matching in its most basic form is concerned with the identification of a fixed

sequence of bytes in a single packet. In addition to the tell-tale byte sequence, most IPS will also match various combinations of the source and destination IP address or network, source and destination port or service, and the protocol. It is also often possible to tune the signature further by specifying a start and end point for inspection within the packet, or a particular combination of TCP flags.

The more specific these parameters can be, the less inspection needs to be carried out against each packet on the wire. However, this approach can make it more difficult for systems to deal with protocols that do not live on well defined ports and, in particular, Trojans, and their associated traffic, which can usually be moved at will.

They are also prone to false positives, since legitimate traffic can often contain the relatively small set of criteria supposedly used to determine when an attack is taking place.

This method is usually limited to inspection of a single packet and, therefore, does not apply well to the stream-based nature of network traffic such as HTTP (Hypertext Transfer Protocol) sessions. This limitation gives rise to easily implemented evasion techniques.

Protocol analysis means analyzing the behavior of protocols to determine whether one host is communicating normally with another. For example, a host might send malformed IP packets, perhaps IP packets in which one or more values in the IP header is out of range. In still another, a malicious code may send malformed "chunks," parcels in which data are transferred from a browser to a web server to provide an orderly way for the web server to encode the input [39].

Stateful protocol analysis can identify unexpected sequences of commands, such as issuing the same command repeatedly or issuing a command without first issuing a command upon which it is dependent. Another state tracking feature of stateful protocol analysis is that for protocols that perform authentication, the *intrusion detection and prevention system* (IDPS) can keep track of the authenticator used for each session, and record the authenticator used for suspicious activity. This is helpful when investigating an incident.

The primary drawback to stateful protocol analysis methods is that they are very resource-intensive because of the complexity of the analysis and the overhead involved in performing state tracking for many simultaneous sessions. Another serious problem is that stateful protocol analysis methods cannot detect attacks that do not violate the characteristics of generally acceptable protocol behavior, such as performing many benign actions in a short period of time to cause a denial of service. Yet another problem is that the protocol model used by an IDPS might conflict with the way the protocol is implemented in particular versions of specific applications and operating systems, or how different client and server implementations of the protocol interact [34].

Rule-based intrusion detection is more of an eclectic approach than the other alternatives to signature-based intrusion detection. In this approach, logic conditions based on possible incident-related observations are defined. Observations could be signatures, irregularities in protocol behavior, unusual system events, changes in files and/or directories, and so on. Rule-based intrusion detection analyzes elements derived from these observations and then uses logic to identify attacks. The rule-based approach is potentially more powerful than signature-based intrusion detection because it relies on multiple variables/indicators—events based on signatures, protocol analysis, target detection indicators, and so on [39].

The main limitation of rule-based intrusion detection is the potential complexity associated with all the rules that are normally created. Only those with advanced technical skills and knowledge are likely to be able to understand the rules in the first place. It is generally difficult to create rules (which can often involve many steps of logic) and also to maintain rules (for example, weeding out obsolete rules). Processing the rules themselves can also cause massive CPU and memory utilization in the host that houses a rule-based intrusion detection system. Still, rule-based detection represents a significant advance over simple signature-based intrusion detection; it is likely to be used increasingly over time [39].

Neural networks are systems that perform pattern recognition on inputs they receive based on models of how neurons in mammals process

information. Although complicated and still somewhat mysterious, the neural networks approach can be applied to a wide range of pattern recognition problems, intrusion detection included. The beauty of neural networks in intrusion detection is that no signatures or even rules are needed. You simply start feeding input—data concerning network- or host-based events—to a neural network, and it does the rest. Neural networks are, therefore, well suited to picking up new patterns of attacks readily, although some learning time is required [39].

4.3. Which detection technique is the best?

Which detection method to choose is a difficult question. Adequate performance to handle the traffic to which the sensor will be exposed, accuracy of alerts, low incidence of false positives, and centralised management and reporting/ analysis tools are far more important than how the packets are processed.

As it has already mentioned, most protocol analysis systems are also reduced to performing some form of pattern-matching process following the protocol decode. Likewise, even the most basic pattern-matching systems perform some form of protocol analysis - even if it is only for a limited range of protocols. In truth, almost all Network IPS systems are already adopting a hybrid architecture.

Intrusion detection systems can be arranged as either centralized (for example, physically integrated within a firewall) or distributed. A distributed intrusion detection system (DIDS) consists of multiple IDSs over a large network, all of which communicate with each other. More sophisticated systems follow an agent structure principle where small autonomous modules are organized on a per-host basis across the protected network [40]. The role of the agent is to monitor and filter all activities within the protected area and — depending on the approach adopted — make an initial analysis and even undertake a response action. The cooperative agent network that reports to the central analysis server is one of the most important components of intrusion detection systems. DIDS can employ more sophisticated analysis tools, particularly connected with the detection of distributed attacks [41]. Another separate role of the agent is associated with its mobility and roaming across multiple physical locations. In addition, agents can be specifically devoted to detect certain known attack signatures. This is a decisive factor when introducing protection means associated with new types of attacks [42]. IDS agent-based solutions also use less sophisticated mechanisms for response policy updating [43].

5. Intrusion Prevention Systems (IPSs)

Intrusion prevention systems are proactive defence mechanisms designed to detect malicious packets within normal network traffic and stop intrusions dead, blocking the offending traffic automatically before it does any damage rather than simply raising an alert as, or after, the malicious payload has been delivered [26, 39].

Intrusion prevention devices are typically inline devices on the network that inspect packets and make decisions before forwarding them on to the destination. This type of device has the ability to defend against single packet attacks on the first attempt by blocking or modifying the attack packet. There are several methods of intrusion prevention technologies [44].

- System memory and process protection -- This type of intrusion prevention strategy resides at the system level. Memory protection consists of a mechanism to prevent a process from corrupting the memory of another process running on the same system. Process protection consists of a mechanism for monitoring process execution, with the ability to kill processes that are suspected of attacks.
- Session sniping -- This type of intrusion prevention strategy terminates a TCP session by sending a TCP RST packet to both ends of the connection. When an attempted attack is detected, the TCP RST is sent and the attempted exploit is flushed from the buffers and thereby prevented. Note that the TCP RST packets must have the correct sequence and acknowledge numbers to be effective.
- Gateway interaction devices -- This type of intrusion prevention strategy allows a detection device to dynamically interact with network gateway devices such as a router or firewall. When an attempted attack is detected, the detection device can direct the router or firewall to block the attack.

- Inline network devices -- This type of intrusion prevention strategy places a network device directly in the path of network communications that has the capability to modify and block attack packets as they traverse the device's interfaces. This acts much like a router or firewall combined with the signature-matching capabilities of an IDS. The detection and response happens in real time before the packet is passed on to the destination network.

Within the IPS market place, there are two main categories of product: *Host IPS* and *Network IPS*.

5.1. Host IPS (HIPS)

As with Host IDS systems, the Host IPS relies on agents installed directly on the system being protected. It binds closely with the operating system kernel and services, monitoring and intercepting system calls to the kernel or APIs in order to prevent attacks as well as log them.

It may also monitor data streams and the environment specific to a particular application (file locations and Registry settings for a Web server, for example) in order to protect that application from generic attacks for which no “signature” yet exists.

One potential disadvantage with this approach is that, given the necessarily tight integration with the host operating system, future OS upgrades could cause problems.

Since a Host IPS agent intercepts all requests to the system it protects, it has certain prerequisites - it must be very reliable, must not negatively impact performance, and must not block legitimate traffic. Any HIPS that does not meet these minimum requirements should never be installed in a host, no matter how effectively it blocks attacks [26].

5.2. Network IPS (NIPS)

The Network IPS combines features of a standard IDS, an IPS and a firewall, and is sometimes known as an *In-line IDS* or *Gateway IDS (GIDS)*. As with a typical firewall, the NIPS has at least two network interfaces, one designated as *internal* and one as *external*. A useful side effect of some NIPS products is that as a matter of course - they will provide “*packet scrubbing*” functionality to remove protocol inconsistencies resulting from varying interpretations of the TCP/IP specification (or intentional packet manipulation).

Thus any fragmented packets, out-of-order packets, or packets with overlapping IP fragments will be re-ordered and “cleaned up” before being passed to the destination host, and illegal packets can be dropped completely.

A true IPS device, however, is sitting in-line - all the packets have to pass through it. Therefore, as soon as a suspicious packet has been detected - and before it is passed to the internal interface and on to the protected network, it can be dropped [26].

Most NIPS products are basically IDS engines that operate in-line, and are thus dependent on protocol analysis or signature matching to recognise malicious content within individual packets (or across groups of packets). These can be classed as *Content-Based IPS* systems [26].

5.2.1. Rate-based IPS

There is, however, a second breed of Network IPS that ignores packet content almost completely, instead monitoring for anomalies in network traffic that might characterise a flood attempt, scan attempt, and so on. These devices are capable of monitoring traffic flows in order to determine what is considered “normal”, and applying various techniques to determine when that traffic deviates from normal. This is not always as simple as watching for high-volumes of a specific type of traffic in a short span of time, since they must also be capable of detecting “stealth “attacks, such as low-rate connection floods and slow port scan attempts.

Since these devices are concerned more with anomalies in traffic flow than packet contents, they are classed as *Rate-Based IPS* systems - and are also known as *Attack Mitigators*, as they are so effective against DOS and DDOS attacks [23].

6. Implementation Challenges

There are a number of challenges to the implementation of an IPS device that do not have to be faced when deploying passive-mode IDS products. These challenges all stem from the fact that the IPS device is designed to work in-line, presenting a potential choke point and single point of failure [26].

If a passive IDS fails, the worst that can happen is that some attempted attacks may go undetected. If an in-line [45] device fails, however, it can seriously impact the performance of the network. Perhaps latency rises to unacceptable values, or perhaps the device fails closed, in which case you have a self-inflicted Denial of Service [39] condition on your hands. On the bright side, there will be no attacks getting through [46].

Even if the IPS device does not fail altogether, it still has the potential to act as a bottleneck, increasing latency and reducing throughput as it struggles to keep up with upto a Gigabit or more of network traffic. Devices using off-the-shelf hardware will certainly struggle to keep up with a heavily loaded Gigabit network, especially if there is a substantial signature set loaded, and this could be a major concern for the network administrator [26, 47].

Network IPS device must perform much like a network switch. It must meet stringent network performance and reliability requirements as a prerequisite to deployment, since very few customers are willing to sacrifice network performance and reliability for security. A NIPS that slows down traffic, stops good traffic, or crashes the network is of little use [46].

Dropped packets are also an issue, since if even one of those dropped packets is one of those used in the exploit data stream it is possible that the entire exploit could be missed. Indeed, it is necessary to design the product to operate as much as a switch as an intrusion detection and prevention device [26].

Most notable is the recurring issue of false positives in today's intrusion detection systems. On some occasions, legitimate traffic will display some attack characteristics similar to those of malicious traffic. This could be anything from inadvertently matching signatures to uncharacteristic, high-volume traffic. Even a finely tuned IDS can present false positives when this occurs [46].

When intrusion prevention is involved, false positives can create a denial of service (DoS) condition for legitimate traffic [23, 39, 45, 47]. Additionally, attackers who discover or suspect the use of intrusion prevention methods can purposely create a DoS attack against legitimate networks and sources by sending attacks with spoofed source IP addresses. A simple mitigation to some DoS conditions is the use of an exclude list, also called a whitelist. It is important to include systems such as DNS, mail, routers, and firewalls in the whitelist [26]. Another potential problem with any Gigabit IPS/IDS product is the amount of alert data it is likely to generate. The ability to tune the signature set accurately is essential in order to keep the number of alerts to an absolute minimum.

Session sniping system identification is another concern when deploying IPSs. When systems terminate sessions with RST packets, an attacker may be able to discover not only that an IPS is involved, but also the type of underlying system. Readily available passive operating system identification tools, such as p0f, analyze packets to determine the underlying operating system. This type of information allows an attacker to potentially evade the IPS or direct an attack at the IPS [46, 48, 49].

Of course, one point in favour of IPS when compared with IDS is that because it is designed to prevent the attacks rather than just detect and log them, the burden of examining and investigating the alerts - and especially the problem of rectifying damage done by successful exploits - is reduced considerably [26, 45, 47].

When deploying an IPS, one should carefully monitor and tune his/her systems and be aware of the risks involved. One should also have an in-depth understanding of his/her network, its traffic, and both its normal and abnormal characteristics. It is always recommended to run IPS and active response technologies in test mode for a while to thoroughly understand their behaviour [26, 47].

7. Requirements for Effective Prevention

IPS must exhibit the following characteristics and features to avoid implementation problems [47].

In-line operation: Only by operating in-line can an IPS device perform true protection, discarding all suspect packets immediately and blocking the remainder of that flow [45].

Reliability and availability: Should an in-line device fail, it has the potential to close a vital network path and thus, once again, cause a DoS condition. An

extremely low failure rate is thus very important in order to maximise up-time, and if the worst should happen, the device should provide the option to fail open or support fail-over to another sensor operating in a fail-over group [34].

Resilience: The very minimum that an IPS device should offer in the way of High Availability is to fail open in the case of system failure or power loss. Active-Active stateful fail-over with cooperating in-line sensors in a fail-over group will ensure that the IPS device does not become a single point of failure in a critical network deployment [26].

Low latency: When a device is placed in-line, it is essential that its impact on overall network performance is minimal. Packets should be processed quickly enough such that the overall latency of the device is as close as possible to that offered by a layer 2/3 device such as a switch [46].

High performance: Packet processing rates must be at the rated speed of the device under real-life traffic conditions, and the device must meet the stated performance with all signatures enabled. Ideally, the detection engine should be designed in such a way that the "signatures" loaded does not affect the overall performance of the device [34, 45, 49].

Unquestionable detection accuracy: It is imperative that the quality of the signatures is beyond question, since false positives can lead to a DoS condition [39]. The user MUST be able to trust that the IDS is blocking only the user selected malicious traffic. New signatures should be made available on a regular basis, and applying them should be quick [34].

Fine-grained granularity and control: Fine grained granularity is required in terms of deciding exactly which malicious traffic is blocked. The ability to specify traffic to be blocked by attack, by policy, or right down to individual host level is vital. In addition, it may be necessary to only alert on suspicious traffic for further analysis and investigation [26].

Advanced alert handling and forensic analysis capabilities: Once the alerts have been raised at the sensor and passed to a central console, someone has to examine them, correlate them where necessary, investigate them, and eventually decide on an action. The capabilities offered by the console in terms of alert viewing and reporting are keys in determining the effectiveness of the IPS product [46].

Understanding the different kinds of protection provided by Network security technologies is helpful in deciding what systems are required for each network. These technologies can be broadly classified into four categories [50]:

- Packet level protection, such as routers' Access Control Lists (ACL) or stateless firewalls
- Session level protection, such as stateful inspection firewalls.
- Application level protection, such as proxy firewalls and intrusion prevention systems
- File level protection, such as gateway antivirus systems

Table 2 compares the four categories of network security technologies. Evaluation of each category by coverage of protocols/applications, level of protection, and relative performance enables organizations to choose the appropriate network security technologies to protect their networks.

Table 2. Comparison of network security technology categories.

	Packet Level Protection	Session Level Protection	Application Level Protection	File Level Protection
Examples	Packet filtering	Stateful inspection firewalls	IPS and proxy firewalls	Gateway antivirus
Mechanism	Examine Packet Header	Examine Packet Header and control fields	Examine application fields	Examine files inside application traffic
Protocol and Application Coverage	N. A. packet level	Large	Medium	Small
Protection Provided	Client-to-server and server-to-client	Client-to-server and server-to-client	Mainly Client-to-server	Mainly server-to-client
Relative Performance	High	High	Medium	Low

Packet level protection: Packet level protection, also known as packet filtering, is one of the most widely used means of controlling access to a

network. The concept is simple: determine whether a packet is allowed by comparing some basic pieces of information in the packet headers. Cisco IOS Access Control List (ACL) is one of the most used packet filters. IPChains is also a popular packet filter application, which comes bundled with many versions of Linux.

Two-way communication presents a challenge for network security based on packet filtering. If one blocks all incoming traffic, one prevents responses to outgoing traffic from coming in disrupting communication. Consequently, one has to open two holes, one for outgoing traffic and one for incoming traffic, without enforcing any association of the incoming traffic with existing outgoing connections in the network. Packet filtering thus can allow in crafted malicious packets that appear to be part of existing sessions, causing damage to protected resources.

Session level protection: Session level protection technologies control the flow of traffic between two or more networks by tracking the state of sessions and dropping packets that are not part of a session allowed by a predefined security policy. Firewalls that implement session-level protection keep state information for each network session and make allow/deny decisions based on a session state table. The most common systems for session level protection are stateful inspection firewalls.

Note that session level protection technologies are “session based,” meaning that firewalls go beyond individual TCP connections to involve many such connections. Session-level firewalls support dynamic protocols by identifying port change instructions in client-server communication and comparing future sessions against these negotiated ports. For instance, to track FTP sessions, the firewall inspects the control connection, used for issuing commands and negotiating dynamic ports, and then allows in various data connections for transferring files.

Because session level protection provides all the benefits of packet level protection without the limitations, it renders packet level protection unnecessary for most networks.

Application level protectiou: Application level protection technologies monitor network traffic and dynamically analyze it for signs of attacks and intrusions. Within the network security infrastructure, two common technologies for application level protection are proxy firewalls and Intrusion Prevention Systems (IPS).

Proxy firewalls and IPS examine control and data fields within the application flow to verify that the actions are allowed by the security policy and do not represent a threat to end systems. By understanding application-level commands and primitives, they can identify content out of the norm and content that represents a known attack or exploit. Proxy firewalls and IPS perform IP de-fragmentation and TCP stream reassembly as well as eliminating ambiguity within traffic, which can be used by malicious users trying to conceal their actions.

Proxy firewalls usually support the common Internet applications, including HTTP, FTP, telnet, rlogin, email and news. Yet, a new proxy must be developed for each new application or protocol to pass through the firewall, and custom software and user procedures are required for each application.

IPS generally support a wider range of protocols and applications, including those required to protect the network against attacks from the Internet. New applications can be allowed through an IPS without requiring changes to the user workstations. In this way, IPS are more transparent to the network than proxy firewalls.

Proxy firewalls and IPS can detect certain viruses or Trojans by looking at application service fields. For instance, IPS can look at the subject field, attachment name, or attachment type within email traffic to detect characteristics of known viruses. However, application level protection does not do a detailed analysis at the file level, which is also required to detect the large number of viruses in existence.

File level protection: File level protection provides the ability to extract files within traffic and inspect them to detect malware, including viruses, worms or Trojans. A common technology for file level protection in a network is gateway antivirus.

Gateway antivirus systems scan files that are embedded in network traffic, including files in HTTP traffic (web downloads) and files in email traffic (attachments). If an infected file is detected, a gateway antivirus system removes it from the

traffic, so it does not affect other users. To scan files within network traffic, gateway antivirus must understand a broad range of file encoding protocols and file compression algorithms.

Figure 1 illustrates the inspection functions that take place as the packets are analyzed bystateful firewall for session level protection, Intrusion Prevention Systems (IPS) for application level protection and gateway antivirus for file level protection.

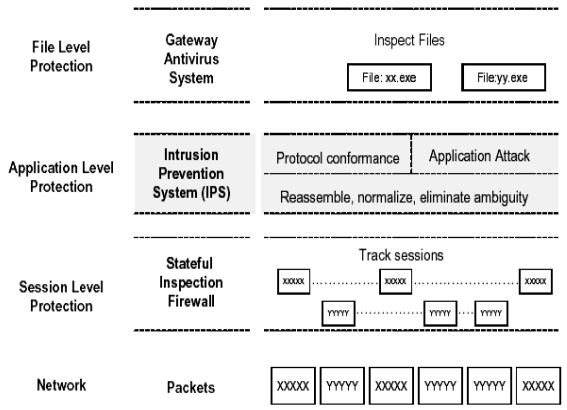

Figure 1. Network packets are inspected by session level protection, application levelprotection and file level protection technologies in order to defend the network from viruses, worms and other network attacks [50].

8. Available Open Source Intrusion Prevention Sytems (IPS)

Achieving a security goal in a networked system requires the cooperation of a variety of devices, each device potentially requiring a different configuration. Many information security problems may be solved with appropriate mangement of these devices and their interactions, giving a systematic way to handle the complexity of real situations [31, 51].

The term "intrusion prevention" has become prevalent in marketing materials and sales presentations as commercial vendors develop an abundance of products (both good and bad) under this umbrella term. While commercial intrusion prevention products are often technologically diverse and contain a rich feature set, they also often come with a hefty price tag [52].

In this section the description of some free, open source alternatives for implementing intrusion prevention systems, has been presented. It looks at intrusion prevention from a defense in depth approach, including not only network methods but also system and application methods [52].

8.1. Snort products

Snort Flexible Response Plug-in: Snort can perform session sniping through its flexible response plug-in. This plug-in adds the response and react keywords to rule creation. When a rule is triggered, the appropriate action is taken based on the keywords. If one is using Snort in stealth mode, he will need an additional interface to send the responses. Snort flexible response is a quick and simple solution that uses sessions sniping. Although it is not an overall enterprise solution, it is a lightweight method for use in simple environments [53], [54].

SnortSam: SnortSam is an active response plug-in for Snort that performs gateway interaction with various router and firewall devices. SnortSam acts at the network layer by instructing the gateway to alter or block traffic for specified amounts of time based on IP address. SnortSam consists of two parts: an intelligent agent that runs on the gateway device and accepts commands, and an output plug-in for Snort that sends commands based on triggered rules. The communication between the output plug-in and agent is secured by an encrypted TCP session.

The SnortSam agent provides several features including:

- The ability to specify a whitelist of IP addresses that will never be blocked.
- The ability to provide per-rule blocking and time interval.
- The ability to prevent repetitive blocking of the same IP address.
- Twofish encrypted sessions between Snort and SnortSam.
- The ability to multi-thread for faster processing and simultaneous blocking on multiple devices.
- The ability to log events and send email notification.

- The ability to scale to larger distributed networks using client/server architecture.

SnortSam is platform independent and is actively developed and maintained. It is a stable, recommended solution for active response [55, 56].

Fwsnort: Fwsnort functions as a transport layer inline IPS, because it is deployed directly within the iptables firewall. It works by translating Snort signatures into their equivalent iptables rulesets; hence, it will only stop attacks for which there are Snort signatures. Not all Snort rules are easily translated, but fwsnort does a good job at translating about 70% of them. Fwsnort also accepts Snort rules by the SID value, so one can add specific rules to iptables ruleset. Iptables can then either log or block the attacks.

Fwsnort is not bulletproof. Since it uses string matching, it can easily be evaded with simple evasion techniques such as fragmentation, URL encoding, and session splicing. It is still a good tool to use, however [57, 58].

Snort Inline: Snort Inline is a true IPS that is deployed between network segments with the capability to alter or drop packets in real time as they flow through the system. It runs on a Linux system and uses iptables packet queuing to collect and make decisions about packets as they traverse the system's interfaces. It can also be used in stealth mode as a bridge between network segments, so it will not be detected as a hop in the network. One of the best features of Snort Inline is its ability to mitigate attacks by altering application layer data as the packet traverses the system [59].

Hogwash: Hogwash is a Gateway IDS and packet scrubber based on the Snort IDS that can detect attacks and filter them out. Hogwash works in three different modes: IDS, inline packet scrubber, and honeypot. In IDS mode, it can detect attack traffic and send TCP resets to end sessions. In inline packet scrubber mode, it can actively filter exploits from network traffic. In this mode, it can send resets, drop the packet, or modify the packet as it traverses the system. The honeypot mode is still experimental; however, the concept is that Hogwash can route attackers to one of several honeypots that are behind the Hogwash system while each of these honeypots can have the same IP and MAC address. Hogwash also has the ability to perform multi-packet signature matching and port-scan detection [60].

Hogwash runs on a Linux system that can be transparently connected to the network. It has the capability of managing up to 16 different interfaces, thus protecting several network segments with a single system. Hogwash handles the packet forwarding for each network segment, so remember to disable the kernel IP forwarding [53].

Hogwash also uses a Stackless Control Protocol to remotely control the Hogwash system. The transactions are secured with either Twofish or AES encryption. Remote actions that can be performed include pinging, gathering statistics, and transferring files [61].

LAk: LAk is an open source intrusion prevention system project. It houses a collection of programs, scripts, and whitepapers on implementing and operating an open source IPS. The goal of the project is for users to be able to easily install and run an IPS in a short amount of time. It also aims to combat the current media hype generated by commercial vendors about IPS.

LAk currently consists of a "Getting Started" guide, a list of prerequisites, and a whitepaper with installation and configuration instructions. The LAk IPS is based on iptables and Snort Inline. It assumes that iptables is installed with IP queuing enabled. LAk also assumes that Snort Inline is installed (although it includes a precompiled binary) and that the latest set of Snort signatures has been downloaded. LAk IPS is a handy tool that automates the process of setting up iptables and Snort Inline [62].

8.2. Miscellaneous products

Modsecurity: Modsecurity is a module that acts as an intrusion detection and prevention engine for Web applications. It increases Web application security by protecting applications from both known and unknown attacks [63]. Modsecurity sits inline between the Web client and server to detect attacks. If it identifies a potential attack, it can reject the request or perform any number of built-in active responses. Modsecurity integrates with the Web server and provides the following features:

- Request filtering -- Incoming Web requests are analyzed inline before being passed to the Web server or other modules.
- Anti-evasion techniques -- Paths and parameters are normalized before analysis takes place.
- Understanding of the HTTP protocol -- The engine has a deep understanding of the HTTP protocol, allowing it to perform very specific and granulated filtering.
- POST payload analysis -- The engine will intercept and analyze POST methods contents.
- Audit logging -- All requests are logged in full detail for later analysis.
- HTTPS filtering -- The engine can operate with encrypted sessions because it has access to the request data after decryption occurs.
- Built-in checks -- Other special built-in checks include URL-encoding validation, Unicode-encoding validation, and byte-range verification to detect and reject shellcode.
- Rule support -- Modsecurity also supports any number of custom rules for attack detection and prevention. These rules are formed using regular expressions. Negated rules are also supported.

Modsecurity rules can analyze headers, cookies, environment variables, server variables, page variables, POST payload, and script output. Modsecurity rules can also intercept files that are being uploaded to the Web server, store uploaded files on a disk, and execute an external binary to approve or reject files [64].

The Modsecurity audit log feature captures data and logs it in text format. This is easy and convenient to work with; however, when analyzing large quantities of data, a better method is needed [65].

Modsecurity is a great tool to use. It is best coupled with an IDS that is monitoring at the network level. Modsecurity fills the gap between the Web server and the application, providing a great open source solution for Web application security [66].

LIDS: The Linux Intrusion Detection System (LIDS) is an intrusion detection and prevention system that resides within the Linux kernel. It is a security enhancement to the Linux kernel consisting of a kernel patch and some admin tools. LIDS implements mandatory access control (MAC), file protection, and process protection on the Linux system by restricting file access, network operations, raw device access, memory use and access, and I/O access. LIDS provides the administrator the ability to define and finely tune access controls. LIDS also contains a port-scan detector [67].

LIDS provides protection, detection, and response within the kernel of the Linux system. It provides protection in the following ways:

- Full file system protection of files and directories from unauthorized users and programs including protection from root.
- Protection of important processes from being terminated.
- Protection of RAW I/O operations from unauthorized programs including hard disk and master boot record (MBR) protection.

LIDS provides detection via the port-scan detector and by monitoring any unauthorized system activity. The port-scan detector functionality is built into the kernel. It can easily detect tools like Nmap and Nessus. The port-scan detector works with raw socket disabled, making it more secure than standard sniffers.

LIDS can provide response in the following ways:

- When a rule violation occurs, LIDS logs a detailed message about the violation to the system kernel log file, which is also protected by LIDS. LIDS logging has an anti-flooding capability.
- Sending log messages via email.
- Automatically terminating a user's session that is in violation of the rules.

The LIDS functionality extends the existing Linux kernel "immutable" attribute by allowing administrator to grant or deny specific rights on a more granular basis with ACLs. He can also use the capabilities, to remove the Linux "immutable"

feature all together, thus using LIDS for all file system protection [68].

The biggest advantage of using LIDS to protect system is that it can prevent the root user from tampering with important system controls. This is significant in the event of a system compromise. Furthermore, its most important features include increased file system protection, protection against direct port access or direct memory access, protection against raw disk access, and protection of log files. LIDS can prevent certain system actions, such as installing a packet sniffer or changing firewall rules [52].

Grsecurity and PaX: Grsecurity is a Linux security project that uses a multi-layered detection, prevention, and containment model. It uses a Role-Based Access Control (RBAC) system that can generate least-privilege policies for the entire system. It also provides the following additional features:

- Change root (chroot) hardening
- Full-featured fine-grained auditing
- Address space modification protection provided by the PaX project
- Additional randomness in the TCP/IP stack and process IDs
- All alerts and audits support a feature that logs the IP address of the attacker with the log
- Restricted viewing of processes
- Integrated local attack response on all alerts

PaX is a separate project that is included in grsecurity as part of its security strategy [52]. The PaX project researches various defenses against the exploitation of software bugs that give the attacker arbitrary read/write access to the target's address space. PaX doesn't focus on finding and fixing the bugs, but rather on prevention and containment of exploit techniques. Exploit techniques can affect the target at three different levels:

- Introduce or execute arbitrary code.
- Execute existing code out of original program order.
- Execute existing code in original program order with arbitrary data.

PaX is a patch for the Linux kernel that implements least-privilege protections for memory. It flags data memory as non-executable and program memory as non-writable, and randomly arranges the program memory. Prevention is implemented through PaX and hardening certain sections of the kernel. One must monitor log files to look for intrusion attempts reported by PaX. An execution attempt of a service could signify an attack or compromise. For effective security, these logs should be correlated with network logs.

PortSentry and PSAD: PortSentry was developed to detect and respond to port scans on a host. Because port scans are often the first step of the attack process, PortSentry monitors the TCP and UDP ports on a system and responds when a scan is identified. It has the ability to detect various types of scans including stealth scans [69].

PortSentry provides three active response choices:

- Insert a null route into the hosts routing table. This will re-route the scan from the attacker to a non-existent IP address. The disadvantage to this type of response is that it increases the size of the routing table on the host, which uses more memory. If the attacker is using random, spoofed source addresses as part of the attack, this could lead to a DoS condition on the host.
- Insert a firewall rule to block traffic from the scanning IP address. PortSentry supports ipfw, ipfilter, ipfwadm, ipchains, and iptables. When it detects a scan, PortSentry can add the appropriate rule to the firewall to block the IP address of the scanning host. Once again, this can also be used to create a DoS condition for the host or network. An attacker could spoof the source address to prevent legitimate connections.
- Add a TCP wrapper rule for the attacking IP address to the /etc/hosts.deny file. This will prevent the attacker from connecting to the target host's services. Although this protection mechanism isn't as strong, it does alleviate the potential DoS conditions from the other options.

The Port Scan Attack Detector (PSAD) runs on Linux and analyzes iptables firewall logs to detect port scans and other suspicious traffic. It is

designed to function as a network IDS that uses iptables firewall log data to block and log packets. It consists of three system daemons written in Perl and C.

PSAD features alert messages that include the source, destination, scanned port range, start and end times, TCP flags and corresponding nmap options, as well as email alerting, DShield reporting, and automatic blocking of the attacking IP address via iptables. It also utilizes Snort signatures to detect backdoor scans, DDoS tools, and advanced port scans such as stealth, FIN, and Xmas [70].

PSAD also has the ability to passively fingerprint the remote operating system of the attacker by using TTL, IP id, TOS, and TCP window sizes. Combining PSAD with FWSnort and the iptables string match extension allows you to detect about 70% of all Snort signatures including application-level attacks [71].

PSAD addresses some of the limitations of PortSentry including:

- Better firewall integration. PortSentry listens on ports to detect scans, causes more administration on the firewall.
- Implemented scoring mechanism for scans to prioritize actions and responses.
- Implemented passive fingerprinting.
- ICMP probe detection.
- Backdoor and DDoS probe detection.
- Integrated Whois lookups.
- Integrated email alerts.

PortSentry is very portable across many different Unix systems [72, 73].

OSSIM: Ossim stands for *Open Source Security Information Management*. Its goal is to provide a comprehensive compilation of tools which, when working together, grant a network/security administrator with detailed view over each and every aspect of his networks/hosts/physical accessdevices/server/etc [74].

Verification, Integration, Risk Assessment may be *OSSIM*'s most valuable contribution at this time. Using its correlation engine, OSSIM screens out a large percentage of false positives, enables to perform a range of tasks from auditing, pattern matching and anomaly detection to forensic analysis in one single platform and offers high level state indicators that allow guiding inspection and measuring the security situation of network. *OSSIM* is a distribution rather than a product. The *OSSIM* aims at intercommunication, making these processes integrate with each other [31].

OSSIM integrates a number of powerful open source security tools in a single distribution. These include: Snort, Nessus, Ntop, Snortcenter, Acid, Riskmeter, Spade, RRD, Nmap, P0f, Arpwatch, etc. These tools are linked together in OSSIM's console giving the user a single, integrated navigation environment. The ability to act as an IPS (Intrusion Prevention System) based on correlated information from virtually any source result in a useful addition to any security professional [74].

Besides these open source products a number of commercially available IPS products like Lanifex's Event Horizon (*EH*) [25, 75], McAfee's Protection-in-Depth [76], NetScreen-IDP [77], Cisco IPS [78], Tipping Point IPS [79] are available.

9. Conclusion

Although the security solutions, mentioned and discussed above, cover a wide domain of difficulties currently addressed and focused by researchers and experts in this area, however, preventive measures from external attacks is still a hot issue. Yet there seems to be no "silver bullet" to the problem. This survey examines the possible solutions to this problem, provides taxonomies to classify those solutions and analyzes the feasibility of those approaches. Applications need to provide strict enforcement of access control policies, assurances of secure data handling, consistent auditing and alarming, secure administration, and pervasive denial of service protection. Taking these measures will help protect against unauthorized access, data loss, and resource theft. Furthermore, intruders and suspicious application access trends can be tracked and reported. Organizations must deploy multiple security technologies to protect networks against viruses, worms and other sophisticated attacks.

References

[1] Technical White Paper, (2002). ISecurity in Converged Networks. Avaya labs and services www.avaya.com

[2] Richard Bejtlich (2004). The Tao of Network Security Monitoring: Beyond Intrusion Detection. Publisher: Addison-Wesley.

[3] Network security resources and reporting problems (2006), http://pangea.stanford.edu/ computerinfo /resources /network/security/

[4] Buyer's Guide for Intrusion Prevention Systems (IPS). Juniper Networks, Inc. 2004, 1194 North Mathilda Avenue Sunnyvale, CA 94089 USA

[5] P. Barford, V. Yegneswaran and J. Ullrich, "Internet intrusions: Global characteristics and prevalence," in Proceedings of the 2003 ACM SIGMETRICS, (2003).

[6] C. Zou, D. Towsley and W. Gong, "The performance of internet worm scanning strategies," (2003).

[7] P. Kazienko and P. Dorosz, (2004). Intrusion Detection Systems (IDS) Part I. http://www. windowsecurity.com/articles/Intrusion_Detection_Systems_IDS_Part_I

[8] J. Wilander and M. Kamkar. A comparison of publicly available tools for static intrusion prevention. In Proc. of 7th Nordic Workshop on Secure IT Systems, 2002.

[9] J. Wilander and M. Kamkar. A comparison of publicly available tools for dynamic buffer overflow prevention. In Proc. of 10th Network and Distributed System Security Symposium, (2003).

[10] L. A. Grenier. Practical code auditing. http://www.daemonkitty.net/lurene, (2002).

[11] R. Jones and P. Kelly. Bounds checking for C. http://www-ala.doc.ic.ac.uk/~phjk/ Bounds Checking.html, July (1995).

[12] T. M. Austin, S. E. Breach, and G. S. Sohi. Efficient detection of all pointer and array access errors. ACM SIGPLAN Notices, **29**, No. 6, 1994.

[13] R. Hastings and B. Joyce. Purify: Fast detection of memory leaks and access errors. In Proceedings of the Winter USENIX Conference (1992).

[14] O. Ruwase and M. S. Lam. A practical dynamic buffer overflow detector. In Proceedings of the 11th Network and Distributed System Security Symposium, (2004).

[15] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, Q. Zhang, and H. Hinton. Stack-Guard: Automatic detection and prevention of buffer overflow attacks. In Proceedings of the 7th USENIX Security Conference, January (1998).

[16] T.-C. Chiueh and F.-H. Hsu. RAD: A compile-time solution to buffer overflow attacks. In Proc. of 21st Intl. Conf.on Distributed Computing Systems, (2001).

[17] Vendicator. StackShield GCC compiler patch. http://www.angelfire.com/sk/stackshield, (2001).

[18] H. Etoh. GCC extensions for protecting applications from stack-smashing attacks. http://www.trl.ibm. com/projects/security/ssp, June (2000).

[19] M. Frantzen and M. Shuey. StackGhost: Hardware facilitated stack protection. In Proceedings of the 10th USENIX Security Sumposium, August (2001).

[20] C. Cowan, M. Barringer, S. Beattie, G. Kroah-Hartman, M. Frantzen, and J. Lokier. FormatGuard: Automatic protection from printf format string vulnerabilities. In Pro. of 10th USENIX Security Symposium, (2001).

[21] S. Savage, D. Wetherall, A. R. Karlin and T. Anderson, "Practical network support for IP traceback," in SIGCOMM, (2000), pp. 295–306.

[22] S. Bellovin, "Icmp traceback messages," http://www.research.att.com/smb/papers/draft-bellovin-itrace-00.txt, (2000).

[23] S. Lin and Tzi-cker Chiueh, (2006) "A Survey on Solutions to Distributed Denial of Service Attacks", RPE report, Department of Computer Science, Stony Brook University, Stony Brook, US.

[24] C. Brenton and C. Hunt, Mastering Network Security. Second edition Sybex Inc., UK (2003).

[25] Technical White Paper, (2003). Event Horizon™: Lanifex Intrusion Detection Solution., ver. 1.5, © 2003 CSO Lanifex GmbH.

[26] Technical White Paper, (2004). Intrusion Prevention Systems. NSS Labs, Inc.733 Lee St.Des Plaines, US.

[27] P.J. Barry, 2002. Intrusion Detection – Evolution beyond Anomalous Behavior and Pattern Matching. Security Essentials Version 1.4.

[28] R.A. Kemmerer and G. Vigna, Computer, **35**, No. 4 (2002) 27.

[29] T. Wang, B. Suckow and D. Brown, 2001. A Survey of Intrusion Detection Systems**.** Department of Computer Science, University of California, San Diego San Diego, CA 92093, USA.

[30] J.P. Anderson, (1980). Computer Security Threat Monitoring and Surveillance. James P. Anderson Co., Fort Washington. Micki Krause, Harold F. Tipton, (2006). Handbook of Information Security Management. Publisher: CRC Press LLC. ISBN: 0849399475.

[31] M. Anwar, M.F. Zafar, Z. Ahmed, (2007). A Proposed Preventive Information Security System. International Conference on Electrical Engineering (ICEE 2007) , UET Lahore, Pakistan.

[32] R. Bace and P. Mell, (2001). Special Publication on Intrusion Detection Systems. Tech. Report SP 800-31, National Institute of Standards and Technology, Gaithersburg, Md.

[33] G. Mansfield, K. Ohta, Y. Takei, N. Kato, Y. Nemoto, Towards trapping wily intruders in the large, Computer Networks **34** (2000), pp 659-670.

[34] K. Scarfone and Peter Me (2007), Guide to Intrusion Detection and Prevention Systems (IDPS). Recommendations of the National Institute of Standards and Technology Computer Security Division, Information Technology Laboratory, Gaithersburg, MD 20899-8930, US.

[35] D.E. Denning, IEEE Trans. Software Eng., **13**, No. 2 (1987) 222.

[36] A.K. Ghosh, J. Wanken, and F. Charron, Detecting Anomalous and Unknown Intrusions Against Programs. Proc. Annual Computer Security Application Conference (ACSAC'98), IEEE CS Press, Los Alamitos, Calif (1998) 259–267.

[37] K. Ilgun, R.A. Kemmerer and P.A. Porras, IEEE Trans. Software Eng. **21**, No.3 (1995) 181.

[38] U. Lindqvist and P.A. Porras,. Detecting Computer and Network Misuse with the Production-Based Expert System Toolset. IEEE Symp. Security and Privacy, IEEE CS Press, Los Alamitos, Calif. (1999) 146–161.

[39] C. Endorf, E. Schultz and J. Mellander; (2004). Intrusion Detection & Prevention. Published by McGraw-Hill.

[40] C. Krügel, T. Toth, Applying Mobile Agent Technology to Intrusion Detection, ICSE Workshop on Software Engineering and Mobility, Toronto 2001, http://www.elet. polimi.it/ Users/DEI/Sections/Compeng/Gian Pietro.Picco/ICSE01mobility/papers/ krugel.pdf.

[41] C. Krügel, T. Toth. Distributed Pattern Detection for Intrusion Detection, Conf. Proc. of the Network and Distributed System Security Symposium NDSS, 2002, http://www.isoc.org/isoc/conferences/ndss/02 /proceedings/papers/kruege.ps.

[42] J.S. Balasubramaniyan, J.O. Garcia-Fernandez, D. Isaco, E. Spafford, D. Zamboni, An Architecture for Intrusion Detection using Autonomous Agents, 14th IEEE Computer Security Applications Conference ACSAC '98, December 1998, pages 13-24, http://www.cs.umbc.edu/cadip/ docs/tr9805.ps.

[43] D.J. Ragsdale, C.A. Carver, J.W. Humphries, U.W. Pooh, Adaptation techniques for intrusion detection and intrusion response systems, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2000, pages 2344-2349, http:// www.itoc.usma.edu/ragsdale/pubs/ adapt.pdf

[44] A. Orebaugh and E. Cole., Sys. Admin. Magazine. **14**, No 3. (2005) 44.

[45] N. Desai (2003). Intrusion Prevention Systems: the Next Step in the Evolution of IDS. Retrieved from www.securityfocus. com/infocus.

[46] F. Gong (2003), White paper on Intrusion Prevention: Myths, Challenges, and Requirements, McAfee Network Protection, www.mcafee.com.

[47] Marc Willebeek-LeMair (2005) Anatomy of an Intrusion Prevention System. Tipping Point, www.tippingpoint.com.

[48] B. Toxen (2003). Real World Linux® Security: Intrusion Prevention, Detection, and Recovery, Second Edition Publisher: Prentice Hall, www.securityfocus. com/infocus.

[49] S. Suehring and R. Ziegler (2005). Linux Firewalls, 3rd Edition Published by Novell Press.

[50] http://www.juniper.net/solutions/literature/white_papers/200063.pdf

[51] J.D. Guttman, A. L. Herzog, Int. J. Inf. Secur. **4** (2005) 29–48, Springer-Verlag.

[52] M. Smith, S. Dukin and K. Tan (2006). A Design for Building an IPS Using Open Source Products. System Admin Magazine, The journal for Unix and Linux system administrators.

[53] C. Brian. (2004). Snort 2.1 Intrusion Detection, Second Edition. Syngress Publishing.

[54] C. Kerry and C. Gerg. 2004. Managing Security with Snort and IDS Tools. O'Reilly & Associates.

[55] Snorthttp://www.snort.org/docs/snort_manual /node21.

[56] SnortSam -- http://www.snortsam.net/.

[57] fwsnort -- http://www.cipherdyne.org/fwsnort/.

[58] Snortconfig:--http://www.shmoo.com/~bmc/ software /snortconfig

[59] Snort Inline -- http://snort-inline. sourceforge. net/

[60] An Introduction to Gateway Intrusion Detection Systems: Hogwash GIDS http://www.cansecwest.com/core02/hogwash .ppt.

[61] Hogwash -- http://hogwash.sourceforge.net/.

[62] LAk-IPS -- http://lak-ips.sourceforge.net.

[63] Better Living Through Mod Security http:// www.hackinthebox.org/article.php?sid=1286 7

[64] Introducing mod_security -- http://www. onlamp.com/.

[65] pub/a/apache/2003/11/26/mod_security.html

[66] Web Security Appliance with Apache and mod_security--http://www.securityfocus.com/ infocus/1739.

[67] ModSecurity -- http://www.modsecurity.org/.

[68] LIDS -- http://www.lids.org.

[69] Overview of LIDS, Part Two http://www. securityfocus.com/ infocus/ 1502.

[70] Sentry Tools--http://sourceforge.net/projects/ sentrytools.

[71] Grsecurity -- http://www.grsecurity.net.

[72] PSAD -- http://www.cipherdyne.com/psad/.

[73] PortSentry for Attack Detection: Part 1 http:// www.securityfocus.com /infocus/1580

[74] PortSentry for Attack Detection: Part Two http:// www.securityfocus.com/infocus/1586.

[75] http:// www.ossim.net.

[76] http:// www.lanifex.com/.

[77] McAfee®Protection-in-Depth,www.mcafee com/.

[78] NetScreen-IDP Juniper Networks www. juniper.net.

[79] Cisco IPS --www.cisco.com/index.html.

[80] Tipping Point IPS -- www.tippingpoint.com.