A Review on Intrusion Detection Systems

Vaishali Anwekar
Khushboo Pawar

Abstract – The rapid growth of Internet malicious activities has become a major concern to network forensics and security community. With the increasing use of IT technologies for managing information there is a need for stronger intrusion detection mechanisms. Critical mission systems and applications require mechanisms able to detect any unauthorized activities. An Intrusion Detection System (IDS) acts as a necessary element for monitoring traffic packets on computer networks, performs analysis to suspicious traffic and makes vital decisions. IDSs allow cybercrime forensic specialists to gather useful evidence whenever needed. This paper presents a review of Intrusion Detection System (IDS).

Keywords – Intrusion Detection System (IDS), HTTP

I. INTRODUCTION

Recent activities highlighted the need for fast reactionary capabilities in network security. According to an analysis from [1], the Code Red worm infected over 359,000 hosts within less than 14 hours, as illustrated by Figure 1. Later theoretical analysis place the time for a complete spread of a network worm as low as 15 minutes [2], or 30 seconds [3].

Current conventional security techniques seem incapable of dealing with these threats (leaving the task to security personnel). In many respects, this mirrors the epidemic spread of diseases and electronic viruses. Intrusion detection systems have the potential to mitigate or prevent such attacks, if updated signatures or novel attack recognition and response capabilities are in place.

Intrusion Detection Systems

Intrusion detection systems have evolved from monolithic batch-oriented systems to distributed real-time networks of components as shown in Figure 2.

Figure 1: Code Red worm spread

In current systems, a number of common building functional blocks can be distinguished:

- Sensor, Probe: These modules form the most primitive data-gathering components of IDS. Implemented in a highly system-specific fashion, they track network traffic, log files or system behaviour - translating raw data into events useable by the IDS monitors.
- Monitor: Monitor components, the main processing segment of IDS, receive events from sensors. These events are then correlated against the IDS behaviour models, potentially producing model updates and alerts. Alerts, events in themselves, indicate occurrences significant to the security of a system, and may be forwarded to higher-level monitors, or to resolver units.
- Resolver: Resolver components receive suspicion reports from monitors (in the form of one or more alerts), and determine the appropriate response - logging, changing the behaviour of lower level components,
reconfiguring other security mechanisms (e.g. adding firewall rules), and notifying operators.

- Controller: Facilitating component configuration and coordination, controller components are most significant in distributed ID systems, where manually upgrading, configuring and starting a network-wide series of components would be infeasible. In addition, controller units offer a single point of administration and interrogation for IDS, and may act in a supervisory capacity, restarting failed components.

II. ANOMALY MODELLING TECHNIQUES

**Statistical Models**

In Denning’s ground laying paper on IDS [5], she described a number of statistical characterizations of events and event counters. These, and more refined techniques, have been implemented in anomaly detection systems [4]:

- Threshold measures: Referred to as the operational model in [5], this scheme applies set or heuristic limits to event occurrences or event counts over an interval. A common example is logging and disabling user accounts after a set number of failed login attempts.

- Mean and standard deviation: By comparing event measures to a profile mean and standard deviation, a confidence interval for abnormality can be established. The profile values are fixed or based on weighted historical data.

- Multivariate model: Calculating the correlation between multiple event measures, relative to the profile expectations.

- Markov process model: This model regards event types to be state variables in a state transition matrix, where an event is considered anomalous if its probability, given the previous state and associated value in the state transition matrix, is too low.

- Clustering analysis: This nonparametric method relies on representing event streams in a vector representation, examples of which are then grouped into classes of behaviours using some clustering algorithm (e.g. k-nearest-neighbour). Clusters represent similar activities or user patterns, where normal and anomalous behaviour can be distinguished.

**Immune System Approach**

Application implementations inherently provide a model of normal behaviour, in the form of application code paths. In the immune system approach, applications are modelled in terms of sequences of system calls for a variety of different conditions: normal behaviour, error conditions and attempted exploits. Comparing this model to observed event traces allows classification of normal or suspicious behaviour. For example, an anomalous exec system call in a web server process may be indicative of a buffer overflow attack. This approach has demonstrated the ability to detect a number of typical attack techniques, but cannot detect attacks based on race conditions, policy violations or masquerading.

**Protocol Verification**

Many attack techniques rely on the use of unusual or malformed protocol fields, which are incorrectly handled by application systems. Protocol verification techniques rigorously check protocol fields and behaviour against established standards or heuristic expectations. Data that violates the relevant boundaries is tagged as suspicious. This approach, used in a number of commercial systems, can detect many commonly-used attacks, but suffers from the poor standards-compliance of many protocol implementations. [6] In addition, using this technique on proprietary or under-specified protocols may be difficult or lead to false positives.

**File Checking**

Best known through its implementation in the Tripwire system2, this technique uses cryptographic checksums of sensitive system data to detect changes - including unauthorised software installations, back-doors left by successful intrusion, and system corruption. These techniques are extremely useful in system recovery and forensic examination. Detection is inherently after the fact, however, and can be bypassed if the cryptographic checksums are modified or the examination process is compromised.

**Taint Checking**

An application-centric approach to anomaly detection lies in creating risk-aware applications. One example of this is in the Perl programming language, commonly used to implement HTTP CGI applications. In this system, all user-provided input is considered ‘tainted’; any attempt to use such tainted data in a sensitive context (such as an exec system call) fails. Un-tainting a variable must be done explicitly, by extracting expected content with a regular expression - avoiding the risk of embedded shell commands, or otherwise unexpected content, from being used.
Neural Nets
A neural net is essentially a network of computational units that jointly implement complex mapping functions. Initially, the network is trained with normal system behaviour traces. Observed event streams are then fed into the network, which classifies these streams as normal (observations match training data), or anomalous. The system may also undergo continuous training using this observed data, allowing the network to learn changes in system behaviour. Since this approach does not rely on preconceptions about the behaviour of the monitored system, it avoids the need to preselect suitable features and thresholds. Similarly, the learning ability of the network allows compensation for behaviour drift - though this may allow undetected violations actions to be included in the model. A greater difficulty with this technique lies in the fact that only an outcome may be observed - the reason for a mismatch between the model and observed behaviour is not evident.

White Listing
White listing is a simple technique, which involves passing a raw event stream (for example, a system log) through a number of filters, each corresponding to known benign patterns. Any remainder after all known events have been filtered out is novel or suspicious. Primarily a data reduction technique, this makes manual review of event streams feasible. Filters are manually incrementally refined, reducing the number of false positives - though sufficiently specific filters may be hard to generate. In addition, recognition of attacks typified by large numbers of otherwise normal events (e.g. login failure) is difficult.

III. CATEGORIES OF INTRUSION DETECTION
IDSs can also be categorized according to the detection approaches they use. Basically, there are two detection methods: misuse detection and anomaly detection. The major deference between the two methods is that misuse detection identifies intrusions based on features of known attacks while anomaly detection analyzes the properties of normal behaviour. IDSs that employ both detection methods are called hybrid detection-based IDSs.

Anomaly Detection Systems
Anomaly-based intrusion detection techniques depict an intrusion as a deviation from normal behaviour. The so called "normal behaviour needs to be defined, or learned. The first step in deploying anomaly-based detectors is therefore building up a knowledge base about the monitored domain (e.g. a flow of packets in a network link, a sequence of system calls, a sequence of HTTP request, etc.) usually derived from long-term system observations and define thresholds beyond which the system activity is considered abnormal. These systems can be learning-based or specification-based, depending on the way in which they determine, what the normal behaviour of a system is.

Misuse Detection Systems
Misuse detection systems define what an intrusion is in the observed system and uses a knowledge-base of system vulnerabilities and patterns of known security violations - so-called signatures - as a model of the intrusive process. Misuse detection systems identify evidence of attacks by searching for patterns of known attacks in the data collected from the monitored system. If an action does not precisely match a signature it is deemed not malicious and considered acceptable.

IV. COMPARISON BETWEEN MISUSE-BASED AND ANOMALY-BASED INTRUSION DETECTION
Anomaly-based systems are able to detect previously unknown attacks, new exploits and unforeseen vulnerabilities, but they traditionally produce more false positives, which are erroneous detections. This is because anomaly detection systems discriminate between intrusions and legitimate activities based on thresholds (set or learned) that characterize the normal behaviour, which need system-specific, hard-to-provide, tuning.

Misuse-based systems, on the other hand, are usually faster and have a low false positive rate, but cannot detect attack for which they do not have a description; therefore they depend on continuous updating of the signature database. The advantage of this approach, in addition to the fact that the potential for false alarms is very low, is the fact that the detailed contextual analysis makes it easy for the security officer to take preventive or corrective action, while, instead, is not easy to trace the identified anomaly back to the causing attack. This makes it difficult to automatically initiate countermeasures.

Because of their low false positive rate and their higher performance, and given that they are simpler to implement and to configure, misuse-based detection approaches are the basis for the majority of the existing network-based intrusion detection systems. Unfortunately, as the speed of
network links increases, keeping up with the pace of events becomes a real challenge.

In order to overcome the shortcomings of both approaches, as pointed out by Desai in [7], a new breed of systems is being developed that combines the low false-positive rate of signature-based intrusion detection systems and the ability of anomaly detection systems to detect novel unknown attacks. These detectors leverage their knowledge of both intrusive and normal behaviours and use that knowledge to classify events. The results reported by Hwang et al. In [11] using such kind of systems are very encouraging.

V. INTRUSIONS AND NETWORK ATTACKS

Intrusion is the attempt by an attacker to access or manipulate restricted information and services or, to prevent legitimate clients to do so (i.e., by compromising a system).

According to the work of Anderson [9], continued by Axelsson in [10, 11], attacks can be classified on the basis of two criteria, that is, whether or not the user is authorized to access the computer system and whether or not she can access a particular resource on the system. This categorization implies that we can divide intruders in three classes:

External Penetrators
Those who are not authorized to access the system.

Internal Penetrators: those who are authorized to access the system but are not authorized to access the information they have accessed. This class can be further divided in:

- Masqueraders: those who operate under another user's identity
- Clandestine Users: who evade logs and audits (usually authenticating the system as an administrator) internal penetrators are the culprits of most of the security breaches [12].
- Misfeasors: those who use their privileges maliciously.

From the perspective of the goal that the intruder attempts to accomplish, attacks can be roughly categorized in four categories:

- Information gathering;
- Unauthorized access;
- Disclosure of information;
- Denial of service.

From the point of view of the technique utilized in the attack, we can enumerate:

Network Attacks

Network attacks that focus on the networks themselves, rather than simply using them as a communication channel. The descriptions of some network attacks follow:

a. Sniffing

It is the technique at the basis of many attacks. The attacker sets her interface in promiscuous mode and can access all the traffic in a network segment. Since many protocols (TELNET, FTP, HTTP, and POP) transfer information in clear text, it is possible to collect sensible data by sniffing the network. There are plenty of tools, most of them in the public domain, to setup sniffing attacks.

b. Spoofing

It is an attack in which one person or program successfully masquerades as another by falsifying data and thereby gaining an illegitimate advantage. For instance in an IP spoofing attack, a host impersonates another host by sending a datagram with the address of some other host as the source address. IP spoofing is used to impersonate sources of security-critical information (e.g., a DNS server or a NIS server) and to bypass address-based authentication.

c. Hijacking

It is an attack in which an intruder interposes herself in a legitimate conversation and steals the role of one of the actors. It can be done at different abstraction levels such as link (ARP), transport (UDP, TCP) and even at the application level (HTTP and NIS).

d. Man-in-the-middle

It is an attack that can be considered a two-sided hijacking, since the attacker interposes herself in between a conversation between two actors, leaving the actors unaware of the interception. The perpetrator of a man-in-the-middle attack usually can access and modify all the information exchanged by the communication partners.

e. Fragmentation

It is an attack whose goal is to cause troubles during the reassembling stages on the receiving host. Bugs (or features) in fragmentation handling can be exploited to cause denials of service (e.g., when the size of the reconstructed packed exceeds the memory available) or to evade firewalls and intrusion detection systems (if the packet reconstruction policy differs from the one used by the end-host, and, therefore malicious packets can be seen as legitimate ones by the IDS under certain conditions [13]).
f. Denial of service
   It is a class of attacks in the attacker compromises or even tears down a service by means of exhausting the resources of the servers she is attacking. There are many known ways to do that: SYN flood, Ping of Death, UDP storm, etc.

g. Reconnaissance
   It is an attack that tries to identify and fingerprint the machines that the attacker wants to compromise and the operating systems running on them. This is done by means of different types of scan such as TCP (connect, half-open, FIN, ACK, idle scan) and UDP port sweeps.

h. Local Attacks
   These are possible when the attacker has already an account on the system she wants to compromise, i.e., the attacker is an internal penetrator. The vast majority of the local vulnerabilities in UNIX systems exploit SUID (Set User ID)-root programs to obtain root privileges. There are many attacks based on this idea: an indicative, yet non-all-encompassing, list follows.

   i. Environment Attacks
      These lie in those attacks, in which the attacker gains restricted privileges maliciously modifying environment variables.

      Input Arguments Attacks: These attacks exploit the lack of size-checking and sanitization in user provided data to perform overflows, command injections, or directory traversal attacks.

      Symlink attacks: These attacks leverage the fact that some applications that create temporary files for logging/locking do not test if the file already exists or whether is a symbolic link.

      Race condition attacks: In this case, the attacker races against the applications by exploiting the gap between testing and accessing a resource. The whole family of the TOCTTOU (Time-Of-Check-To-Time-Of-Use) attacks is based on this idea.

      File descriptor attacks: These attacks exploit the fact that sometimes SUID applications open files to perform their tasks and fork external processes. If the close-on-exec flag is not set, the new process will inherit the open file descriptors of the original process. If the spawned program is interruptible, or worse, is interactive, a malicious user can use the file descriptors of the father process and write to them.

      Format string attacks: This is a complicate attack based on the simple idea that the C*printf functions are not type safe, and, if the arguments are interpreted as a format specifier, it is possible to write arbitrary values in the process memory by providing a carefully crafted format string.

j. Overflow Attacks
   These attacks encompass more a common usage pattern than a technique per se, but we treat them separately since they are one of the most popular types of attacks. The lack of boundary checking is one of the most common mistakes in C/C++ applications. Buffer overflows can be exploited both locally and remotely and can modify both the data flow and the control flow of an application. Recent tools have made the process of exploiting buffer overflows easier if not completely automatic. Much research has been devoted to finding vulnerabilities, design prevention techniques, and developing detection mechanisms. The most common overflows attacks are:

      Stack-based overflows: Such as return address overflows, jump into libc, off by one, long jump overflows, and array overflows.

      Heap based overflows: This exploit in-band control structures for dynamic memory management and C++ vtables.

      Integer overflows: It caused by unexpected results when comparing, casting, and adding integers.

k. Web Attacks
   Web attacks against web-based authentication, authorization, and HTTP protocols implementations. Beyond the classical techniques of eavesdropping and brute-force guessing, examples of web-specific attack are:

      SQL injections: In these types of attacks, arbitrary SQL commands are injected into database servers by exploiting poor sanitization of client-provided data. Similar concepts are used to inject LDAP and XPath based application as well.

      Session fixation: These forces the user's session ID to a known value. The ID can be a fixed value or can be obtained by the attacker through a previous interaction with the vulnerable system.

      Cross-site scripting: In such types of attacks, the privacy of a client is violated by executing commands on the client impersonating critical servers.

      Web spoofing: It is also known as a man-in-the-middle attack, which exploits URL rewriting techniques to hijack the user browsing session.

      Response splitting and request smuggling: It attacks to HTTP protocol implementations. The aforementioned categorizations are not in contrast with each other and are aimed at characterizing attacks from different points of view. It is remarkable to notice that many real-world attacks do not fit in any of the categories above but, instead, use a combination of those attacks.
VI. CONCLUSION

Intrusion detection systems (IDSs) play an important role in computer security. IDS users relying on the IDS to protect their computers and networks demand that an IDS provides reliable and continuous detection service. However, many of the today’s anomaly detection methods generate high false positives and negatives. All anomaly-based intrusion detection systems work on the assumption that normal activities differ from the abnormal activities (intrusions) substantially. In the case of IDS models that learn a program’s behaviour, these deference may manifest in the form of (a) the frequency of system calls (Src_bytes, Dst_bytes), and (b) the duration of system calls used by the processes under normal and abnormal execution.

REFERENCE


