

Introduction

In the last three years, the networking revolution has finally come of age. More than ever before, we see that the Internet is changing computing as we know it. The possibilities and opportunities are limitless; unfortunately, so too are the risks and chances of malicious intrusions.

It is very important that the security mechanisms of a system are designed so as to *prevent* unauthorized access to system resources and data. However, completely preventing breaches of security appear, at present, unrealistic. We can, however, try to detect these intrusion attempts so that action may be taken to repair the damage later. This field of research is called **Intrusion Detection**.

Anderson, while introducing the concept of intrusion detection in 1980, defined an **intrusion attempt** or a **threat** to be the potential possibility of a deliberate unauthorized attempt to

- access information,
- manipulate information, or
- render a system unreliable or unusable.

Since then, several techniques for detecting intrusions have been studied. This paper discusses why intrusion detection systems are needed, the main techniques, present research in the field, and possible future directions of research.

SECURITY POLICY

A **Security Policy** defines what is permitted and what is denied on a system. There are two basic philosophies behind any security policy:

1. **Prohibitive** where everything that is not expressly permitted is denied.
2. **Permissive** where everything that is not expressly denied is permitted.

Elements of a System's Security

A computer system can be considered as a set of resources which are available for use by authorized users. A paper by Donn P outlines six elements of security that must be addressed by a security administrator. It is worth evaluating any tool by determining how it address these six elements.

1. **Availability** - the system must be available for use when the users need it. Similarly, critical data must be available at all times.
2. **Utility** - the system, and data on the system, must be useful for a purpose.
3. **Integrity** - the system and its data must be complete, whole, and in a readable condition.
4. **Authenticity** - the system must be able to verify the identity of users, and the users should be able to verify the identity of the system.
5. **Confidentiality** - private data should be known only to the owner of the data, or to a chosen chosen few with whom the owner shares the data.
6. **Possession** - the owners of the system must be able to control it. Losing control of a system to a malicious user affects the security of the system for all other users.

The need for Intrusion Detection Systems

A computer system should provide confidentiality, integrity and assurance against denial of service. However, due to increased connectivity (especially on the Internet), and the vast spectrum of financial possibilities that are opening up, more and more systems are subject to attack by intruders. These subversion attempts try to exploit flaws in the operating system as well as in application programs and have resulted in spectacular incidents like the Internet Worm incident of 1988.

There are two ways to handle subversion attempts. One way is to prevent subversion itself by building a completely secure system. We could, for example, require all users to identify and authenticate themselves; we could protect data by various cryptographic methods and very tight access control mechanisms. However this is not really feasible because:

1. In practice, it is not possible to build a completely secure system because bug free software is still a dream, & no-one seems to want to make the effort to try to develop such software. Apart from the fact that we do not seem to be getting our money's worth when we buy software, there are also security implications when our E-mail software, for example, can be attacked. Designing and implementing a totally secure system is thus an extremely difficult task.
2. The vast installed base of systems worldwide guarantees that any transition to a secure system, (if it is ever developed) will be long in coming.
3. Cryptographic methods have their own problems. Passwords can be cracked, users can lose their passwords, and entire crypto-systems can be broken.
4. Even a truly secure system is vulnerable to abuse by insiders who abuse their privileges.

5. It has been seen that that the relationship between the level of access control and user efficiency is an inverse one, which means that the stricter the mechanisms, the lower the efficiency becomes.

We thus see that we are stuck with systems that have vulnerabilities for a while to come. If there are attacks on a system, we would like to detect them as soon as possible (preferably in real-time) and take appropriate action. This is essentially what an Intrusion Detection System (IDS) does. An IDS does not usually take preventive measures when an attack is detected; it is a reactive rather than pro-active agent. It plays the role of an informant rather than a police officer.

The most popular way to detect intrusions has been by using the audit data generated by the operating system. An audit trail is a record of activities on a system that are logged to a file in chronologically sorted order. Since almost all activities are logged on a system, it is possible that a manual inspection of these logs would allow intrusions to be detected. However, the incredibly large sizes of audit data generated (on the order of 100 Megabytes a day) make manual analysis impossible. IDS automate the drudgery of wading through the audit data jungle. Audit trails are particularly useful because they can be used to establish guilt of attackers, and they are often the only way to detect unauthorized but subversive user activity.

Many times, even after an attack has occurred, it is important to analyze the audit data so that the extent of damage can be determined, the tracking down of the attackers is facilitated, and steps may be taken to prevent such attacks in future. An IDS can also be used to analyze audit data for such insights. This makes IDS valuable as real-time as well as post-mortem analysis tools.

Anderson also classified intruders into two types, the *external* intruders who are unauthorized users of the machines they attack, and *internal* intruders, who have permission to access the system, but not some portions of it. He further divided internal

intruders into intruders who *masquerade* as another user, those with *legitimate* access to sensitive data, and the most dangerous type, the *clandestine* intruders, who have the power to turn off audit control for themselves.

Classification of Intrusion Detection Systems

Intrusions can be divided into 6 main types

1. *Attempted break-ins*, which are detected by atypical behavior profiles or violations of security constraints.
2. *Masquerade attacks*, which are detected by atypical behavior profiles or violations of security constraints.
3. *Penetration* of the security control system, which are detected by monitoring for specific patterns of activity.
4. *Leakage*, which is detected by atypical use of system resources.
5. *Denial of service*, which is detected by atypical use of system resources.

6. Malicious use, which is detected by atypical behavior profiles, violations of security constraints, or use of special privileges.

Characterization of Intrusion Detection Systems Based on Data Source

1. *Host based:*

Audit data from a single host is used to detect intrusions.

2. *Multihost based:*

Audit data from multiple hosts is used to detect intrusions.

3. *Network based:*

Network traffic data, along with audit data from one or more hosts, is used to detect intrusions.

Characterization of Intrusion Detection Systems Based on Model of Intrusions

1. *Anomaly detection model:*

This intrusion detection system detects intrusions by looking for activity that is different from a user's or system's normal behavior.

2. *Misuse detection model:*

This intrusion detection system detects intrusions by looking for activity that corresponds to known intrusion techniques (signatures) or system vulnerabilities

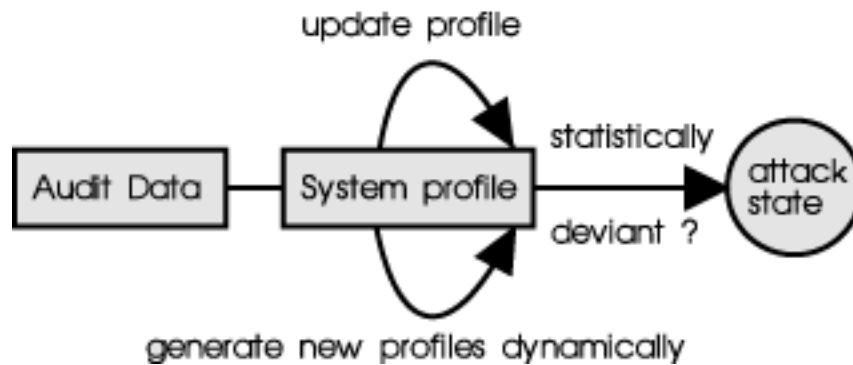
However, we can divide the techniques of intrusion detection into two main types.

Anomaly Detection

Anomaly detection techniques assume that all intrusive activities are necessarily anomalous. This means that if we could establish a "normal activity profile" for a system, we could, in theory, flag all system states varying from the established profile by statistically significant amounts as intrusion attempts. However, if we consider that the set of intrusive activities only intersects the set of anomalous activities instead of being exactly the same, we find a couple of interesting possibilities: (1) Anomalous activities that are not intrusive are flagged as intrusive. (2) Intrusive activities that are not anomalous result in false negatives (events are not flagged intrusive, though they actually are). This is a dangerous problem, and is far more serious than the problem of false positives.

The main issues in anomaly detection systems thus become the selection of threshold levels so that neither of the above 2 problems is unreasonably magnified, and the selection of features to monitor. Anomaly detection systems are also computationally expensive because of the overhead of keeping track of, and possibly updating several system profile metrics. Some systems based on this technique are discussed in Section 4 while a block diagram of a typical anomaly detection system is shown in figure below.

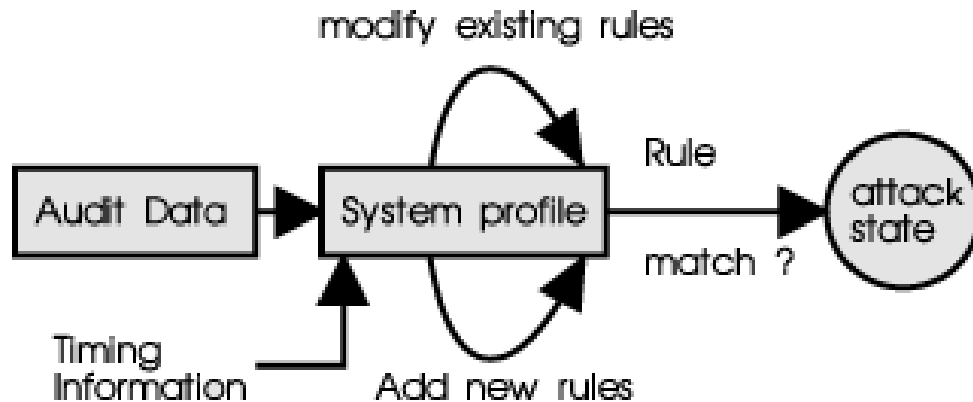
A typical anomaly detection system



Misuse Detection

The concept behind misuse detection schemes is that there are ways to represent attacks in the form of a pattern or a signature so that even variations of the same attack can be detected. This means that these systems are not unlike virus detection systems -- they can detect many or all *known* attack patterns, but they are of little use for as yet unknown attack methods. An interesting point to note is that anomaly detection systems try to detect the complement of "bad" behavior. Misuse detection systems try to recognize known "bad" behavior. The main issues in misuse detection systems are how to write a signature that encompasses *all* possible variations of the pertinent attack, and how to write signatures that do not also match non-intrusive activity. Several methods of misuse detection, including a new pattern matching model are discussed later. A block diagram of a typical misuse detection system is shown in figure below.

A typical misuse detection system



Anomaly Detection Systems

There have been a few major approaches to anomaly intrusion detection systems, some of which are described below.

Statistical approaches:

In this method, initially, behavior profiles for subjects are generated. As the system continues running, the anomaly detector constantly generates the variance of the present profile from the original one. We note that, in this case, there may be several measures that affect the behavior profile, like activity measures, CPU time used, number of network connections in a time period, etc. In some systems, the current profile and the previous profile are merged at intervals, but in some other systems profile generation is a one time activity. The main advantage to statistical systems is that they adaptively learn the behavior of users; they are thus potentially more sensitive than human experts. However there are a few problems with statistical

approaches: they can gradually be trained by intruders so that eventually, intrusive events are considered normal, false positives and false negatives are generated depending on whether the threshold is set too low or too high, and relationships between events are missed because of the insensitivity of statistical measures to the order of events.

An open issue with statistical approaches in particular and anomaly detection systems in general, is the selection of measures to monitor. It is not known exactly what the subset of all possible measures that accurately predicts intrusive activities is. Static methods of determining these measures are sometimes misleading because of the unique features of a particular system. Thus, it seems that a combination of static and dynamic determination of the set of measures should be done. Some problems associated with this technique have been remedied by other methods, including the method involving Predictive Pattern Generation, which takes past events into account when analyzing the data.

Predictive pattern generation:

This method of intrusion detection tries to predict future events based on the events that have already occurred. Therefore, we could have a rule

$E1 - E2 \rightarrow (E3 = 80\%, E4 = 15\%, E5 = 5\%)$

This would mean that given that events E1 and E2 have occurred, with E2 occurring after E1, there is an 80% probability that event E3 will follow, a 15% chance that event E4 will follow and a 5% probability that event E5 will follow. The problem with this is that some intrusion scenarios that are not described by the rules will not be flagged intrusive. Thus, if an event sequence A - B - C exists that is intrusive, but not listed in the rule base, it will be classified as unrecognized. This problem can be partially solved by

flagging any unknown events as intrusions (increasing the probability of false positives), or by flagging them as non-intrusive (thus increasing the probability of false negatives). In the normal case, however, an event is flagged intrusive if the left hand side of a rule is matched, but the right hand side is statistically very deviant from the prediction.

There are several advantages to this approach. First, rule based sequential patterns can detect anomalous activities that were difficult with traditional methods. Second, systems built using this model are highly adaptive to changes. This is because low quality patterns are continuously eliminated, finally leaving the higher quality patterns behind. Third, it is easier to detect users who try to train the system during its learning period. And fourth, anomalous activities can be detected and reported within seconds of receiving audit events.

Neural networks:

The idea here is to train the neural network to predict a user's next action or command, given the window of n previous actions or commands. The network is trained on a set of representative user commands. After the training period, the network tries to match actual commands with the actual user profile already present in the net. Any incorrectly predicted events (events and commands are used interchangeably in this discussion) actually measure the deviation of the user from the established profile. Some advantages of using neural networks are: they cope well with noisy data, their success does not depend on any statistical

assumption about the nature of the underlying data, and they are easier to modify for new user communities. However, they have some problems. First, a small window will result in false positives while a large window will result in irrelevant data as well as increase the chance of false negatives. Second, the net topology is only determined after considerable trial and error. And third, the intruder can train the net during its learning phase.

Misuse Detection Systems

There has been significant research in misuse detection systems in the recent past, including attempts at SRI, Purdue University and the University of California-Davis. Some of these systems are explained in depth in this section.

Expert systems:

They are modeled in such a way as to separate the rule matching phase from the action phase. The matching is done according to audit trail events. The Next Generation Intrusion Detection Expert System (NIDES) developed by SRI is an interesting case study for the expert system approach. NIDES follow a hybrid intrusion detection technique consisting of a misuse detection component as well as an anomaly detection component. The anomaly detector is based on the statistical approach, and it flags events as intrusive if they are largely deviant from the expected behavior. To do this, it builds user profiles based on many different criteria (more than 30 criteria, including CPU and I/O usage, commands used, local network activity, system errors etc.). These profiles are updated at periodic intervals. The expert system misuse detection component encodes known intrusion scenarios and attack patterns (bugs in old versions of send-mail could be one vulnerability). The rule database can be changed for different systems. One advantage of the NIDES approach is that it has a statistical component as well as an expert system component. This means that the chances of one system catching intrusions missed by the other increase. Another advantage is the problem's control reasoning is cleanly separated from the formulation of the solution.

There are some draw backs to the expert system approach too. For example, the expert system has to be formulated by a security professional and thus the system is only as strong as the security personnel who program it. This means that there is a real chance that expert systems can fail to flag intrusions. It is for this reason that NIDES has an anomaly as well as a misuse detection component. These two components are loosely coupled in the sense that they perform their operations independently for the most part. The NIDES system runs on a machine different from the machine(s) to be monitored, which could be unreasonable overhead. Furthermore, additions and deletions of rules from the rule-base must take into account the inter-dependencies between different rules in the rule-base. And there is no recognition of the sequential ordering of data, because the various conditions that make up a rule are not recognized to be ordered.

Model Based Intrusion Detection:

The system states that certain scenarios are inferred by certain other observable activities. If these activities are monitored, it is possible to find intrusion attempts by looking at activities that infer a certain intrusion scenario. The model based scheme consists of three important modules. The anticipator uses the active models and the scenario models to try to predict the next step in the scenario that is expected to occur. A scenario model is a knowledge base with specifications of intrusion scenarios. The planner then translates this hypothesis into a format that shows the behavior as it would occur in the audit trail. It uses the predicted information to plan what to search for next. The interpreter then searches for this data in the audit trail. The system proceeds this way, accumulating more and more evidence for an intrusion attempt until a threshold is crossed; at this point, it signals an intrusion attempt. This is a very clean approach. Because the planner and the interpreter know what they are searching for at each step, the large amounts of noise present in audit data can be filtered, leading to excellent performance improvements. In addition, the system can predict the attacker's next move based on the intrusion model. These predictions can be used to verify an intrusion hypothesis, to take preventive measures, or to determine what data to look for next.

Pattern Matching:

This model encodes known intrusion signatures as patterns that are then matched against the audit data. Like the state transition analysis model, this model attempts to match incoming events to the patterns representing intrusion scenarios. The implementation makes transitions on certain events, called labels, and Boolean variables called guards can be placed at each transition. The difference between this and the state transition model is that the state transition model associates these guards with states, rather than transitions. The important advantages of this model are:

1. Declarative Specification: It only needs to be specified what patterns need to be matched, not how to match them.
2. Multiple event streams can be used together to match against patterns for each stream without the need to combine streams. This means that streams can be processed independently, and their results can be analyzed together to give evidence of intrusive activity.
3. Portability: Since intrusion signatures are written in a system independent script, they need not be rewritten for different audit trails. The patterns' declarative specifications enable them to be exchanged across different Operating Systems and different audit trails.
4. It has excellent real-time capabilities. Kumar reports a CPU overhead of 5-6% when scanning for 100 different patterns, which is excellent.
5. It can detect some attack signatures like the failed logins signature that the state transition model cannot do.

One problem with this model is that, it can only detect attacks based on known vulnerabilities. In addition, pattern matching is not very useful for representing ill-defined patterns and it is not an easy task to translate known attack scenarios into patterns that can be used by the model. Also, it cannot detect passive wire-tapping intrusions, nor can it detect spoofing attacks where a machine pretends to be another machine by using its IP address.

An interesting fact about this IDS is that it is called IDIOT (Intrusion Detection In Our Time), and we leave it to the reader to ponder the appropriateness of the name for the state of the art in intrusion detection.

Characteristics of a Good Intrusion Detection System

An intrusion detection system should address the following issues, regardless of what mechanism it is based on:

1. It must **run continually** without human supervision. The system must be reliable enough to allow it to run in the background of the system being observed. However, it should not be a "black box". That is, its internal workings should be examinable from outside.
2. It must be **fault tolerant** in the sense that it must survive a system crash and not have its knowledge-base rebuilt at restart.
3. On a similar note to above, it must **resist subversion**. The system can monitor itself to ensure that it has not been subverted.
4. It must impose **minimal overhead** on the system. A system that slows a computer to a crawl will simply not be used.
5. It must **observe deviations** from normal behavior.
6. It must be **easily tailored** to the system in question. Every system has a different usage pattern, and the defense mechanism should adapt easily to these patterns.
7. It must cope with **changing system behavior** over time as new applications are being added. The system profile will change over time, and the IDS must be able to adapt.
8. Finally, it must be **difficult to fool**.

Conclusion

Intrusion Detection is still a fledgling field of research. However, it is beginning to assume enormous importance in today's computing environment. The combination of facts such as the unbridled growth of the Internet, the vast financial possibilities opening up in electronic trade, and the lack of truly secure systems make it an important and pertinent field of research. Future research trends seem to be converging towards a model that is a hybrid of the anomaly and misuse detection models; it is slowly acknowledged that neither of the models can detect all intrusion attempts on their own. This approach has been successfully adopted in NIDES, and we can expect more such attempts in the future.

Tomorrow's IDS

Due to the inability of NIDES to see all the traffic on switched Ethernet, many companies are now turning to Host-based IDS (second generation). These products can use far more efficient intrusion detection techniques such as heuristic rules and analysis. Depending on the sophistication of the sensor, it may also learn and establish user profiles as part of its behavioral database. Charting what is normal behavior on the network would be accomplished over a period of time.

.