The Development of a Intrusion Detection/Defense System for Linux Hosts

Banji K. Lawal
Department of Microcomputer Studies
St. Cloud State University
laba0001@stcloudstate.edu

Dr. Dennis Guster
Department of Microcomputer Studies
St. Cloud State University
guster@stcloudstate.edu

Abstract

As distributed computer systems perform an increasing number of essential services the consequences of the failure of these mission critical networks become more catastrophic. A significant number of networks are susceptible to performance degradations due to the malicious acts of intruders.

Some of the current approaches to decreasing the Mean Time Between Failures (MTBF) Of computer networks have focused on designing feedback systems based on intrusion detection and defense to increase the survivability of networks. The paper describes requirements, techniques and issues pertaining to the development of an ID/DS capability for Linux hosts.

Introduction

One of the most important networking problems is assuring that the network and its resources perform as expected. If network's behavior is unpredictable or unexpected, such a network is insecure [1]. Therefore, to assure that a network performs as expected it must be secure.

Many researchers in the field of network security have stated that network security is a process that tries to optimize two characteristics of secure systems. The two characteristics are integrity and confidentiality. Having integrity means the system and its information remain unaltered by accidents or malicious attacks while confidentiality means the data that is in the system must be available only to users who are authorized to access and manipulate the information or resources on the network [2]. Once a computer system has both integrity and confidentiality, we can claim that the system is trusted. If we trust the system, we can then consider it secure.

To assure that the network and its services are secure recommended practices call for System Administrators to develop or implement a security system generally referred to as Intrusion Detection/Defense System (ID/DS) [3].

Intrusion detection (ID) systems are prevalent in distributed systems; for example, telephone networks use intrusion detection systems to prevent the large-scale misuse of corporate phone services. Other distributed systems where ID technologies are implemented are in the satellite industry and power networks [4].

Intrusion Detection Systems (IDS) are popular in these different infrastructure networks for a number of reasons. The first one is that as distributed systems increase in size and complexity the ability of human operators to detect system breaches in real time decreases. However, with the aid of an automated system monitoring tool administrators are able to maintain the integrity of the network.

Another factor that leads to the popularity of IDS is that the majority of these distributed systems run continuously; however, personnel to watch the network are not always available or on site. Having an automated system monitor alert administrators allows the network to be administered remotely.

The last reason for deploying a IDS in a computer network is that currently most attacks on computer networks are initiated with the aid of scripts, which can launch thousands of automated stacks on a computer network. Most non-automated defensive and detective methods would be overwhelmed by the scripted attacks. Parties responsible for maintaining a network's integrity and trust have found that using a IDS the network will not be overwhelmed.

Research Project

Our current research project is on implementing an ID/DS for Linux networks. We hope to achieve this goal by using ID technology on as many of the OSI layers of the network as is possible. It is expected that after developing and implementing ID/DS that the network will be more resistant to intrusion. The ID/DS should also have the side effect of reporting problems that could undermine network performance.

The goal of this paper is to discuss some of the factors that must be considered when attempting to implement such a system.

Expected Characteristics of ID/DS

Our first concern in designing a Linux based ID/DS system is what characteristics the application must have to meet the user's requirements [5]. Once we know what they are we will try to develop a system that will provide defense in depth and by using multiple detectors and defenses [6]. The desired characteristics are described in the following pages.

Modifiability

The system must be upgradeable. This is essential since the types of threats keeps increasing; it will be easier to add new detection and defensive components to the Intrusion Detection/Defense System if it is made upgradeable from the ground up. Also as the detection methods improve if the system is easily modified it will be able administer patches and bug fixes to the code or replace hardware.

Another aspect of a modifiable ID/DS application is that it should scale well from a single host to large LANs or WANs. Many architects of system design have found that building modularity into the architecture makes maintenance of the system easier. Hence in designing and our ID/DS we must decide on a scheme for organizing the components. We will probably have to use all of the following schemes.

- Residence of component in the OSI model
- The component's function
- Attack phase that the component works under

Reliability

For users to rely on ID/DS they should be able to trust it will behave in a manner that is predictable to a person who has no knowledge of the ID/D system's inner workings. To get reliability there are a number of things that must be done.
Firstly the results generated by ID/DS must be correct, precise and accurate. Correctness alone is not enough since many intrusion detection systems report minor network problems or failures as security breaches. Although they are correct, the results are meaningless [6]. The result must not only be correct but accurate in its description of the seriousness of the event and also precise in noting under what conditions the breach occurred. A record of where the breach occurred is also needed. If a response does not have these three qualities then operators of ID/DS will disregard any information that it provides.

Since the users and designers of the security system cannot predict the nature of all possible future attacks the solutions provided by ID/DS must be general enough that the system does not fail when it encounters conditions outside it's specifications or boundary conditions. In other words, the design of the Intrusion Detection/Defense System must be robust. One of the problem of building a robust IDS is that defining what an intrusion is mathematically is for networks, a nontrivial problem, hence most IDS work on specific cases [4].

Many attackers strive to overload a network's defenses or bring about failures by exploiting security holes, one way to make the intrusion detection system survivable is by making it fault-tolerant [7]. Fault-tolerance is achieved by creating redundant components that will come online if one system fails. Since fault-tolerant systems consist of modules that perform the same function in different ways a fault-tolerant system may be more robust than a fault-intolerant one.

Other techniques such as gating, methods, or use of association metrics have been used to lower error rate of ID/D systems [6].

Cost

For an intrusion detection system to be useful, the cost of implementing it must be low in terms of installation, support training, and money. If these factors are too high, the monetary and non-monetary cost to organizations in using ID/DS will be prohibitive.

To be widely used, it is also essential that the intrusion detection systems do not use a great deal of bandwidth or system resources. If performance of hosts or the network is reduced by ID/DS organizations will remove it from their systems and thus leave them open to attack. For these reasons, it is preferred that ID/DS uses mainly passive sensors. Furthermore, all its software should run as background processes. The best approach is to run the ID/DS services under a client-server model where there will be specific machines that will run the ID/DS applications that will be available to clients via remote procedure

calls or anonymous FTP [8]. By using the existing Linux networking utilities such as NIS/NFS or protocols such as RPC and UDP it should be possible to run ID/DS on a network with minimal performance loss.

ID/DS Software Design Considerations

Having examined the principles that will guide the design of ID/DS, we can now conclude that the best architectural approach to building a ID/DS is to use a modular approach with an object oriented language such as C++ or Java on the software side. The OOP languages are preferred since with them adding more functionality is easier than with procedural based languages such as C.

Java is also a good candidate since it has it's own native security features and it's memory management strategies help make more robust code. Many IDS designers have found that using intelligent agents is a very prudent design approach. If we are going to use expert systems it may be more efficient to use Lisp.

ID/DS Hardware Design Considerations

On the hardware level, we can use various switches and other electronic devices that will monitor the wires and their traffic. In terms of the architecture, the best way to make the system reliable and provide defense/detection in depth is to place different components on all seven layers of the OSI model.

Layer 1: Physical

Here we simply watch for wire taps in the media and detect things like voltage leaks and changes in frequency or signal intensity. Since most networks are LANs where the administrators control the media, intruders find it very difficult to carry out wiretaps. Hence, for LANs most attacks will not be at this level.

Layer 2: Data Link

Security practioners recommend that switches be used to route packets to the MAC address of each node instead of hubs [9], which simply broadcast to all points. By using switches, attackers see a limited amount of network traffic. We can also prevent connections from unauthorized MAC addresses.

Layer 3: Network

Current best practices recommend that traffic filtering schemes like firewalls be installed. We can also provide some protection from snoopers by using security options and encrypting our datagrams [9].

Layers 4-7: Transport to Presentation

On the transport layer the best thing to do is to encrypt traffic and make sure that it meets expected parameters of legal traffic. The same applies to other layers.

ID/DS Detection Levels

There are two levels of detection in ID/DS. Level one is detection on the network layer while level two is detection under the operating system.

Level One: Network Intrusion Detection

Network intrusion detectors have the following parts

- Network Sensors (monitors)
- Network Analyzers
- Traffic Database
- Encryption System

Network Sensors

In IDS there are two categories of sensors. IDS detectors are either signature or anomaly based.

Signature Sensors

These types of sensors respond to a specified signal or pattern that matches the known profile of an attack [8]. The sensor verifies that the signal matches the signal associated with an attack by using pattern matching techniques. McHugh, Christie and Christie [10] report that pattern-matching methods are based on the following approaches

- Comparison to known intrusive packets.
- Creation of state machines describing attacks.

For redundancy and robustness, we should have signal monitors that operate on any two of the techniques described above. Sensors that use the same methods will poll each other to verify that they each detect the same thing. If any of them have reports that differ from the others, it will be assumed that the network has failed in some form and this will then be investigated.

Since signature sensors are only sensitive to known threats if they are exposed to a novel attack they will give a false-negative response. For this reason it is a good idea to Have anomaly monitors also.

Anomaly Sensors

Anomaly sensors analyze a signal's noise for deviations from its usual noise Distribution [10]. Anomalies from the signal's expected noise distribution indicate that an attack is in progress. Unlike signature monitors, anomaly sensors can detect new types of attacks since they do not use static libraries to predict network breaches [10]. However, anomaly detectors report a higher number of false positives [11] than signature sensors. Researchers in this field mainly use one of the following approaches for detecting noise anomalies, they are:

- Statistical modeling of system behavior to generate a distribution of the network's noise.
- Use of a neural nets to create an abstract model of the attack
- Approaches based on the human immune system.

Once again, we need at least two differing methods to create a fault-tolerant system. The only time when it is concluded that the network is secure is when all the stations report that there is no unexpected changes in the noise distribution.

Anomaly detection has one problem, which is that if the attacker makes small intrusions over time the breaches may fall beneath the threshold of the sensor [11]. This can lead to a massive and deep breach of the network over a long period.

Traffic Database

This is where the information collected from the network scanners is collected. The default data in the database will consist of a library of known attack signatures. To build a distribution model for anomaly detection the data will have to be collected for a certain time interval. The best method to take for the data collections is to collect samples of packets at various time intervals. To build an accurate noise distribution model we must collect data during the following times.

- Peak times

- Nonpeak times
- Every day of the week
- Days and nights

By collecting data at these times we can build more sensitive models, which will be run at specific time intervals. By doing this we can generate more reliable results for our ID/DS.

Network Analyzer

The network analyzer generates the signatures for signature detectors. It also generates new signatures whenever attacks occur by data mining the database. It will also generate models of noise distributions for anomaly detectors. The best design will be to create a separate analyzer for each database.

Developing the signatures and noise distributions is a complicated task. It may be possible to simplify this by encrypting legitimate traffic. By doing this hopefully legal traffic will have a different profile from hostile traffic. This may make it easier to detect intrusions.

Level Two: Operating System Intrusion Detection

Under Linux, confidentiality is maintained by restricting the access rights that users have to mission-critical processes. Hence, we have different levels of users. User confidentiality is maintained by a variety of authentication schemes [1]. Integrity is maintained by allowing only trusted users to modify critical system processes.

As a result of this the OS based ID/DS must detect unauthorized access to mission-critical process. This can be done in the following ways:

- Verify file integrity using cryptographic checksums
- Monitor logs to see what events have been reported
- Monitor users who may try to modify mission critical resources
- Build profiles of every user. Deviations from the user's norm may signal that an attack is taking place
- Detect and log changes that are made to mission critical files.

ID/DS Defensive Methods

Traditionally site administrators have found that the developing and following a strong and sound security policy is the best way to defend their network. Most of these policies have focused on restricting access to network services and preventing unwanted changes to the system. These defensive procedures depend on the output of the ID system. Since

the intrusion detection is done on both the network and operating system we must enact defenses on these levels.

Network Defense

The current state of the art of network defense mainly consists of two approaches. The first approach is to shut down vulnerable services and nodes. The second is to deny access to sensitive resources.

One problem with shutting down systems is that a wily attacker can use the network's defense mechanism to institute a denial of service (DoS) attack. DoS attacks can be started by triggering the defensive net shutdown essential services, this will then lead to a cascade of network failures [4]. The only way to prevent this is to have a backup network that uses a completely different security scheme. Such a network will be prohibitively expensive for most organizations, which makes this unfeasible. The best strategy is to backup the network's data.

Other ways of prevent network failure include rerouting traffic through secure nodes, routers, and switches [9]. This may be more feasible than having a separate network; it may be best to have switching mechanisms that can handle very high capacities even for a small network so that in the event of a failure they can cope with the total network load.

Other things to do include tracing the source of the attack. This can be hard to do in real time since most attackers use dummy accounts on numerous ISPs and direct their attacks through circuitous paths that can span continents [12]. Though the perpetrators can be found tracing the source of the breach can stop future ones. If the attacker is stealing data or lurking on the site, tracing the source may be more useful since their activity is more likely to be long term.

Cohen [13] reports that the use of deceptive devices such as deception toolkits and honey pots Is an effective way to defend the network. By using deception, the attacker cannot make an effective breach since they do not posses true information about the network's structure.

Operating System Defenses

To defend the OS the most important thing that the administrator does is to keep on top of security announcements and patches. The reason why this is important is that many times the users of software have no control of what it does hence they must rely on the developers.

Researchers in the field of survivable systems have also proposed incorporating dynamic system adaptation into hardware and software [14]. Other approaches to designing

survivable systems include building black box modules, and engineering self-aware systems [15].

Other software defenses rely on the accuracy of detection techniques to work well a number of them have bee documented by Cohen [7]. A number of the methods incorporate cryptographic checksums, fault-tolerant software design, or using integrity shells. As research in building survivable systems continues, [15] it will be easier to defend software from failure.

ID/DS Limitations

ID/DS has a number of limitations many of them are common to all IDS. One of these is that although we can assure that the IDS will detect events and mark them as attacks the detectors frequently report benign events as hostile. Hence the accuracy of the IDS is limited. By using a fault tolerant model of different detectors we may be able to reduce the amount of erroneous attacks that are reported.

Since ID/DS is used to verify the security of the network it is a logical target for attackers who will try to compromise ID/DS's, which will then generate spurious reports and carry out other acts such as corrupt models, signatures, logs and checksums. Attackers may also fake packets to match the signatures or noise patterns of legitimate packets.

Encrypting all the network traffic introduces two other problems. The first one is that encryption consumes a great deal of processing time and bandwidth. The other limitation of encryption is that it may be possible to guess what the contents of packets are by conducting brute force attacks and traffic analysis.

A complete ID/DS system will consume a great deal of network resources the database will require massive storage and encryption uses a great deal of processing cycles and bandwidth. An intrepid attacker could use this knowledge to bring about a DoS attack by overwhelming the detection system with a large number of packets.

Another problem with ID/DS and other ID systems is that a great deal of what they report turns out to be trivia, and discounted by the administrator. This leads to lack of faith in the ID system and operators will then ignore it, which can lead to disaster when the reports are correct.

The last limitation of ID/DS and other IDS systems is that since each network will have different policies there are different actions that they may consider intrusions. For example in an academic setting a student copying the file "/etc/services" file or examining "rc.local" would be considered a benign use of the network. However in an enterprise a user viewing this same file may be considered a hostile act. Due to the fact that there is no general description of what constitutes unauthorized behavior in all cases

it is hard for ID developers to express a general mathematical description of what an intrusion is and propose an effective general deterrent to all attacks [4].

Conclusion

Although ID/DS cannot provide complete system security implementing it will decrease the probability that the network will fail; if it is used with other security tools the networks resources should be hardened against failures from attacks. By providing an automated means of defending computer systems it should make it easier for administrators to manage the network.

Currently there are many intrusion detection applications on the market. All these differing ID systems vary in their approach to intrusion detection. A number of them are such as ISS, and SATAN are also used by attackers [11] so their effectiveness is limited. At present the best way to acquire detection/defense in depth is by using a number of ID/D systems on the network.

Currently most researchers in the field of IDS and system survivability have are carrying out research on combining defensive mechanisms with deductive ones. By doing this they hope to build distributed systems, which are more responsive to change and have lower MTBFs than current networks.

Acknowledgements

We would like to acknowledge the assistance and support St. Cloud State University, in conducting this research.

References

[1] Garfinkel, S. & Spafford, E. (1996) Practical Unix and Internet Security. Sebastopol, O'Reilly and Associates, Inc.,

[2] Howard, J. (2000) "An Analysis of Security Incidents on the Internet 1989-1995". Retrieved February 10,2001 from the World Wide Web: http://www.cert.org/research/JHThesis/Start.html.

[3] McHugh, J.& Christie, A. & Allen, J. (2000). "Defending Yourself: The Role of Intrusion Detection Systems". Retrieved Febuarary 25,2001 from the World Wide Web: http://www.computer.org/software/so2000/pdf//s5042.pdf .

[4] Lawrence Livermore National Laboratory. (1996). "Intrusion Detection and Response". Retrieved Febuarary 20,2001 from the World Wide Web: http://www.all.net/journal/ntb/ids.html.

[5] Abbot, R. (1986). An Integrated Approach to Software Development. NewYork, John Wiley and Sons, Inc.,

[6] Bass, T. (2000) "Intrusion Detection Systems and Multisensor Data Fusion". Communications of the ACM, 43 99-105.

[7] Cohen, F. (1994). A Short Course On Computer Viruses. (2nd ed.) New York, John Wiley and Sons , Inc.,

[8] Allen, J. et al. (1999). "State of the Practice of Intrusion Detection Technologies". Retrieved March 2,2001 from the Word Wide Web: http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028abstract.html.

[9] Guster, D. (2000). "Lecture Notes for MCS 526 (Computer Networking II). St. Cloud State University.

[10] McHugh J, et al. (2001). "Intrusion Detection[1]: Implementation and Operational Issues". Retrieved February 22,2001 from the World Wide Web: http://www.stsc.hill.af.mil/crosstalk/2001/jan/mchugh.asp.

[11] Ranum, M. (1998). "Intrusion Detection: Challenges and Myths". Retrieved February 22,2001 from the World Wide Web: http://secinf.net/info/ids/ids_mythe.html.

[12] Power, R. (2000). Tangled Web: Tales of Digital Crime from the sadows of Cyberspace. Indianapolis, Que,

[13] Cohen, F. (1999). "A Mathematical Structure of Simple Defensive Network Deceptions". Retrieved February 25,2001 from the World Wide Web: http://www.all.net/journal/ntb/mathdeception/mathdeception.html.

[14] Ellison, R. et al. (1997). "Survivable Network Systems: An Emerging Discipline". Retrieved February 25,2001 from the World Wide Web: http://www.cert.org/research/tr13/97tr013title.html.

[15] Linger, R. et al. (1998). "Requirements Definition for Survivable Network Systems". Retrieved February 25,2001 from the World Wide Web: http://www.sei.cmu.edu/organization/programs/nss/icre.html.