ANOMALY-BASED INTRUSION DETECTION

by XXXXXXXX

March 15, 2012

Introduction

There was a time in the not too distant past when information security did not seem to be a major concern for many organizations. Bypassing security systems usually required specialized knowledge, and countermeasures for certain systems were relatively expensive, if they were available at all. Some enterprises were inclined to “roll the dice” and gamble that they would not be compromised. Over the past decade or so, however, as hacker tools became more available to amateurs, as major corporations found themselves the victims of data breaches, and as regulatory requirements from government and professional groups such as the Gramm-Leach-Bliley Act and PCI-DSS placed increasing responsibility on organizations to keep their data safe from unauthorized access, information security has become a much greater part of enterprise systems planning.

Current best practices suggest a layered approach, where someone attempting to compromise a system is forced to defeat multiple levels of protection in order to gain access. For example, a server might require a password to use a console to control the server; a locked door blocks physical access to the console; and an intrusion alarm monitors the door for unauthorized use. If the server has access to a network outside of its immediate physical location, measures should be taken to protect the server from illicit digital access in similar layered fashion to the physical protections. One of the tools available to prevent unauthorized network access is an Intrusion Detection System (IDS) that monitors for signs of attack.

Important Ideas

Information Security researcher Dorothy E. Denning, in one of the early foundational papers in the field of intrusion detection [1] pointed out four reasons for the importance of an intrusion detection system to watch other systems for signs of attack: most systems have vulnerabilities; the user may have operational or financial reasons for not wanting to upgrade to more secure versions; if a decision to upgrade is made, the new system is likely to have its own vulnerabilities; and even the most secure systems are susceptible to privilege abuse. Denning suggested that one way to detect intrusion attempts might be to look for anomalies in the use of the system. Her paradigm would use system logs to build a statistical model of "normal" behavior and then use mathematical analysis to discover outliers in the data that might indicate an attempt to perform unauthorized actions on the system.

This anomaly-based approach offers a major advantage over the signature-based method, which scans for behavior that matches pre-defined patterns: searching for anomalies does not require that the attack be of a type seen before, making it much more likely that a zero-day attack, that is, a new type of exploit, will be discovered, because it does not match normal patterns. Denning's method [1] calculates things like the number of a given type of event that occurs within a certain length of time, then compares this number to events per period counted at other times. From this series of counts can be derived a statistical mean and standard deviation for the sample, which can be used to define the "normal" range, outside of which an event can be considered abnormal, and examined as a possible attack.

Another approach to discovering abnormal behavior deals with state analysis. One of the models used to represent computers is that of a finite-state machine. This is an abstraction that says there are only a certain (finite) number states that a computer can be in. By discovering patterns in the progression of states that lead to an insecure or compromised

states, anomaly-based intrusion detection that uses state analysis can perhaps spot intermediate machine states that result from an intrusion attempt before the machine gets to the compromised state.

In 1995, three researchers with connections to the University of California at Santa Barbara—Koral Ilgun, Richard Kemmerer, and Phillip Porras—suggested using state transitions to create a rule-based approach to intrusion detection [2]. The trio pointed out limitations on rule-based approaches, namely that similar attacks could produce a slightly different series of log entries and go unnoticed, the tendency of rule-based systems to ignore an attack until the compromise is already complete, and the necessity to rely on humans to write complex rules to fire on a given audit input. They suggested that monitoring for the machine states that lead up to the compromise may result in discovery of an attack before it becomes successful.

They gave the example of an attacker who has command-line shell access, but not root access, to a BSD UNIX system, exploiting a vulnerability in a mail application. The vulnerability permits setting up a shell that has root access, but is executable by anyone. The example requires five steps to allow a non-root user to illicitly run commands as root. A traditional rule-based intrusion detection system might detect that a non-root user has run a process that is usually run only by root. The authors maintained however that the actual security violation occurred a few steps before the shell access was created, when the vulnerability permitted a non-root user to make one of root's files world-executable. The state-transition method would be geared to watch for signs of this “setuid” step rather than the final result.

The type of intrusion detection described above is classified as host-based (HIDS), because it examines logs created by the hosts being protected. Another variety is known as network-based intrusion detection (NIDS), which examines the network traffic to look for signs

of attack. In 2002 Austrian researchers Christopher Kruegel, Thomas Toth and Engin Kirda looked at how NIDS algorithms might be improved [3]. Traditional rule-based NIDS systems might look for things like numbers of connections to a given IP address and port (layers 3 and 4 of the OSI model), but other attacks, such as Ping of Death attacks and SYN attacks depend on the data payload inside the TCP segment to deliver the attack (OSI layers 5-7).

Kruegel, Toth, and Kirda suggested building a NIDS made up of a Packet Processing Unit that inspects the payload of network traffic for the service type being requested, and passes the payload and service type to a Statistical Processing Unit, which over time learns the types of payloads found in normal traffic. The analysis could even extend to the characters that make up the payloads of each type, as well as the size of the payload for a given service [3]. Examining payload size, for example, could be valuable in detecting a Ping of Death attack before it reached the target system, whereas a log auditing HIDS probably would only find the attack after the fact, if at all.

In the year following the work mentioned immediately above, Krueger teamed with Giovanni Vigna from the University of California at Santa Barbara to continue the theme of payload analysis, this time dealing specifically with analysis of attacks on web servers. In [4] they noted that the sheer number of web applications available to the public generates a huge number of attacks. At the time of their writing, the signature-based IDS, Snort, devoted almost half of its detection signatures to web attacks.

Krueger and Vigna noted the limitations of signature (rule) based IDS solutions such as constantly expanding vulnerability lists, and recommended that the system also include anomaly-based analysis. They set up a mathematical model to evaluate web traffic, and had the system learn the patterns in legitimate traffic, noting that web queries have a “high regularity in the number, name, and order of parameters.”[4] In the testing phase of developing their system, they discovered that their system would detect the several types of

attacks they looked for, such as buffer overflow and cross-site scripting, with a low rate of false positive alerts.

Polish computer scientists Jaroslav Skarus and Franciszek Seredynski developed a similar study [5] surrounding the problem of protecting web applications against SQL attacks such as SQL injection attacks, where SQL commands are embedded into web forms with the hope that the web application will accept them and process the queries. Skarus and Seredynski proposed a system of parsing SQL queries into tokens and then training Recurrent Neural Networks to predict the structure and sequence of tokens to alert on illegitimate queries.

Several years earlier, researchers Magnus Almgren and Ulf Lindqvist also investigated the idea of developing intrusion detection at the application level as opposed to simply examining operating system logs, writing that "...it is important for the IDS to collect data at the most meaningful abstraction level(s) for the event in question." and referred to their idea as an "application-based IDS"[6]. They mention that a network-based solution is limited by platform considerations, inconsistent methods of dealing with atypical payloads, and the inability to evaluate encrypted traffic. Likewise they point out that host-based approaches where multiple hosts are monitored have administrative and performance costs and because they use logs as input, the events have already occurred as they are being evaluated, and the system cannot act proactively to stop attacks.

Almgren and Lindqvist believed these limitations could be surmounted by an IDS that operates within the application itself. They developed an Apache web server module that had access to all data being processed by the web application, and found that on average it only delayed delivery of a test web page by less than a tenth of a second. [6]

Common to most of the literature in the field of anomaly-based intrusion detection is the problem of developing patterns of normal usage. Most systems create vast amounts of

data that an IDS must comb through to build models to compare traffic with. Attackers that are more interested in data theft than destructive attacks usually do their best not to be detected, so their actions are likely to be low-profile and sparsely generated rather than aggressive and easily noticed. What if surreptitious actions simply get lost in the noise of hundreds of thousands of other legitimate actions?

Purdue University researchers Terran Lane and Carla E Brodley tackled the issue in [7] by examining user actions as opposed to system or application activities. Like Skarus and Seredynski [5], they parsed and tokenized input. They then used evaluation methods such as k-nearest-neighbor to develop sequences that can be expected in normal behavior. This information was used to develop user behavior profiles that not only could be used to detect a low-profile attack, but also spot illicit activity by legitimate users that went against their normal patterns of usage.

Wenke Lee from Georgia Institute of Technology and Salvatore J. Stolfo of Columbia University looked at the issue of extensibility in [8]. They realized that accuracy was important, but their interest was in developing a strategy for developing new IDS solutions in a more automated fashion. They used data mining techniques to find tipping points within data sets—numerical points at which the likelihood of an action being illicit increases dramatically, finding patterns such as the number of bad logins that tends to indicate a guessing attempt rather than someone simply typing their password incorrectly. These patterns were then used to extend the monitoring capabilities of the system to incorporate newly discovered anomalies.

Three computer scientists from California, Calvin Ko, Manfred Ruschitzka, and Karl Levitt in [9] examined the idea of spotting undesirable behavior in the use of applications by developing a specification within the application, using a grammar that defines the legitimate activities of the program. They used the same example as Ilgun, Kemmerer, and Porras [2] of the flaw that permitted a non-root user to use setuid to execute root-only functions. Their

approach,however, instead of watching the command line history logs for bad activity, used a per-system agent approach to monitor actions in real time. The agents were useful when monitoring distributed applications because they could report to a single system to watch the application's activity as a whole. They developed trace policies from combinations of process ID, user ID, program, and host, which instructed the agents on what to monitor.

Conclusion

Anomaly-based intrusion detection may hold the greatest promise for making systems more secure by discovering attempts to use the system in illegitimate ways. Its ability to spot abnormal behavior without prior knowledge of the vulnerability makes it much better at detecting zero-day attacks than signature-based systems that must wait for the vulnerability to be discovered, then wait for a new signature to detect that attack method to be created by developers, then download and install the signature, during which time the host being monitored may already be compromised. Using algorithms to calculate rolling means and standard deviations and thus automatically and continually revising the definition of “normal” behavior gives the system a “learning” feature that can account for legitimate changes in the way the system is used while helping to reduce false reports of attacks.

Unfortunately, many attackers are responding to event-counting by dropping the rate at which they perform actions designed to compromise systems. This causes their actions in many cases to be lost in the statistical noise and not seen as anomalous because of the low numbers of suspicious actions per time period. In this case more extensive data mining methods are required to define patterns

Future development of IDS systems will probably make use of a combination of signature-based and anomaly-based detection, using certain features of host-based IDS as well as network-based and application-based systems.

As attacks get more sophisticated, signatures may be developed for well-known attack vectors but it is probable that anomaly-based systems will be the front line of defense against unauthorized actions, both from outside and from within.

References

1. D. E. Denning, “An Intrusion-Detection Model,” *IEEE Transactions on Software Engineering,* vol. 13, (2), pp. 222-232, February 1987.

2. K. Ilgun, R. A. Kemmerer, and P. A. Porras, “State Transition Analysis: A Rule-Based Intrusion Detection Approach,” *IEEE Transactions on Software Engineering,* vol. 21, (3), pp. 181-199, March 1995.

3. C. Kruegel, T. Toth, and E. Kirda, “Service specific anomaly detection for network intrusion detection,” in *Proceedings of the 2002 ACM Symposium on Applied Computing*, New York, NY, USA: ACM, 2002, pp. 201-208.

4. C. Kruegel and G. Vigna, “Anomaly detection of web-based attacks,” In *Proceedings of the 10th ACM conference on Computer and Communications Security,* New York, NY, USA: ACM, 2003, pp. 251-261.

5. J. Skaruz and F. Seredynski, “Some Issues on Intrusion Detection in Web Applications,” in *Proceedings of the 9th International Conference on Artificial Intelligence and Soft Computing*, Leszek Rutkowski, Ryszard Tadeusiewicz, Lotfi A. Zadeh, and Jacek M. Zurada, Ed. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 164-174.

6. M. Almgren and U. Lindqvist, “Application-Integrated Data Collection for Security Monitoring,” in *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, Wenke Lee, Ludovic Me, and Andreas Wespi, Ed. London, UK: Springer-Verlag, 2001, pp. 22-36.

7. T. Lane and C. E. Brodley, “Temporal Sequence Learning and Data Reduction for Anomaly Detection,” in *Proceedings of the 5th ACM Conference on Computer and Communications Security, New York, NY, USA: ACM, 1*998, pp. 150-158.

8. W. Lee and S. J. Stolfo, “A Framework for Constructing Features and Models for Intrusion Detection Systems,” *ACM Transactions on Information and System Security*, vol. 3, (4), pp. 227-261, November 2000.

9. C. Ko, M. Ruschitzka, and K. N. Levitt. “Execution Monitoring of Security-Critical Programs in Distributed Systems: a Specification-Based Approach,” in *Proceedings of the 1997 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 1997, pp. 175-.