Interested in learning more about security?

SANS Institute InfoSec Reading Room

Protocol Anomaly Detection for Network-based Intrusion Detection

A taxonomy was developed by Axelsson to define the space of intrusion detection technology and classify IDSs. The taxonomy categorizes IDSs by their detection principle and their operational aspects. The two main categories of detection principles are signature detection and anomaly detection. The remainder of this paper will compare the two categories of detection principles and describe a new type of anomaly detection based on protocol standards. While the taxonomy applies to both host-based and network-based IDSs, t...

Protocol Anomaly Detection for Network-based Intrusion Detection

Kumar Das
GSEC Practical Assignment Version 1.2f (amended August, 13, 2001)

Introduction

Intrusion detection has become an essential component of computer security in recent years. Security administrators are complementing existing security measures with intrusion detection systems (IDSs) to achieve defense in depth [1]. To be useful, an IDS must be selected and configured with its environment in mind, and it must be monitored by knowledgeable personnel. An effective IDS provides accurate and timely information about ongoing intrusions which is necessary to protect today's networks.

The role of an IDS is to warn administrators of suspicious and potentially malicious computer activity. Most IDSs are passive warning devices that must be monitored by trained professionals. Some IDSs, however, feature active response mechanisms to automate the intrusion recovery process. Of the active response IDSs, some target the attacker and launch countermeasures to inhibit the attacker system. The difficulty with launching countermeasures is ensuring the authenticity of the attacker. Hackers often spoof IP addresses or use intermediary hosts during a break-in. Disabling an alleged attacking system in such a scenario disrupts an innocent machine and leaves the real attacker system unaffected. Another approach to active response is to target the victim system. Typical victim responses attempt to repair the system, stop the current attack, or prevent the machine from being exploited in the future. Unfortunately, measures taken against the victim system are not always effective because it is difficult to determine the success of an attack and the resulting damage. Incorrect responses may have ill effects on the victim system. Regardless, repair takes a machine offline which is costly for production systems. The biggest problem with active response mechanisms, however, is the current state of intrusion detection technology.

Measuring IDSs

Current IDSs generate too many inaccurate alarms. Acting automatically on such alarms is very dangerous. Simply stated, IDSs aren't good enough yet. Worse yet, the concept of "good" isn't well defined for the intrusion detection problem. There are many factors to consider when evaluating IDSs such as speed, cost, effectiveness, ease-of-use, scalability, and interoperability. Without taking specific environment details into consideration, effectiveness and ease-of-use can be used as general metrics to compare IDSs. Both factors measure general aptitude because they are determined by the detection algorithm of the IDS.

Intrusion detection systems use sensors to collect data which is processed into events. Sensors can process information from a variety of sources including network taps, syslog records, and audit records. Sensor-generated events are then processed by a detection algorithm that determines whether an event corresponds to an attack. In some IDSs, the algorithm does not perform a binary classification into the categories "attack" or "not attack," but assigns a n-valued

variable to describe event severity. For such IDSs, extra operator labor is required to determine if an event is malicious based on the type of the event and the severity.

The detection algorithm maps incoming events to attacks and normal activity. The resulting classification can be used to determine the effectiveness of an IDS. Effectiveness, as defined in [2], is the ability of an IDS to maximize the detection rate while minimizing the false alarm rate (false positive rate). In other words, a good IDS reports intrusions when they occur, and does not report intrusions when they do not occur. Stefan Axelsson analyzed the intrusion detection problem with Bayesian statistics and determined that the base-rate fallacy governs the effectiveness of IDSs. The basic rate of incidence of an event, or the base-rate, is often overlooked when calculating conditional probabilities. Simply stated in terms of intrusion detection, the probability that an intrusion is actually occurring, given that an IDS reports an intrusion, is dominated by the false alarm rate of the IDS. The important measure of an IDS is not how frequently it detects attacks, but how infrequently it produces false alarms. He also concluded that high false alarm rates found in modern IDSs show that none the products he surveyed lived up to real-world expectations of effectiveness.

Given that intrusion detection is still a relatively new field, it is acceptable that none of the products available today are extremely effective. Systems that are somewhat effective at determining attacks are still useful, especially because of the computing power required to do intrusion detection that makes it impossible to perform manually. Another important factor for measuring IDSs is its ease-of-use. Because active response is not yet an acceptable technology, human intervention is necessary to use IDSs. It is therefore necessary for IDSs to be intuitive and easy to manage. Alarms from an IDS must be investigated by a security officer to separate the real threats from the false alarms. The less false alarms an IDS generates, the easier it is for an operator to find the real intrusions in his network. In terms of false alarms, ease-of-use and effectiveness are closely related. However, ease-of-use also includes the user interface, interoperability with other products, reporting capabilities, and investigation capabilities.

IDS Classification

A taxonomy was developed in [3] to define the space of intrusion detection technology and classify IDSs. The taxonomy categorizes IDSs by their detection principle and their operational aspects. The two main categories of detection principles are signature detection and anomaly detection. The remainder of this paper will compare the two categories of detection principles and describe a new type of anomaly detection based on protocol standards. While the taxonomy applies to both host-based and network-based IDSs, this paper will focus on network-based IDSs because protocol anomaly detection is unique to analyzing network traffic.

Signature detection, also known as misuse detection, attempts to identify events that misuse a system. Signature detection is achieved by creating models of intrusions. Incoming events are compared against intrusion models to make a detection decision. When creating signatures, the model must detect an attack without any knowledge of normal traffic in the system. Attacks and only attacks should match the model, otherwise false alarms will be generated. The simplest form of misuse detection uses simple pattern matching to compare network packets against binary signatures of known attacks. A binary signature may be defined for a specific portion of

the packet, such as the TCP flags. For instance, an attack signature for the land attack [4] would match packets that had the SYN flag set and had the same source and destination IP. The remaining content of the packet is irrelevant.

The signature detection method is good at detecting known attacks. A well crafted signature will always detect the attack it represents. However, other packets may match the signature and generate false alarms. Signature systems are typically easily customizable and knowledgeable users can create their own signatures. Poorly formed signatures, however, are dangerous because they cause false alarms. Another problem with signature detection is the large number of signatures required to effectively detect misuse. Additional signatures detect more attacks but also raise the probability that normal traffic will be incorrectly matched to a signature, thereby reducing the system's effectiveness. In practice, increasing the signature database reduces the bandwidth of a signature-based IDS. Each packet must be compared against many signatures and each comparison uses computational resources. At some point, when a signature database grows too large, not all packets can be processed and some will be dropped, which also reduces the detection rate and the overall effectiveness. With a constraint on bandwidth, operators are forced to select only those signatures which are most important for their network. In addition, new signatures are being developed more rapidly than new attacks are discovered. Recent polymorphic attacks, such as ADMmutate [5] create the need for multiple signatures for a single attack. Multi-functional attacks such as Nimda [6] also require multiple signatures. Changing a single bit in some attack strings can invalidate a signature and create the need for an entirely new signature. It is difficult for operators to incorporate such a multitude of new signatures into their performance-limited signature databases. Despite such problems with signature-based intrusion detection, they are very popular and work well in practice when configured correctly and monitored frequently.

Anomaly detection, or not-use detection, differs from signature detection in the subject of the model. Instead of modeling intrusions, anomaly detectors create a model of normal "use" and look for activity that does not conform. Deviations are labeled as attacks because they do not fit the "use" model, thus the name, not-use detection. The difficulty in creating an anomaly detector is creating the model of normal "use." The traditional method, called statistical or behavioral anomaly detection, selects key statistics about network traffic as features for a model trained to recognize normal activity. Unfortunately, live networks have very little invariants and are anything but normal. Vern Paxson did a study of internet traffic in [7] and concluded that statistics such as packet arrival times, connection arrival times, and website hits have much variation. Too much statistical variation makes models inaccurate and events classified as anomalies may not always be malicious. For example, a company's employees might return to their desks and check e-mail immediately following a company-wide meeting. The resulting spike in SMTP activity is not normal for that time of the day or week but is not necessarily a denial of service attempt against the mail server either, as a statistical anomaly detector might label it. Another problem with this approach is the inability to train a model on a completely "normal" network. Anomaly detection models must be trained on the specific network to be monitored. It is naive to assume that a network with a connection to the Internet is clean when the anomaly detector is being trained. If a burglar breaks into a house and leaves everything as he found it, how will the owners know he was there? An administrator cannot be sure his network is clean simply because it appears "normal." Traditionally, anomaly detection systems

aren't as popular as signature detection systems because of high false alarm rates created by inaccurate models of normal "use."

Protocol Anomaly Detection

A new variant of anomaly detection has been incorporated into IDSs in recent years. Instead of training models on normal behavior, protocol anomaly detectors build models of TCP/IP protocols using their specifications [8,9]. Statistical anomaly detection is plagued by the inability to create a normal model of network traffic statistics. Protocol anomaly detection is much easier, however, because protocols are well defined and a normal "use" model can be created with greater accuracy. Protocols are created with specifications, known as RFCs, to dictate proper use and communication. All connection oriented protocols have state. Certain events must take place at certain times. As a result, many protocol anomaly detectors are built as state machines. Each state corresponds to a part of the connection, such as a server waiting for a response from a client. The transitions between the states describe the legal and expected changes between states. For example, legal transitions from the "server waiting" state are the "client sends data" state, "client cuts connection" state, or the "server timeout" state.

While the specifications in RFCs are not always complete, they give a good starting point for building a model. One common addition while building a "use" model is to allow for accepted deviations from the RFCs [8,9]. For instance, not all software applications are created with the rules of a protocol in mind. Microsoft may not follow all of the SMTP standards when creating a mail server. However, if the mail server works, and non-malicious traffic for the mail server is frequently seen on the network, it is acceptable to program the model to recognize such traffic as normal.

Most of the benefits of protocol anomaly detection come from the simplicity and elegance of the "use" method of detection. It is much simpler to model the correct use of a protocol than to model the misuse of a protocol. New attack methods and exploits are constantly being discovered, many of which violate protocol standards. The space of malicious attack signatures is growing at an incredible rate. As such, attack signature databases must be updated frequently to effectively detect attacks. In comparison, new protocols and extensions to existing protocols are being developed at a much slower rate. The space of network protocols is well-defined and changing slowly. Protocol anomaly detectors are able to detect most new attacks without being updated because the new attacks deviate from the protocol specifications.

The elegance of the "use" model of detection has other benefits as well. Protocol anomaly detection systems are easier to use because they require no signature updates. Occasionally, new protocols will be developed or new protocol extensions will become popular and it is necessary to update or add new protocol state machines to an IDS. However, the frequency of such updates is much less than the frequency of current attack signature updates. Protocol anomaly detectors also differ from traditional IDSs in how they present alarms. Because attacks are detected by deviations from the "use" model, the best way to present such information is to describe what part of the state machine was violated. To interpret this type of information requires in-depth knowledge of protocol design which is unrealistic of typical IDS operators. It is therefore necessary for such IDSs to provide documentation to help operators interpret alarms.

It is also helpful for the IDS to map well-known attacks to their protocol violations. Despite the difficulty of use, protocol anomaly detection systems are essential for understanding new attacks. Without names and prior documentation, new attacks can only be defended against by understanding their intrusion methods and their effects. Protocol violation information is crucial to determining the intrusion method. Another benefit of protocol anomaly detection is increased efficiency. Well designed anomaly detectors use less rules to describe acceptable behavior than signature detectors use to describe the numerous malicious behaviors. Fewer rules to check each event against increases the bandwidth of an IDS which increases its detection rate and effectiveness.

Protocol Anomaly Detection Example: Nimda

Analyzing a portion the Nimda attack will help illustrate the principle of protocol anomaly detection. On September 18th the Internet was infected with Nimda, a virus and a worm that proliferated networks and denied service to many sites. Nimda, as described in [6] exploited holes in multiple Microsoft products including IIS and IE. Once a machine became infected, Nimda attempted to spread itself by probing other computers with multiple attack mechanisms. Some of the attempts targeted IIS web servers with the Extended Unicode Directory Traversal Vulnerability [10]. Two of the sixteen Nimda web probes are shown below:

GET /scripts/..**%c0%af**../winnt/system32/cmd.exe?/c+dir
GET /scripts/..**%c1%9c**../winnt/system32/cmd.exe?/c+dir

Both of the probes exploit the Directory Traversal Vulnerability using overlong UTF-8 characters. IIS incorrectly interprets the Uniform Resource Identifiers (URIs) above and translates the highlighted UTF-8 encoded characters to "/" and "\" respectively. Once decoded, the URIs allow remote access to cmd.exe on the local machine. The URI specification [11] allows the use of escaped characters when interpreting URIs. However, as noted in the Unicode Standard [12], applications should only interpret "shortest from" unicode strings. Neither of the strings above are the shortest forms of the characters they represent.

A well built anomaly detection engine would include a model for HTTP connections. The HTTP protocol design has been supported by numerous RFCs. Part of the protocol specification allows for encoded URIs, however, unicode encoded URIs are governed by the Unicode Standard. In the HTTP model, when the connection is in the "client request" state, the detection engine would validate the transition to the "server send" state. During validation, the rules built from the Unicode and URI standards would indicate that the requests deviated from the model because of imbedded overlong UTF-8 characters. At the outbreak of Nimda, when it proliferated the quickest, signature detection systems were unable to detect the attack because the signature did not exist. Some protocol anomaly detection systems, however, were able to detect the attack and provide operators early warning and information about the attack required for further investigation. Such educated operators were able to defend their networks against Nimda before other operators even knew Nimda existed.

Conclusion

Protocol anomaly detection is a new variant on the existing IDS technology of anomaly detection. Protocol anomaly detection improves on signature detection by modeling the smaller, more well-defined space of "use" instead of "misuse." It also eliminates the need for frequent signature updates and provides enhanced performance. Traditional anomaly detectors have been mostly ineffective because of the inability to create an accurate "use" model. Protocol anomaly detection solves this problem by placing restrictions on what is being modeled. Statistical anomaly detection systems attempt to model erratic network statistics, whereas protocol anomaly detection systems model well-defined protocol standards.

Protocol anomaly detection is not a perfect generalized solution to the intrusion detection problem, however. In restricting the nature of the model, many attacks are detected because of intrinsic protocol violations. However, some attacks, such as some viruses, conform to protocol standards and are undetectable by protocol anomaly detectors. In addition, attacks will remain undetected that are encrypted over the network or visible only to host-based IDSs. To increase the quantity of attacks detected, security administrators must exercise defense in depth with their IDSs. Multiple intrusion detection principles must be employed together. Not until multiple intrusion detection technologies can be correlated and analyzed together will security administrators have an accurate view of their network using intrusion detection.

References

[1] J. Allen, A. Christie, W. Fithen, et al., "State of the Practice of Intrusion Detection Technologies," CMU/SEI-99-TR-028, http://www.sei.cmu.edu/publications/documents/99.reports/99tr028/99tr028abstract.html.

[2] S. Axelsson, "The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection," in Proceedings of the 6th ACM Conference on Computer and Communications Security, pp. 1-7, November 1-4, 1999, Kent Ridge Digital Labs, Singapore, Copyright ACM 1999, http://www.ce.chalmers.se/staff/sax/difficulty.pdf.

[3] S. Axelsson, "Intrusion Detection Systems: A Taxonomy and Survey," Technical Report No 99-15, Dept. of Computer Engineering, Chalmers University of Technology, Sweden, March 2000, http://www.ce.chalmers.se/staff/sax/taxonomy.ps.

[4] CERT Advisory CA-1997-28 IP Denial-of-Service Attacks, http://www.cert.org/advisories/CA-1997-28.html.

[5] ADMmutate 0.8.4 by K2, http://www.ktwo.ca/readme.html.

[6] NIMDA Worm/Virus Report – Final, http://www.incidents.org/react/nimda.pdf.

[7] V. Paxson, "Why Understanding Anything About The Internet Is Painfully Hard," UCB Berkeley MIG Seminar, April 1999, http://www.icir.org/vern/talks/vp-painfully-hard.UCB-mig.99.ps.gz.

[8] E. Lemonnier, “Protocol Anomaly Detection in Network-based IDSs,” June 2001, http://erwan.lemonnier.free.fr/exjobb/report/protocol_anomaly_detection.pdf.

[9] Beetle, Sasha, “A Strict Anomoly Detection Model for IDS,” Phrack, vol. 10, issue 56, May 2000, http://www.opennet.ru/base/sec/p56-0x0b.txt.html.

[10] Nimda Worm Analysis, http://aris.securityfocus.com/alerts/nimda/010919-Analysis-Nimda.pdf.

[11] T. Berners-Lee, R. Fielding, L. Masinter, “Uniform Resource Identifiers (URI): Generic Syntax,” RFC 2396, August 1998, http://www.ietf.org/rfc/rfc2396.txt.

[12] Corrigendum to Unicode 3.0.1, http://www.unicode.org/unicode/uni2errata/UTF-8_Corrigendum.html.

Last Updated: August 24th, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS San Francisco Fall 2017	OnlineCAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced