Most security experts agree that a completely secure system is impossible to achieve. So we must stay alert for attacks.

Intrusion Detection: A Brief History and Overview

Richard A. Kemmerer and Giovanni Vigna

Reliable Software Group, Computer Science Department, University of California Santa Barbara

Suppose a strange man is standing in front of your house. He looks around, studying the surroundings, and then goes to the front door and starts turning the knob. The door is locked. He moves to a nearby window and gently tries to open it. It, too, is locked. It seems your house is secure. So why install an alarm?

This question is often asked of intrusion detection advocates. Why bother detecting intrusions if you've installed firewalls, patched operating systems, and checked passwords for soundness? The answer is simple: because intrusions still occur. Just as people sometimes forget to lock a window, for example, they sometimes forget to correctly update a firewall's rule set.

Even with the most advanced protection, computer systems are still not 100 percent secure. In fact, most computer security experts agree that, given user-desired features such as network connectivity, we'll never achieve the goal of a completely secure system. As a result, we must develop intrusion detection techniques and systems to discover and react to computer attacks.

Intrusion detection: A brief history

Originally, system administrators performed intrusion detection by sitting in front of a console and monitoring user activities. They might detect intrusions by noticing, for example, that a vacationing user is logged in locally or that a seldom-used printer is unusually active. Although effective enough at the time, this early form of intrusion detection was ad hoc and not scalable.

The next step in intrusion detection involved audit logs, which system administrators reviewed for evidence of unusual or malicious behavior. In the late '70s and early '80s, administrators typically printed audit logs on fan-folded paper, which were often stacked four- to five-feet high by the end of an average week. Searching through such a stack was obviously very time consuming. With this overabundance of information and only manual analysis, administrators mainly used audit logs as a forensic tool to determine the cause of a particular security incident after the fact. There was little hope of catching an attack in progress.

As storage became cheaper, audit logs moved online and researchers developed programs to analyze the data.[1] However, analysis was slow and often computationally intensive, and, therefore, intrusion detection programs were usually run at night when the system's user load was low. Therefore, most intrusions were still detected after they occurred.

In the early '90s, researchers developed real-time intrusion detection systems that reviewed audit data as it was produced. This enabled the detection of attacks and attempted attacks as they occurred, which in turn allowed for real-time response, and, in some cases, attack preemption.

More recent intrusion detection efforts have centered on developing products that users can effectively deploy in large networks. This is no easy task, given increasing security concerns, countless new attack techniques, and continuous changes in the surrounding computing environment.

Intrusion detection overview

The goal of intrusion detection is seemingly simple: to detect intrusions. However, the task is difficult, and in fact intrusion detection systems do not detect intrusions at all—they only identify evidence of intrusions, either while they're in progress or after the fact.

Such evidence is sometimes referred to as an attack's "manifestation." If there is no manifestation, if the manifestation lacks sufficient information, or if the information it contains is untrustworthy, then the system cannot detect the intrusion.

For example, suppose a house monitoring system is analyzing camera output that shows a person fiddling with the front door. The camera's video data is the manifestation of the occurring intrusion. If the camera lens is dirty or out of focus, the system will be unable to determine whether the person is a burglar or the owner.

Data collection issues

For accurate intrusion detection, we must have reliable and complete data about the target system's activities. Reliable data collection is a complex issue in itself. Most operating systems offer some form of auditing that provides an operations log for different users. These logs might be limited to the security-relevant events (such as failed login attempts) or they might offer a complete report on every system call invoked by every process. Similarly, routers and firewalls provide event logs for network

activity. These logs might contain simple information, such as network connection openings and closings, or a complete record of every packet that appeared on the wire.

The amount of system activity information a system collects is a trade-off between overhead and effectiveness. A system that records every action in detail could have substantially degraded performance and require enormous disk storage. For example, collecting a complete log of a 100-Mbit Ethernet link's network packets could require hundreds of Gbytes per day.

Collecting information is expensive, and collecting the right information is important. Determining what information to log and where to collect it is an open problem. For example, having your house alarm system monitor the water for pollution levels is an expensive activity that doesn't help detect burglars. On the other hand, if the house's threat model includes terrorist attacks, monitoring the pollution level might be reasonable.

It's now time for operating systems and network software to integrate intrusion detection sensors.

Detection techniques

Auditing your system is useless if you don't analyze the resulting information. How intrusion detection systems analyze collected data is an important system characteristic.

There are two basic categories of intrusion detection techniques: anomaly detection and misuse detection.

Anomaly detection. Anomaly detection uses models of the intended behavior of users and applications, interpreting deviations from this "normal" behavior as a problem.[2-4]

A basic assumption of anomaly detection is that attacks differ from normal behavior. For example, we can model certain users' daily activity (type and amount) quite precisely. Suppose a particular user typically logs in around 10 a.m., reads mail, performs database transactions, takes a break between noon and 1 p.m., has very few file access errors, and so on. If the system notices that this same user logs in at 3 a.m., starts using compilers and debugging tools, and has numerous file access errors, it will flag this activity as suspicious.

The main advantage of anomaly detection systems is that they can detect previously unknown attacks. By defining what's normal, they can identify any violation, whether it is part of the threat model or not. In actual systems, however, the advantage of detecting previously unknown attacks is paid for in terms of high false-positive rates. Anomaly detection systems are also difficult to train in highly dynamic environments.

Misuse detection. Misuse detection systems essentially define what's wrong. They contain attack descriptions (or "signatures") and match them against the audit data stream, looking for evidence of known attacks.[5-7] One such attack, for example, would occur if someone created a symbolic link to a Unix system's password file and executed a privileged application that accesses the symbolic link. In this example, the attack exploits the lack of file access checks.

The main advantage of misuse detection systems is that they focus analysis on the audit data and typically produce few false positives.

The main disadvantage of misuse detection systems is that they can detect only known attacks for which they have a defined signature. As new attacks are discovered, developers must model and add them to the signature database.

Response: After the intrusion

An intrusion detection system's *response* is its output or action upon detecting a problem. A response can take many different forms; the most common is to generate an alert that describes the detected intrusion. There are also more aggressive responses, such as paging a system administrator, sounding a siren, or even mounting a counter-attack.

A counterattack might include reconfiguring a router to block the attacker's address or even attacking the culprit. Obviously, aggressive responses can be dangerous, since they could be launched against innocent victims. For example, a hacker can attack a network using *spoofed traffic*—traffic that appears to come from a certain address, but that is actually generated elsewhere. If the intrusion detection system detected the attack and reconfigured the network routers to block traffic from that address, it would effectively be executing a denial-of-service attack against the impersonated site.

Open issues

Although intrusion detection has evolved rapidly in the past few years, many important issues remain. First, detection systems must be more effective, detecting a wider range of attacks with fewer false positives. Second, intrusion detection must keep pace with modern networks' increased size, speed, and dynamics. Finally, we need analysis techniques that support the identification of attacks against whole networks.

System effectiveness

The challenge for increased system effectiveness is to develop a system that detects close to 100 percent of attacks with minimal false positives. We are still far from achieving this goal.

Today's intrusion detection systems primarily rely on misuse detection techniques. The freely available Snort[8] (www.snort.org) and the commercially available RealSecure (www.iss.net) are two products that use signatures to analyze network traffic. Because they model only known attacks, developers must regularly update their signature sets. This approach is insufficient. We need anomaly detection's ability to detect new attacks, but without the approach's accompanying high rate of false positives. Many researchers advocate using a hybrid misuse–anomaly detection approach, but further investigation is needed.[9]

Performance

Simply detecting a variety of attacks is not enough. Intrusion detection systems must also keep up with the input-event stream generated by high-speed networks and high-performance network nodes.

Gigabit Ethernet is common, and fast optical links are becoming popular. The network nodes are also getting faster, processing more data and generating more audit logs. This takes us back to the historical problem of a system administrator confronting a mountain of data. There are two ways to analyze this amount of information in real-time: split the event stream or use peripheral network sensors.

In the first approach, a "slicer" component splits the event stream into slimmer, more manageable streams that the intrusion detection sensors can analyze in real-time. To do this, the whole event stream must be accessible at a single location. Therefore, researchers typically advocate stream splitting for centralized systems or network gateways.

The problem with this approach is that the slicer must divide the event stream in a way that guarantees the detection of all relevant attack scenarios. If an event stream is divided randomly, sensors might not receive sufficient data to detect an intrusion, because different parts of the attack manifestation might be assigned to different slices.

A second approach is to deploy multiple sensors at the network periphery, close to the hosts the system must protect. This approach assumes that by moving the analysis to the network's periphery, a natural partitioning of traffic will occur.

The problem with this approach is that it's difficult to deploy and manage a highly distributed set of sensors. First of all, correct sensor positioning can be difficult to do. Attacks that depend on the network topology, such as routing- and spoofing-based attacks, require that detection sensors be placed in a specific position in the network. Second, there is a control-and-coordination issue. Networks are dynamic entities that evolve through time, and the threats evolve, too. New attacks are invented every day; the sensing infrastructure must evolve accordingly.

Network-wide analysis

Placing sensors at critical network locations lets administrators detect attacks against the network as a whole. That is, the sensing network is able to provide an integrated, "big picture" view of the network security status. Attacks that might appear irrelevant in the context of a single host might be extremely dangerous when considered across the network.

Consider, for example, an attack that involves multiple steps. Suppose each step is carried out on a different host, but because the system under attack has a shared file system, the effects are evident throughout the network. The system might not identify an individual step as malicious when analyzing a single sensor's information, but a more comprehensive analysis of network activity could reveal the attack pattern. This alert correlation or fusion—identifying intrusion patterns based on different sensor alerts—is one of the most challenging problems in intrusion detection today.

Even as networks become more secure, intrusion detection will always be an integral part of any serious security solution. The current trend to distribute and specialize sensors will result in systems composed of hundreds, possibly thousands, of intrusion detection sensors connected by an infrastructure that supports communication, control, and reconfiguration. Although the infrastructure type and characteristics might vary, all will have to be able to scale up to large numbers.[10] Also, analysis will gradually shift its focus from low-level sensors to high-level analyzers that will give administrators a better, more concise picture of the entire network's important security events.

In the near future, sensor technology will be integrated into our everyday computing environment. We've seen something similar with firewalls, which are now an integral part of operating systems: both Unix and Windows provide some form of host-based firewalling. It's now time for operating systems and network software to integrate intrusion detection sensors. Intrusion detection will no doubt become a default feature, rather than an esoteric option.

That said, a pervasive, ubiquitous sensor network can be deployed only if we can integrate different types of sensors running on different platforms, environments, and systems. We thus need standards that will support interoperability. A first step in this direction is the current Intrusion Detection Message Exchange Format standard proposed by the Internet Engineering Task Force's Intrusion Detection Working Group.[11] IDMEF defines the format of alerts and an alert exchange protocol. Additional effort is needed to provide a common ontology that lets sensors agree on what they see. Without this common way of describing the involved entities, sensors will continue to disagree when detecting the same intrusion.

Pushing the evolution even further, software-based intrusion detection might evolve into hardware-based sensing technology. New types of pervasive sensors might also open new directions for intrusion detection. Perhaps in the future, our sensor-woven clothing will be capable of detecting pickpockets; the possibilities are intriguing, if not endless.

References

1. J.P. Anderson, *Computer Security Threat Monitoring and Surveillance*, James P. Anderson Co., Fort Washington, Pa. 1980.
2. D.E. Denning, "An Intrusion Detection Model," *IEEE Trans. Software Eng.*, vol. 13, no. 2, Feb. 1987, pp. 222–232.
3. C. Ko, M. Ruschitzka, and K. Levitt, "Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-Based Approach," *Proc. 1997 IEEE Symp. Security and Privacy*, IEEE CS Press, Los Alamitos, Calif., 1997, pp. 175–187.
4. A.K. Ghosh, J. Wanken, and F. Charron, "Detecting Anomalous and Unknown Intrusions Against Programs," *Proc. Annual Computer Security Application Conference (ACSAC'98)*, IEEE CS Press, Los Alamitos, Calif., 1998, pp. 259–267.
5. K. Ilgun, R.A. Kemmerer, and P.A. Porras, "State Transition Analysis: A Rule-Based Intrusion Detection System," *IEEE Trans. Software Eng.* vol. 21, no. 3, Mar. 1995, pp. 181–199.
6. V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," *Proc. Seventh Usenix Security Symp.*, Usenix Assoc., Berkeley, Calif., 1998.
7. U. Lindqvist and P.A. Porras, "Detecting Computer and Network Misuse with the Production-Based Expert System Toolset," *IEEE Symp. Security and Privacy*, IEEE CS Press, Los Alamitos, Calif., 1999, pp. 146–161.
8. M. Roesch, "Snort—Lightweight Intrusion Detection for Networks," *Proc. Usenix Lisa '99 Conf.*, Usenix Assoc., Berkeley, Calif., 1999.

9. R. Bace and P. Mell, "Special Publication on Intrusion Detection Systems," Tech. Report SP 800-31, National Institute of Standards and Technology, Gaithersburg, Md., Nov. 2001.

10. G. Vigna, R.A. Kemmerer, and P. Blix, "Designing a Web of Highly Configurable Intrusion Detection Sensors," *Proc. Fourth Int'l Symp. Recent Advances in Intrusion Detection (RAID 2001)*, Lecture Notes in Computer Science, vol. 2212, Springer Verlag, New York, 2001, pp. 69–84.

11. D. Curry and H. Debar, "Intrusion Detection Message Exchange Format: Extensible Markup Language (XML) Document Type Definition," http://www.ietf.org/internet-drafts/ draft-ietf-idwg-idmef-xml-06.txt, Dec. 2001.

Richard A. Kemmerer is a professor in and past chair of the Department of Computer Science at the University of California, Santa Barbara. His research interests include computer security, formal specification and verification, and software engineering. Contact him at kemm@cs.ucsb.edu.

Giovanni Vigna is an assistant professor in the Department of Computer Science at the University of California, Santa Barbara. His research interests include network and computer security, intrusion detection systems, security of mobile code systems, penetration testing, and distributed systems. Contact him at vigna@cs.ucsb.edu.

Research vs. Practical Security, *continued from pg. 32*

of these prove the resistance of the cryptosystem to attack. Scholarly research into provably secure cryptosystems aims to develop techniques, and cryptosystems, that provide such assurance. The emphasis here is on proof, not opinion.

Scholarly research explores new technologies as well. The goal is to go beyond existing technologies to find new ways and new mechanisms to improve the state of computer security. The Eros operating system is a good example.[4] Eros is a capability-based system exploring an unusual approach to building systems designed with security in mind. While capability-based systems are not new, current operating system technology generally focuses on access control list technology. So Eros is exploring an approach that could prove fruitful. The approach could fail. But, whether it succeeds or fails, our knowledge and understanding of protection and the technologies that support it will increase. This is the mark of a scholarly research project.

The drawback of this type of research lies in its uncertainty and its long range. Scholarly research explores new avenues of ideas and principles, and any particular research project might not produce useful new results. In some cases, the usefulness of the results might not be apparent for years, decades, or even centuries. But in other cases, the usefulness could be obvious; intrusion detection systems, first proposed in 1986,[5] were being marketed by 1989 and are now a very popular technology). All this work uncovers material that can be incorporated into scholarly education to foster a deeper understanding of computer security.

A common perception is that training is superior to (or inferior to) scholarly education. The truth of this perception depends on the education's purpose. If the goal is to train someone how to use specific systems, work in specific environments, or perform specific tasks, training will achieve this goal more quickly than scholarly education. If the goal is to train someone to understand the general principles and ideas underlying a subject, or a technology, scholarly education will achieve this goal more quickly than training. But each form of learning enhances the other. A student of scholarly education sees how current technology applies the principles when she attends a training class. A student of training who takes a scholarly class will learn about the principles that guide the methodologies and technologies she learned. The ideal student will have both training and scholarly education.

Similarly, a common perception is that research used to support training classes is more beneficial (or less beneficial) than scholarly research. Again, the goal of the research determines its usefulness. If the goal is to provide short-term results leading to improving existing technology, the research supported by (or influenced by) training organizations is appropriate. If the goal is to foster a deeper understanding of the problems, to obtain more effective and long-term solutions, or to explore new approaches or technologies, scholarly research is the more appropriate venue.

Scholarly education and training complement each other. Frequently, data from the research of training organizations provides information that scholarly research projects can use to test their ideas or techniques, or to suggest alternate paths of research. Similarly, training organizations can build research, and education, around the results of scholarly research projects to better understand new technologies and how those can be used to improve the state of security.

References

1. J. Saltzer and M. Schroeder, "The Protection of Information in Computer Systems," *Proc. IEEE*, vol. 63, no. 9, 1975, pp. 1278–1308.

2. Bastille Unix Web page, http://www.bastille-linux.org/ (current Mar. 2002).

3. M. Harrison, W. Ruzzo, and J. Ullman, "Protection in Operating Systems," *Comm. ACM*, vol. 19, no. 8, Aug. 1976, pp. 461–471.

4. EROS: The Extremely Reliable Operating System Web page, http://www.eros-os.org (current Mar. 2002).

5. D. Denning, "An Intrusion Detection System," *Proc. Symp. Security and Privacy*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1986, pp. 118–131.

Matt Bishop is an associate professor of computer science at the Department of Computer Science, the University of California at Davis. He is a charter member of the National Colloquium on Information Systems Security Education. Contact him at bishop@cs.ucdavis.edu.